Identification of Cellular Automata

Witold Bołt
(witold.bolt@hope.art.pl)

Systems Research Institute PAS, Warsaw

January 18, 2017

Cellular automaton: general definition

A cellular automaton $\mathscr{C}$ is a quintuple $\mathscr{C} = \langle \mathcal{T}, S, s, N, \Phi \rangle$:

- countably infinite tessellation $\mathcal{T}$ of $\mathbb{R}^n$, consisting of cells c_i, $i \in \mathbb{N}$
- finite set S of k states
- output function s yields the state $s(c_i, t)$ of cell c_i at step t
- neighborhood function N maps every cell c_i to a finite sequence $N(c_i) = (c_{i_j})_{j=1}^{|N(c_i)|}$
- transition functions $\Phi = (\phi_i)_{i \in \mathbb{N}}$ govern the dynamics:

$$\boxed{s(c_i, t+1) = \phi_i(\tilde{s}(N(c_i), t))}$$

where $\tilde{s}(N(c_i), t) = (s(c_{i_j}, t))_{j=1}^{|N(c_i)|}$

[J.M. Baetens and B. De Baets. Dynamical Systems, 27:411-430, 2012.]

1D CAs considered in our experiments

- M cells arranged in a circular array ($c_{i+M} = c_i$), k states, deterministic
- Symmetric neighborhood with radius r (size $2\,r+1$)
- Local rule: unique transition function $f\colon \{0,\ldots,k-1\}^{2\,r+1} \to \{0,\ldots,k-1\}$
  - a local rule can be represented as a lookup table (LUT)
  - general form of a LUT for an Elementary CA (ECA) ($k=2$, $r=1$ and $\ell_i \in \{0,1\}$)

111	110	101	100	011	010	001	000
ℓ_8	ℓ_7	ℓ_6	ℓ_5	ℓ_4	ℓ_3	ℓ_2	ℓ_1

LUT of local rule with number $n = (\ell_8,\ell_7,\ell_6,\ell_5,\ell_4,\ell_3,\ell_2,\ell_1)_2$

  - 256 ECAs (88 independent ones)

Global rules, configurations and space-time diagrams

- Configuration: $X \in \{0, \ldots, k-1\}^M$, e.g. $s(\cdot, t)$
- Global rule: $A\colon \{0, \ldots, k-1\}^M \to \{0, \ldots, k-1\}^M$ such that

$$s(\cdot, t+1) = A(s(\cdot, t))$$

  expressing the evolution of configurations in time:

$$X \to A(X) \to A(A(X)) \to \ldots \to A^t(X)$$

- Initial configuration: first configuration
- Space-time diagram: entire diagram
  - space-time diagram with M cells and N time steps can be seen as a matrix $D \in \{0, k-1\}^{N \times M}$
  - t-th row in D represents the configuration at time stamp t (after $t-1$ applications of A)

Problem formulation

- CA-based models can solve complex computational/modelling problems
- To build a CA-based model one needs to define the transition function, a non-trivial task
- In many domains designing the transition function by hand is key to model development
- Building CA-based models automatically from observations (real-world data) is very compelling

CA identification problem

The problem of algorithmically/automatically finding the local rule of CAs fulfilling a set of conditions on the global/local dynamics

Specific CA identification problems

- Majority Problem or Density Classification Problem:
  - Only the initial configuration and the desired final configuration are known
  - Pre-defined state set, geometry and typically also neighborhood
  - Problem known to be unsolvable, only “approximate” or weak solutions exist

  several approaches, including genetic algorithms, genetic programming, machine learning methods, . . .

- Global Synchronization Problems such as the firing squad synchronization problem:
  similar approaches as above

General CA identification problem

- Extracting CAs from experimental data: Richards et al. (1990)
  - building nondeterministic CA on the basis of photographs of dendritic solidification of NH_2Br
  - approach based on genetic algorithms (GA)
  - fitness based on joint information measure
  - some form of temporal incompleteness of observations (unknown time scale)
- General identification formulation: Adamatzky (1994)
  - extraction of local transitions from complete observations
  - reading/estimating the LUT of deterministic/nondeterministic CAs
  - touches upon radius selection and state set selection in various settings

Informal introduction

- What we have: recorded behavior of some system, which we "believe" is a CA
- What we want: the underlying CA rule

Note

Sometimes the problem might be **very easy**. Example...

Incomplete observations of space-time diagrams

- Incomplete observation of a space-time diagram (observation): matrix $I \in \{0, 1, \, ?\}^{N \times M}$
  - spatial incompleteness: **?** denotes an unknown state
  - temporal incompleteness: specific time stamp of a given row is unknown, i.e. in between two consecutive rows of an observation, there might be a **time gap** of unknown length

Identification problem

Motivation:

- faulty or/and limited observation capabilities
- lack of synchrony: different clocks

Problem statement

Given an incomplete observation I, find a CA A such that there exists a space-time diagram D of A, for which there exists a sequence of integers $(\tau_i)_{i=1}^{N}$ such that $\tau_1 = 1$, $\tau_n < \tau_{n+1}$ and for all n, m:

$$I[n,m] = ? \qquad \text{or} \qquad D[\tau_n, m] = I[n,m]$$

In words

Find a CA with a space-time diagram D such that we can assign any row of the observation I to a specific row in D for which all non-? states agree

Identification as an optimization problem

- Technical assumptions:
  1. initial configuration $I[1]$ is completely observed
  2. there is a maximal time gap T between two consecutive rows of any observation
- A-completion of an (incomplete) observation I: given a CA A and a sequence of time stamps $\tau = (\tau_i)_{i=1}^{N}$, we construct a complete observation

$$\bar{I}_{\tau}^{A} \in \{0, 1\}^{N \times M}$$

  by replacing every occurrence of ? in I by the result of evaluating A taking into account τ

Identification as an optimization problem

- Consider $\tau = (\tau_i)_{i=1}^N$ and let $t_i = \tau_{i+1} - \tau_i$, $i = 1, \ldots, N-1$, then we define the following error

$$E_I(A, \tau) = \sum_{i=1}^{N-1} d_H(A^{t_i}(I_\tau^A[i]), \bar{I}_\tau^A[i+1])$$

where d_H stands for the Hamming distance

- In view of the maximal time gap, we can define an error that does not depend on the choice of τ:

$$E_I(A) = \min_\tau \sum_{i=1}^{N-1} d_H(A^{t_i}(I_\tau^A[i]), \bar{I}_\tau^A[i+1])$$

where the minimum only covers sequences $(\tau_i)_{i=1}^N$ satisfying $t_i \leq T$, $i = 1, \ldots, N-1$

Fitness function

- The fitness function is defined with the use of E_I as:

  $$\mathrm{fit}_I(A) = C - M - E_I(A)$$

  where C is the total number of entries in observation I having a value in $\{0, 1\}$

  - The term $C - M$ is the total number of completely observed states, excluding the initial configuration
  - In words: the fitness function counts the number of matching cells when comparing the space-time diagram of A with observation I, considering the best possible sequence of time stamps
  - The goal of the evolutionary algorithm is to maximize the given fitness function
  - In practice, we use a small subset of set of observations $\mathcal{I}$, which is updated during GA evolution, to overcome local optima and limit computing costs

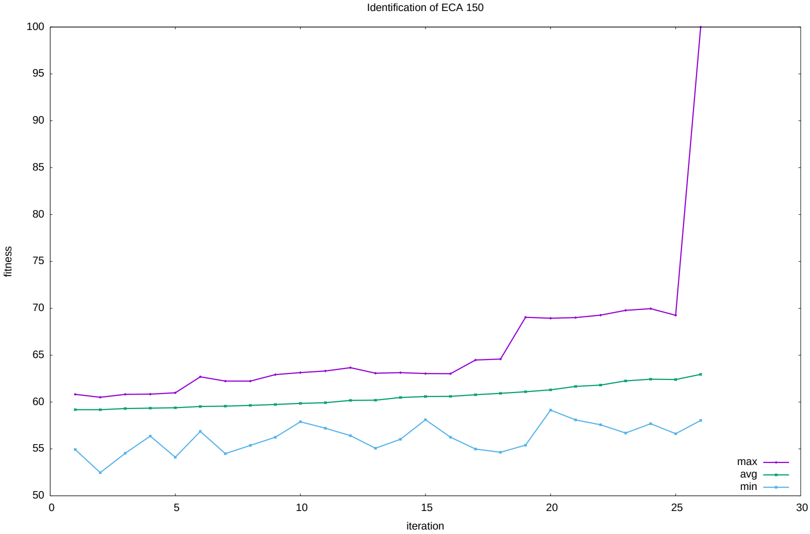
Genetic algorithm - summary of design

- Individuals: variable length bit-strings corresponding to LUTs; fixed size population
- Selection: fitness proportional, with replacement
- Crossover: uniform with radius selection
- Mutation: radius decrease and increase, random bit-flip
- Elite survival: top performers preserved
- Restart: population reinitialization in case of local optima
- Halting condition: perfect solution found

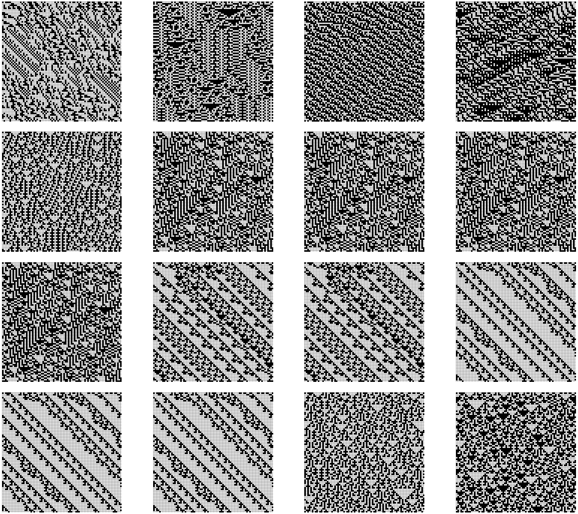
GA setup

- Population: 192 individuals, elite: 24
- Radius min: 2, max: 5
- Mutation probabilities: up/down-scale: 0.001, bit-flip base: 0.001
- Max time gap: 10, random length
- Observations: 64 observations (8 used for fitness approx.), 69×69 pixels

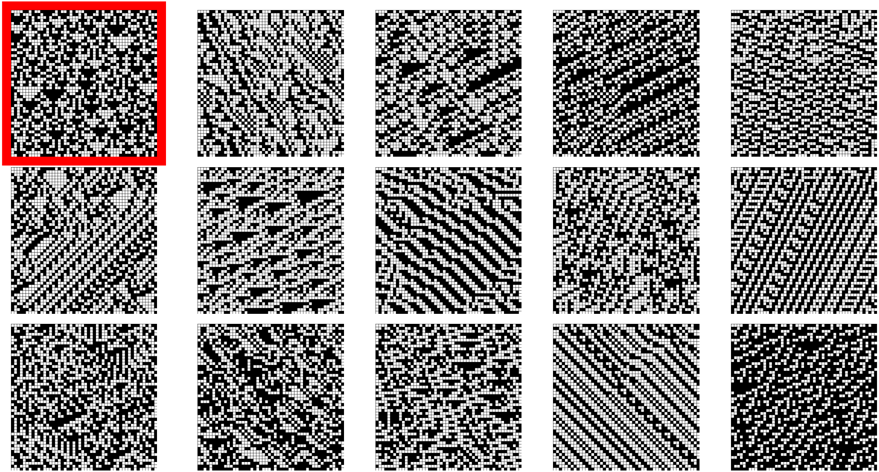
Fitness evolution in time



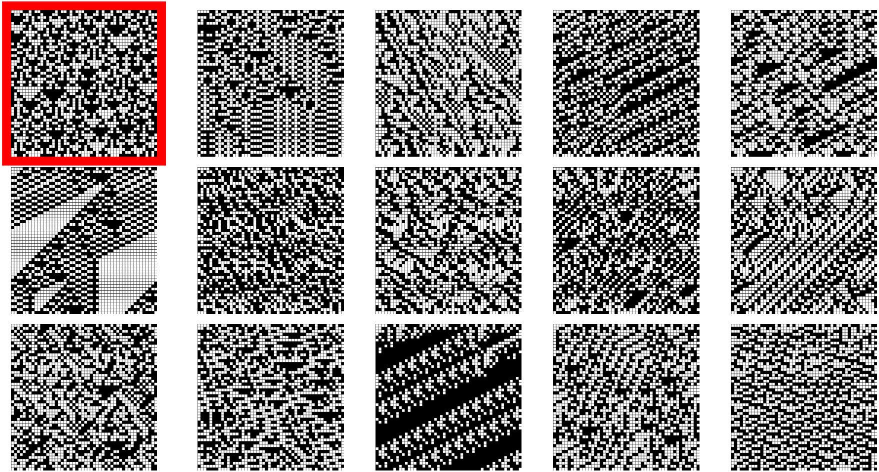
Best individuals



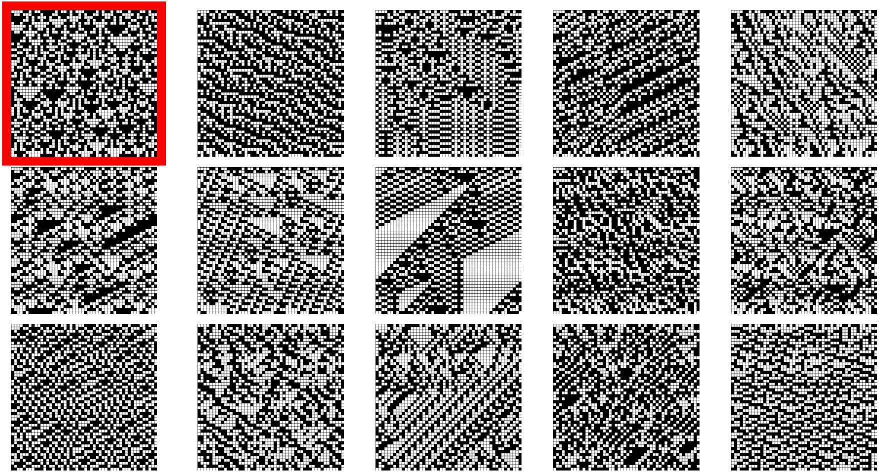
Iteration no. 1



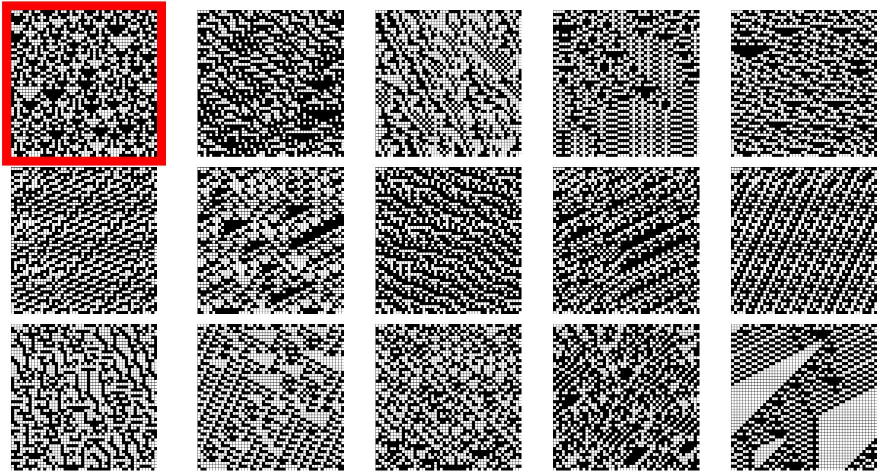
Iteration no. 2



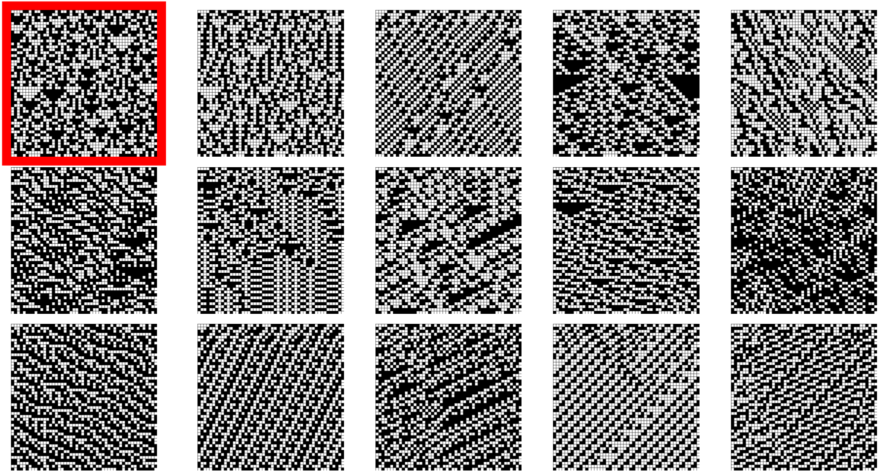
Iteration no. 3



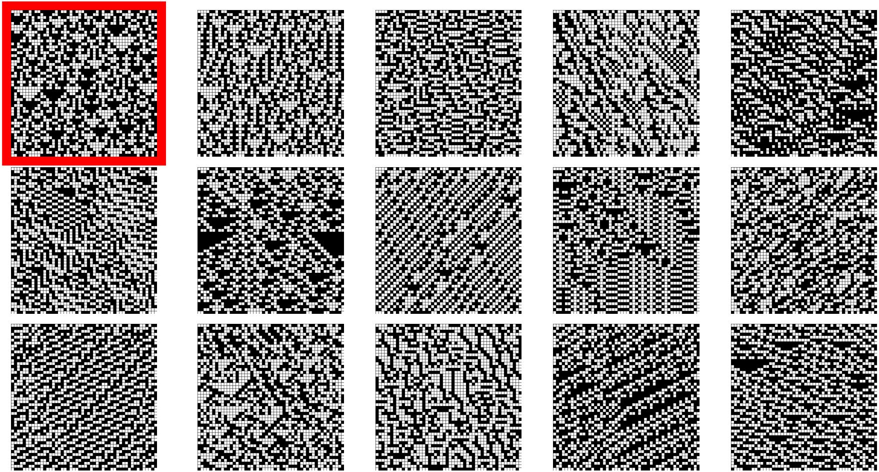
Iteration no. 4



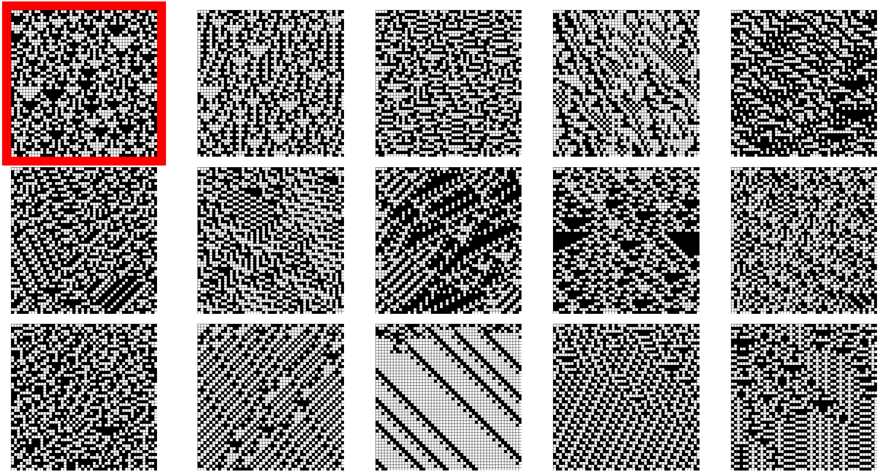
Iteration no. 5



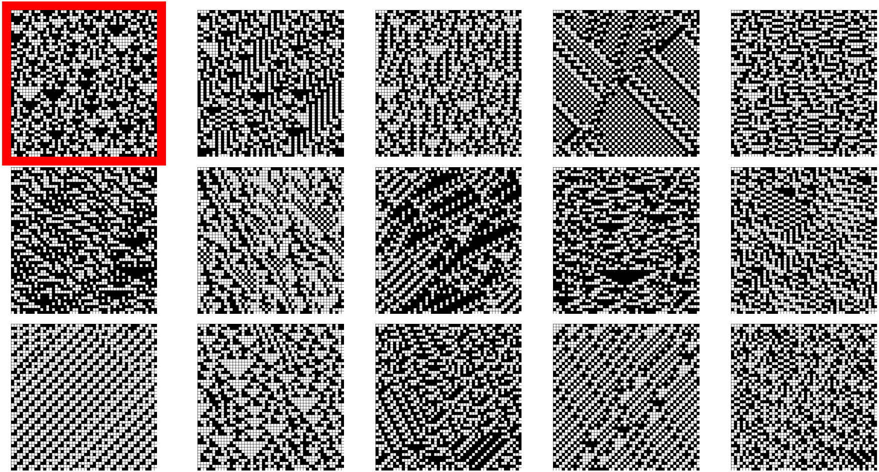
Iteration no. 6



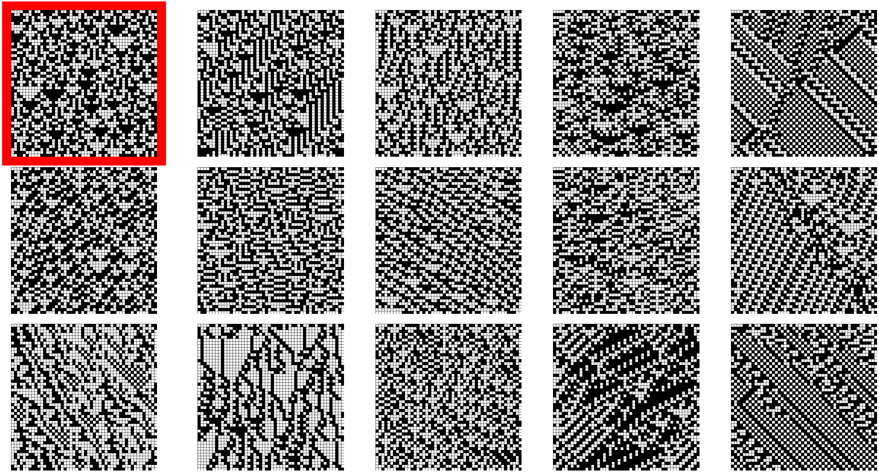
Iteration no. 7



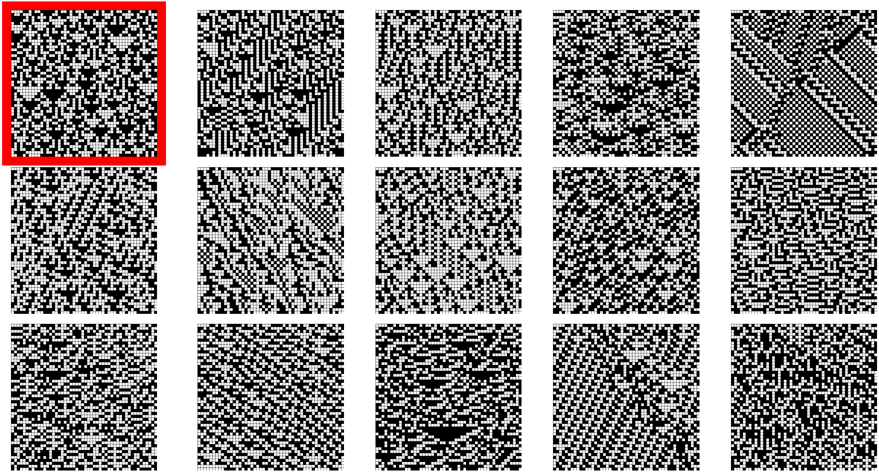
Iteration no. 8



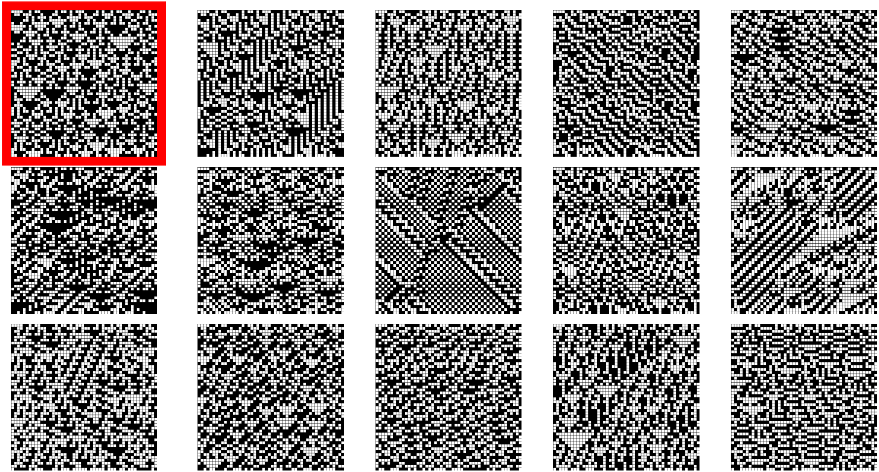
Iteration no. 9



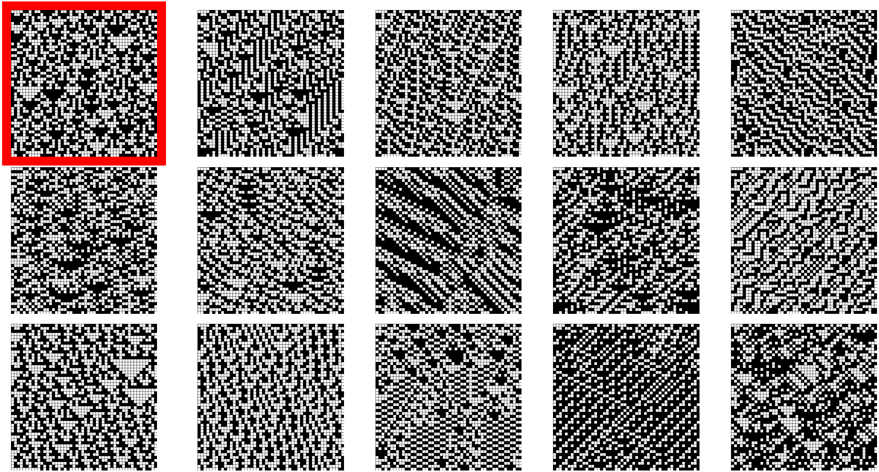
Iteration no. 10



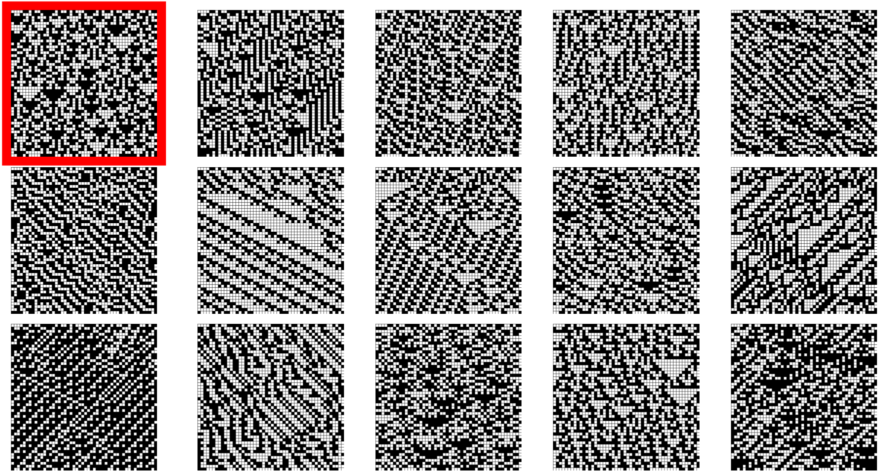
Iteration no. 11



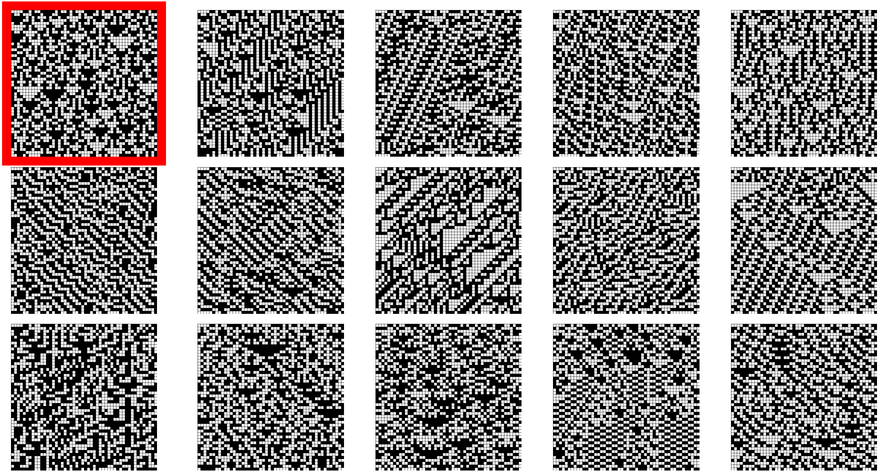
Iteration no. 12



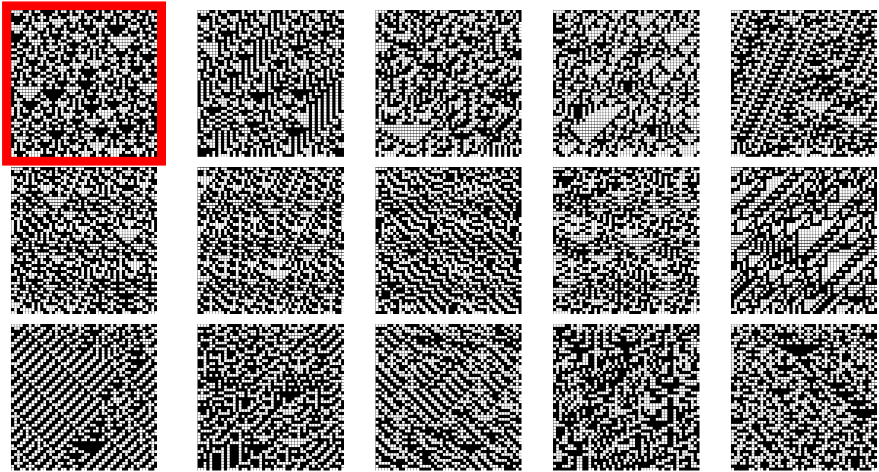
Iteration no. 13



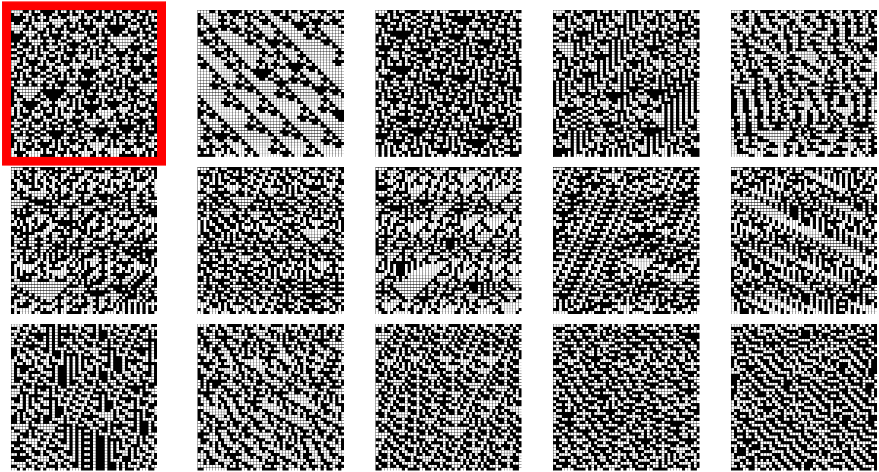
Iteration no. 14



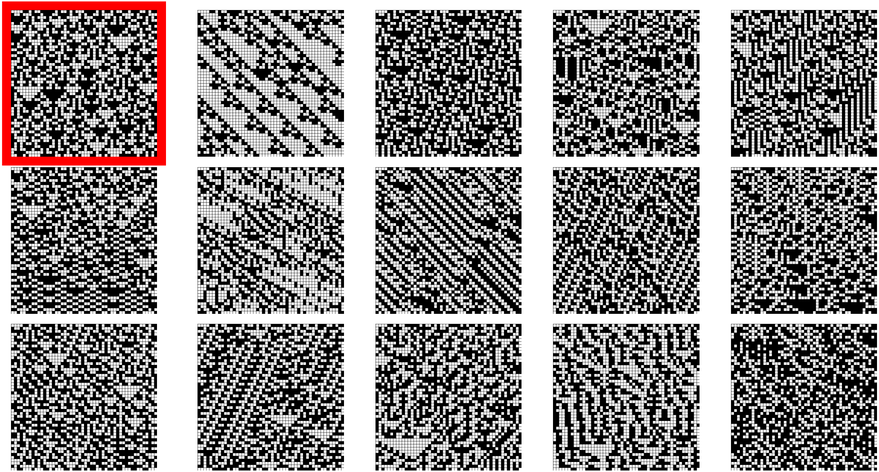
Iteration no. 15



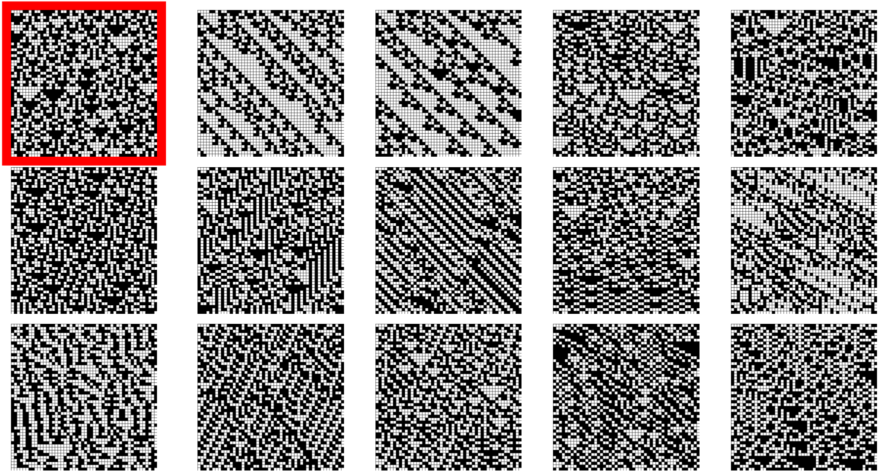
Iteration no. 16



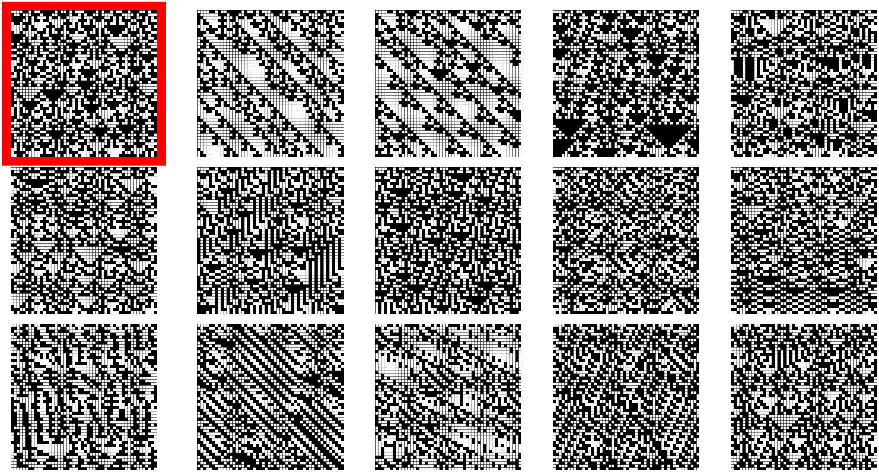
Iteration no. 17



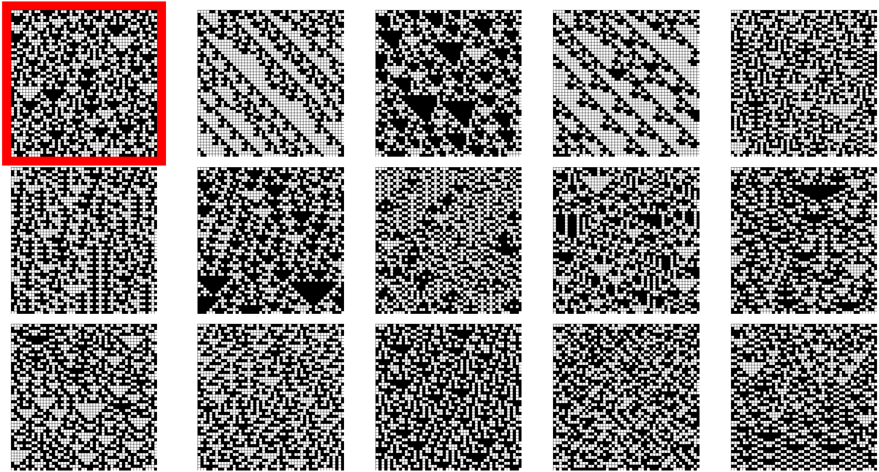
Iteration no. 18



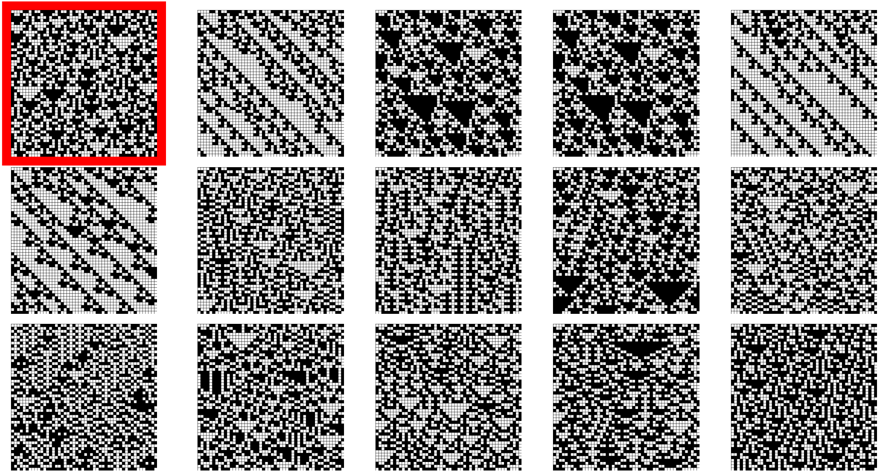
Iteration no. 19



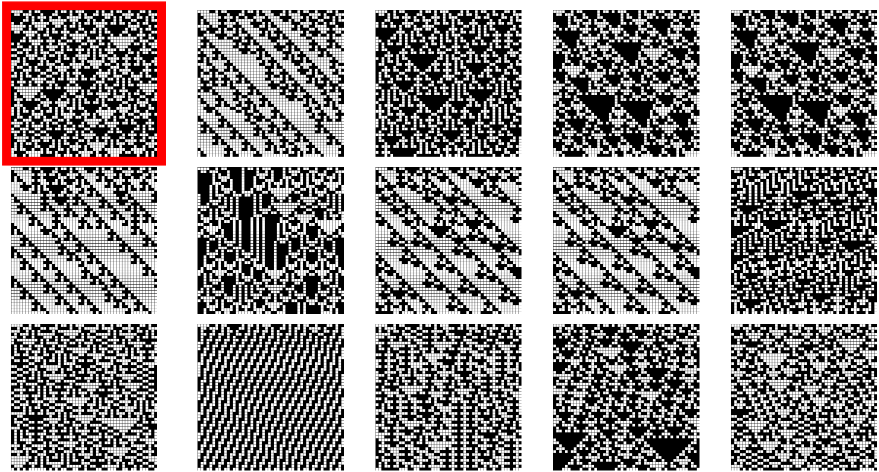
Iteration no. 20



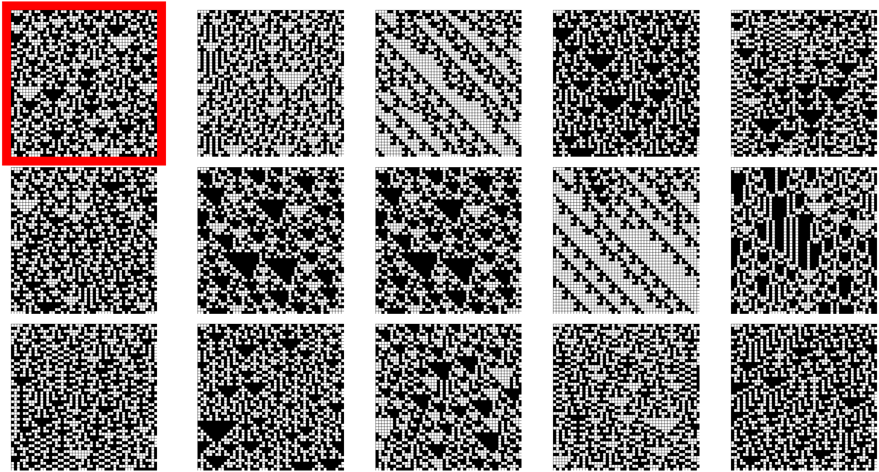
Iteration no. 21



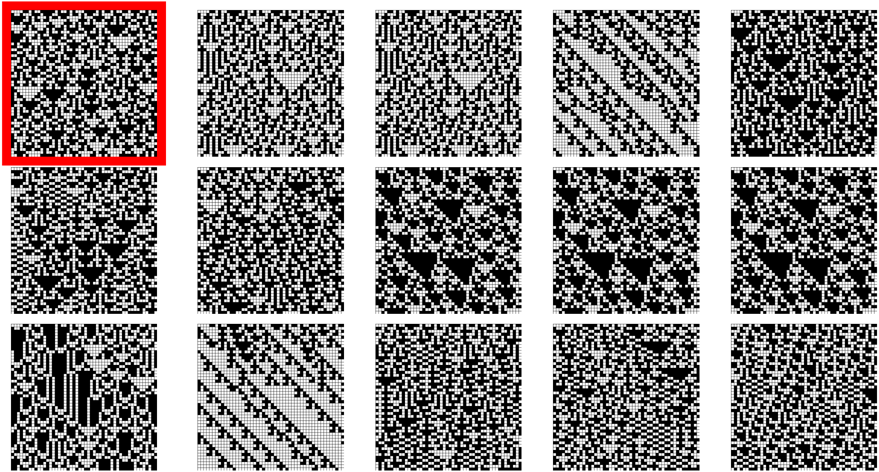
Iteration no. 22



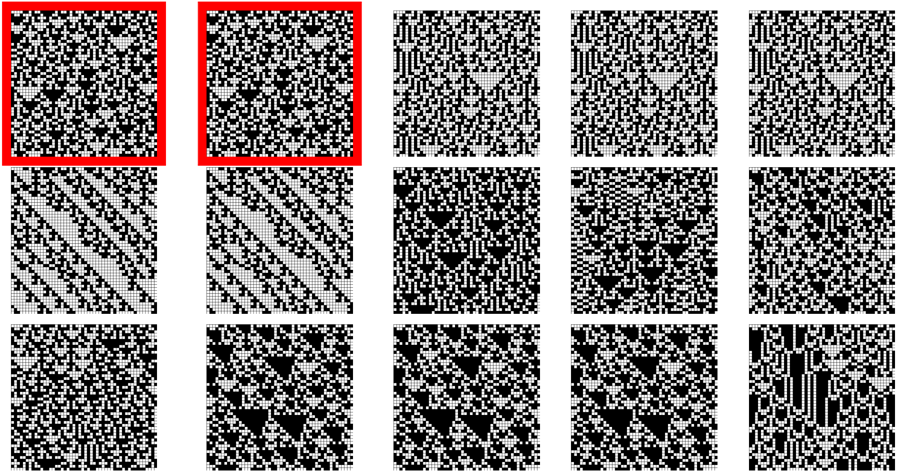
Iteration no. 23



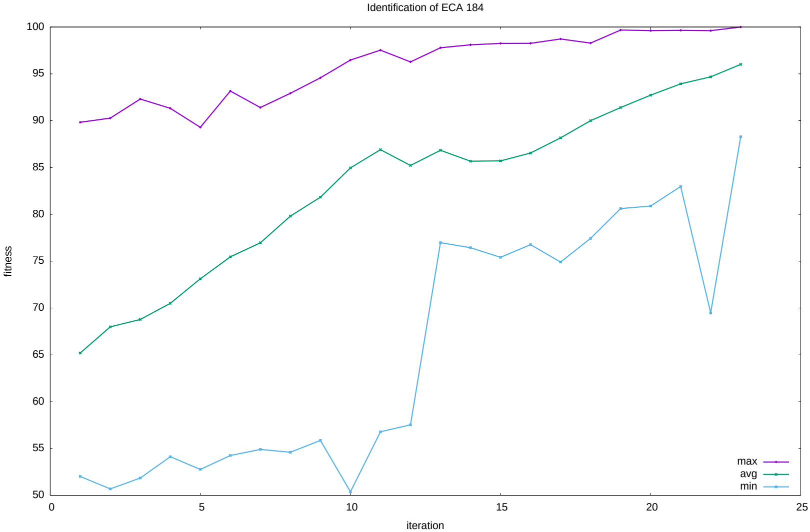
Iteration no. 24



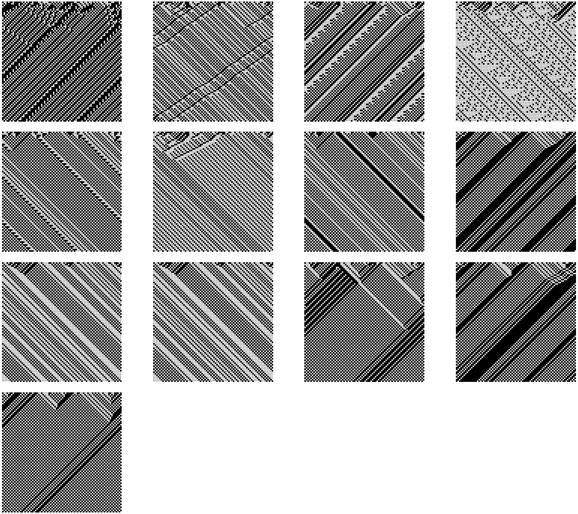
Iteration no. 25



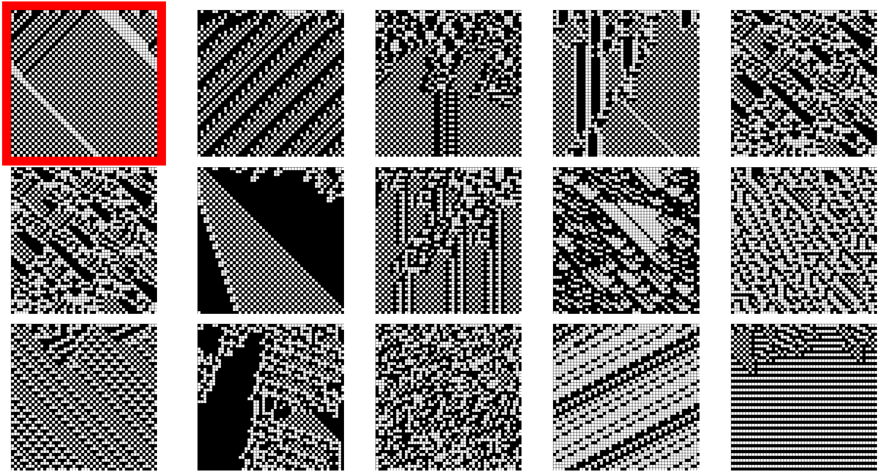
Fitness evolution in time



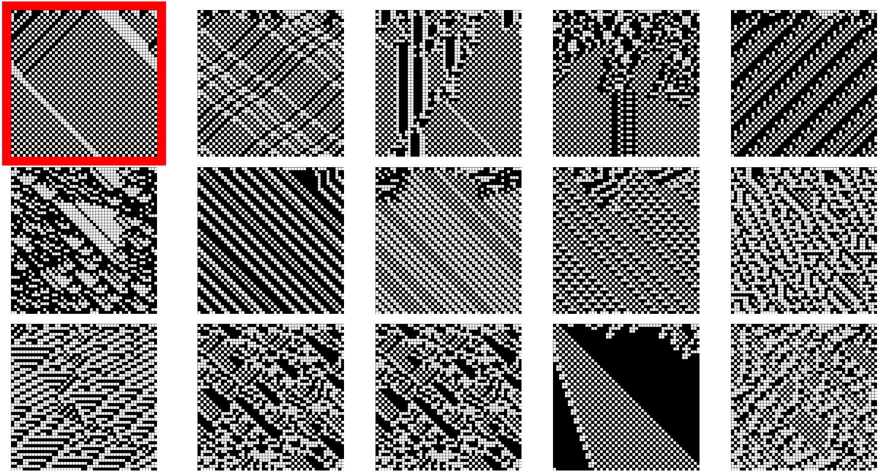
Best individuals



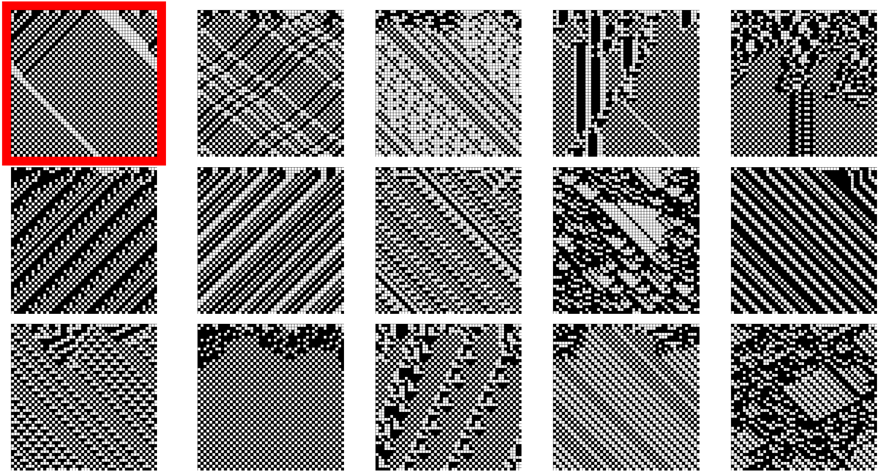
Iteration no. 1



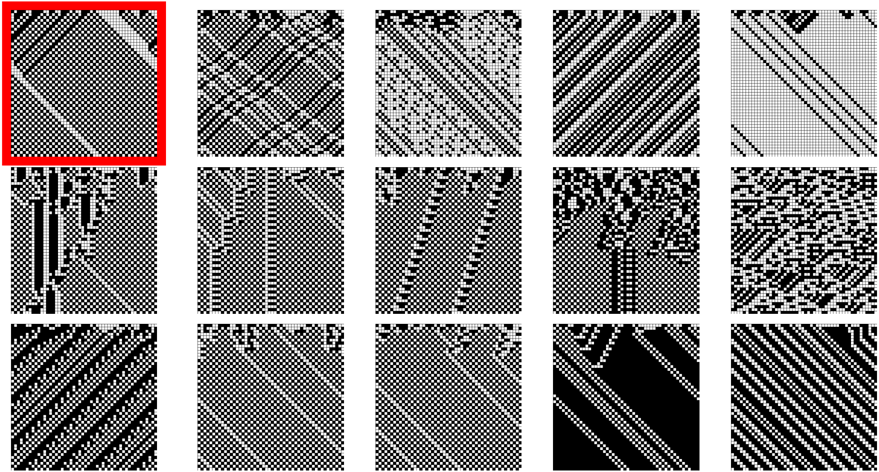
Iteration no. 2



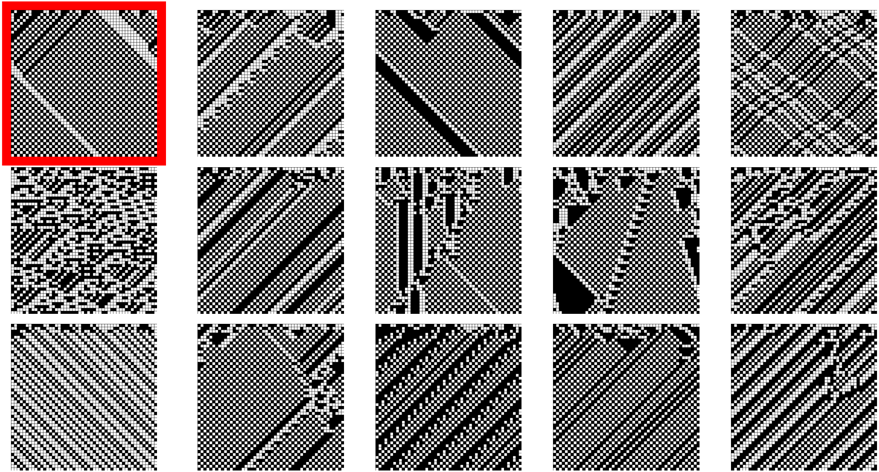
Iteration no. 3



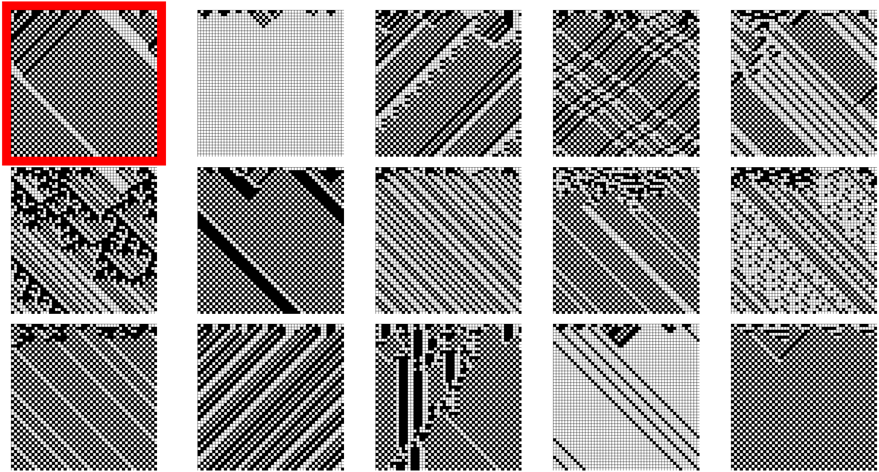
Iteration no. 4



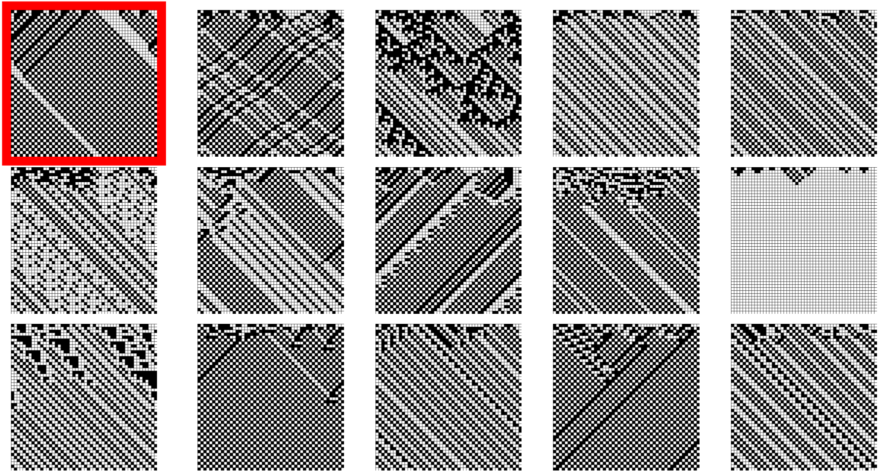
Iteration no. 5



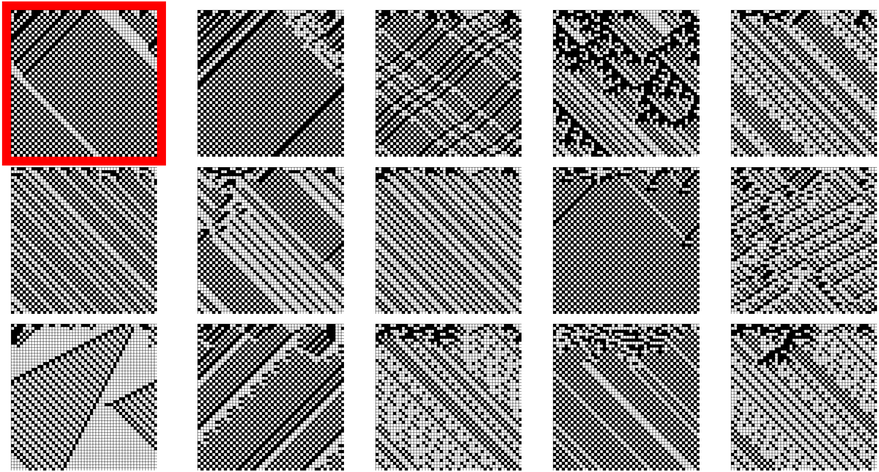
Iteration no. 6



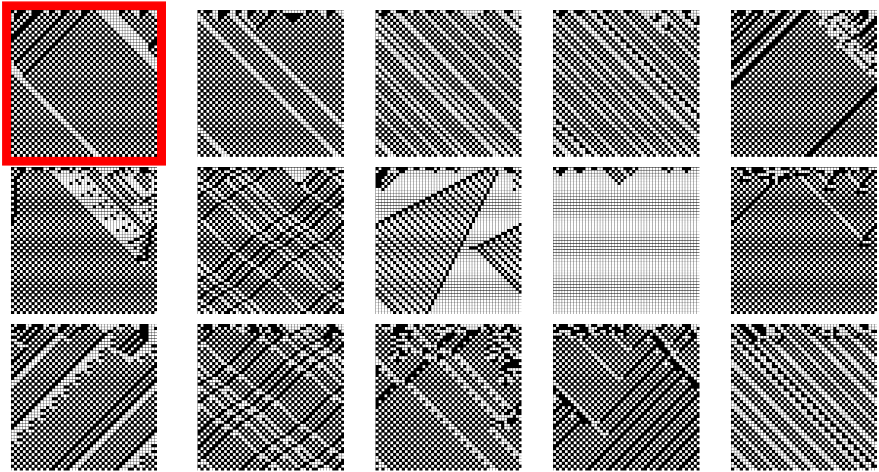
Iteration no. 7



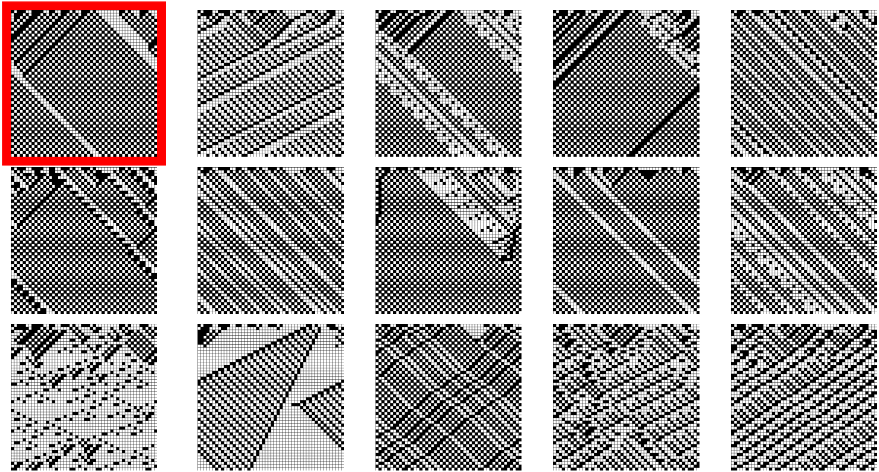
Iteration no. 8



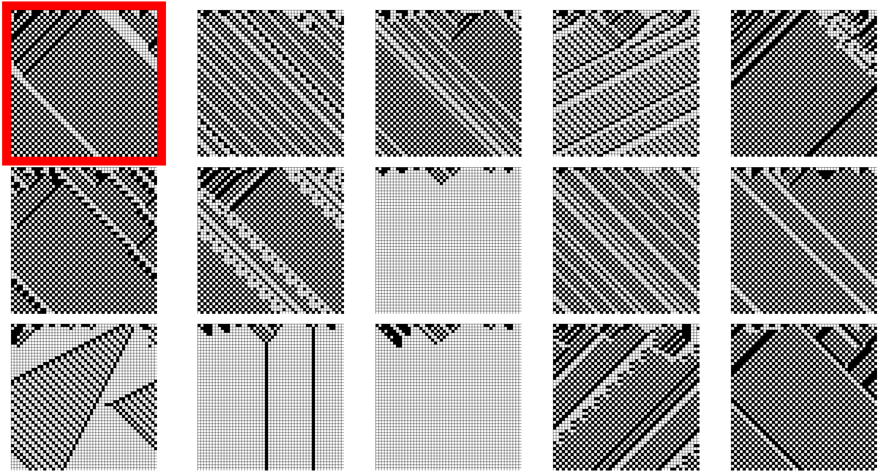
Iteration no. 9



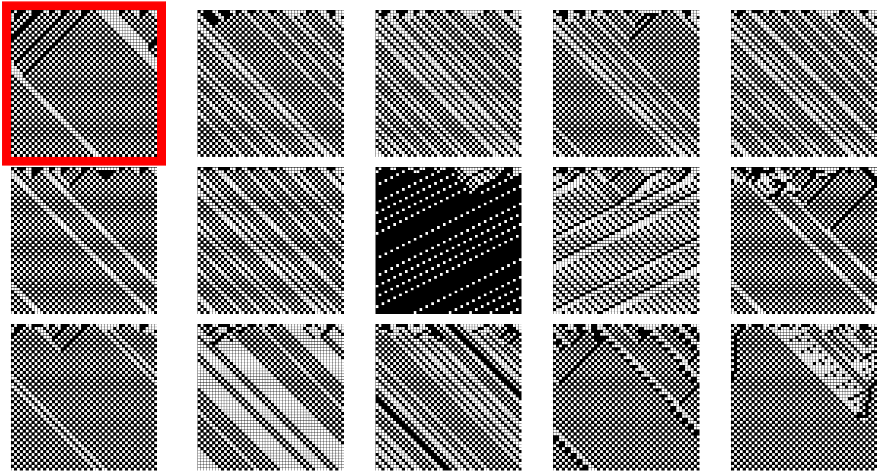
Iteration no. 10



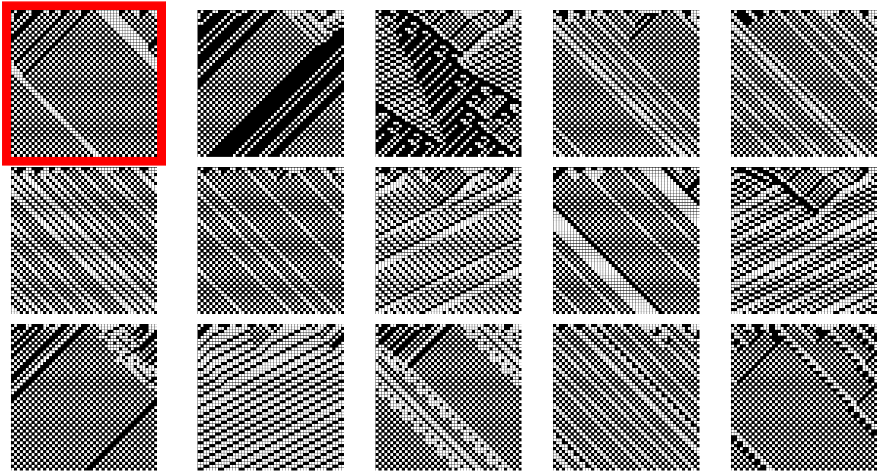
Iteration no. 11



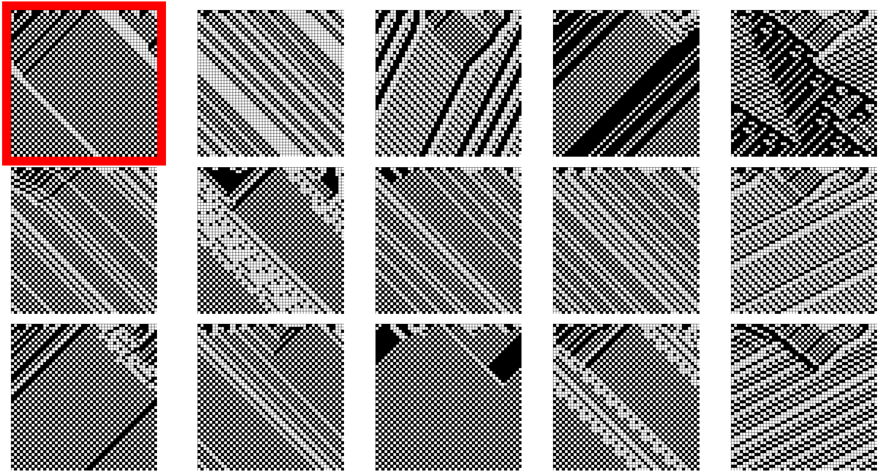
Iteration no. 12



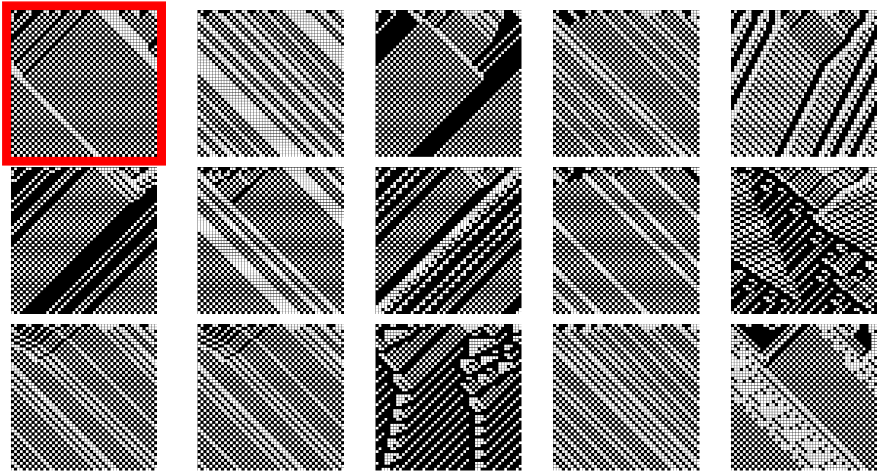
Iteration no. 13



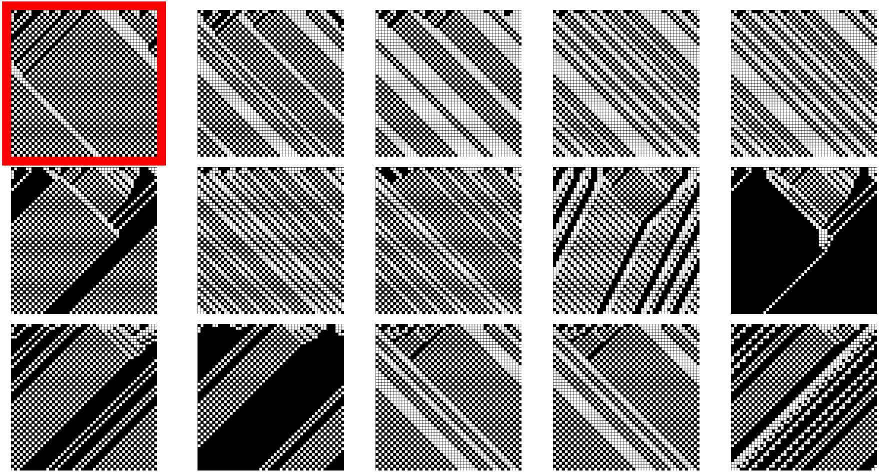
Iteration no. 14



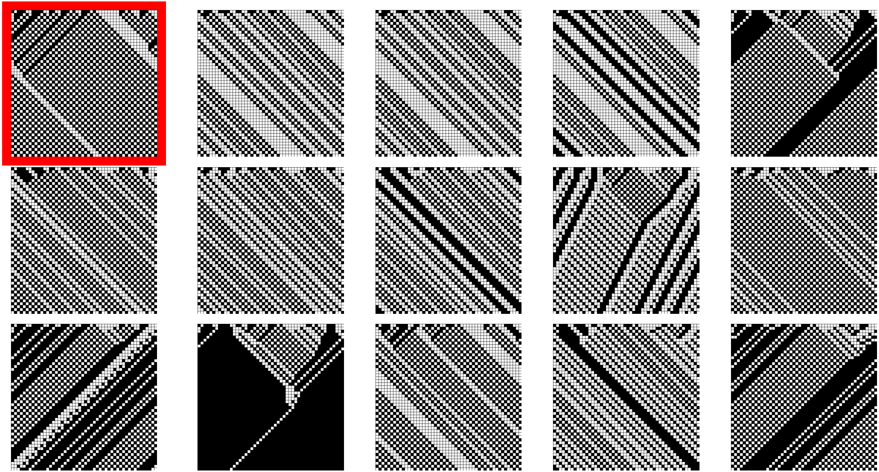
Iteration no. 15



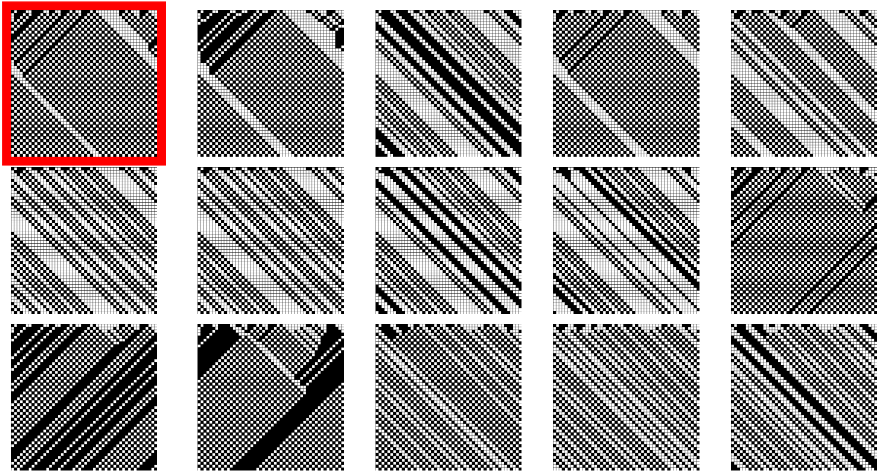
Iteration no. 16



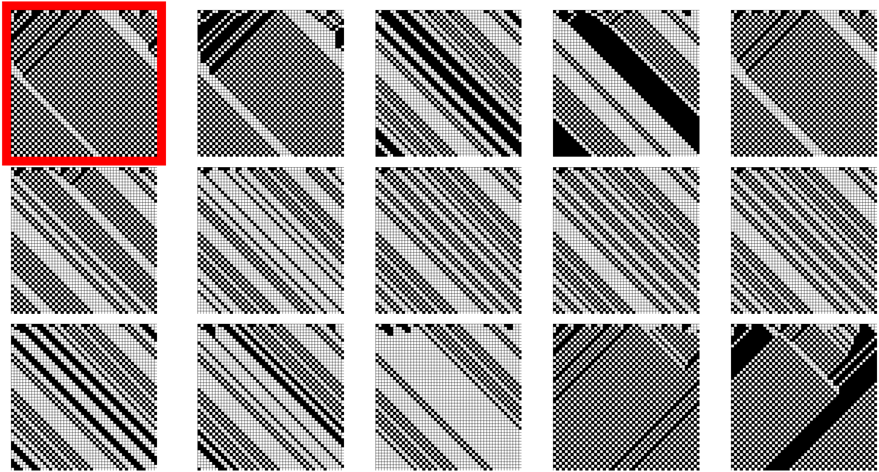
Iteration no. 17



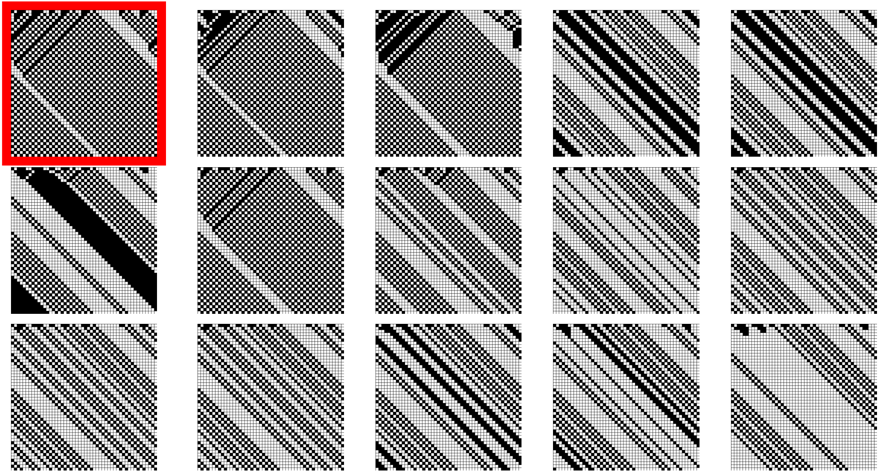
Iteration no. 18



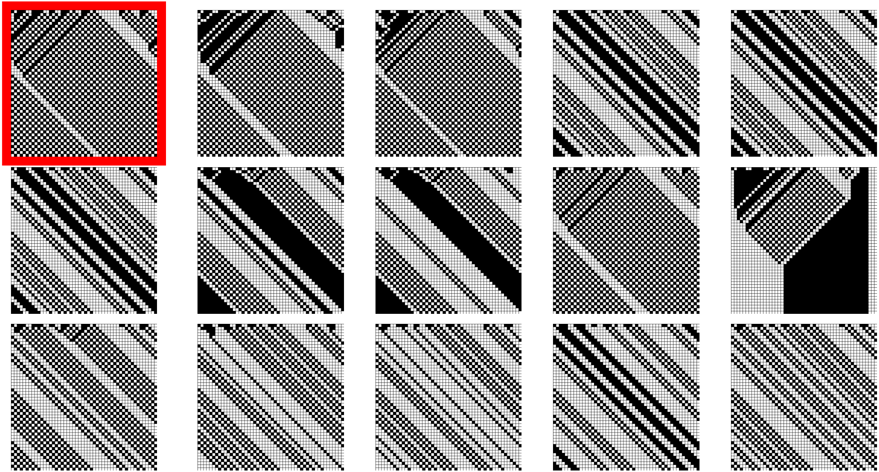
Iteration no. 19



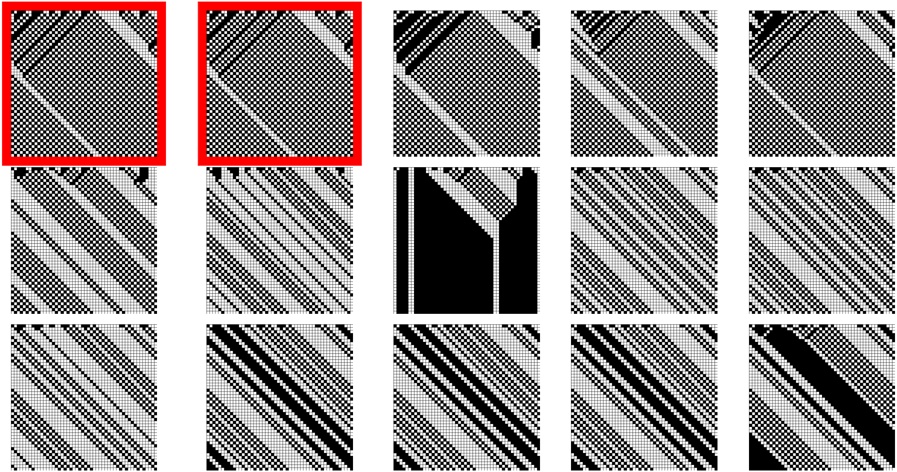
Iteration no. 20



Iteration no. 21



Iteration no. 22



Overview of experiments

Set-up

- Input: sets of observations of known CAs with randomly introduced incompleteness
- Performance: number of GA iterations needed to find the solution
- Goal: evaluate the performance of the GA in various settings

Experiments (first three on ECAs)

1. Relation between performance and dynamical properties in case of temporal incompleteness
2. Impact of constant time gaps on the performance
3. Impact of spatial incompleteness on the performance
4. Radius detection

Experiment 1: Design

- 64 spatially complete observations per ECA
- Same set of 64 random initial conditions
- Upper bound of 10 for independent time gaps
- Number of cells and number of observed time steps: 69 (square observations)
- GA population size: 32 individuals (elite: 8)
- Radii of CAs in populations: from 2 to 5, i.e. the search space has more than $2^{2^{11}}$ elements
- Simulations repeated 50 times per ECA

Experiment 1: Relation with dynamical properties

- In each GA run, a solution was found
- Wolfram classification:

class	avg(min-iter)	avg(avg-iter)	avg(max-iter)
I	38	207	666
II	47	244	915
III	40	296	1264
IV	57	690	2650

- normalized maximum Lyapunov exponent:

nMLE	avg(min-iter)	avg(avg-iter)	avg(max-iter)
$-\infty$	44	257	934
$\mathbf{0}$	38	176	665
$> \mathbf{0}$	49	308	1187

Experiment 1: Easiest and hardest ECAs

- Easiest:

ECA	nMLE	min-iter	avg-iter	max-iter
255	$-\infty$	1	3	9
0	$-\infty$	4	30	183
221	$-\infty$	25	47	86
207	0	25	55	156
246	0	27	57	112

- Hardest:

ECA	nMLE	min-iter	avg-iter	max-iter
137	0.6577	48	1679	8270
193	0.6561	52	1625	6221
110	0.6574	49	1348	4269
25	0.5168	43	1262	5690
124	0.6567	59	1231	4431

Experiment 2: Design & Results

- Compared to Experiment 1, each observation I has a randomly selected constant time gap (the GA is unaware of this)
- The nature of the time gaps can greatly affect the performance and even prevent the GA to successfully identify a solution

description	**avg(avg-iter)**
even time gaps	158
odd time gaps	7074

Experiment 3: Design

- Same as Experiment 1
- We gradually introduce spatial incompleteness by changing 2000 randomly selected entries in the set of observations to $?$
- This process is repeated multiple times until we end up with observations of which only the first row is known
- Simulations repeated 50 times per ECA after each step

Performance

Maximal percentage S_A of $?$ symbols in the set of observations for which the identification is successful for at least one of the 50 runs in fewer than $G = 9 \times 10^5$ iterations

Experiment 3: Results

- Wolfram classification:

class	avg(S_A)
I	98.14 %
II	74.46 %
III	58.10 %
IV	55.17 %

- normalized maximum Lyapunov exponent:

nMLE	avg(S_A)
$-\infty$	81.63 %
$\mathbf{0}$	83.74 %
$> \mathbf{0}$	66.61 %

Experiment 4: Design

- Randomly selected CAs with radii 2, 3, 4 (100 CAs per radius)
- 64 spatially complete observations per CA
- GA population size: 512 individuals (elite: 128) with radii from 1 to 5
- Algorithm is stopped when finding a solution or after $G = 10^4$ iterations
- No repetitions

Experiment 4: Results

- Only 6 runs were unsuccessful
- Breakdown according to radius:

radius	min(iter)	avg(iter)	max(iter)
2	25	217	4988
3	77	325	2457
4	250	572	2173

- For radii 3 and 4, the solution always has the desired radius, and a strict majority of the population has this radius some time before finding the solution
- For radius 2, the same happens for 93% of the cases; otherwise, the final solution and the majority of the population have a larger radius

Experiment 4: Results

Performance

Percentage of the number of GA iterations before finding the solution, during which a strict majority of the population has the desired radius

radius	avg(% iter)
2	57.74%
3	63.26%
4	75.00%

Conclusion

1. Measures from dynamical systems theory allow to gain insight into the interplay between CA dynamics and model design
   - design parameters can have a dramatic impact on the dynamical properties
   - further research needed: multi-state Lyapunov exponents, Lyapunov profiles, ...
2. Evolutionary techniques are able to discover CA rules based on sets of observations
   - the GA can handle various types of temporal incompleteness very effectively
   - the GA is very effective in case of no or limited spatial incompleteness; in case of higher spatial incompleteness, the performance depends on the rule
   - the performance differs greatly among rules, and is related with the dynamical characteristics of the rules

Thank you for your attention!