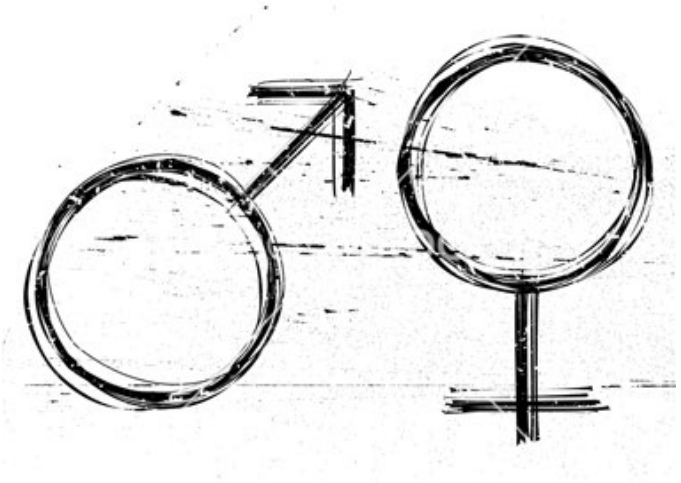
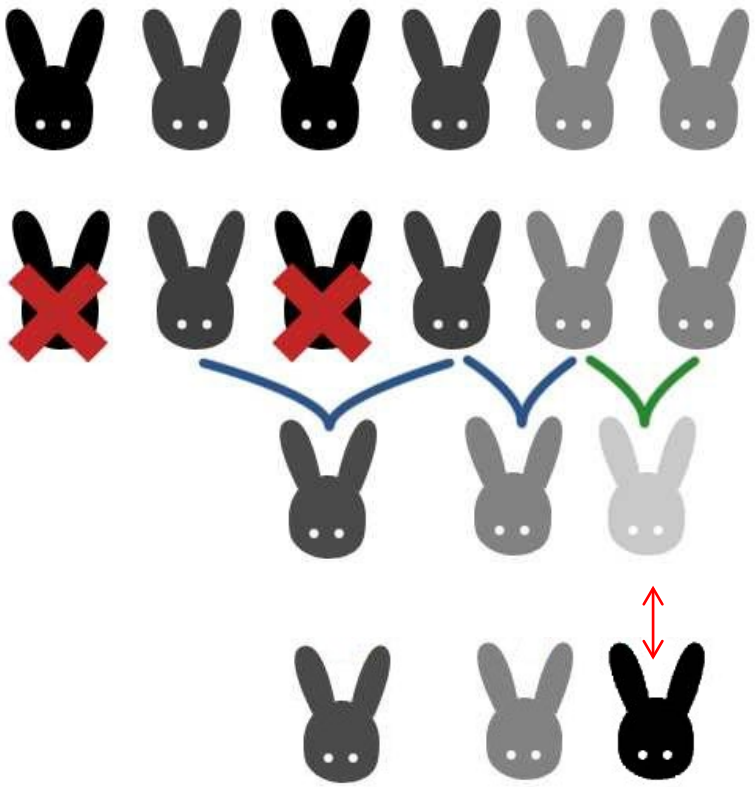
Computer modeling of physical phenomena



Lecture V – Evolution and genetic algorithms

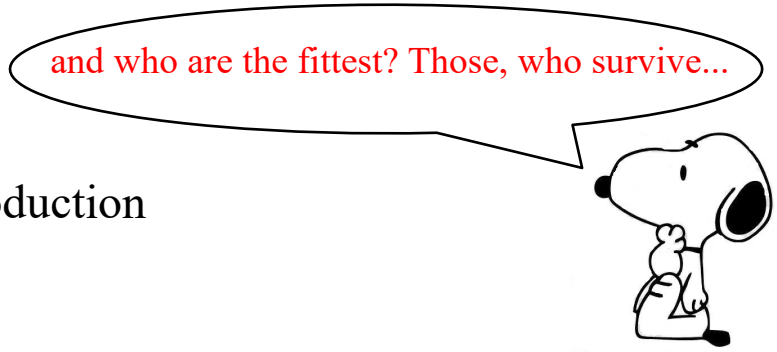
Lab V – breeding cellular automata

Evolution in 5 seconds



initial population

natural selection (the fittest survive)



reproduction

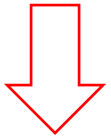
mutation

Evolution is a consequence of

- heredity (memory);
- variability (mutations);
- each generation providing more individuals than can survive (surplus)

Uniqueness:

- evolution acts on a very small subset of all possible individuals
- new variants arise by small variations of what is already present



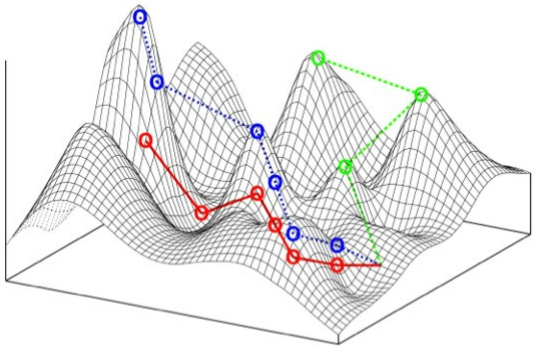
if life started again, the outcome would be enitrely different

Evolution is successful

in finding (nearly) optimal solutions to complex problems

Genetic algorithms try to mimic that...

- and find a (nearly) optimal solution in a high-dimensional, rugged fitness landscape

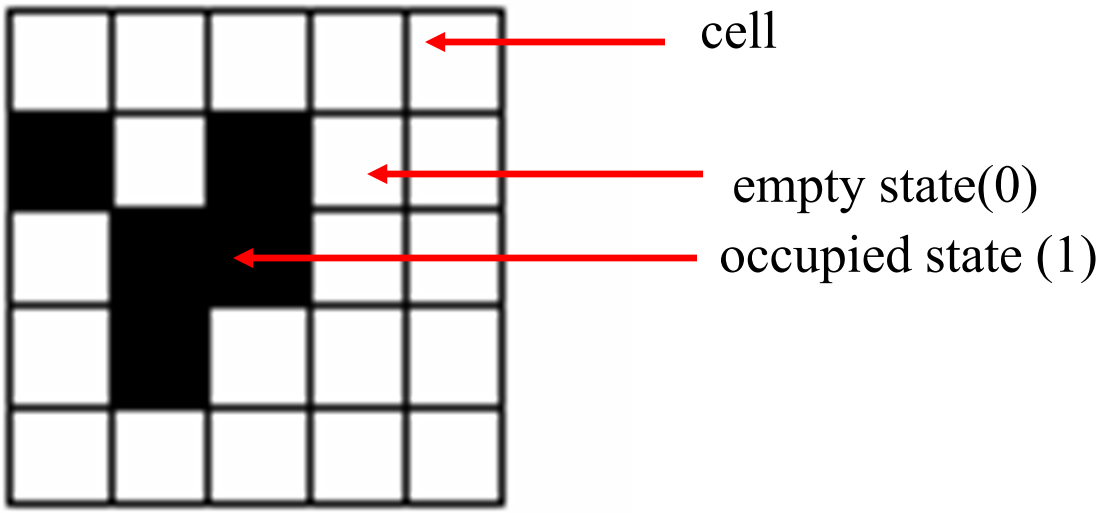


- particularly useful in cases when the complexity of the problem of interest makes it impossible to search every possible solution or combination
- instead, the aim is to find good, feasible solutions in an acceptable timescale
- there is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why (if it does work)

Genetic algorithms

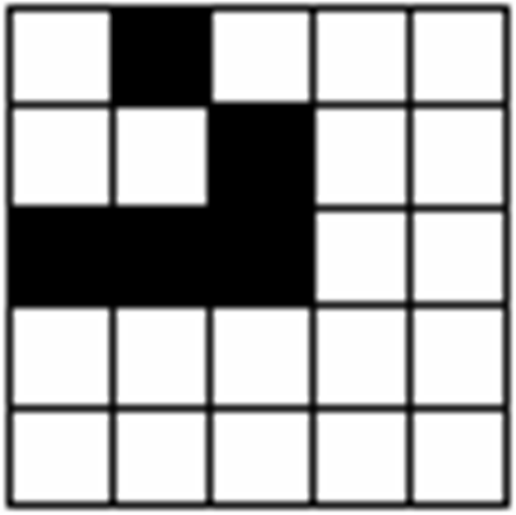
- Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals or phenotypes) to an optimization problem evolves toward better solutions.
- Traditionally, genotype represented in binary as strings of 0s and 1s, but other encodings are also possible.
- The evolution usually starts from a population of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population.
- The new population is then used in the next iteration of the algorithm.
- Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

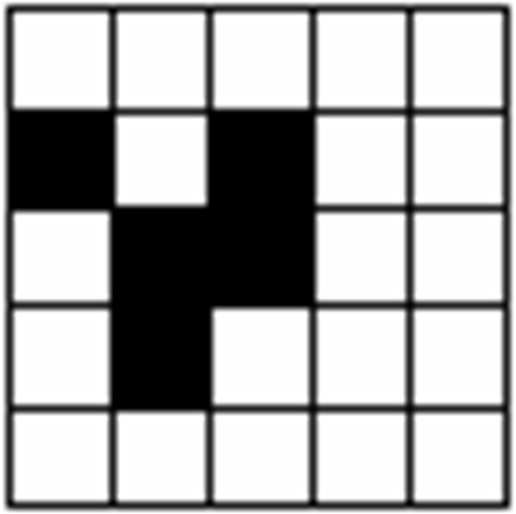
Cellular automaton



- Automaton is defined on a grid of cells, each of which can attain a discrete number of states

Evolution



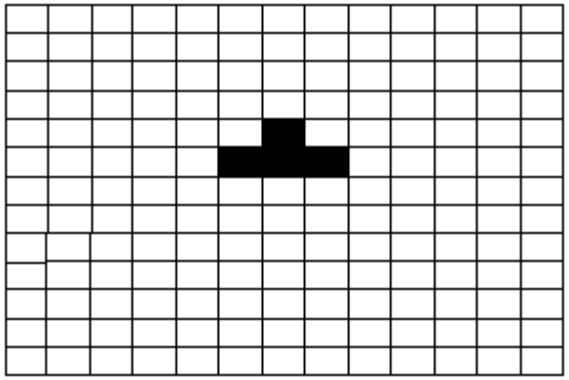


Cellular automaton evolves according to the *transition rules*, which determine the future state of the cell based on the present state of both the cell and its neighbourhood.

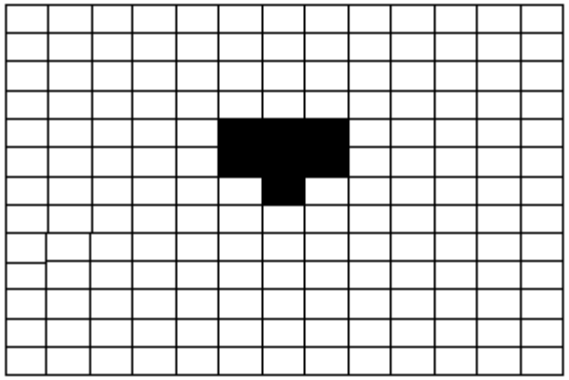
Example (Game of life)

Transition rules :

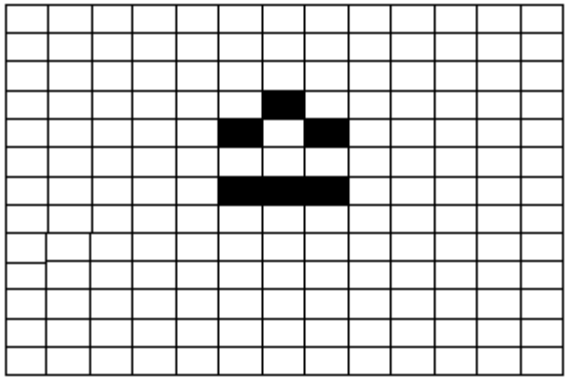
- the cell remains in state 1 (black), if it has 2 or 3 black neighbours the cell gets transformed to black, if it has exactly 3 black neighbours (birth)
- in other cases the cell remains (or gets transformed to) white



initial state

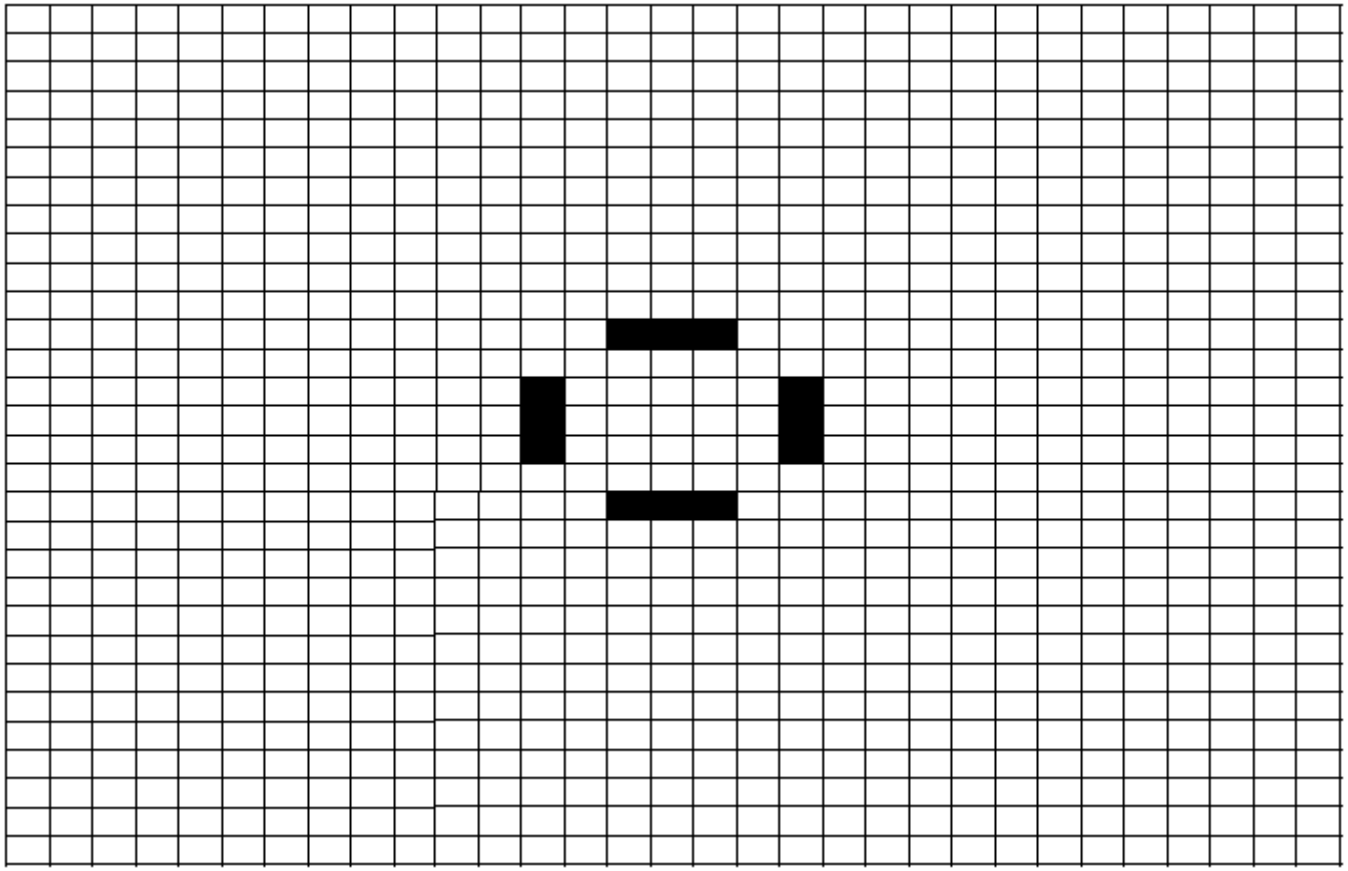


step 1



step 2

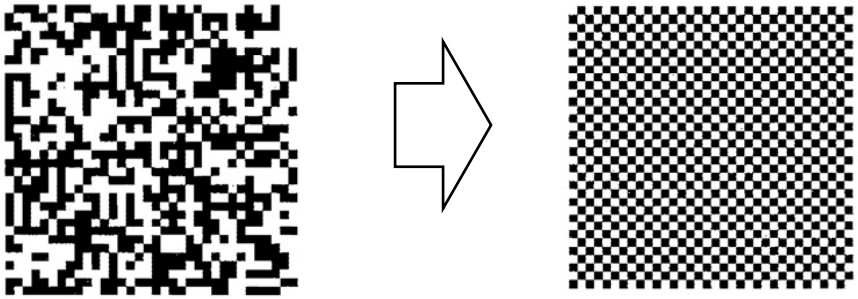
Evolution



Example: Breeder

Find checkerboard CA

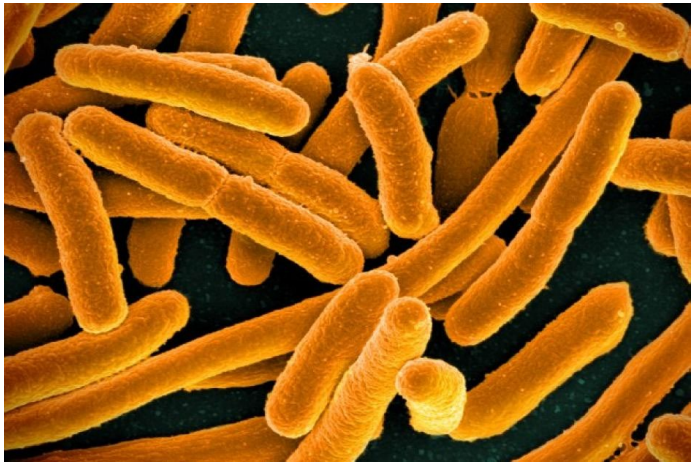
- Start from an array in an initial random configuration of black (=1) and white (=0) cells
- The aim is to tum the whole array into a checkerboard pattern



- Can you find such a cellular automaton?
- There are $2^9 = 512$ different neighborhood-states, thus each rule is encoded by 512-digit binary string

There are 2^{512} automatons, how to find the one we are looking for?

Use genetics!



- Breed cellular automata like bacteria
- In their genome, encode the rule of CA

- In each generation, give them the task and let the best ones survive
- Then recombine them and mutate to form a new population

Chromosomes: rule encoding

a_0	a_1	a_2
a_3	a_4	a_5
a_6	a_7	a_8

Neighbourhood state:

$$N = \sum_{i=0}^{8} 2^i a_i$$

Each rule is encoded by 512-digit binary string

1001 101 . ..110 ⟵ chromosome

genes: Nth place in the rule string determines the state of the center cell in the next timestep (0 or 1)

Iterations: new=Rule[N(old)]

where Rule is a rulestring written in form of a 512-element list

Periodic b.c

Use *roll*, e.g. *np.roll(np.roll(Z, -1, axis=0),-1,axis=1)* shifts the matrix (periodically) by (-1,-1)

Fitting function

each chromosome is given a fitness test in the form of how well
it converts an array of random black and white cells into an approximation of a checkerboard:

- make 100 timesteps with random initial conditions and then
- For each cell (x, y):
  - Subtract 3 points if cell (x+1,y) or (x,y+1) is in the same state as (x,y)
  - Otherwise, add 8 points if cell (x+1, y + 1) is in the same state as (x, y) and subtract 5 points if it is not, and likewise add 8 points if cell (x+1,y-1) is in the same state as cell (x, y) and subtract 5 points if it is not
- repeat this for a few random configuration and take an average
- use different set of configurations each time (i.e. each generation)

Cloning

- Clone each chromosome a number of times that depends on its level of fitness

$$f_i = \frac{F(c_i)}{\sum_k F(c_k)}$$

fraction of ith chromosome in the cloned population

fitness of chromosome k

Reproduction

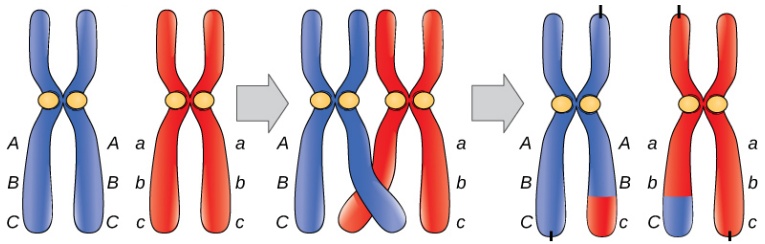
pick up pairs of chromosomes at random from the population and let them be "parents"

1001 101 . ..110

1110 110 . ..000

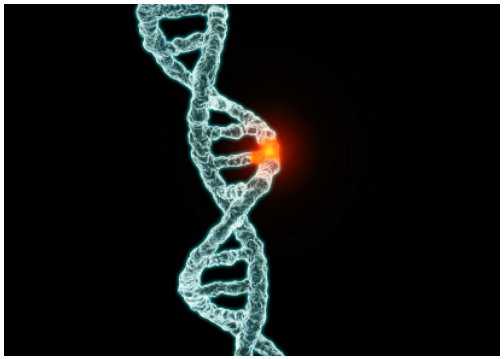
1101 110 . ..000

align their chromosomes and create a new "baby" string by selecting one of the state values at each of the 512 digit positions randomly from either one of the parents.



Mutation

- mutate randomly 1-2 bits in some of the chromosomes



The evolution in action

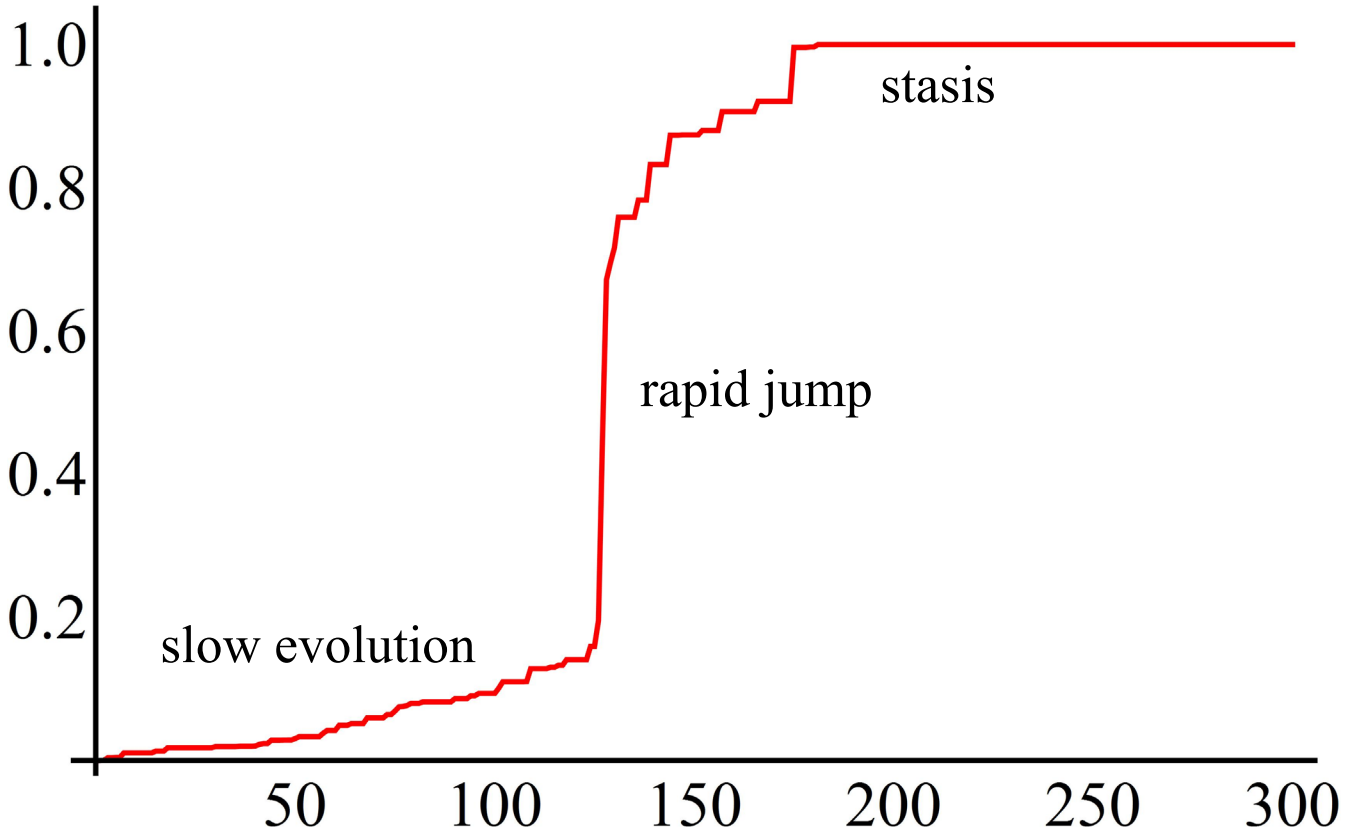
30th generation

80th generation

140th generation

200th generation

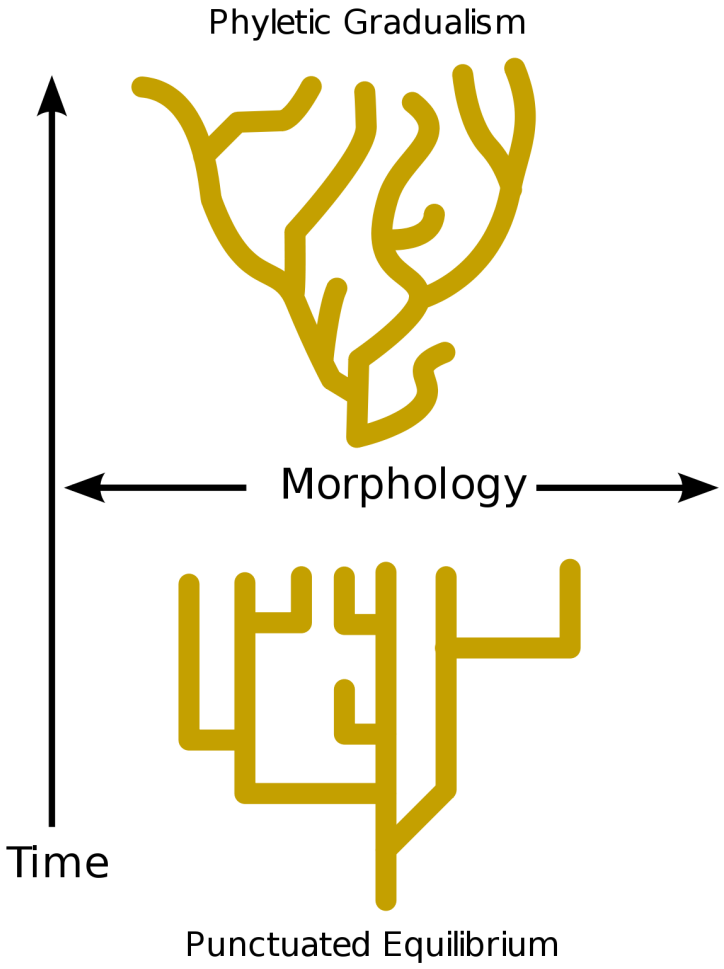
Fitting function vs time

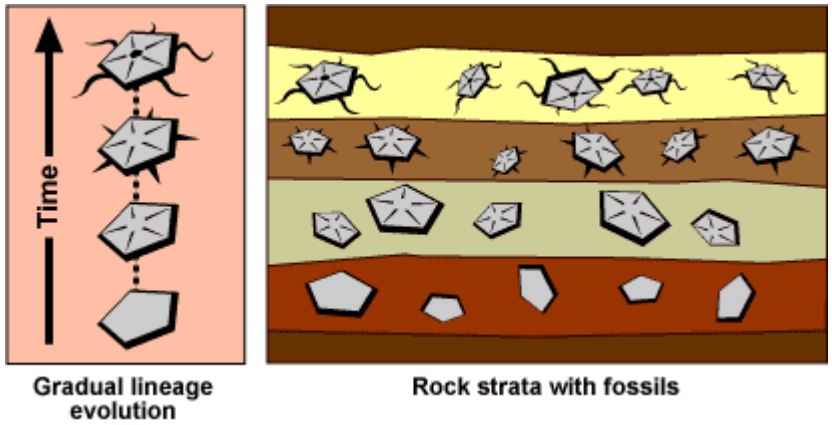


Punctuated equlibrium: significant evolutionary changes are restricted to rare and geologically rapid events followed by periods of morphological stability

(Eldredge, Gould, 1972)

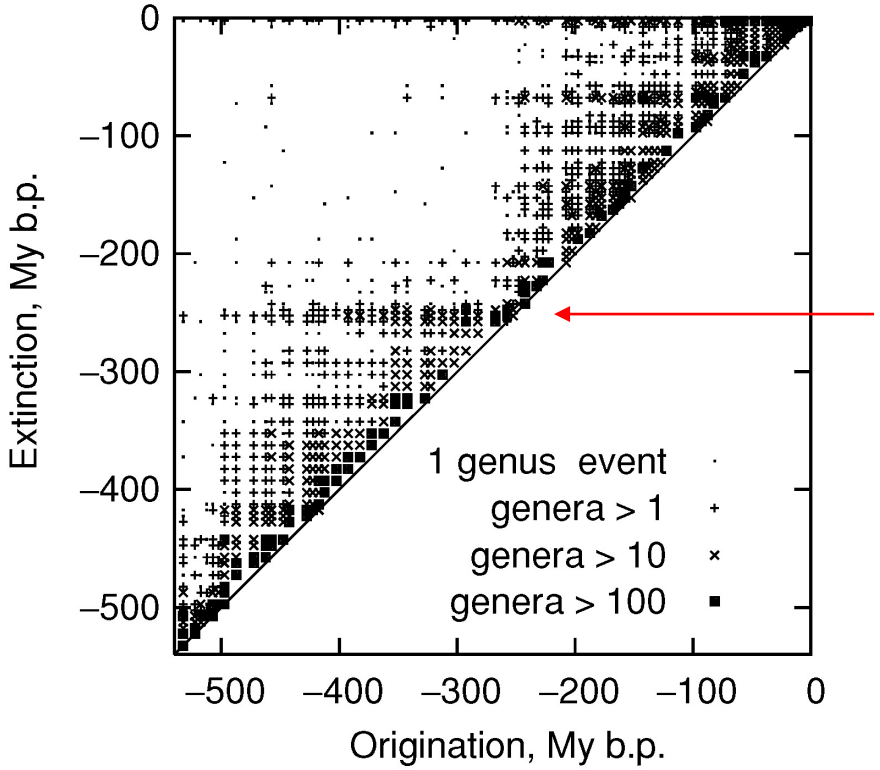
Punctuated equilibrium





species are generally morphologically stable, changing little for millions of years.
This leisurely pace is "punctuated" by a rapid burst of change that results in a new species.

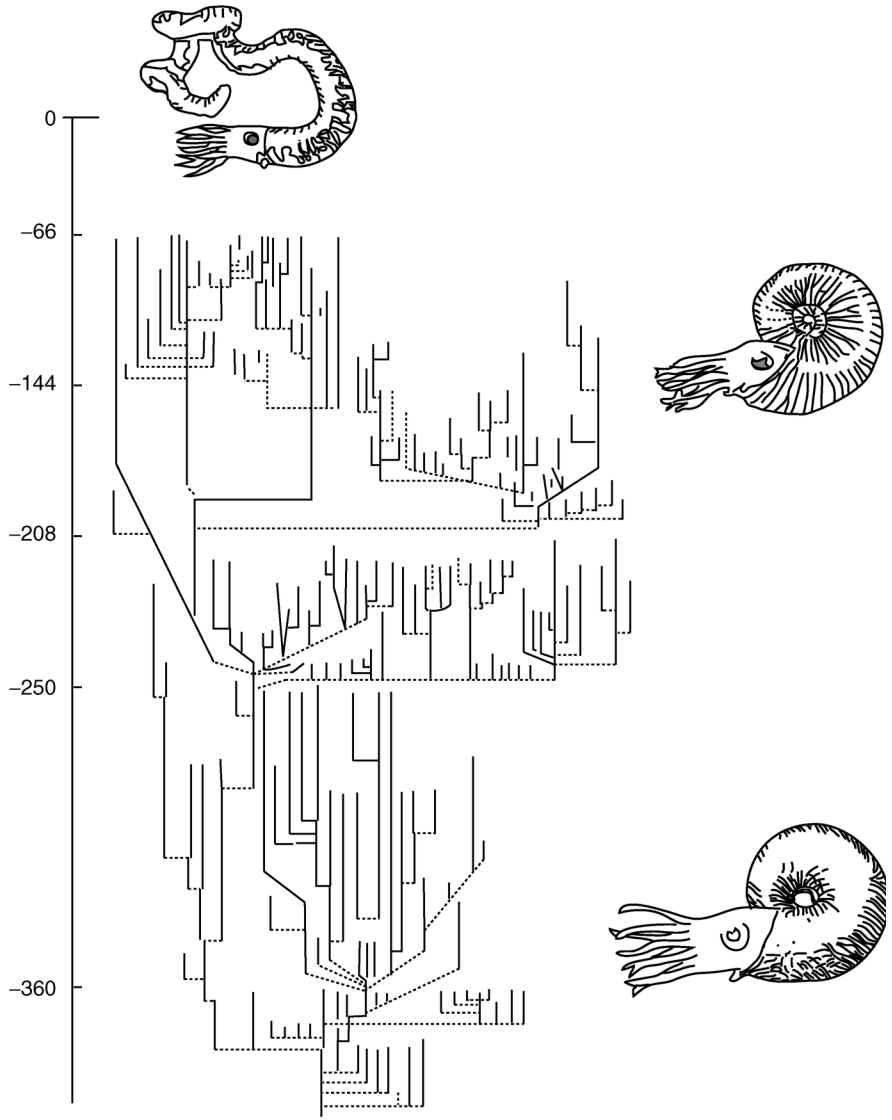
Cooperativity



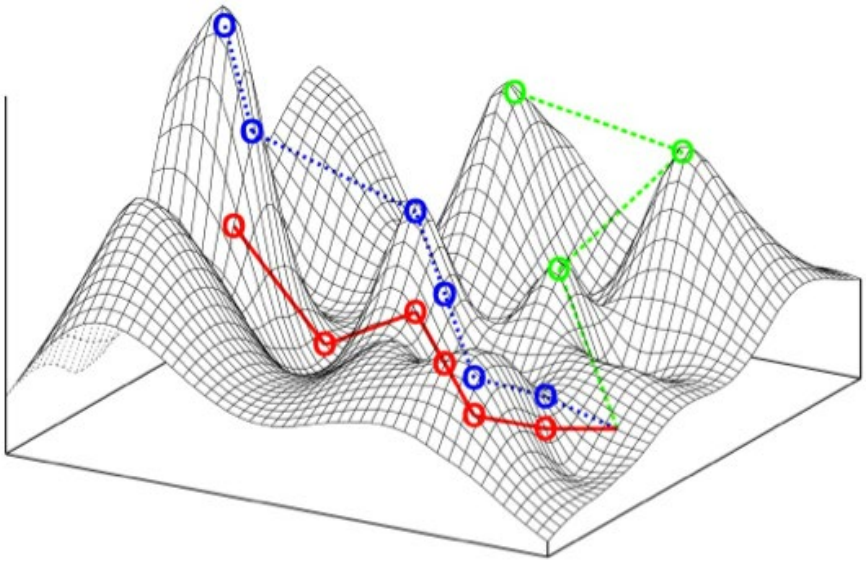
Permian extinction

The species are interlinked within ecosystems, which can stably coexist over millions of years.

Ammonoid family tree



Dynamic landscape



Due to cooperativity, the fitness landscape itself evolves with the species - the mere concept of the fitness becomes then questionable. Such interlinked dynamics can increase the stability of the ecosystems but also lead to great extinctions when this stability is finally disturbed

Your task

- try to find as effective „checkerboard CA” as possible
- experiment with parameter values (populations size, number of chromosomes mutated etc)
- plot a fitness function (max over the populations) vs time

Extra 1 point for the best CA (for the winner or to split between the winners)

Some hints

- It is good to keep the size of the population constant – balancing cloning, reproduction and deaths
- The evolution algorithm can be changed – for example, you can give up cloning but instead use the reproduction probability which depends on the fitness level
- When assessing the fitness it is good to average the value over last few timesteps
- The size of the population does not need to be large – very good results were obtained even with the population of just 10 chromosomes, 5 of them (above the median) surviving and 5 offsprings; but there were some algorithms based on the population of 50 chromosomes also performing very well – it is all up to you
- You can begin by breeding the cellular automata on 50x50 lattice; but if you want to take part in the competition then make sure that it also performs well on larger lattices (you can move the breeding to a larger lattice when you have obtained a strong population, performing well on smaller lattices)