Evolving Two-Dimensional Cellular Automata to Perform Density Classification: A Report on Work in Progress

Francisco Jimenez Morales[a], James P. Crutchfield[b], and Melanie Mitchell[b]
(a)Departamento de Física de la Materia Condensada. Universidad de Sevilla.
P. O. Box 1065, 41080-Sevilla, Spain.
(b) Santa Fe Institute, 1399 Hyde Park Road,
Santa Fe, New Mexico, 87501, USA.

March 12, 1998

Abstract

We present results from experiments in which a genetic algorithm (GA) is used to evolve two dimensional cellular automata (CA) to perform a particular computational task ("density classification") that requires globally-coordination information processing. The results are similar to that of earlier work on evolving one-dimensional CAs. The behavior of the evolved two-dimensional CAs is analyzed, and their performance is compared with that of several hand-designed two-dimensional CAs.

1 Introduction

In many natural systems, simple, locally-interacting components give rise to coordinated global information processing. In both natural and human-constructed information-processing systems, allowing global coordination to emerge from a decentralized collection of simple components has important potential advantages—e.g., speed, robustness, and evolvability—as compared with explicit central control. However, it is difficult to design a collection of individual components and their interaction in a way that will give rise to useful global information processing or "emergent computation". The term "emergent computation" refers to the appearance in a system's temporal behavior of information-processing capabilities that are neither explicitly represented in the system's elementary components.

In order to understand the mechanisms by which an evolutionary process can discover methods of emergent computation a simplified framework was proposed and studied by Crutchfield, Mitchell, and their colleagues [9, 2, 4, 3] in which a genetic algorithm (GA) evolved one-dimensional cellular automata (CAs) to perform computations. In their work the GA was able to discover CAs with high performance on tasks requiring cooperative collective behavior. In this paper we describe extending this work to two-dimensional CAs.

2 Cellular Automata

Cellular Automata (CAs) are regular lattices of variables, each of which can take a finite number of values ("states") and each of which evolves in discrete time steps according to a local rule that may be deterministic or probabilistic. Physical, chemical and biological systems with many discrete elements with local interactions can be modeled using CAs. The CAs discussed in this paper all use square lattices with L cells in each row and column. We denote the lattice size (i.e., number of cells) as $N = L\text{x}L$. A CA has a single fixed rule ϕ used to update each cell; the rule maps from the states in a neighborhood of cells to a single state s, which is the update value for the central cell in the neighborhood. The lattice starts out with an initial configuration of states and this configuration changes in discrete time steps. We use the term "state" of site i ($s_i(t)$) to refer to the local state of a single site at time t, and the term "configuration" ($\mathbf{s}(t)$) to refer to the entire lattice configuration of states. The transition rule ϕ that can be expressed as a look-up table (a "rule table"), which lists for each local neighborhood the updated state of the neighborhood's central cell. The CAs discussed here use the Moore neighborhood, which is depicted in the following picture:

1	2	3
4	**5**	6
7	8	9

Table 1: The Moore neighborhood

In this way a neighborhood can be thought of a string of 9 bit in the following way:

9	8	7	6	**5**	4	3	2	1

Table 2: The neighborhood as a 9-bit-string

The CA we will discuss here are two dimensional with two possible states per cell (0 or 1) and periodic boundary conditions. Note that the number of different neighborhood configurations is $2^9 = 512$.

3 A density-classification task for cellular automata

In [9], one-dimensional CAs were evolved by a GA to perform a density classification task called the "$\rho_c = 1/2$" task. (This built on earlier work by Packard [10].) A successful CA for this task will decide whether or not the initial configuration (IC) contains more than half 1s. If it does, the whole lattice should eventually iterate to the fixed point configuration of all cells in state 1; otherwise it should eventually iterate to the fixed-point configuration of all 0s. More formally, let ρ_0 denote the density of 1s in the initial configuration. If $\rho_0 > 0.5$ then within M time steps the CA should go to the fixed-point

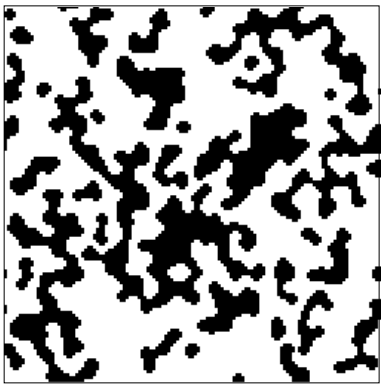

Figure 1: Final state (fixed point) of the majority rule ϕ_{maj} starting from an initial state with $\rho_0 \approx 0.51$. Lattice size is 127x127.

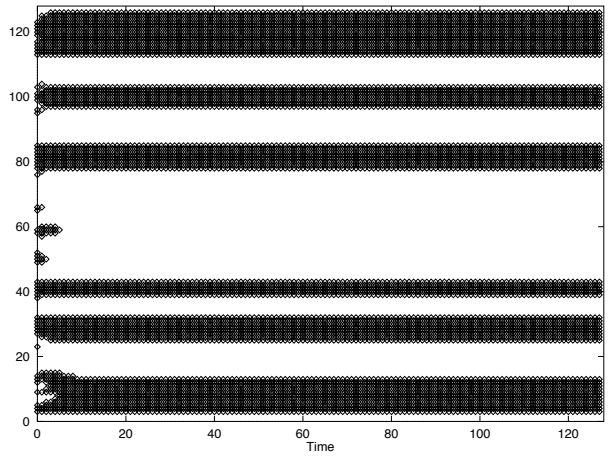

Figure 2: Cross-section of ϕ_{maj} at $x = 64$

configuration of all 1s; otherwise within M time steps the CA should go to the fixed-point configuration of all 0s.

As described in [9], designing an algorithm to perform the $\rho_c = 1/2$ task is trivial for a system with a central controller. However the task is not trivial for small-radius CA, since a small-radius CA relies only on local interactions. Because the 1s can be distributed throughout the CA lattice, the CA must transfer information over large distances ($\approx N$). To do this requires the global coordination of cells that are separated by large distances and that cannot communicate directly. The need for a coordination is illustrated by the two-dimensional "majority" rule ϕ_{maj}, in which a cell's state at time t is 1 if a majority of its neighbors are in state 1 and 0 otherwise. Figure 1 shows the final state of ϕ_{maj} beginning with $\rho_0 \approx 0.51$. When an all-1s region and an all-0s region border each other, there is no way to decide between them, and both persist.

In order to understand the computation performed by different CA rules evolved by the GA, we will use the tools of the "computational mechanics" framework developed for one-dimensional CAs by Crutchfield and Hanson [1, 6]. This framework describes the "intrinsic" computation embedded in the CA space-time configurations in terms of regular

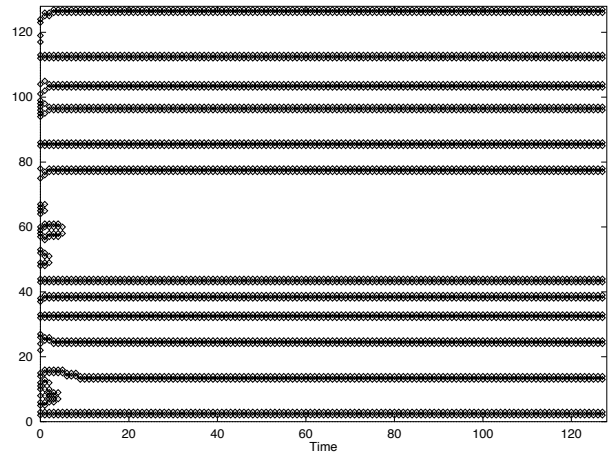

Figure 3: Filtered diagram of the cross-section for ϕ_{maj}.

domains, particles and particle interactions. Regular domains are homogeneous regions that are computationally simple to describe—i.e., they correspond to simple regular languages. In particular, a domain's "pattern" is described using the minimal deterministic finite automaton (DFA) that accepts all and only those configurations that appear in the domain. Such domains are called "regular" since their configurations are members of the regular language recognized by the DFA. Once a CA's regular domains have been detected, non-linear filters can be constructed to filter them out, leaving just the deviations from those regularities. The resulting filtered space-time diagram reveals the propagation of domain "walls". If these walls remain spatially-localized over time, then they are called particles.

Particles are one of the main mechanisms for carrying information over long space-time distances. In case of one-dimensional CAs, the space-time diagram is a two-dimensional surface, with particles forming a one-dimensional surface that can be easily displayed (see, e.g., [1, 2]). However, in the case of two-dimensional CAs, the space-time diagram is a three-dimensional surface, and particles form a two-dimensional surface, making display difficult. However when the regular domains consist simply of regions of all 1s or all 0s (black (B) and white (W) domains), as in Figure 1 as well as several of the evolved rules to be described here, some information can be obtained a space-time diagram of a one-dimensional cross-section of the CA that shows a subset of the time-iteration of the two-dimensional domains. For example, in Figure 2 the space-time diagram of a one-dimensional cross-section for the majority rule is shown. As can be seen, local neighborhoods with majority of 1s map to regions of all 1s and similarly for 0s, but when an all 1s region and an all-0s region border each other, there is no way to decide between them, and both persist. Though the exact shape of the domain wall can be very complicated, Figure 2 illustrates the existence of differentiated regions that persist over the time. Figure 3 displays a space-time diagram in which the regular domains (B and W) have been filtered out. For ϕ_{maj} the cross section filter diagram shows straight lines that do not cross each other. This illustrates the lack of coordination between different regions or information transfer needed to perform the $\rho_c = 1/2$ task.

An interesting variation of the majority rule was constructed by Gerard Vichniac [11].

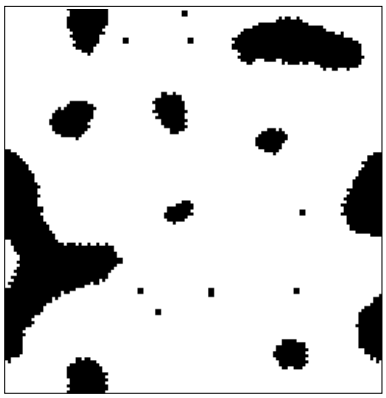

Figure 4: Snapshot at time $t = 2000$ of the evolution of the anneal rule ϕ_{anneal} beginning with $\rho_0 \approx 0.51$. Lattice size is 127x127.

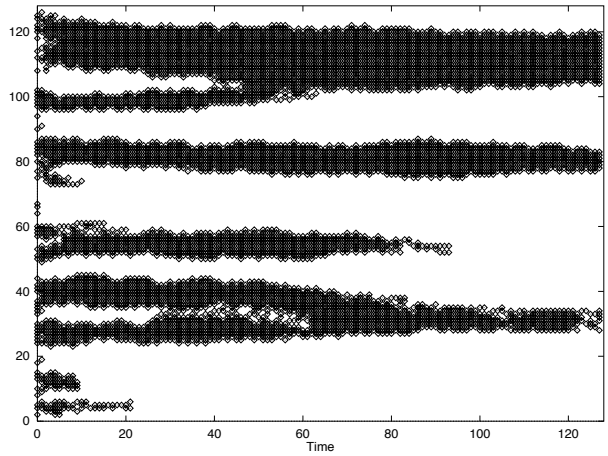

Figure 5: Cross-section of ϕ_{anneal} at $x = 64$.

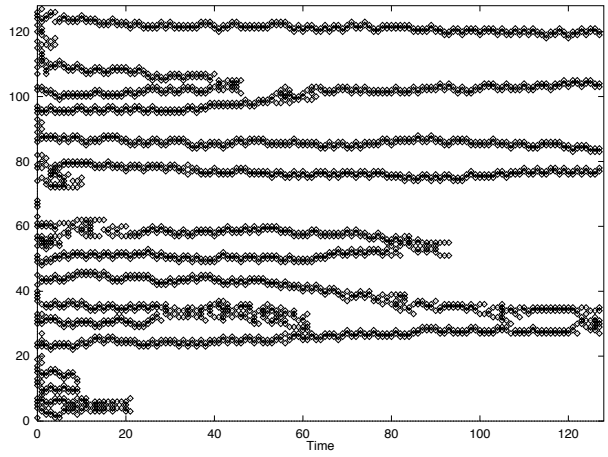

Figure 6: Filtered cross-section of ϕ_{anneal}.

It is defined as follows:

$$s_i(t+1) = \begin{cases} 0 & \text{if } S(t) < 4 \text{ or } S(t) = 5 \\ 1 & \text{if } S(t) > 5 \text{ or } S(t) = 4 \end{cases}$$

where $S(t) = \left[\sum_{neighbors} s_i(t)\right]$. By swapping the two table entries that are adjacent to the threshold (i.e., a neighborhood of four cells in state 1 will be mapped to 1 and a neighborhood of five cells in state 1 will be mapped to 0) one encourages re-shuffling at the boundary between 1s and 0s regions. The net effect is one of gradual annealing of domains: in the long term, each cell behaves as if the vote reflected not only the state of the immediate neighbors, but also that of cells that are further away from it. Domains form as in the majority rule but now the boundaries are in continual ferment (Figures 4–6). We call this rule ϕ_{anneal}. The filtered cross-section (Figure 6) shows how different domains are growing. This corresponds to a "block expanding"strategy, in which blocks of 0s or 1s grow so as to take over the lattice [9]. In contrast to ϕ_{maj}, in ϕ_{anneal} there is some interaction between different domains. But usually the final state of the CA under ϕ_{anneal} will contain some small islands of 1s inside a sea of 0s or small islands of 0s in a sea of 1s. Thus neither the majority rule nor the anneal rule perform the $\rho = 1/2$ task.

4 Details of Experiments

We used a genetic algorithm (GA) to evolve two-dimensional, binary state CAs to perform the $\rho_c = 1/2$ task. GA are search methods inspired by biological evolution. In a typical GA, candidate solutions to a given problem are encoded as bit strings ("chromosomes"). A population of such strings is chosen at random and evolves over several generations under selection, crossover and mutation. At each generation, the fitness of each bit string is calculated according to some externally imposed fitness function, and the highest-fitness bit strings are selected preferentially to be the "parents" who form a new population via crossover and mutation. Under crossover, pairs of parents exchange bits to form offspring, which are then subject to a small probability of mutation at each bit position. After several generations, the population often contains high-fitness bit strings representing high-quality solutions to the given problem. The search mechanism of a GA requires balancing two objectives: exploiting the best solution and exploring the search space.

The GA that we used begins with a population of 100 randomly generated chromosomes listing the rule-table output bits in lexicographic order of neighborhood patterns. The size of the rule space the GA searches in this case is 2^{512}. The fitness evaluation for each CA rule is carried out on a lattice of 21x21=441 cells (other experiments have been done with lattices sizes of 31x31, 41x41, 51x51, 61x61 and 71x71). A rule's fitness is estimated by running the rule on I randomly generated initial configurations (ICs) that are uniformly distributed over $\rho \in [0.0, 1.0]$. We allow each rule to run for a maximum number M of iterations. Here $M = 20 * L$. The rule's fitness $F_I(\phi)$ is the fraction of the ICs on which the rule produces the correct final pattern. No partial credit is given for partially correct final configurations.

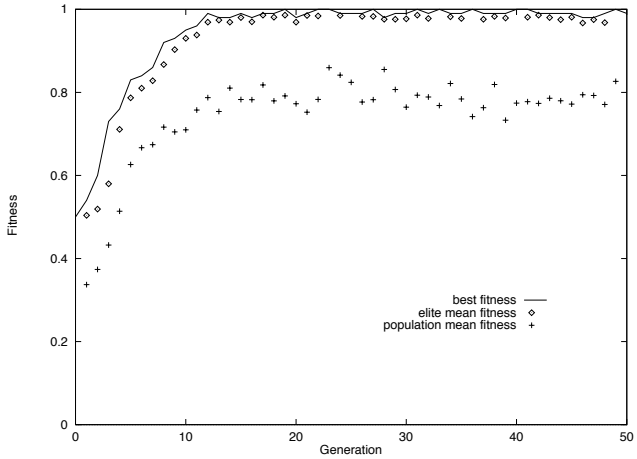

Figure 7: Best fitness, elite mean fitness, and population mean fitness versus generation for one typical run maded on a lattice of 21x21 cells; other lattice sizes gives the same plot. Only the firts 50 generations are shown.

In each generation: (i) A new set of ICs is generated. (ii) $F_I(\phi)$ is calculated for each rule ϕ in the population. (iii) The population is ranked in order of fitness. (iv) A number E of the highest fitness ("elite") rules is copied without modification to the next generation. (v) The remaining $P - E$ rules for the next generation are formed by single-point crossover between randomly chosen pairs of elite rules. The parent rules are chosen from the elite with replacement. The offspring from each crossover are each mutated with a probability m, where mutation consists of flipping a randomly chosen bit in a string. This defines one generation of the GA; it is repeated G times for one run of the GA.

Since $I << 2^N$, the fitness function $F_I(\phi)$ is only an estimate of the "exhaustive performance "— the performance that would be measured by exhaustively testing a CA on all 2^N ICs. $F_I(\phi)$ is a random variable, since the precise value it returns for a given rule depends on the particular set of ICs used to test the rule. Thus, a rule's fitness can vary stochastically from generation to generation. For this reason, at each generation the entire population, including the elite rules, is re-evaluated on a new set of ICs. See [9] for a discussion of this version of the GA.

5 Results

We performed more than 100 different runs of the GA with the following parameters: for each CA in the population $P = 100$; $E = 10$; $I = 100$; $m = 0.016$; $G = 100$ (in some runs G was set to 400), each with a different random-number seed. The dynamics of a typical run is shown in Figure 7 which plots the best fitness rule, the elite mean fitness and the population mean fitness versus the generation for the first 50 generations. Before the GA discovers high fitness rules, the fitness of the best CA rule increases in rapid jumps. Qualitatively, the rise in performance can be divided into several "epochs", each corresponding to the discovery of a new, significantly improved strategy. In our experiments the different epochs appear in the first 30 generations, by the end of which the GA

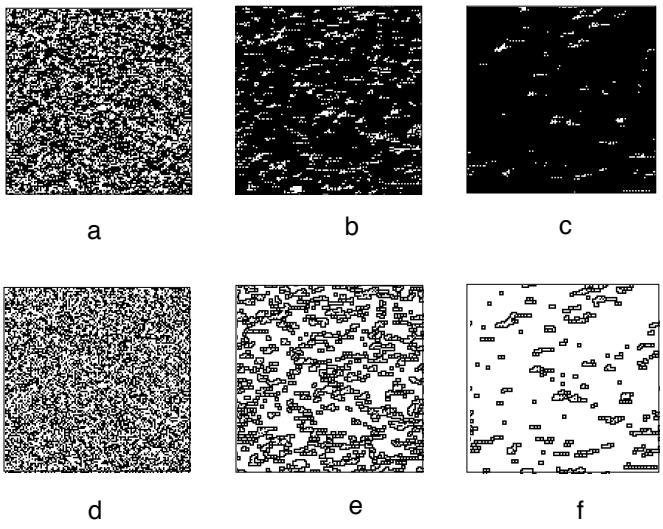

Figure 8: Snapshots of the evolution of rule ϕ_3 and a filtered diagram of the evolution at time: (a-d) t=0 , (b-e) t=4, (c-f) t=8. Initial condition is $\rho_0 \approx 0.51$.

typically has discovered rules with fitness ≈ 0.9. In general we found that running the GA for more generations did not result in improved fitness.

On most runs the GA evolved a rather unsophisticated class of strategies that we have called "block-expanding" strategies. We will compare these with ϕ_{maj} and ϕ_{anneal}.

Here we describe the behavior of the best rules at each epoch in the run plotted in Figure 7.

Epoch 1

The first epoch starts at generation 0, when the best fitness in the initial generation is 0.5 and the λ values are uniformly distributed between 0.0 and 1.0 (The λ parameter [7] of a given CA rule is the fraction of non zero output states in the rule table). In this epoch rules default to either all 0s or all 1s, respectively making correct classifications on half the ICs, and then the highest fitness is 0.5.

In the following figures the lattice size is taken as 127x127 and the CA starts from a random initial concentration or from some other configurations such as a black square or a black triangle. We display snapshots of the evolution and some space-time diagrams of the best fitness rule of the generation in question.

Epoch 2

In generation 3, a new strategy (ϕ_3) ($\lambda \approx 0.61$) is discovered, resulting in significantly better performance: $F_{100}(\phi_3) = 0.73$. The behavior of this rule reveals that it always defaults to the all 1s configuration except when $\rho_0 \approx 0$, in which case it iterates to the all 0s configuration. This second epoch begins when a rule is discovered in which most neighborhood patterns in the rule table that have $\rho < \rho_c$ map to 0 and most neighborhood

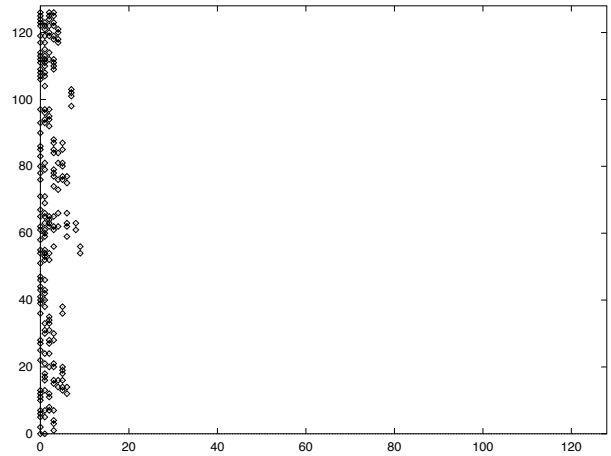

Figure 9: Space-time diagram of a filtered cross section of rule ϕ_3 at $x = 64$ starting with a random initial condition.

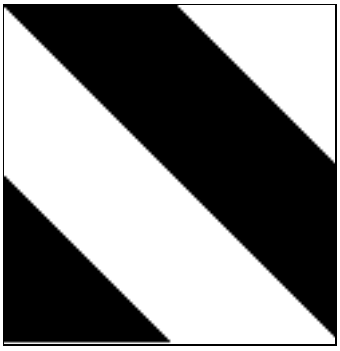

Figure 10: Space-time diagram of a cross section of rule ϕ_3 at $x = 64$ starting with an initial condition that consist of a vertical black and a white domain. Time goes from left to right.

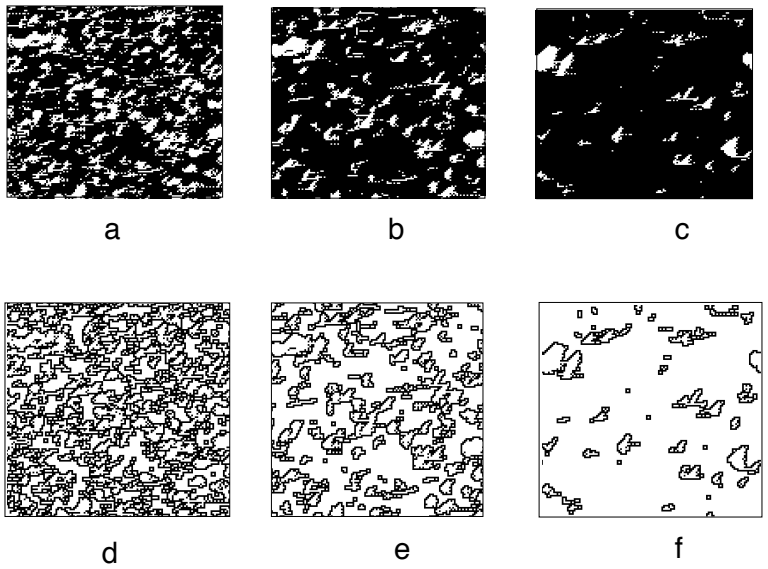

Figure 11: Snapshots of the evolution of rule ϕ_8 and a filtered diagram of the evolution at time: (a-d) t=3 , (b-e) t=6, (c-f) t=20. Initial condition is $\rho_0 \approx 0.51$.

patterns in the rule table that have $\rho > \rho_c$ map to 1. Snapshots of the evolution for the fittest generation 3 rule ϕ_3, starting from a random initial condition, is shown in Figure 8. There are small white domains that move along the diagonal with velocity of **-(i+j)** using vectorial notation. In Figure 9 the filtered diagram of a cross-section for the space-time behavior of ϕ_3 is shown. The time that the CA needs to go to a fixed configuration is less than 20 time steps.

In terms of computational mechanics, the strategy used by ϕ_3 can be explained as follows: ϕ_3 creates two domains: all 1s (B) and all 0s (W). Though the particles that are created between these domain walls depend on the angle between them, some information can be obtained from simple arrangements such as domain walls at 180 degrees (horizontal walls, configurations of cells: 000-111-111 , 111-000-000, 000-000-111,111-111-000) and at 90 degrees (vertical walls, configurations of cells: 100-100-100 , 110-110-110, 011-011-011, 001-001-001). The velocities of these particles are given in Table 3. For ϕ_3 we can see that the particles P_{HWB} and P_{HBW} both have a velocity of **-j** and particles P_{VWB} and P_{VBW} a velocity of **-i** . This combination of velocities between these domain walls causes the irregular domains to move along the diagonal with velocity **-(i+j)**. In Figure 10 a cross-section from an initial condition that consists only of a vertical B and W domain can be seen. In this plot time increases from left to right and the vertical axis represents the positions at which the cross-section is taken. This configuration is dynamically stable but irregularly shaped W domains inside a larger B domain begin to move along the diagonal and this is accompanied by a shrinking until the W domain disappears.

Epoch 3

In generation 8 a new epoch begins with the discovery of ϕ_8, where the sharp jump in fitness ($F_{100}(\phi) = 0.92$) corresponds to a significant innovation. The typical evolution

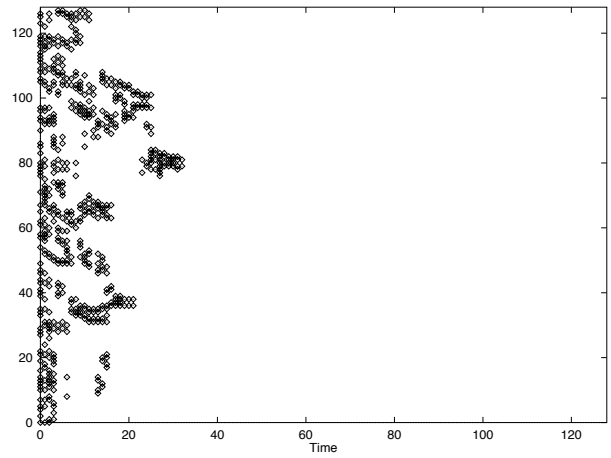

Figure 12: Space-time diagram of a filtered cross section of rule ϕ_3 at $x = 64$ starting with a random initial condition $\rho_0 \approx 0.51$.

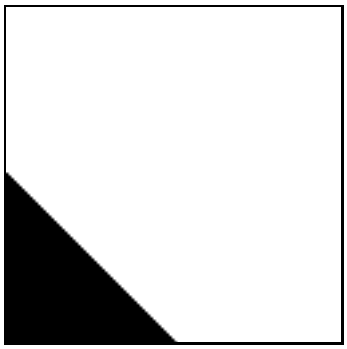

Figure 13: Space-time diagram of a cross section of rule ϕ_8 at $x = 64$ starting with an initial condition that consist of a vertical black and a white domain. Time goes from left to right.

ϕ_8 is illustrated in Figure 11. The small white regions are now larger than in the previous epoch, and they still move along the diagonal. In Figure 12 the filter diagram of a cross-section for ϕ_8 is shown. The time that the CA needs to reach a fixed configuration is increased (≈ 50 time steps). For ϕ_8 the velocities of P_{HWB}, P_{HBW} and P_{VWB} are the same as for ϕ_3, but the velocity of P_{VBW} is now 0. In this way small islands of black in a sea of white will shrink and disappear. This can be seen in Figure 13, which started from an initial condition with a vertical B and W domain. Where the strategy of ϕ_3 was to allow some white islands to survive, ϕ_8 discovers a way to expand white islands and thus correctly classify more ICs with $\rho_o < 1/2$.

Wall type	**Particle**	ϕ_3	ϕ_8	ϕ_{12}	ϕ_{208}	ϕ_{301}
Horizontal-WB	P_{HWB}	**-j**	**-j**	**-j**	**j**	**-j**
Horizontal-BW	P_{HBW}	**-j**	**-j**	**-j**	**j**	**-j**
Vertical-WB	P_{VWB}	**-i**	**0**	**0**	**-i**	**i**
Vertical-BW	P_{VBW}	**-i**	**-i**	**-i**	**-i**	**0**

Table 3: The particles and their velocities generated by the best rules in different generations. **i** and **j** are unit vectors along the x and y axe respectively.

Epoch 4

The best rule of generation 12, ϕ_{12}, has $\lambda = 0.52$ and $F_{100}(\phi) = 0.99$. Figure 14 illustrates its typical evolution. Now the stability of W domains is higher than in the previous epochs, and W domains continue to move along the diagonal. The boundaries between black and white domains are now in "ferment" as in the anneal-rule. The filtered space-time diagram of a cross-section is shown in Figure 15; this displays the domain walls. In previous generations a fixed state is obtained quickly because λ is a high value but for ϕ_{12} there is a more balanced situation and the time that the system needs to go to a fixed state is higher. The velocities of the particles P_{HWB}, P_{HBW}, P_{VWB} P_{VBW} for ϕ_{12} are the same that for ϕ_8. The differences between ϕ_{12} and ϕ_8 concern different types of domain walls. Consider domains with the shape of a square triangle, i.e., there is a domain wall at 45 degrees and at 135 degrees. There are four different arrangements with the squared angle of the triangle at position (0,0), (0,1),(1,1) and (1,0) (these arrangements will be labeled as t1, t2, t3, and t4). Figure 16 displays the space-time behavior of ϕ_8 starting with an initial configuration containing a black t1 inside a sea of white cells. Here, a cross-section of the two-dimensional CA has been taken at a position that coincides with the domain wall W-B that is fixed. As the P_{HBW} moves with velocity **+j** and the new P_{t1WB} moves down, the island of black cells shrinks until it disappears. The strategy implemented by ϕ_{12} (Figure 17) is completely different. Here the new particle P_{t1WB} transforms into different particles until a vertical P_{VBW} is formed. This has a velocity of **-i** that quickly reaches the other side and the t1 vanishes abruptly.

In summary: In this run, all of the evolved rules have a $\lambda > 0.5$. This means that most neighborhoods are mapped to 1. Thus, initial conditions with $\rho_o > \rho_c$ will be easily

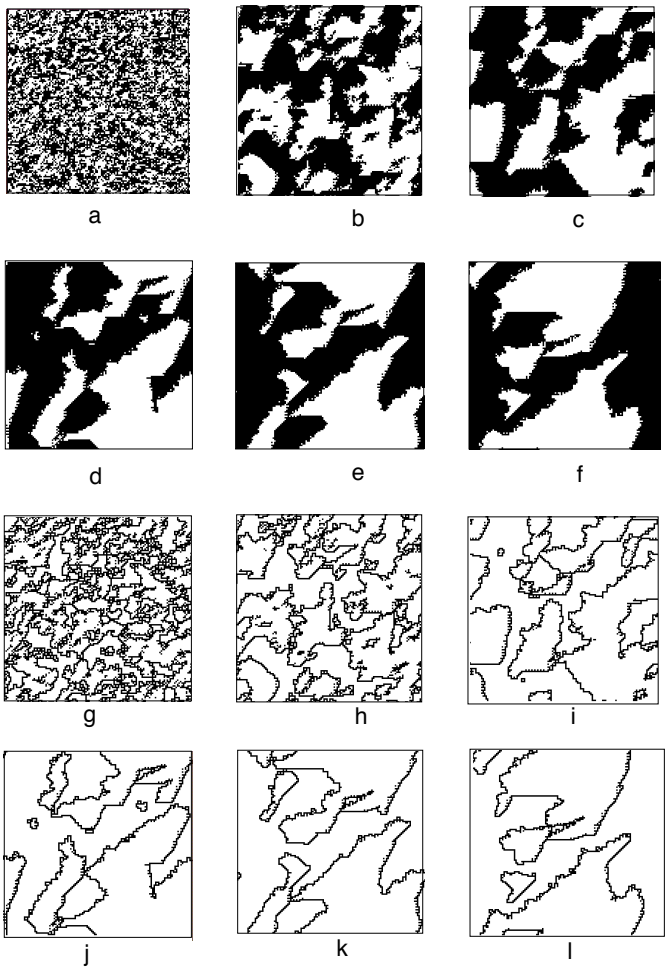

Figure 14: Snapshots of the evolution of rule ϕ_{12} and a filtered diagram of the evolution at time: (a-g) t=0 , (b-h) t=20, (c-i) t=45, (d-j) t=75, (e-k) t=100, (f-l) t=150. Initial condition is $\rho_0 > 0.5$.

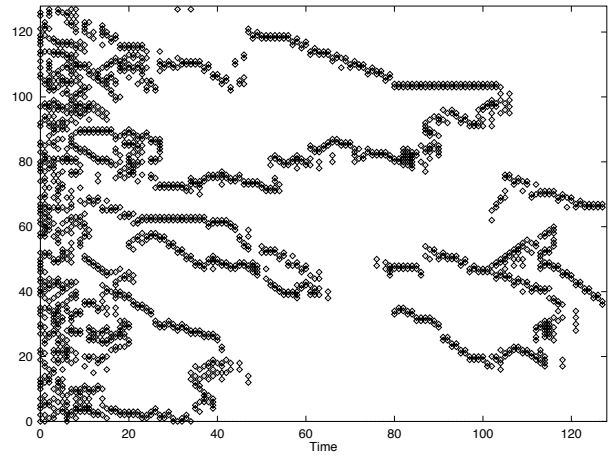

Figure 15: Space-time diagram of a filtered cross section of rule ϕ_{12} at $x = 64$ starting with a random initial condition.

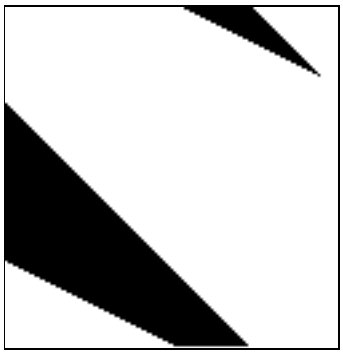

Figure 16: Space-time diagram of a cross section of rule ϕ_8 at $x = 64$ starting with an initial condition that consist of a black triangle t1 into a sea of white cells. Time goes from left to right.

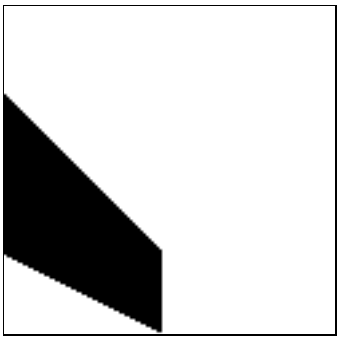

Figure 17: Space-time diagram of a cross section of rule ϕ_{12} at $x = 64$ starting with an initial condition that consist of a black t1 into a sea of white domain. Time goes from left to right.

classified correctly, but many initial conditions with $\rho_o < \rho_c$ will be misclassified. The best evolved rules are those that have found some kind of mechanism to correctly classify initial conditions with $\rho_o < \rho_c$. As can be seen in Table 3, the different particles described allow W domains to survive or expand inside larger B domains.

CA	**Rule Table Hexadecimal code**	**Symbol**	**N = 441**	**N = 961**	**N = 1681**
Majority	00000001-00010117-00010117-0117177f 00010117-0117177f-0117177f-177f7fff 00010117-0117177f-0117177f-177f7fff 0117177f-177f7fff-177f7fff-7fffffff	ϕ_{maj}	0	0	0
Anneal	00010116-01161669-01161669-16696997 01161669-16696997-16696997-6997977f 01161669-16696997-16696997-6997977f 16696997-6997977f-6997977f-977f7fff	ϕ_{ann}	0.359	0.278	0.215
B-Expand-1	00000000-00102010-00000041-00246004 20050032-00048320-12adee6e-d7d2d6f7 4fb7fba8-51d7f7ff-7ca7f5f7-cfdf7fff 5fe37fe5-fffffff-ed6fefff-95ffff9f	ϕ_{12}	0.660	0.653	0.639
B-Expand-2	0122043b-0134057d-453c380f-2374df9d 122ee407-56d07dcb-6247c506-07df27ff 04255459-0d8b91cf-44617bb7-5c7f477f 2b1e45ed-e3870f27-2b7c2337-3fff877f	ϕ_{208}	0.679	0.654	0.611
B-Expand-3	0003606c-00cd4096-1117160a-87272841 10110225-759e457f-29d96537-9f6fbf7f 173652be-45ab1369-913f8849-7b7cb3df 00723337-537ffd7f-b38f97fd-575fddd7	ϕ_{301}	0.690	0.652	0.641
Non-Local 3-majority		$\phi_{NL}(r = 10)$	0.794	0.789	0.782
GKL		ϕ_{GKL}	0.873	0.854	0.842

Table 4: Measured values of $P_{10^4}^N$ at various values of N for different rules: the majority rule, the anneal rule, three of the rules discovered by the GA in different runs, a non-local majority vote among three different cells taken inside a Moore neighborhood of radius $r = 10$, and the two-dimensional GKL rule. To recover the 512-bit string giving the output bits of the rule table, expand each hexadecimal digit to binary. The output bits are then given in lexicographic order.

The best rules discovered by the GA in all runs are : ϕ_{301}, $\lambda = 0.51$ and ϕ_{208}, $\lambda = 0.51$. In Figure 18 snapshots of the evolution of ϕ_{208} starting from a random initial condition is shown. For this rule the previous movement of white regions inside a sea of black cells no longer exists. The filtered space-time diagram, shown in Figure 19, shows that at the boundaries between domains there is some kind of transmission of information in the sense of complex structures that propagate during short periods of time and for small distances (Figure 18d,f). This can be seen clearly from an initial condition that consists of a triangle t3 inside a white sea (Figure 20) where a group of three vertical straight lines of (0-0-0-0),(1-1-1-1) and (0-1-0-1) cells is propagating from right to left. In Figures 21 and 22, a cross section of ϕ_{208}'s space-time diagram made at positions (y=94 and y=98) is shown. In 22 it can be observed that when the structure arrives to the other side of the B domain the boundary is destroyed more rapidily.

The typical evolution of ϕ_{301} is shown in Figure 23. As in the previous rules ϕ_{301}'s strategy primarily consists of expanding blocks of white(black) inside a sea of black(white)

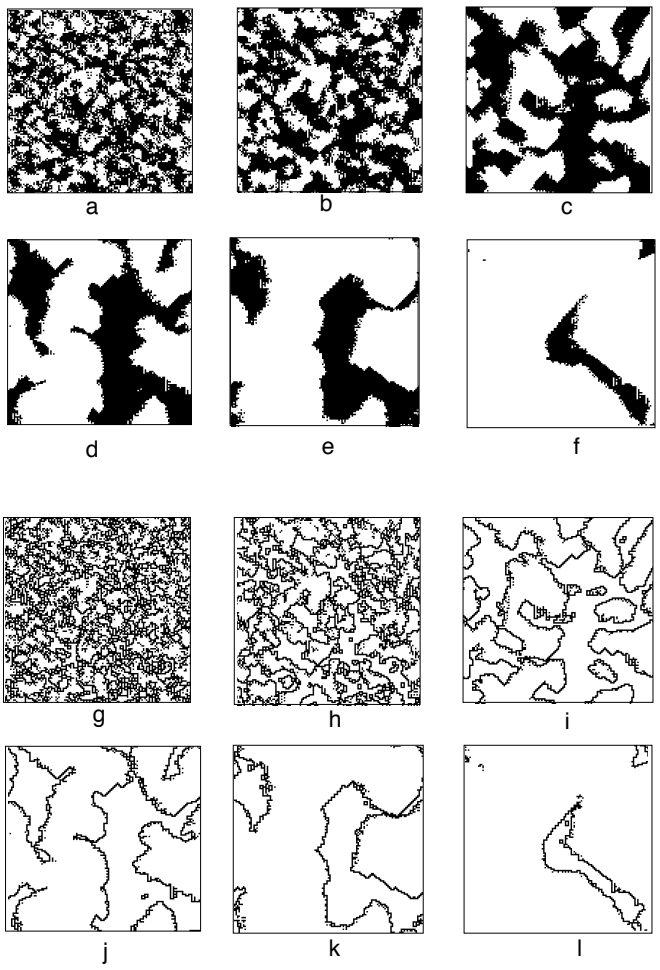

Figure 18: Snapshots of the evolution of rule ϕ_{208} and a filtered diagram of the evolution at time: (a-g) t=5 , (b-h) t=10, (c-i) t=30, (d-j) t=60, (e-k) t=140, (f-l) t=330. Initial condition is $\rho_0 < 0.5$.

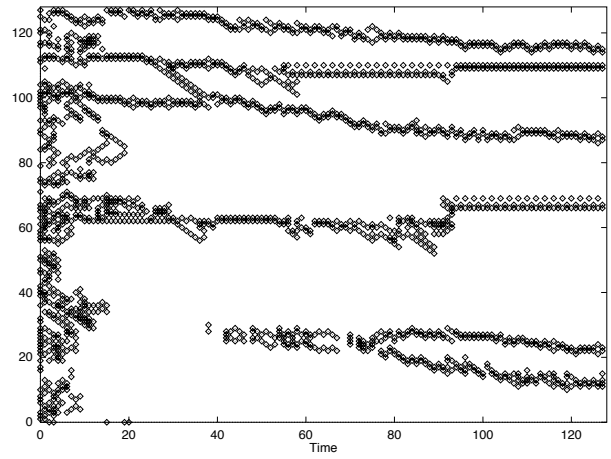

Figure 19: Space-time diagram of a filtered cross-section of rule ϕ_{208}.

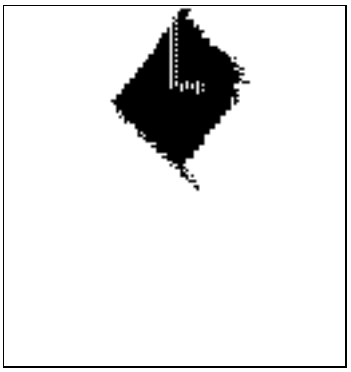

Figure 20: Snapshot of the evolution of rule ϕ_{208} starting with an initial condition that consists on a black t3 triangle into a sea of white cells at time $t = 34$. Inside the black domain it can be observed a structure that is moving to the left (**-i**).

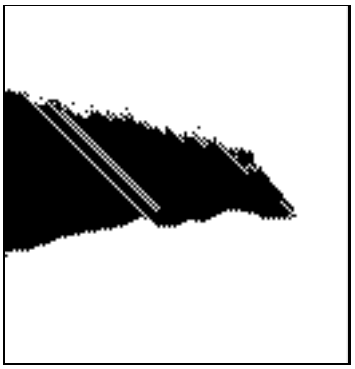

Figure 21: Cross section of rule ϕ_{208} at position $y = 94$ corresponding to Figure 20

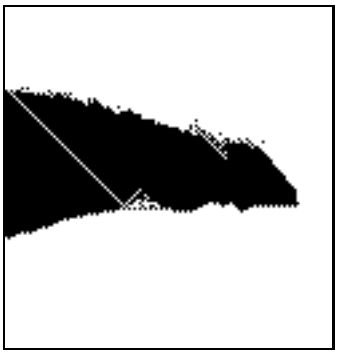

Figure 22: Cross section of rule ϕ_{208} at position $y = 98$ corresponding to Figure 20

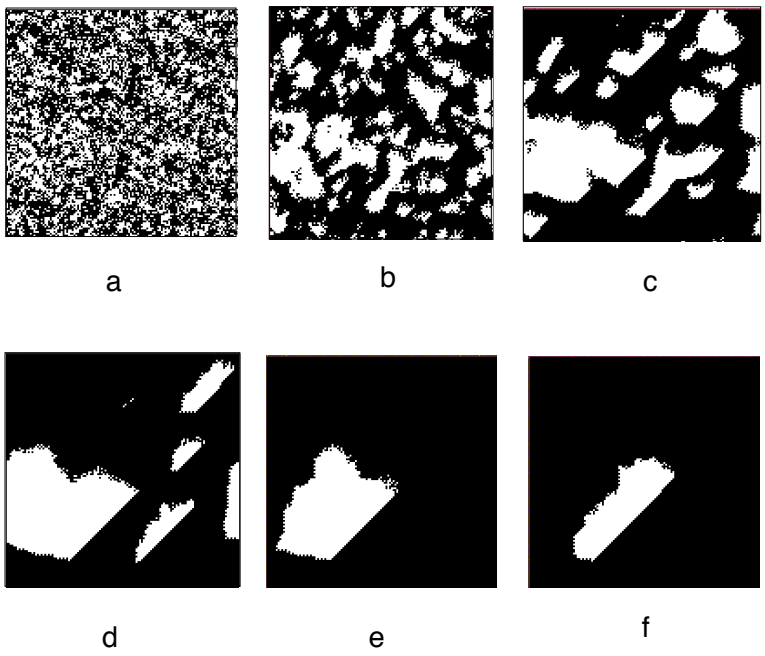

Figure 23: Snapshots of the evolution of rule ϕ_{301} at time: (a)t=0, (b) t=10, (c) t=40, (d) t=70, (e) t=160, (f) t=350. Initial condition is $\rho_0 > 0.5$.

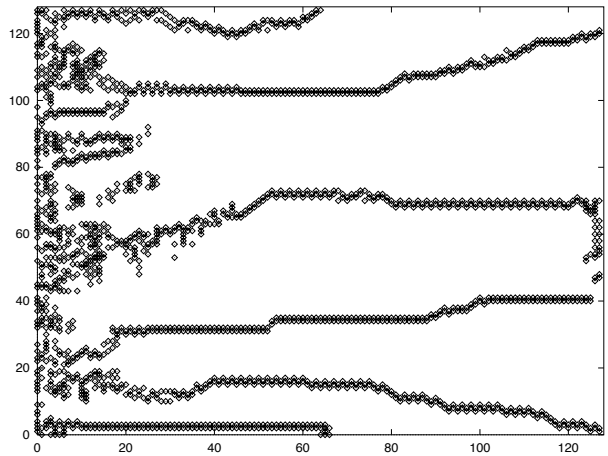

Figure 24: Space-time diagram of a cross section of rule ϕ_{301} at $x = 64$

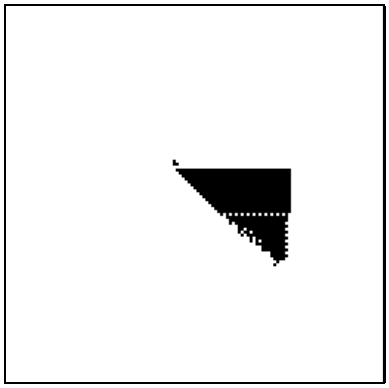

Figure 25: Snapshot of the evolution of rule ϕ_{301} starting with an initial condition that consists on a black t3 triangle into a sea of white cells at time $t = 24$. Inside the black domain it can be observed a structure that is moving up with velocity $(+\mathbf{j})$

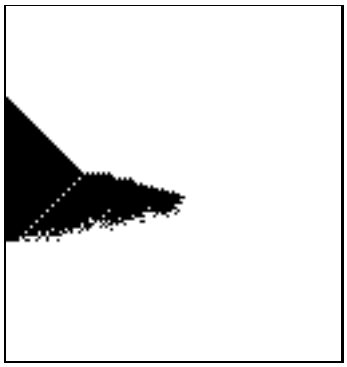

Figure 26: Cross section of rule ϕ_{301} at position $x = 90$ corresponding to Figure 25

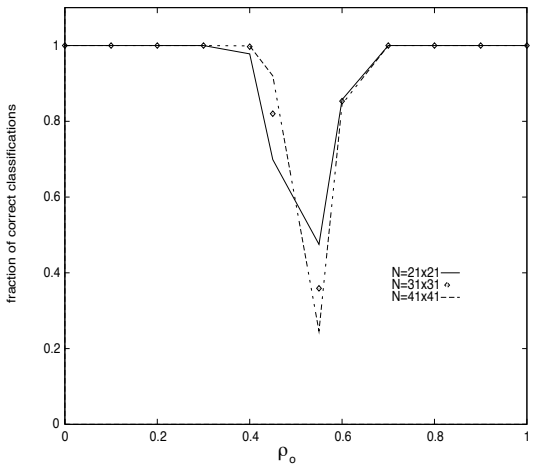

Figure 27: Fraction of correct classifications versus ρ_0 maded by rule ϕ_{301} for three lattice sizes: $21x21$, $31x31$ and $41x41$.

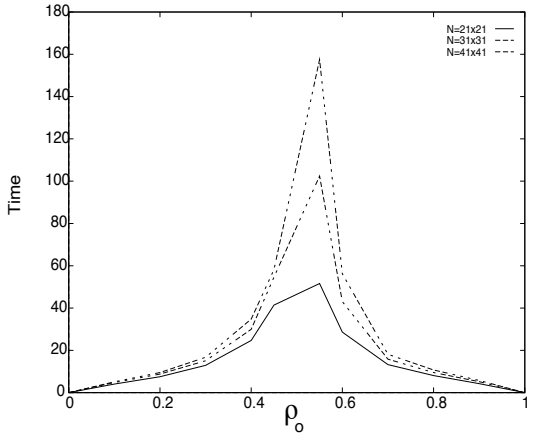

Figure 28: Averaged time to go to a fixed configuration versus ρ_0 for rule ϕ_{301} for three different lattice sizes: $21x21$, $31x31$ and $41x41$.

("block-expanding"). The difference between ϕ_{301} and previous rules is the existence of fixed boundaries. As can be seen in Figure 23d,e, and f, in order to eliminate a white domain inside a black sea, the rule has found a fixed boundary; thus the process of elimination is not done along the whole perimeter of the domain. For this type of block-expanding rule, information must be processed along the boundaries between white and black domains, and the strategy of ϕ_{301} consists mainly in reducing such boundaries.

Starting from an initial condition such a black triangle (t3) inside a white sea it can be observed (Figure 25) that the particles P_{VBW} and P_{t3BW} have zero velocity, and the shrinking of the black domain is mainly done through the movement of P_{HWB}. There is also a horizontal straight line of cells (1-0-1-0) that propagates inside the black domain with a velocity of **j**. The effect of this propagating structure is interesting because it is a way of sending information between two sides of a black domain, and in this way the information is not only processed along the boundaries. In Figure 26 a cross-section taken at the plane x=90 shows the way in which the structure propagates from side to side. The lower part of the Figure 26 corresponds to the vertical B and W boundary and the upper to the horizontal W and B which is moving down in Figure 25. When the propagating structure collides with P_{HWB} , the structure is absorved and the velocity of P_{HWB} is lowered. Plot of the fraction of correct classifications made by this rule versus ρ_0 for three different lattice sizes are shown in Figure 27. All the misclassifications occur around ρ_c and the error region decreases as lattice size increases. In Figure 28, the average time that the CA needs to iterate to a fixed point is also shown. As can be seen from the Figure 27 and Figure 28 rule, $F_{100}(\phi_{301})$ does not scale well with lattice size. The performance decreases for larger lattice sizes since the block to expand was tuned by the GA for $N = 441$.

Performance of the Best Rules

Under $F_{100}(\phi)$ the fittest rules evolved by the GA obtained fitnesses between 0.9 and 1.0 for different sets of ICs. A more indicative performance measure is the "unbiased performance", $P_I^N(\phi)$, defined as the fraction of I ICs chosen from an unbiased distribution over ρ on which ϕ produces the correct final pattern on a lattice of size N after $2 * N$ time steps. With an unbiased distribution most ICs chosen have $\rho \approx 0.5$. Table

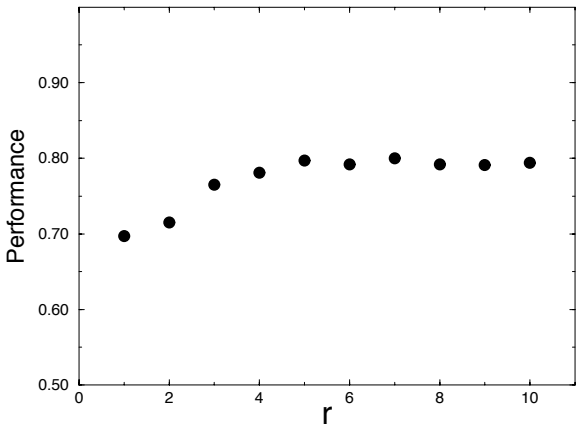

Figure 29: Performance of the non-local majority rule ϕ_{NL} versus r.

4 shows the performance of some of the rules evolved by the GA, as well as ϕ_{maj} and ϕ_{anneal}. The highest measured $P^N_{10^4}(\phi)$ of the GA evolved rules was approximately 0.69 for a lattice of 441 cells for the rule ϕ_{301}. This value is higher than the performances of the block-expanding rules found in one dimension, whose values are around 0.65 for a lattice of 149 cells [9]. In both one and two dimensions, as N increases the performance of the rules decreases.

Table 4 also lists the performance of a non-local majority rule vote among 3-cells ϕ_{NL} that is defined as follows:

$$s_i(t+1) = \text{majority } [\ s_i(t), s_j(t), s_k(t)]$$

where $s_j(t)$ and $s_k(t)$ $(i \neq j \neq k)$ are picked randomly inside a Moore-neighborhood of radius r around $s_i(t)$. Many properties of non-local CAs were investigated by Li [8]. The performance of ϕ_{NL} versus r is shown in Figure 29. For $r = 5$ the performance of this rule seems to be not dependent on r and has a value ≈ 0.79. However the best performance rule for the $\rho = 1/2$ task seems to be a two dimensional version of the GKL rule [5, 9].

The Two-Dimensional GKL rule

The one-dimensional Gacs,Kurdyumov and Levin (GKL) CA is defined by the following rule:

$$\text{If } s_i(t) = 0, \text{ then } s_i(t+1) = \text{majority}[\ s_i(t), s_{i-1}(t), s_{i-3}(t)]$$
$$\text{If } s_i(t) = 0, \text{ then } s_i(t+1) = \text{majority}[\ s_i(t), s_{i+1}(t), s_{i+3}(t)]$$

For the two-dimensional version, this rule is applied during one time step in the horizontal direction and in the next time step in the vertical direction. The rule has a $\lambda = 1/2$ and two absorbing states, i.e., the final state consists of every cell 0 or every cell 1. From an initial condition the GKL evolves towards one of the absorbing states and there is a

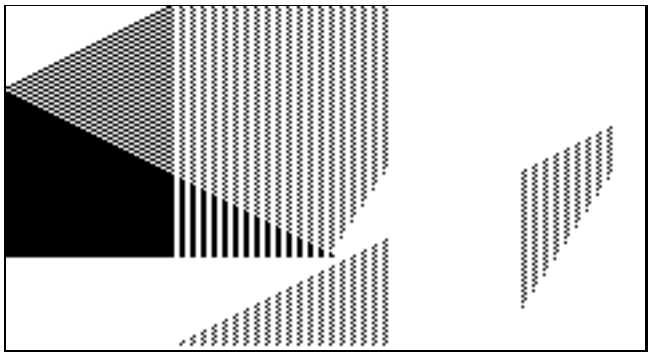

Figure 30: Space-time behavior of a cross section of the GKL rule maded at position $x = 64$, starting from an initial condition that consists of a square black inside a sea of white cells. The initial condition has $\rho_0 < 0.5$. Time goes from left to right.

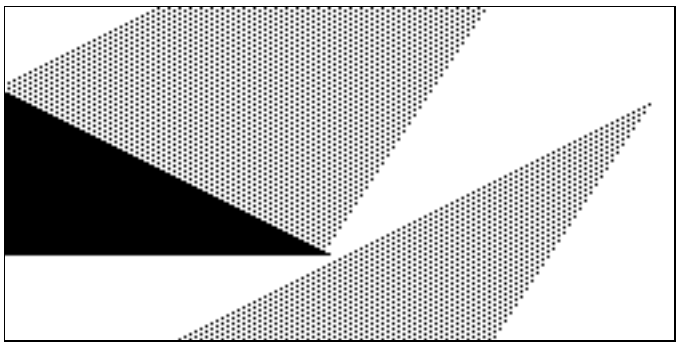

Figure 31: Space-time behavior of a cross section of the GKL rule maded at the diagonal ($x = y$) corresponding to Figure 30

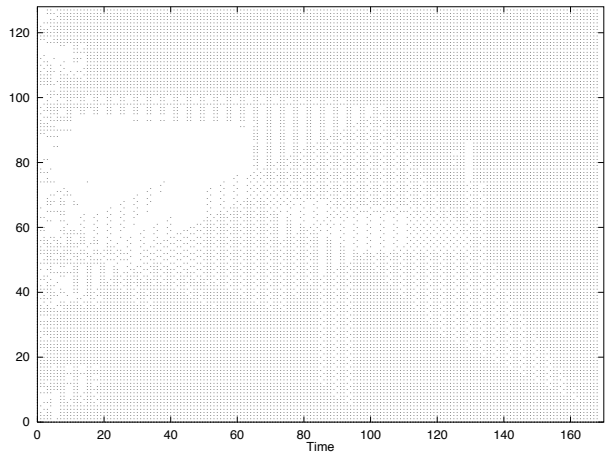

Figure 32: Space-time behavior of a cross section of ϕ_{GKL} at position $x = 64$ starting from a random initial condition $\rho_0 > 0.5$.

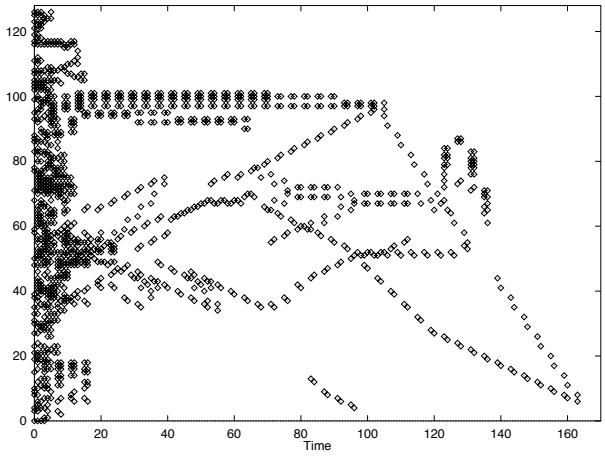

Figure 33: Filtered diagram corresponding to Figure 32

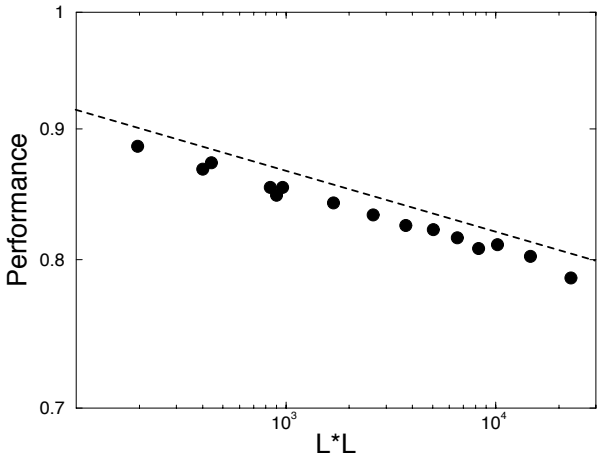

Figure 34: Log-log plot of the performance of ϕ_{GKL} versus the number of cells. $P^N_{10^4}(\phi_{GKL}) \sim N^\alpha$ with $\alpha \approx -0.06$.

transient during which spatial and temporal transfers of information take place around local neighborhoods. The GKL CA classifies local regions and the classified region grows with time. In regions where there is some ambiguity a "signal" is propagated. This is seen either as checkerboard patterns or as white-to-black boundaries , both with $\rho = \rho_c$. These signals indicate that the classification is to be made at a larger scale.

In Figure 30 the initial condition consists of a square black region inside a sea of white cells (the initial condition has $\rho < 1/2$ so that the final configuration should be a W domain). We took a cross-section at the mean point while in Figure 31 the cross-section is taken along the diagonal. As can be seen, at the boundary between B and W domains a particle P_{BW} is formed and produces a checkerboard-like pattern. For the GKL rule there are at least 18 different regular domains. A cross-section space-time diagram and a filtered version of that diagram for a random initial condition are shown in Figures 32 and 33 . For this rule there is transmission of information between distant regions of the CA. The performance of the GKL is 0.873 for a lattice size of 441 and as the lattice size increases the performance decreases slowly with a scaling exponent of ≈ -0.06 as can be seen in Figure 34. We are currently working on a computational-mechanics analysis of this rule's computational strategy.

6 Conclusion

To date, the best two-dimensional rules evolved by the GA in our experiments implement unsophisticated "block-expanding" strategies Although they have higher performance than that of simple hand-designed rules such as the majority or the anneal-rule, their performance is lower than that of the two-dimensional GKL rule and the non-local majority rule, and that of the high-performance rules evolved by the GA in the one-dimensional case [2, 4].

Why did the GA not find higher-performance rules with much better performance ?

Some of the impediments that the GA has in evolving one-dimensional CAs to perform the $\rho = 1/2$ have been discussed in [10]. In the case of two-dimensional CAs, some of the possible impediments are:

a)The breaking of symmetries in early generations for short-term fitness gain [9]. In two-dimensional CAs this is amplified by the fact that the block-expanding rules in two dimensions have higher fitness than in one-dimensional CAs.

b) The symmetries in the bit-string encoding may influence that the search strategy of the GA. For example, consider neighbors such as those represented in Table 5 and 6:

1	0	0
0	0	0
0	0	0

Table 5: Neighbor at the 2-position

0	0	0
0	0	0
0	0	1

Table 6: Neighbor at the 257-position

These two neighborhood configurations are 255 positions apart from one another other in the rule-table, but considering a simple rotation symmetry they represent the same neighborhood.

c) The rules that have the highest performance, the GKL and the non-local majority vote rules, have radius r greater than 1 The CAs in our GA experiments had $r = 1$. It may be the case that there are no $r = 1$ CAs that perform the $\rho = 1/2$ task well.

What needs to be done next?

One possible way to improve the results of our GA experiment is to modify the encoding of look-up tables so that neighborhoods that are the same when considering the different symmetries are mapped to the same value in the rule table. Also the neighborhood ordering in the look up table could be modified so that very high and very low density neighborhoods are close together. Another possibility is to encode CA look-up tables as arrays instead of as simple bit-strings and to have crossover and mutation could work directly on such structures. Finally, experiments should be done using CAs with an increased neighborhood radius. The next step in this work-in-progress is to experiment with these modifications.

Acknowledgments

Many thanks to Rajarshi Das and Wim Hordijk for their assistance and suggestions. This research was supported by the Santa Fe Institute, under grants NSF-IRI-9705830 and ONR N00014-95-1-1000. FJM was also supported by the Spanish Ministerio of Educacion y Ciencia grant no. PR97-2854545R.

References

[1] J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, 69:279–301, 1993.

[2] J. P. Crutchfield and M. Mitchell. The evolution of emergent computation. *Proceedings of the National Academy of Science U.S.A.*, 92:10742–10746, 1995.

[3] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.

[4] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature—PPSN III*, volume 866, pages 344–353, Berlin, 1994. Springer-Verlag (Lecture Notes in Computer Science).

[5] P. Gacs, G. L. Kurdyumov, and L. A. Levin. One-dimensional uniform arrays that wash out finite islands. *Probl. Peredachi. Inform.*, 14:92–98, 1978.

[6] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of Statistical Physics*, 66(5/6):1415–1462, 1992.

[7] C.G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D*, 42:12–37, 1990.

[8] W. Li. Phenomenology of non-local cellular automata. *J. Stat. Phys.*, 68(5/6):829–882, 1992.

[9] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361 – 391, 1994.

[10] N. H. Packard. Adaptation toward the edge of chaos. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293–301, Singapore, 1988. World Scientific.

[11] G. Vichniac. Cellular automata models of disorder and organization. In Bienenstock et al., editor, *Disordered Systems and Biological Organization*, pages 1–20, Berlin, 1986. Springer-Verlag.