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

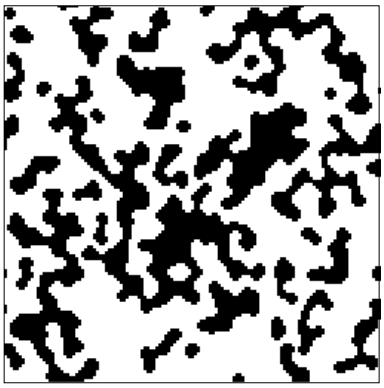

Figure 1: Final state (fixed point) of the majority rule ϕ_{maj} starting from an initial state with $\rho_0 \approx 0.51$. Lattice size is 127x127.

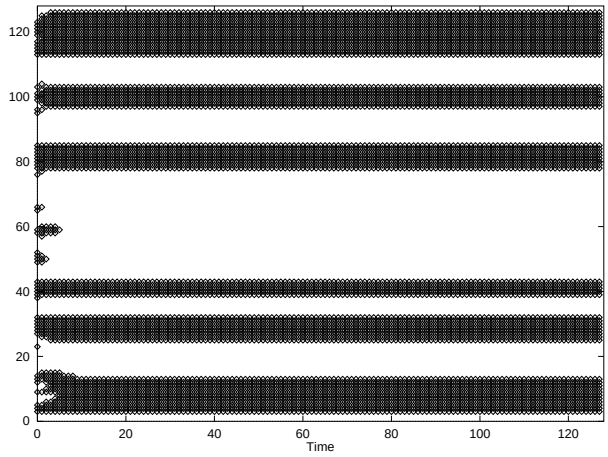

Figure 2: Cross-section of ϕ_{maj} at $x = 64$

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

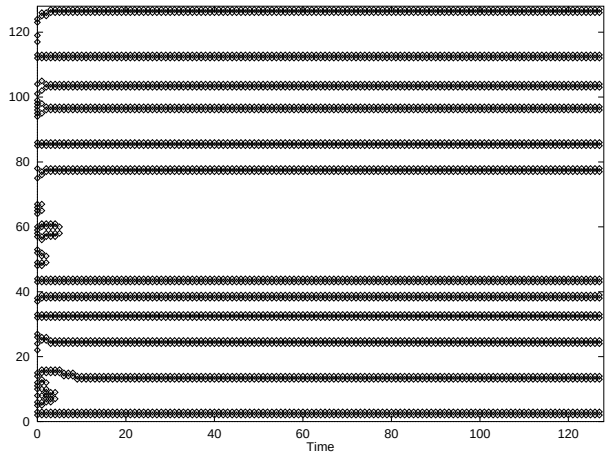

Figure 3: Filtered diagram of the cross-section for ϕ_{maj}.

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].

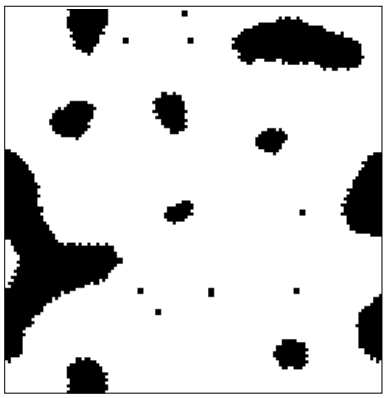

Figure 4: Snapshot at time $t = 2000$ of the evolution of the anneal rule ϕ_{anneal} beginning with $\rho_0 \approx 0.51$. Lattice size is 127x127.

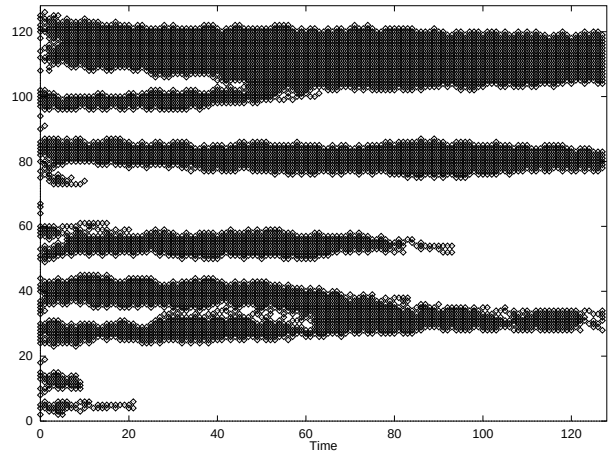

Figure 5: Cross-section of ϕ_{anneal} at $x = 64$.

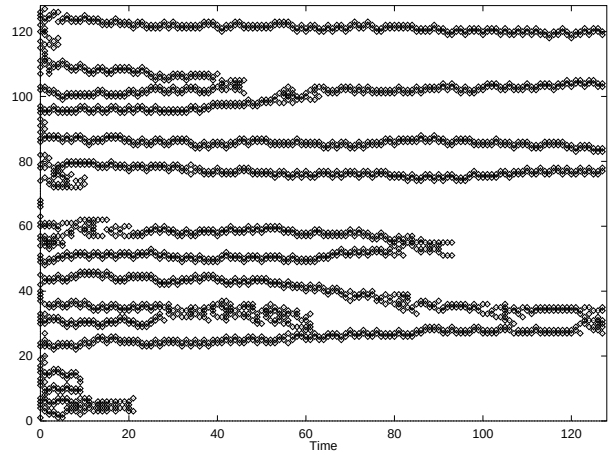

Figure 6: Filtered cross-section of ϕ_{anneal}.

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.

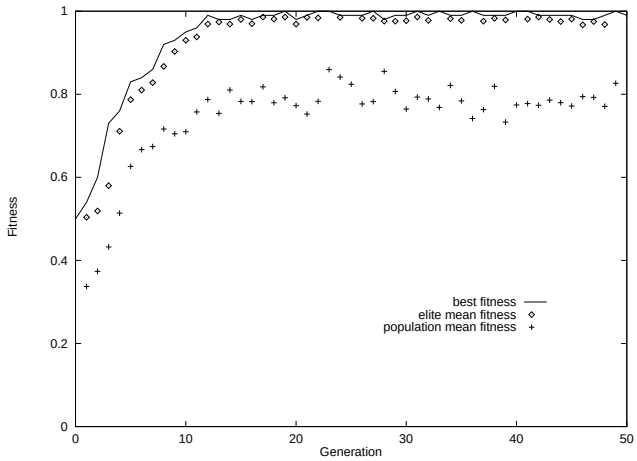

Figure 7: Best fitness, elite mean fitness, and population mean fitness versus generation for one typical run maded on a lattice of 21x21 cells; other lattice sizes gives the same plot. Only the firts 50 generations are shown.

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

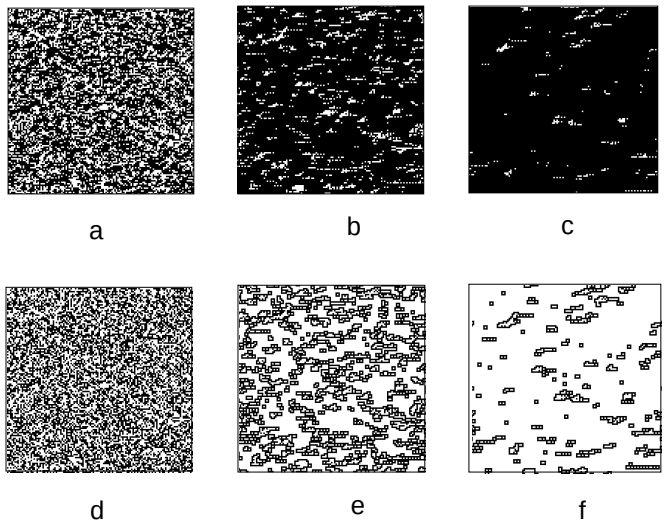

Figure 8: Snapshots of the evolution of rule ϕ_3 and a filtered diagram of the evolution at time: (a-d) t=0 , (b-e) t=4, (c-f) t=8. Initial condition is $\rho_0 \approx 0.51$.

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

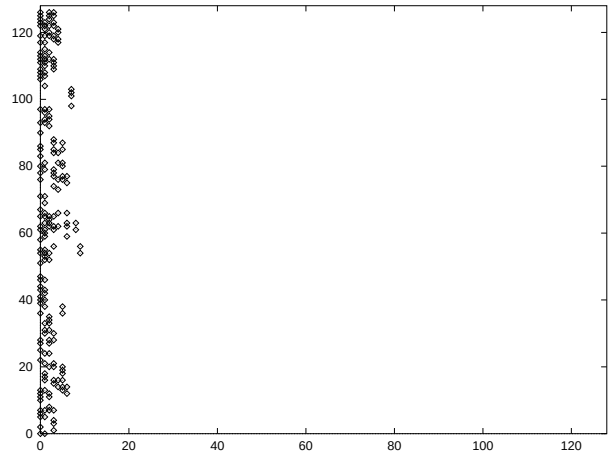

Figure 9: Space-time diagram of a filtered cross section of rule ϕ_3 at $x = 64$ starting with a random initial condition.

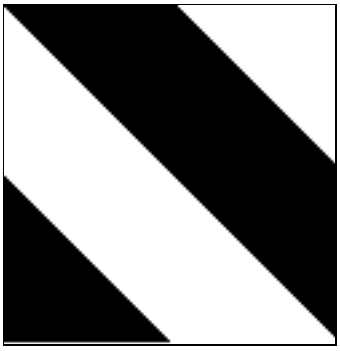

Figure 10: Space-time diagram of a cross section of rule ϕ_3 at $x = 64$ starting with an initial condition that consist of a vertical black and a white domain. Time goes from left to right.

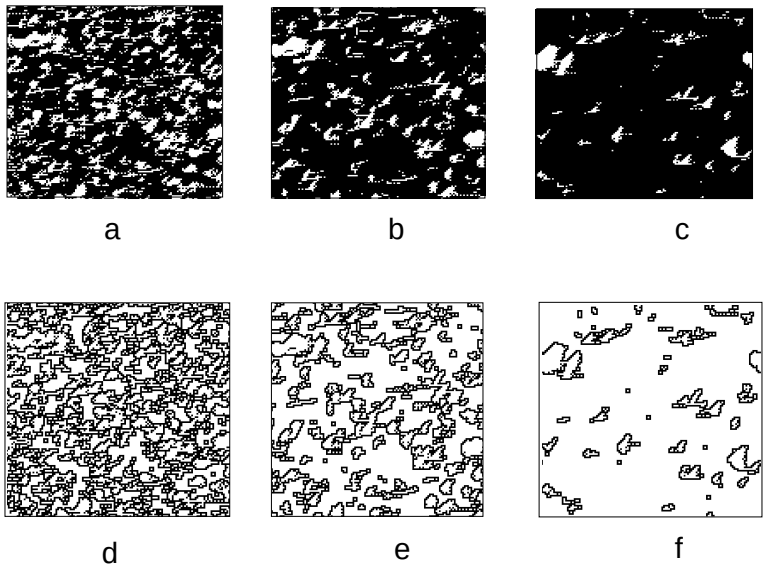

Figure 11: Snapshots of the evolution of rule ϕ_8 and a filtered diagram of the evolution at time: (a-d) t=3 , (b-e) t=6, (c-f) t=20. Initial condition is $\rho_0 \approx 0.51$.

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

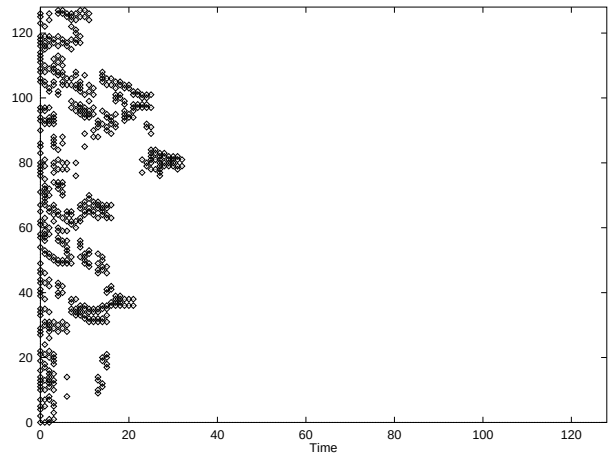

Figure 12: Space-time diagram of a filtered cross section of rule ϕ_3 at $x = 64$ starting with a random initial condition $\rho_0 \approx 0.51$.

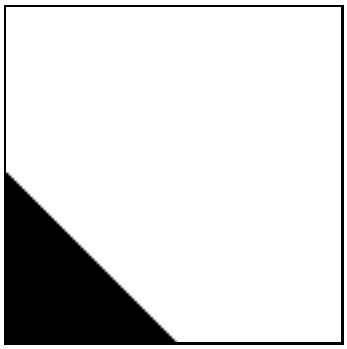

Figure 13: Space-time diagram of a cross section of rule ϕ_8 at $x = 64$ starting with an initial condition that consist of a vertical black and a white domain. Time goes from left to right.

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 ($\approx$ 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

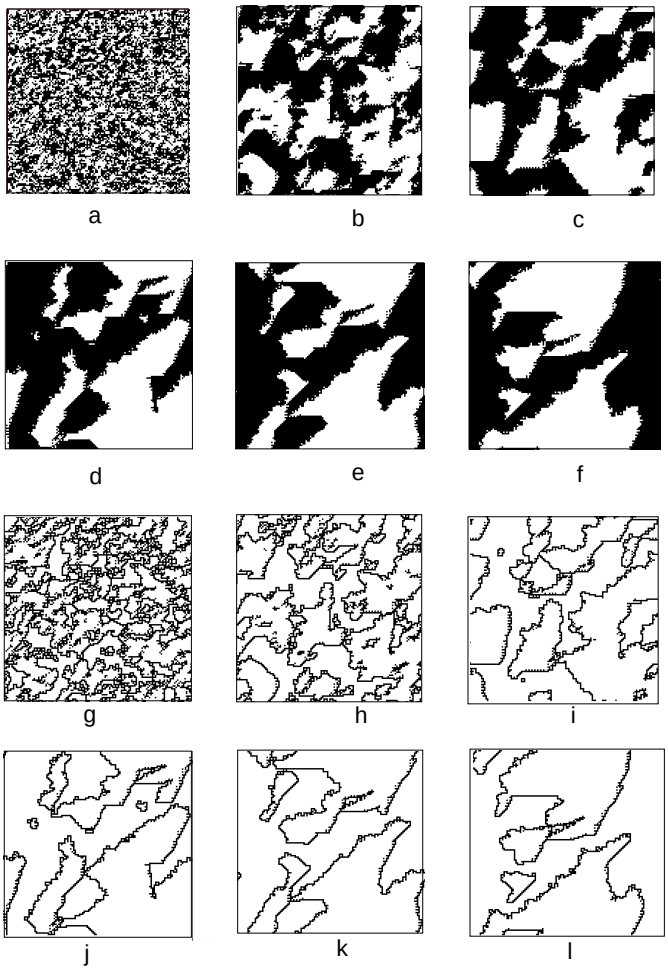

Figure 14: Snapshots of the evolution of rule ϕ_{12} and a filtered diagram of the evolution at time: (a-g) t=0 , (b-h) t=20, (c-i) t=45, (d-j) t=75, (e-k) t=100, (f-l) t=150. Initial condition is $\rho_0 > 0.5$.

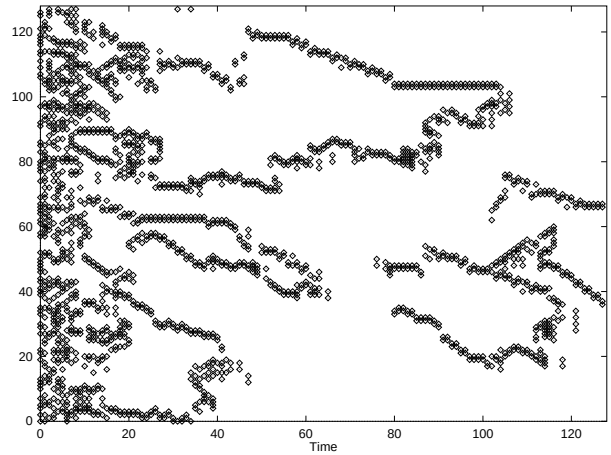

Figure 15: Space-time diagram of a filtered cross section of rule ϕ_{12} at $x = 64$ starting with a random initial condition.