Evolving Cellular Automata with Genetic Algorithms

Aureliano Rama

Dipartimento di Informatica
Università di Pisa, Italy

Pisa – July 23^{rd}, 2009

Outline

Outline

Genetic Algorithms

A Genetic Algorithm (GA) is a search algorithm on a string space: each string encode a chromosome, the instruction on how to perform calculation.
The name comes from the way in which new candidate solutions are produced:

- a set of candidate solutions is valued using a fitness function
- a new set is produced by selecting only the fittest solutions (selection)
- some genetic operators (mutation, cross-over, split, swap, . . .) are applied to them

This seminar

This seminar will show how GAs can be used to find Cellular Automata that perform global computation without need for global coordination.

Cellular Automata, as we will see, are small, local machines whose relative position are fixed and who can communicate with a few of their neighbors.

The goal is to obtain from this local machines, some kind of collective computation.

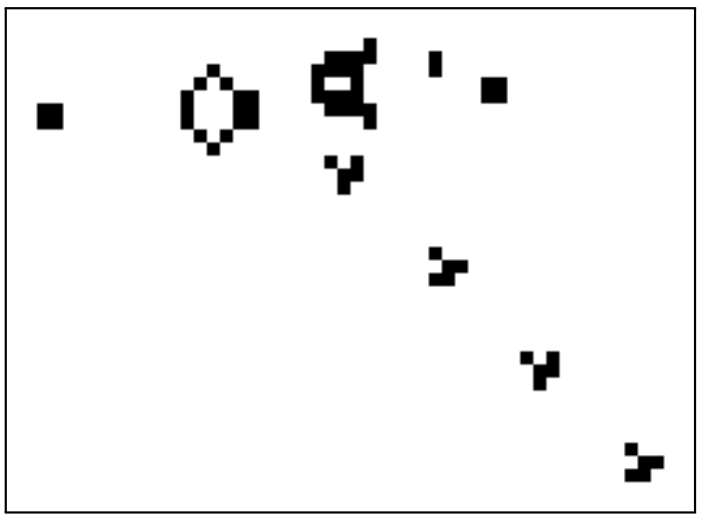
Even more, we want this behavior to emerge "naturally", as the machines compete for the best performance.

Outline

Cellular Automaton

A Cellular Automaton

- is a discrete computation model, first detailed by *Von Neumann*
- consisting of a regular grid of cells, each with
  - a state from a finite (often small) set of States
  - a finite subset of the cells, called its *neighborhood*
- and a state update function, producing next state from the from the cell's and its neighborhood's states



Gosper's Glider Gun
creating "gliders"
in Conway's Game of Life.

One-Dimension CA

A one-dimensional cellular automaton consists of

- a lattice of N identical Finite State Machines (cells)
- with an identical topology of local connections to other cells (neighborhood, η)
- along with boundary conditions.

Each cell obeys the same transition rule ϕ that gives the updated states $s_i^{t+1} = \phi(\eta_i^t)$.

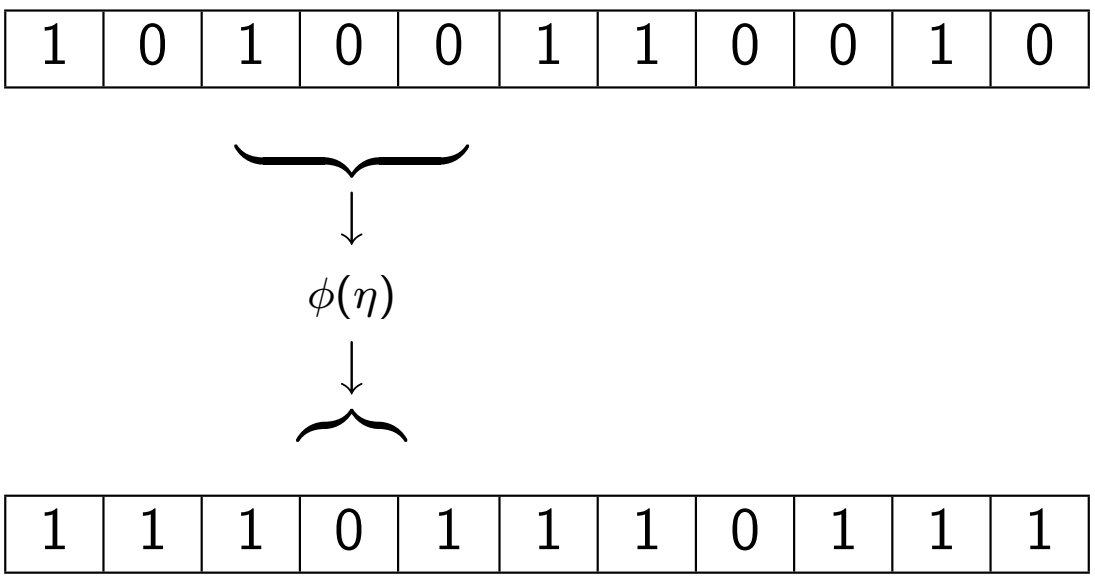
We use $\mathbf{s}^t = s_0^t s_1^t \cdots s_{N-1}^t$ to denote a configuration of cell state, called a CA state.

One-Dimension CA

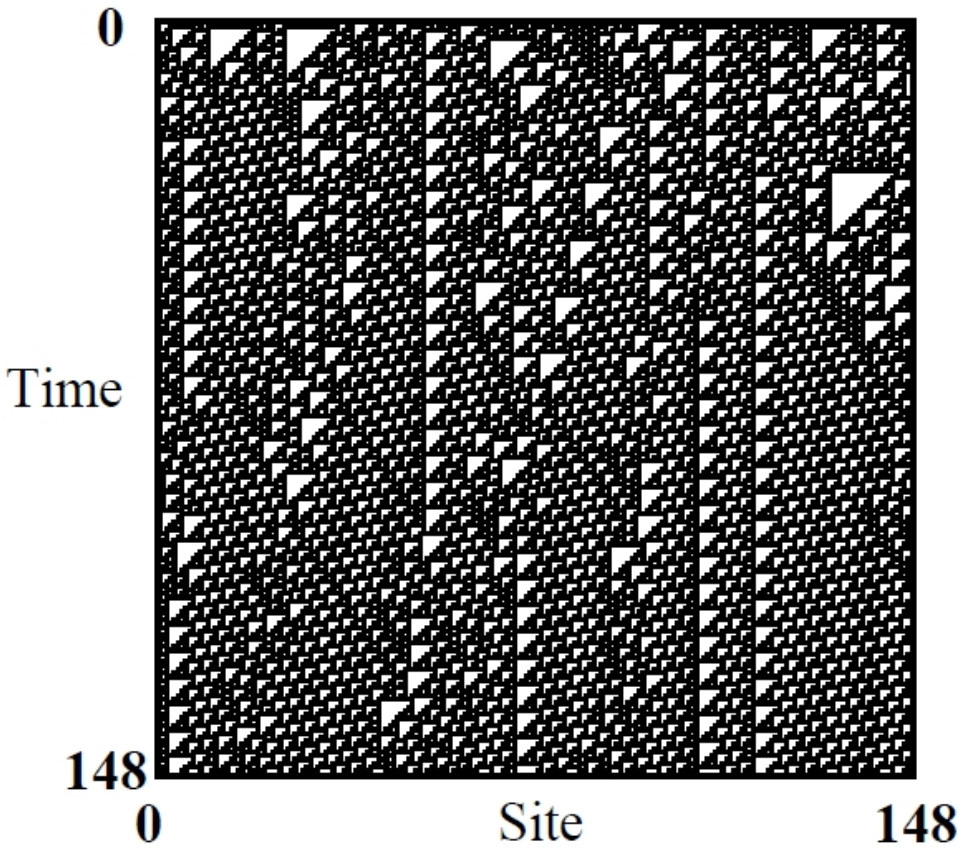
Rule Table ϕ:

Neighborhood η:	000	001	010	011	100	101	110	111
Output bit $\phi(\eta)$:	0	1	1	1	0	1	1	0

Lattice Configuration (with N=11):



Space-Time Diagrams



A space-time diagram illustrating the typical behavior of the well studied elementary CA (ECA) 110.

Outline

Density Classification

For One-Dimensional Cellular Automata, the Density Classification Task is defined as follows:

Definition

- Density Classification is the task of determining if $\rho_0 > \rho_c$ for a fixed ρ_c
- where ρ_0 is the fraction of 1s in the initial configuration IC

and within a maximum of T_{max} time steps:

- if $\rho_0 > \rho_c$, the entire lattice should relax to a fixed point of all 1s
- otherwise, it should relax to a fixed point of all 0s

. The task is undefined for $\rho_0 = \rho_c$.

The task is obviously trivial for Von-Neumann achitectures where the IC is in a global state (like an array).

Other Distributed Problems

Other distributed problems like

- Consensus
- Leader Election
- Byzantine Agreement

are inherently different from DCT: they need to agree on a particular value held initially by one of the processes.

They do not consider the whole initial state as a parameter for the output.

- Moreover, many solutions to these problems assume sophisticated computational and memory capabilities on the single process.

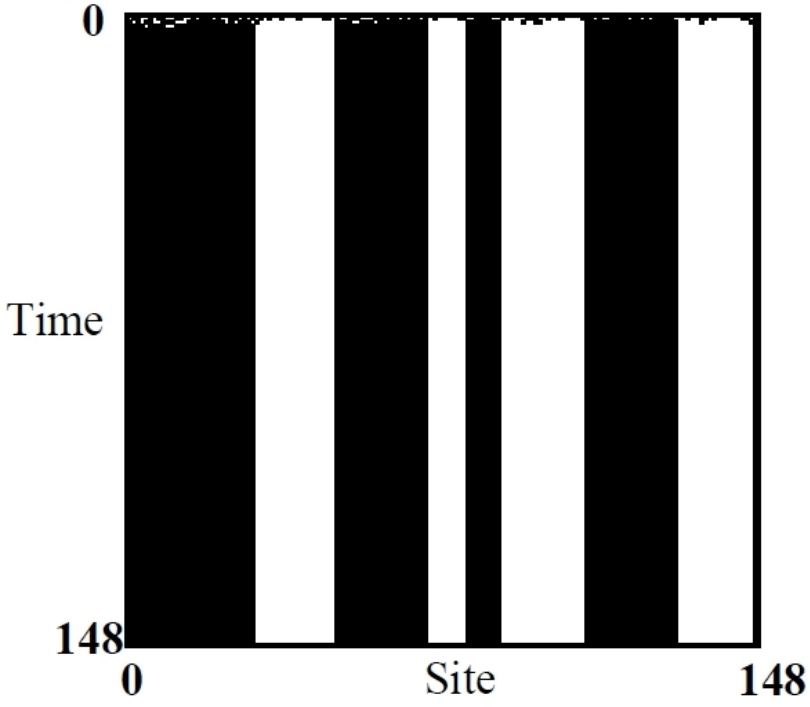
DCT: A Non-Trivial Distributed Problem

- The Density Classification is a non trivial task for Cellular Automata.
- It is actually believed to be unsolvable.
- We are interested in the best partial solutions: CAs that solves a lot of cases out of a big selection.
- We use a performance measure called $\mathcal{P}_N^I$, that is the fraction of ICs correctly classified by the CA over I randomly choosen ICs and with a lattice of N.

Naïve Candidate Solution: ϕ_{maj}

One naïve candidate solution for the $\rho_c = 0.5$ problem is the "majority vote" CA:

Every cell output the majority state of its neighborhood.



$\mathcal{P}_{149}^{10^4} = 0.0$: it doesn't classify correctly any IC (other than the fixed point themselves).

Outline

Genetic Algorithms

A Genetic Algorithm (GA) is a search algorithm with the following properties:

- a population of chromosomes
- a fitness function
- a selection method wrt the fitness
- genetic operators (like mutation and crossover) to apply to the new population

- The chromosomes are usually strings and each represents a candidate solution in the search space

Evolving Cellular Automata Rules

We evolve Cellular Automata rules ϕ by enconding their outputs as sequences of 128 bit.

- We use a population of 100, with initial population chosen at random.
- For each generation, we choose $I = 100$ random IC and we iterate every CA's ϕ function over it.
- The fitness function used for selection is the fraction of those IC correctly classify at the end (at most T_{max} time steps) by each CA.
- A number of the highest fitness CAs (the Elite) is copied to the new population
- The remaining part of the new population is created by applying Crossover and Mutation to the Elite CAs.

Evolving Cellular Automata Rules

- While the evolution used $\mathcal{P}_{149}^{100}$ as performance meter
- the actual performance of the population is valued on a much larger set of ICs: $I = 10^4$.

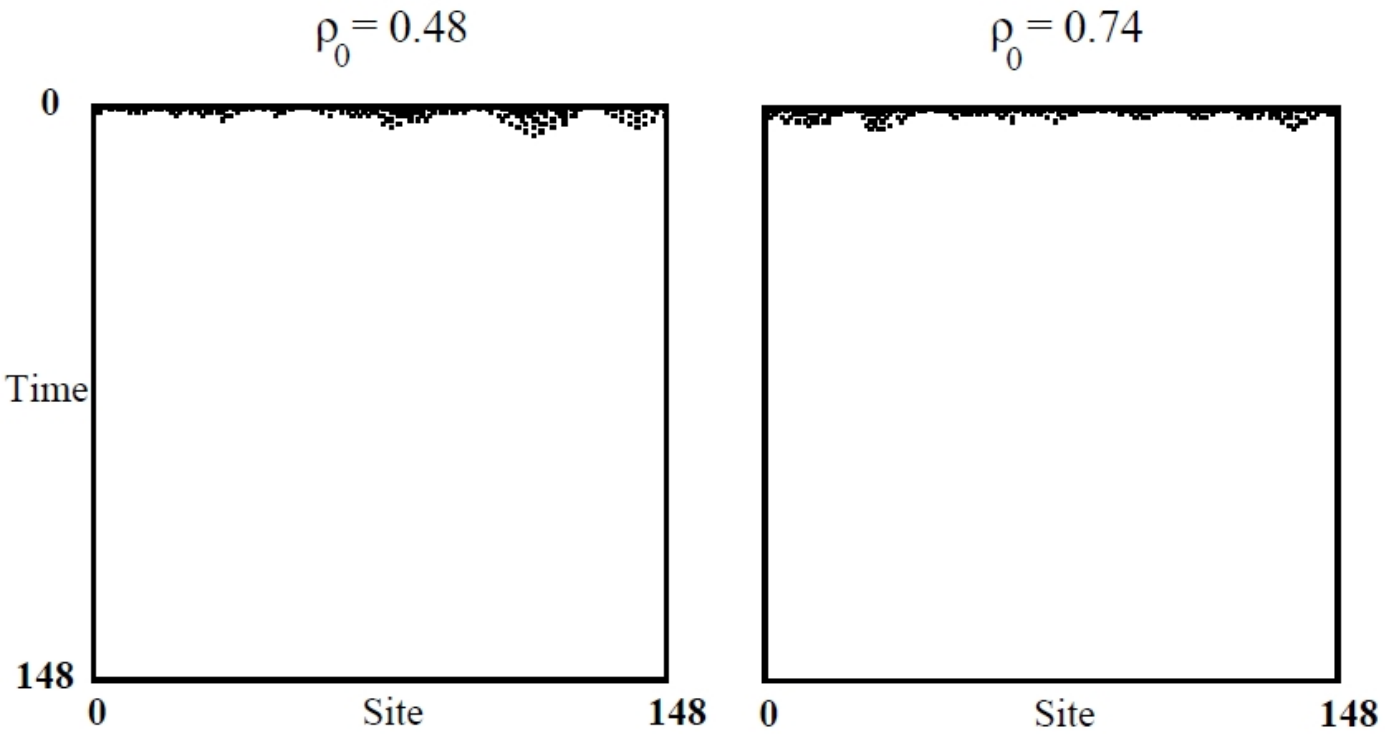
A trivial CA whose rule outputs always 0 (or 1) will correctly classify half of the population

- $\mathcal{P}_N^I = 0.5$

We want to use GAs to evolve CAs that have performance well above chance.

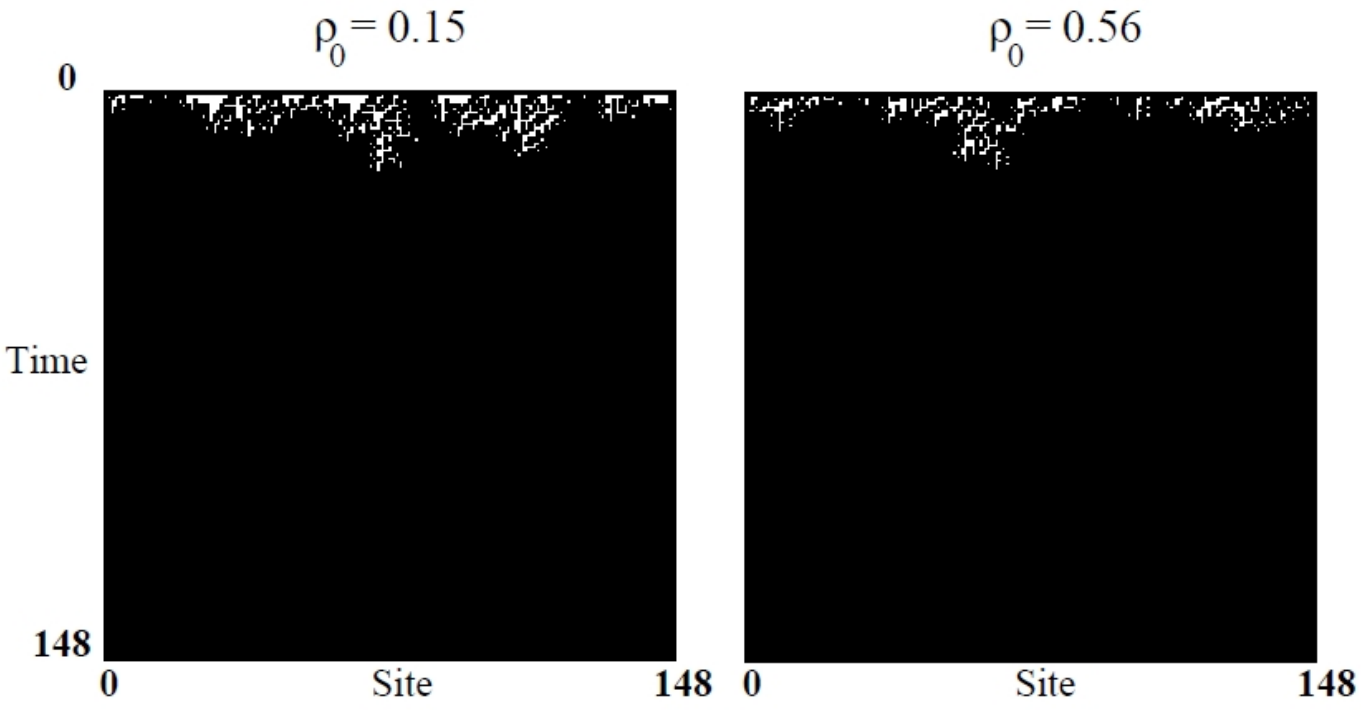
Default Strategy

In the initial population there's always some high-(or low-)ρ individual whose immediately "high" performance of 0.5 will spread quickly.



Default Strategy

Sometimes evolution is unable to get anything better than this in the given time frame.



Block-Expanding Strategy

The block expanding strategy

- will default to a fixed point of all 1s (or all 0s)
- unless there is a large enough block of 0s (or of 1s).

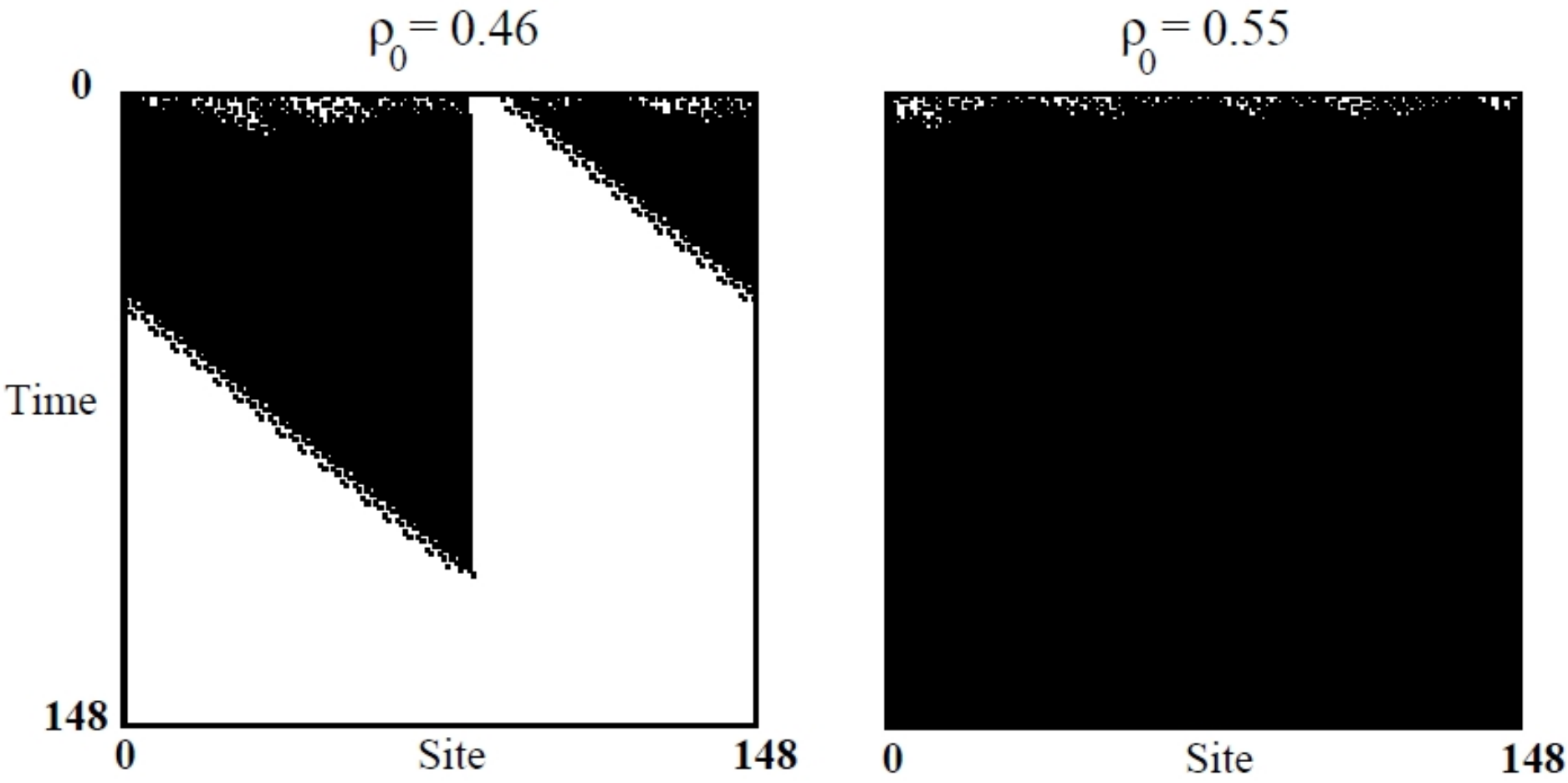
"Large enough" is tipically of the order of the neighborhood.

- Block-expanding strategies rely on the presence of these blocks to be an heuristic for the value of ρ_0.

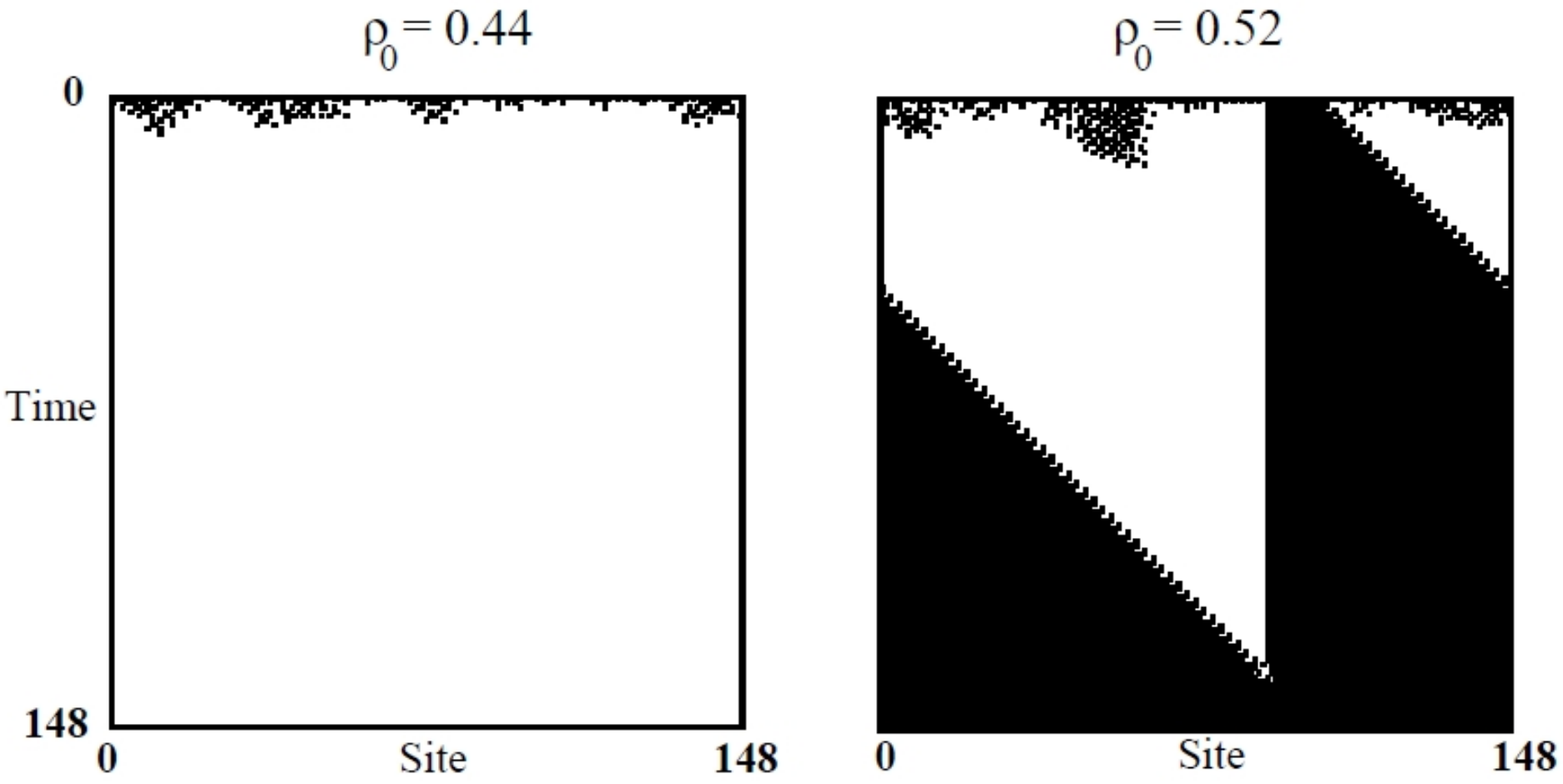
Since there is a statistical correlation, once found these strategies quickly take over the population.

But these are not examples of collective computation.

Block-Expanding Strategy



Block-Expanding Strategy



Embedded-Particle Strategy

To have true Collective Computation (and to perform with high performances in the $\rho_c = 1/2$ task) our CAs must achieve:

- sophisticated coordination
- information transfert

Embedded-Particle Strategies, evolved only rarely by the GA, do achieve these traits.

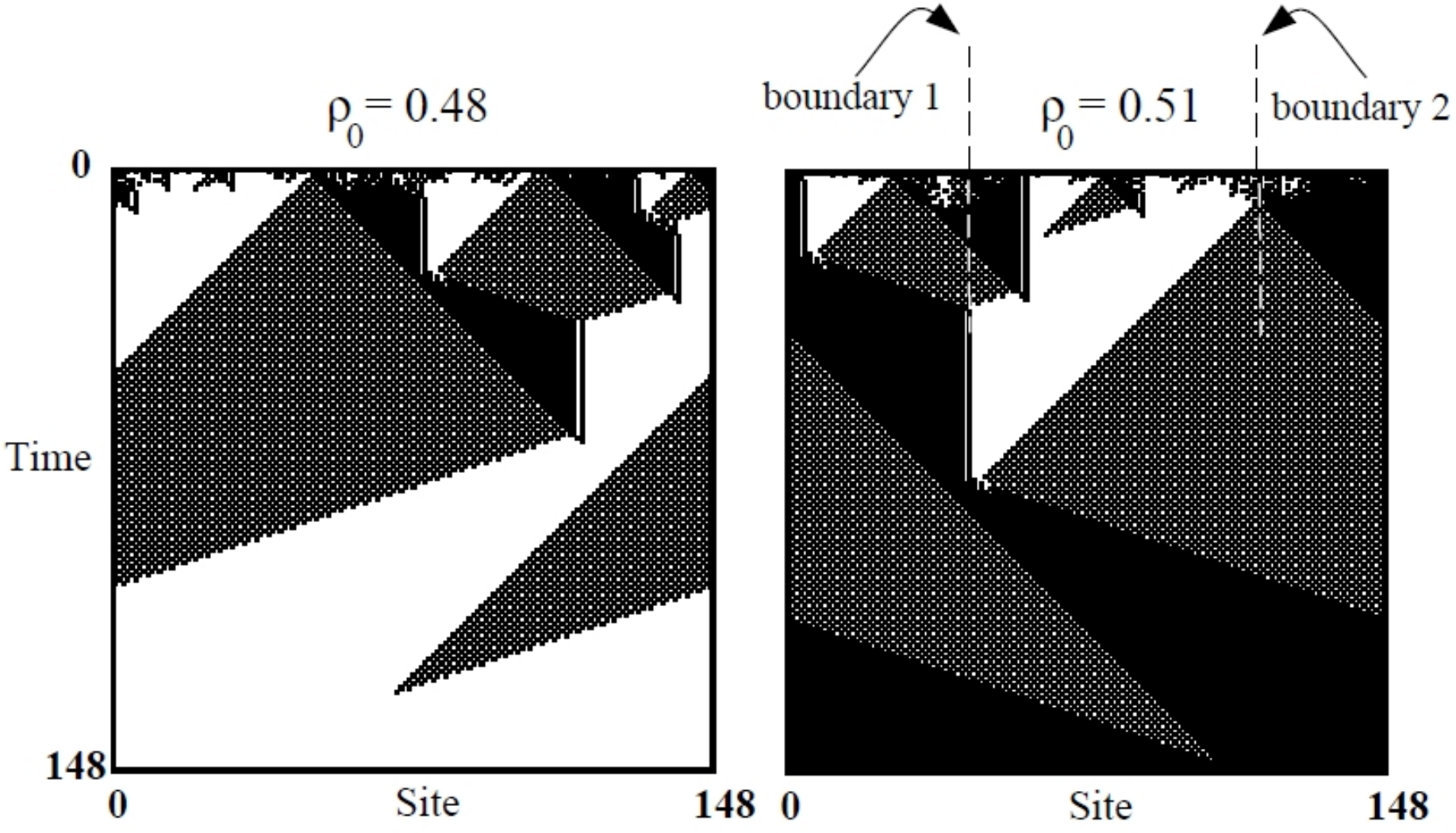
This CAs reach performances over 75% by means of stationary boundaries and expanding regions.

Embedded-Particle Strategy

Details of the EP Strategy behavior:

- The stationary boundaries appears when a region of 1s on the left encounters a region of 0s on the right.
- Whenever a region of 0s on the left encounters a region of 1s on the right, a checkerboard region grows with equal speed in both directions.
- When this checkerboard region encounters a vertical boundary, it start to collapse, faster than the expanding rate.
- At the end, only a larger black (or white) region will remain.

Embedded-Particle Strategy



Outline

Understanding Collective Computation

How (and why) does this work?

At a closer analysis the role of the checkerboard region $(01)^+$ is to decide which of two adiacent region (0s and 1s) is larger.

This is obtained by cutting off the smaller, so that the larger can continue to expand.

- The net decision is that the density in the region was in fact below or above $\rho_c = 1/2$!

The black-white boundary and the checkerboard region can be seen as signals indicating "ambiguous" density regions.
Each of these boundaries has a local density at exactly $\rho_c = 1/2$.

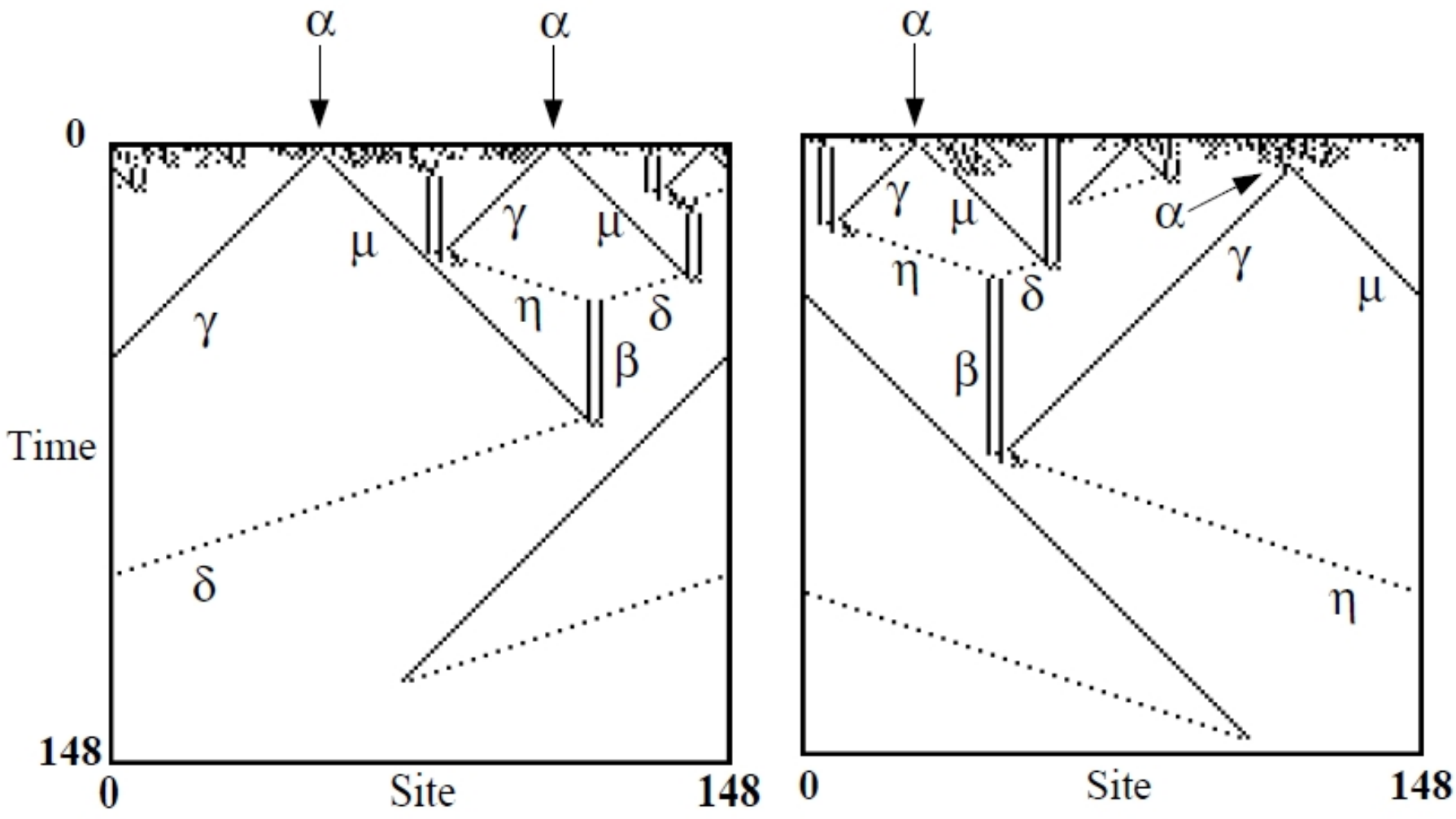
Understanding Collective Computation

With a method based on a computational mechanics framework [Crutchfield, 94] we can classify different pattern that appears in CA space-time behaviou, using concepts from computation and dynamical systems theories.
They aim to reveal the intrinsic information-processing structures embedded in dynamical processes.

We want to discover patterns in terms of which a CA's behavior can be decomposed.
Then the boundaries between patterns (also called domains) can be seen as particles with a proper dynamics.

Understanding Collective Computation

After a short condensation time, domains (and hence particles) become evident to the eye, once we filter out the domains themselves.



Computational Strategy for ϕ_{par}

The interactions between all these particles are complex, transferring geometrical information and allowing classification of the density of large regions.

Regular Domains		
$\Lambda^0 = \{0^+\}$	$\Lambda^1 = \{1^+\}$	$\Lambda^2 = \{(01)^+\}$
Particles (Velocities)		
$\alpha \sim \Lambda^0\Lambda^1$ $(-)$ $\delta \sim \Lambda^2\Lambda^0$ (-3)	$\beta \sim \Lambda^1\Lambda^1$ (0) $\eta \sim \Lambda^1\Lambda^2$ (3)	$\gamma \sim \Lambda^0\Lambda^2$ (-1) $\mu \sim \Lambda^2\Lambda^1$ (1)
Interactions		
decay	$\alpha \rightarrow \gamma + \mu$	
react	$\beta + \gamma \rightarrow \eta$, $\mu + \beta \rightarrow \delta$, $\eta + \delta \rightarrow \beta$	
annihilate	$\eta + \mu \rightarrow \emptyset$, $\gamma + \delta \rightarrow \emptyset$	

Significance of the Particle-Level Description

- In principle, these CAs are completely described by the 128 bits in their lookup tables.

However this is a too low-level description to be useful understanding of how a given CA performs the $\rho_c = 1/2$ task.

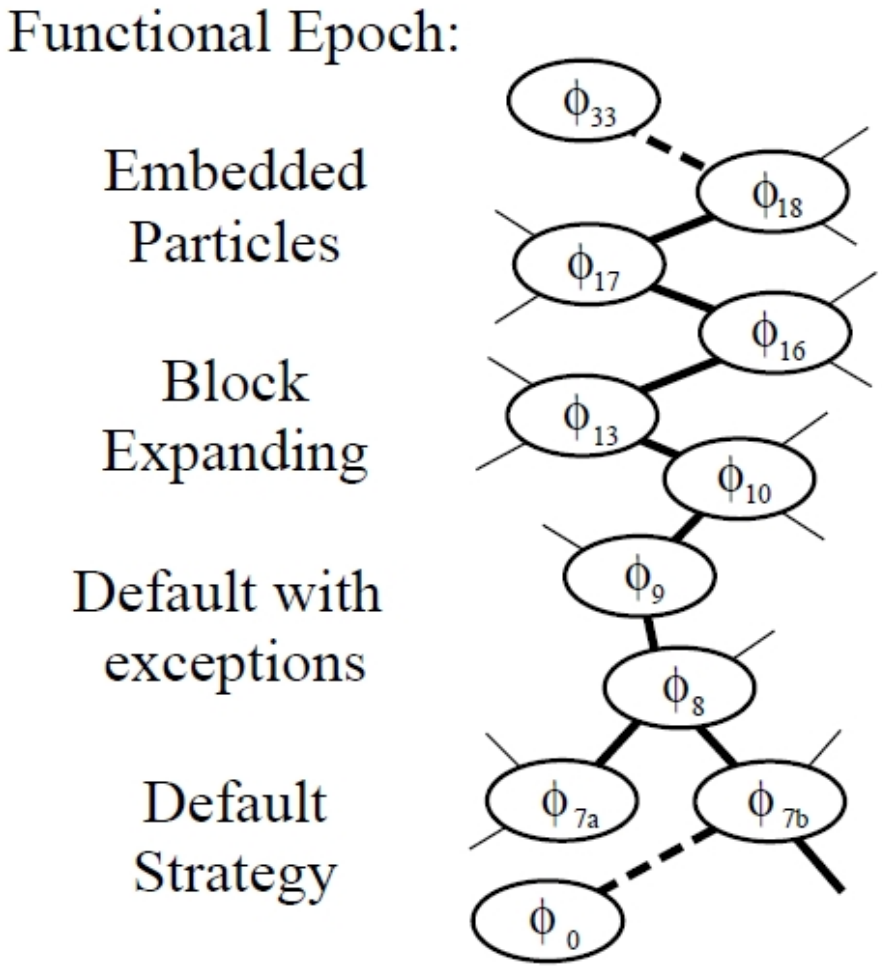
In Dynamical System Theory, a basic principle is that (for non-linear systems) the local space-time equations of motion do not **directly** determine the system's long term behavior.

- Only over a number of iteration, there emerge domains, particles and interactions.

Outline

Evolutionary History of ϕ_{par}

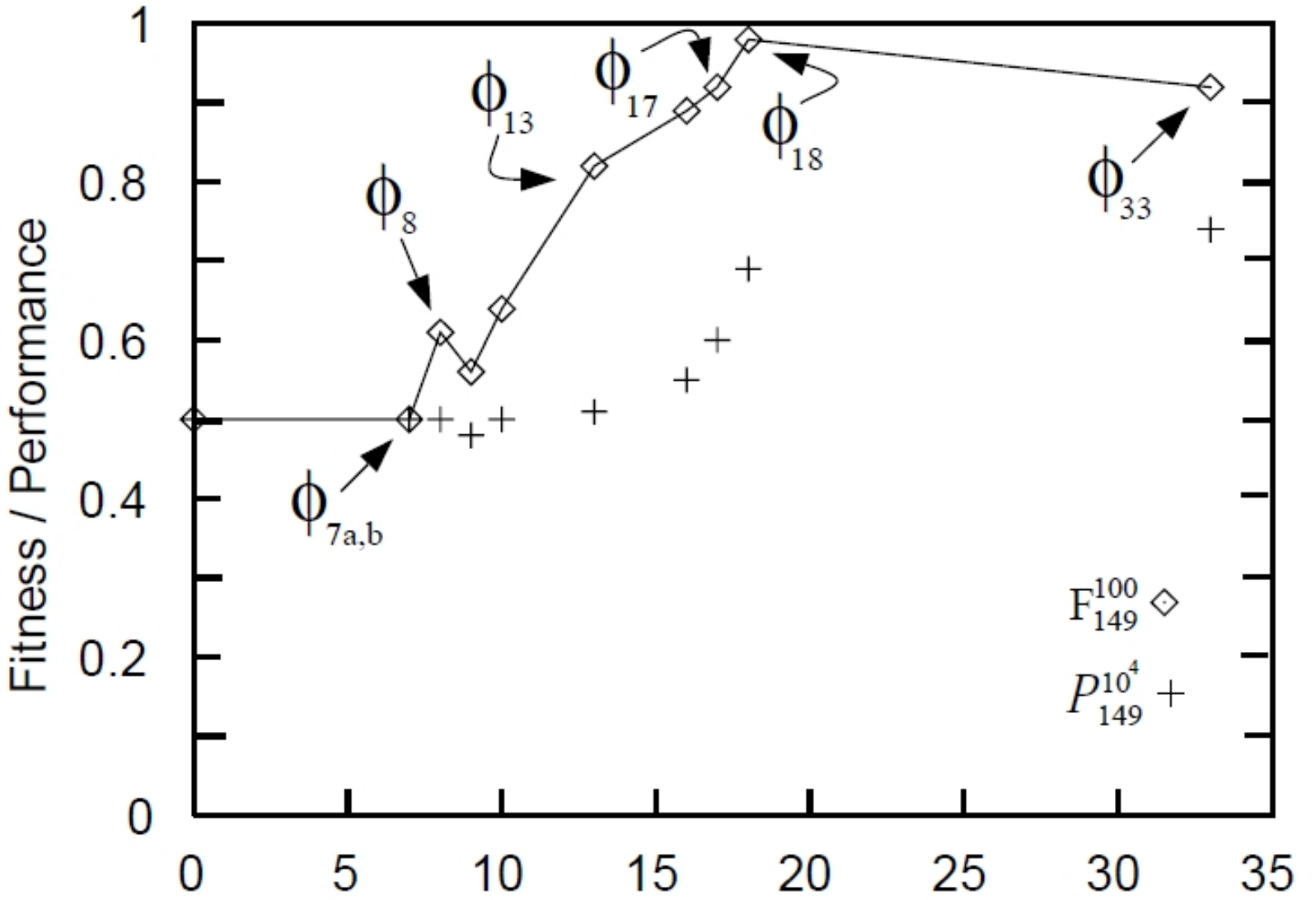
ϕ_{par}'s ancestor's humble origin was as a default CA.



This partial tree of the ϕ_{par}'s evolutionary history shows the advent of different strategies

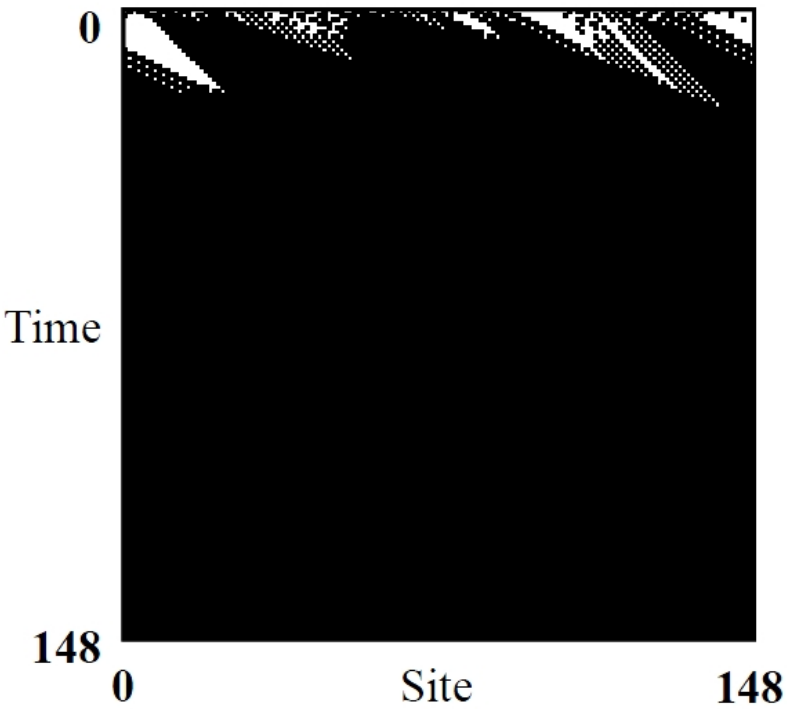
Evolutionary History of ϕ_{par}

As the GA discover each new strategy, the overall performances of the CAs increase sharply.



Exaptation: adaptively neutral traits becomes important

As last note, let's look at a space-time diagram for ϕ_8:



There can be seen the first appearance of the $\Lambda^2 = \{(01)^+\}$ domain. Here is completely neutral (removing it does not change the performance) but it's going to become the essential trait to develop the particle-strategy later on.

Outline

Wrap-up

Genetic Algorithms are very special serch methods: they can forge their way in uncharted lands.

Even more, they are able to discover and exploit hidden structures in the domain they are exploring.

As the evolutionary biologists have discovered in the last 150 years (since Darwin's Origin of Species), the incremental evolution is an incredible powerful force, especially in environment where all parts (preys and predators, friends and foes) try their best to survive at others' expenses: this last idea has started to be esplored by advanced topics like Spatial CoEvolution and Conceptual Spaces.

Quick References

- Mitchell - An introduction to Genetic Algorithm
- Crutchfield, Mitchell, Das - The Evolutionary Design of Collective Computation in Cellular Automata
- Crutchfield - The Calculi of Emergence: Computation, Dynamics and Induction
- Horijik, Crutchfield, Mitchell - Mechanisms of Emergent Computation in Cellular Automata
- Marques-Pita, Mitchell, Rocha - The Role of Conceptual Structures in Designing Cellular Automata to Perform Collective Computation

End

Thank you.