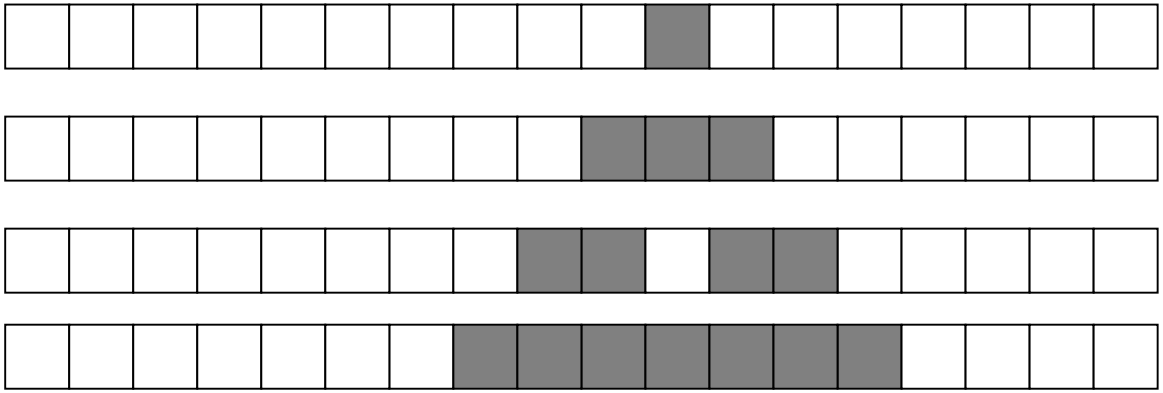
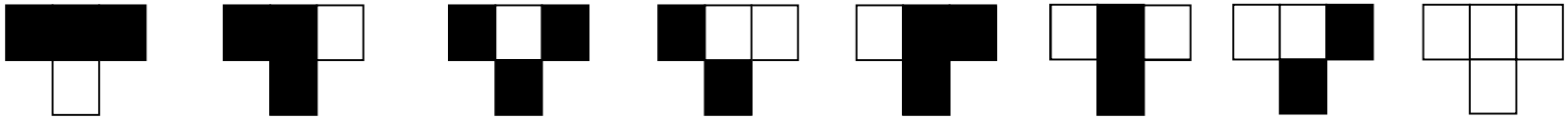
Genetic Algorithms

- Proposed by J. Holland in 60's during his work on *cellular automata*
- First „big" publication for GA function optimization is a book by D. Goldberg from 80's
- 2015: sear Google for keywords „Genetic algorithms" leads to 2 920 000 links (in comparison to „Simulated annealing"1 430 000 results)
- As a *standard genetic algorithm* a binary version of GA from those times is used

Cellular Automata (CA)



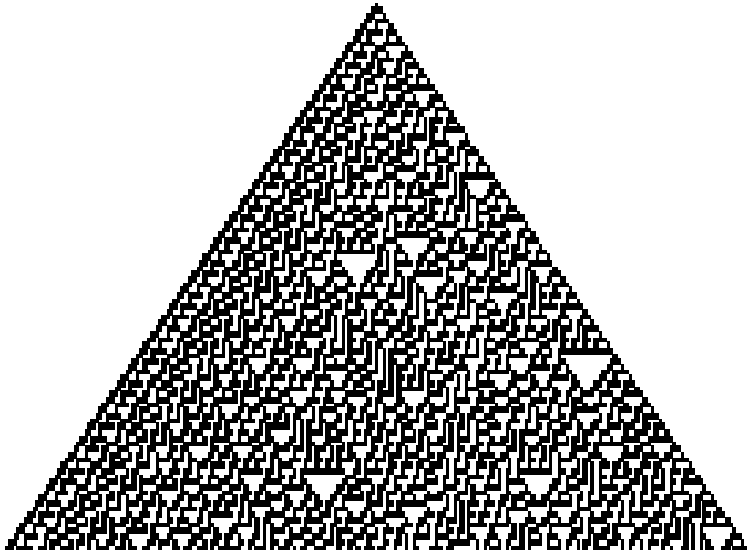


Rules:

- 3 Black = White
- 2 Black = Black
- 1 Black = Black
- 3 White = White

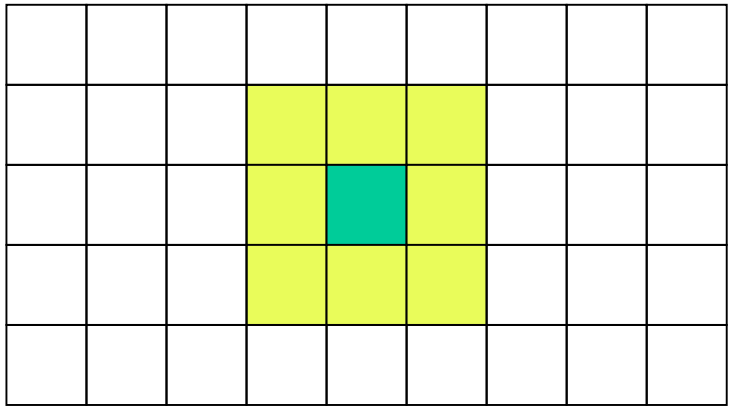
1D Cellular Automata

1-dimensional CA is based on a row of cells. Cells are changing their status / state based on the given rules.



2D Cellular Automata

2-dimensional CA is based on finite (infinite) net of cells, where each of them is in one given *state*. Time is *discrete* and the state in time *t* is given by statuses of its neighbors in time *t-1*.

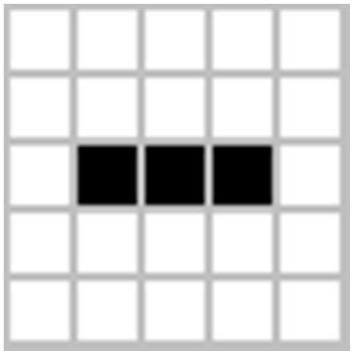


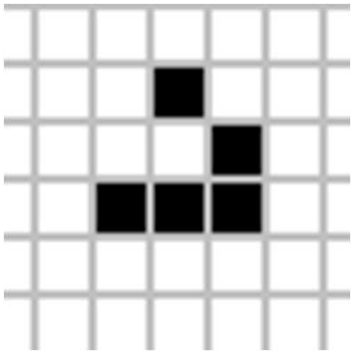
Example: Conway's Game of Life

The universe of the *Game of Life* is an infinite, two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, *alive* or *dead*. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

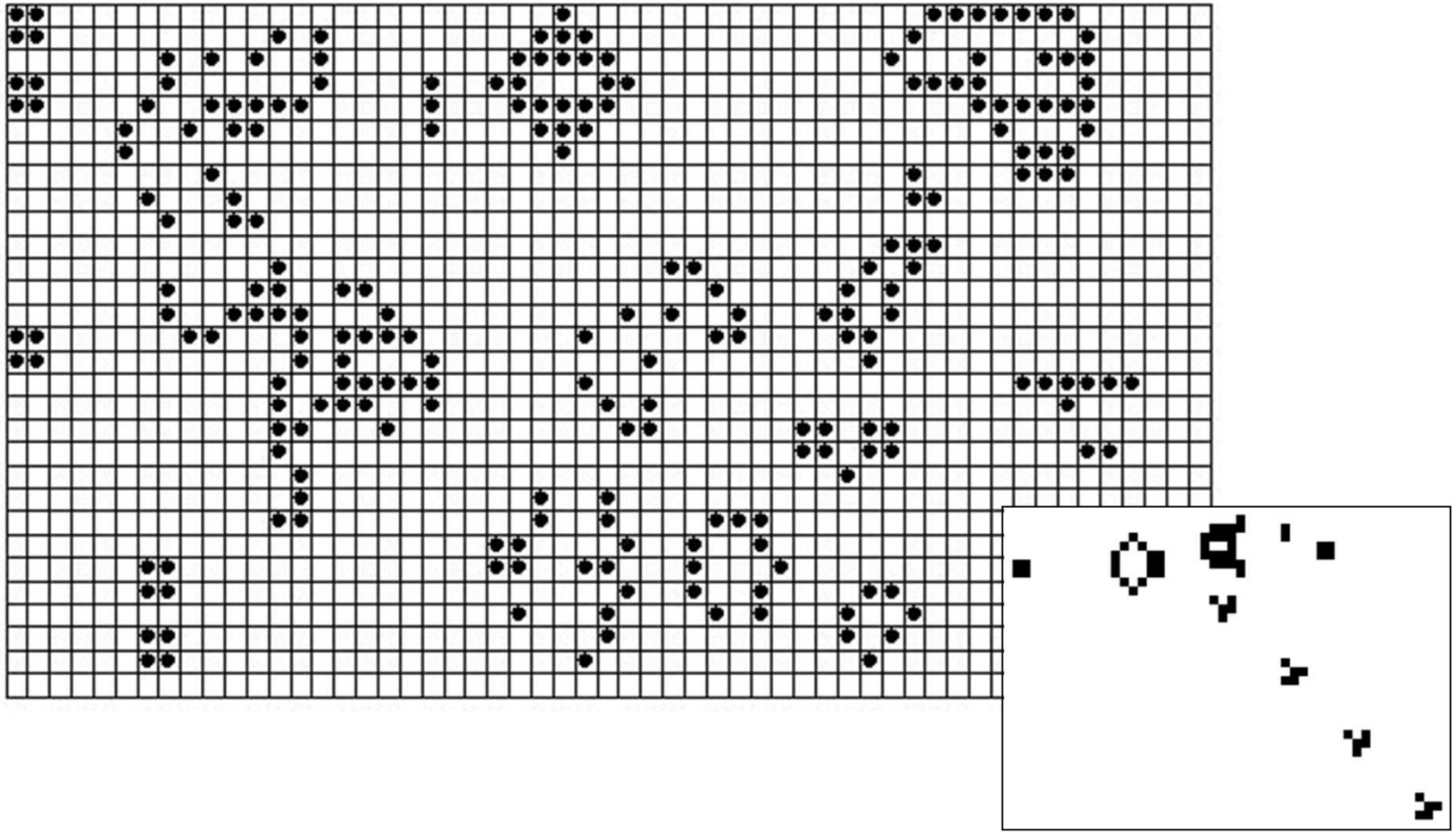
1. Any live cell with fewer than two live neighbors dies, as if by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

[Wikimedia]





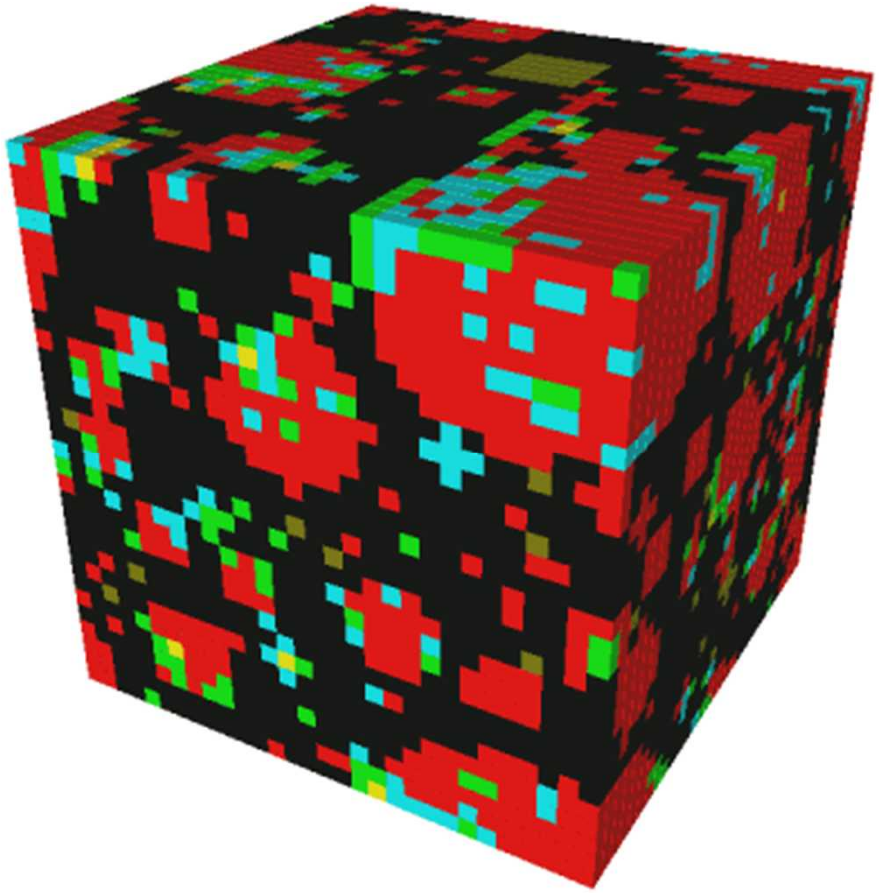
Example: Conway's Game of Life



Př.: Conway's Game of Life

Practical usage: Modeling of concrete microstructural development

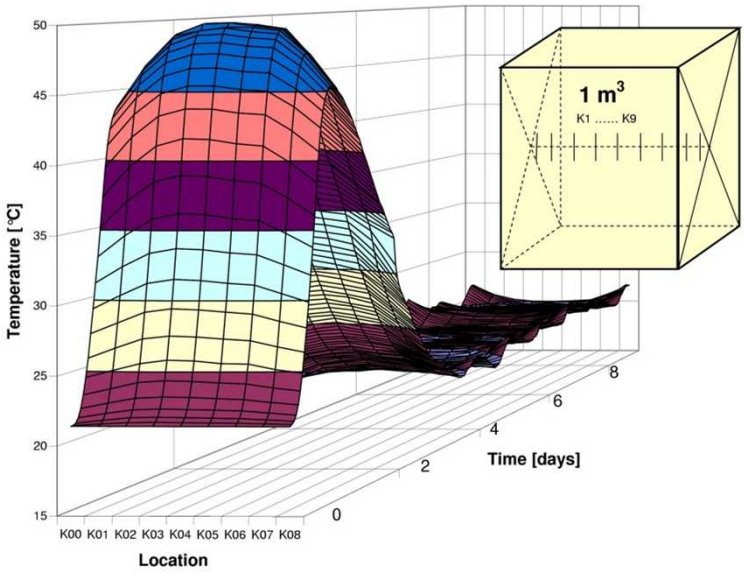
W/c = 0.4, RVE 30 x 30 x 30 μ m

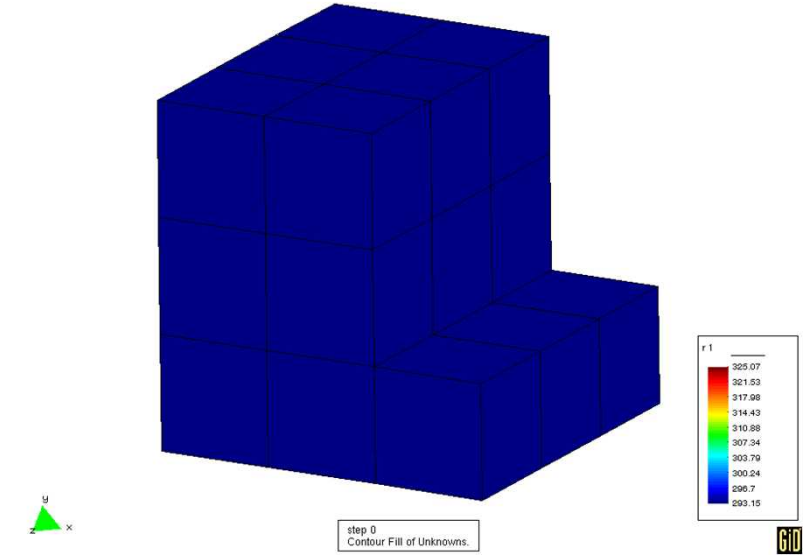


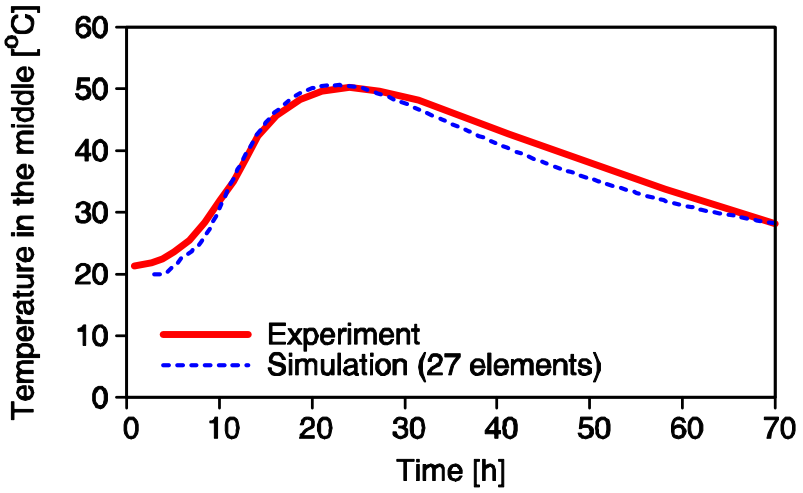
[Vít Šmilauer, Zdeněk Bittnar]

Example cont.: Development of hydration heat

- SCC, CEM I 42.5 R, 350 kg / m3

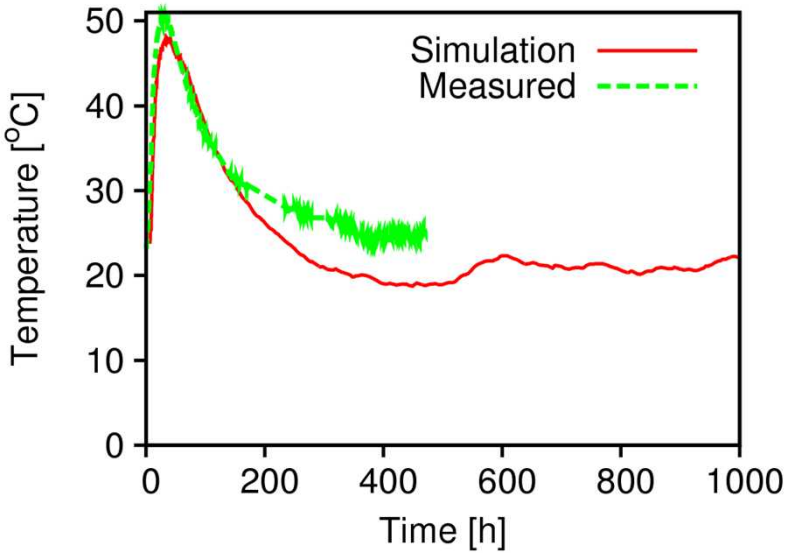




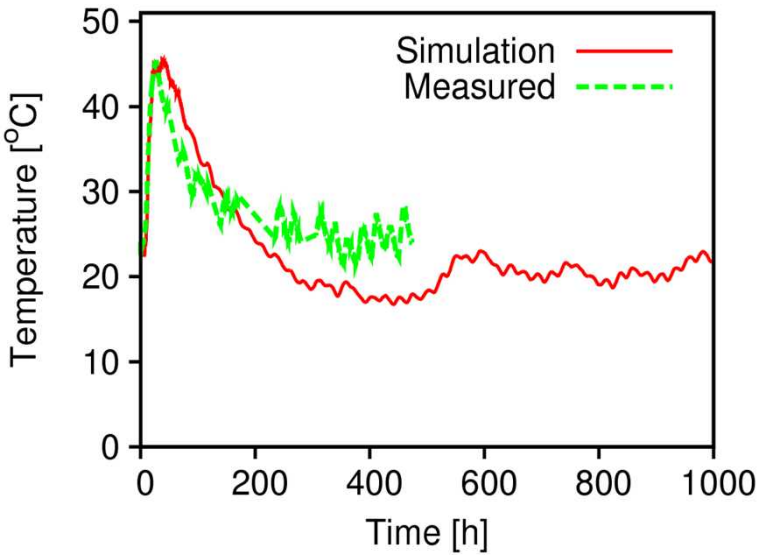


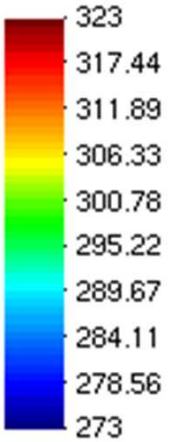
Example cont.:
Development of hydration heat

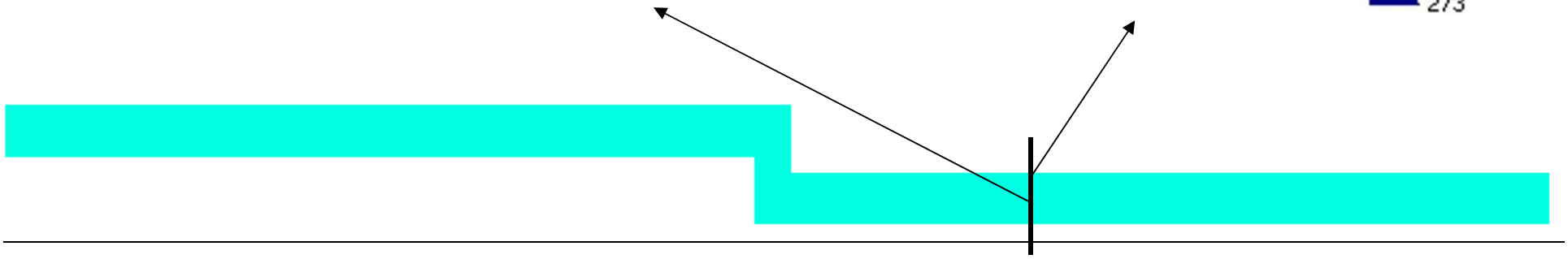
Middle



Top



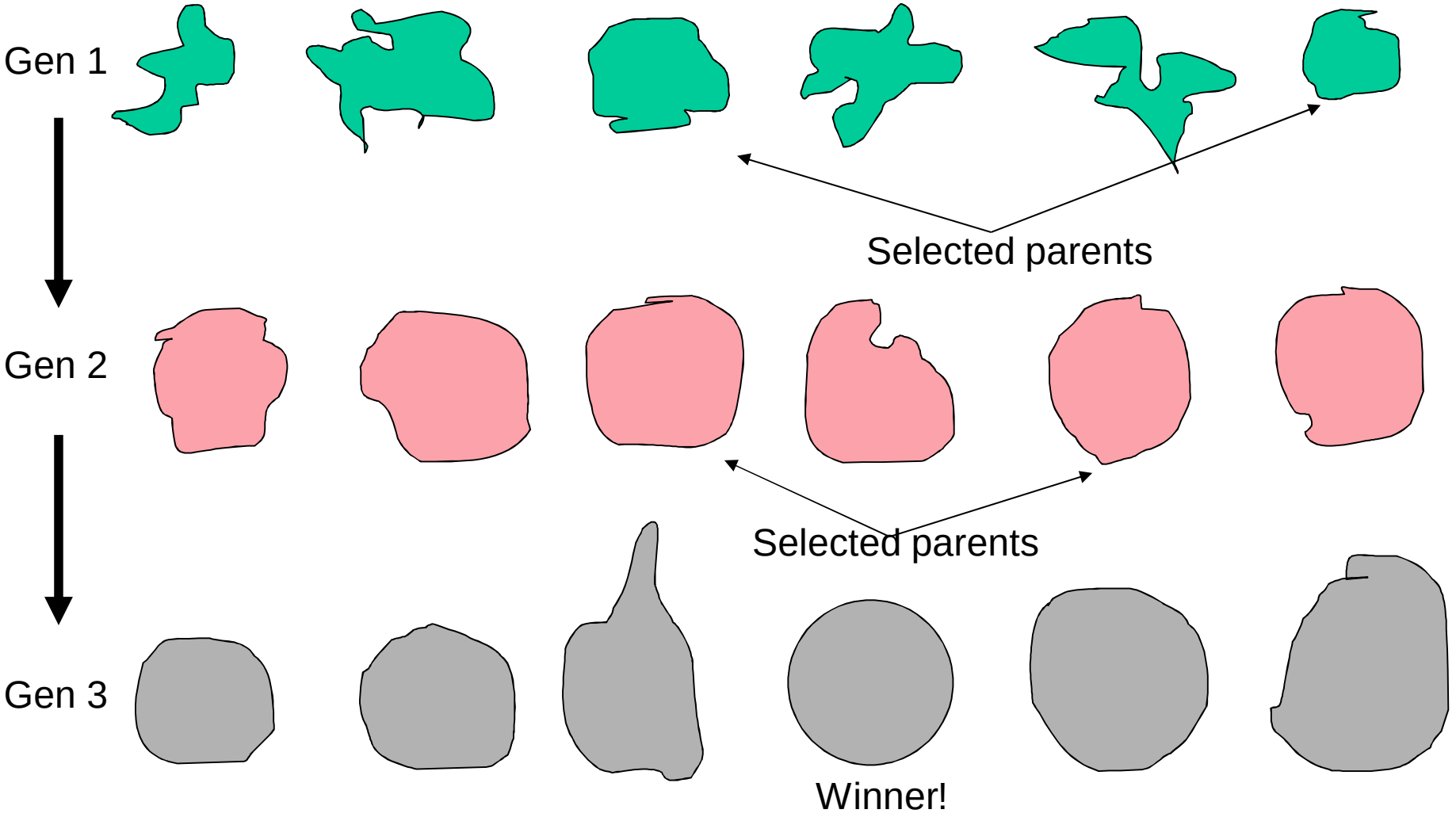




Properties of GA

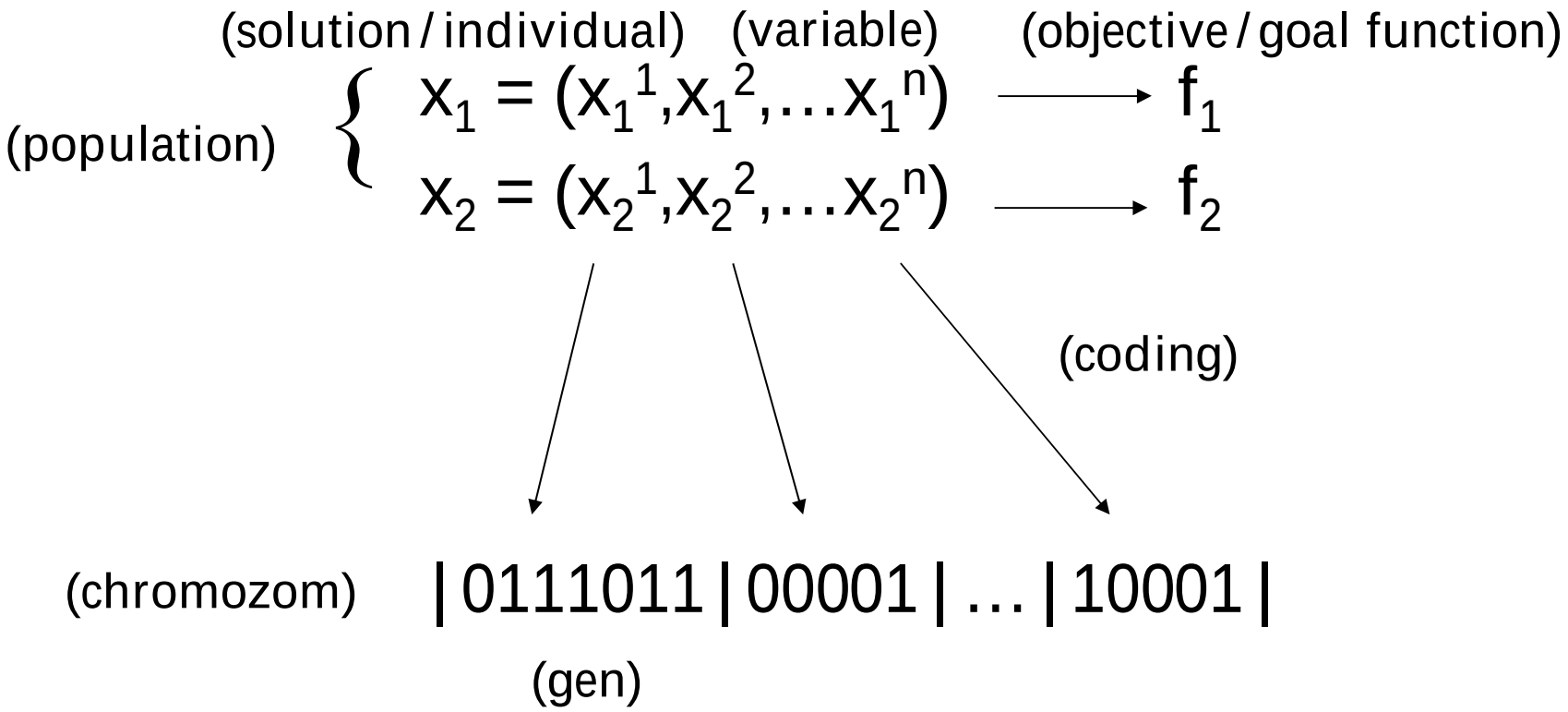
- Based on Darwinian idea „survival of the fittest“
- Stochastic optimization methods – produce random behavior
- Non-gradient methods – even, no need for function continuity
- Proof of convergence based on so-called *Schema theorem*
- Usually the methods of the last resort

Survival of the roundest



Genetic Algorithms

- Notations from biology



Genetic Algorithms

(genotype)	(individual)	(phenotype)	(fitness / force)
B	x	$f(x) = x^2$	F = rank
|0000000000011|	3	9	F = 1
|0000000001001|	9	81	F = 2

E.g.:

DNA → Muscle-man → Fast runner → Winner of competition

Note: Nowadays the *objective function* (phenotype) and *fitness* are usually equal, i.e. $f \equiv$ F.

Motivation

- Theoretical example:

$$\max f(x) = x^2,\ x[\,0\,;\,33\,]$$

Solution	Initial Pop.	x value	Fitness	Prob.	Exp. count	Actual count
1	0 1 1 0 1	13	169	0,14	0,58	1
2	1 1 0 0 0	24	576	0,49	1,97	2
3	0 1 0 0 0	8	64	0,05	0,22	0
4	1 0 0 1 1	19	361	0,31	1,23	1
Sum			1170	1	4	4
Avg.			292,5	0,25	1	1
Max			576	0,49	1,97	2

Solution	Mating pool	Cross pos.	Children	x value	Fitness
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Avg.					438,5
Max					729

Solution	Children	After mut.	x value	Fitness
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	0 1 0 1 1	0 1 0 1 1	27	729
4	1 0 0 0 0	1 0 0 1 0	18	324
Sum				2354
Avg.				588,5
Max				729

Standard genetic algorithm

1 $t = 0$

2 Create P_0, evaluate P_0

3 while (not stopping_criterion) {

4 $t = t+1$

5 Choose M_t from P_{t-1} (Selection)

6 Alter M_t (Genetic operators)

7 Create P_t from M_t and evaluate P_t (New population)

8 }

Where P is population, or set of possible solutions, M is so-called mating pool, t is number of iterations, here called generation.

Binary coding

- Integer numbers are stored in computers in easy accessible binary format, e.g. using binary operators in C programming language.
- If programming language does not provide tools for binary access, transformation is given by:

1. Length of binary string *q*

$$q = \left\lfloor \frac{\ln(max_i - min_i) - \ln(p_i)}{\ln\ 2} \right\rfloor + 1\ ,$$

Binary coding

2. Binary string $\mathrm{B}^i \in \{0;1\}^q$, corresponding to the positive integer value z_i, stands

$$B_j^i = z_i/2^{j-1} \mod 2\,, \quad j = 1, \ldots, q\,.$$

3. With an inversion

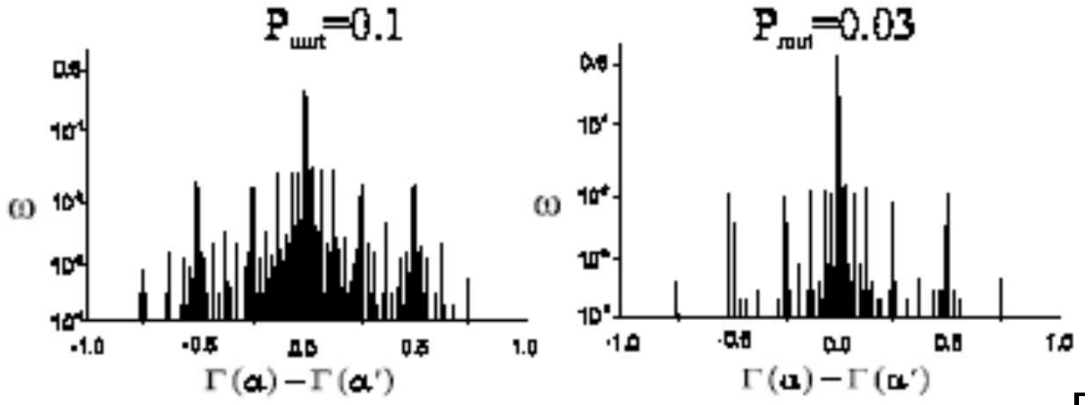
$$z_i = \sum_{j=1}^{q} B_j^i \, 2^{j-1}\,.$$

Hamming's barrier

two numbers:

000000010000000 and
000000001111111

are as integers neighbors, but in binary code they are very distant
(they have big so-called Hamming's distance)

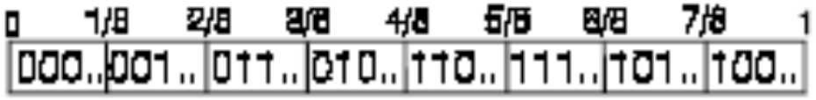


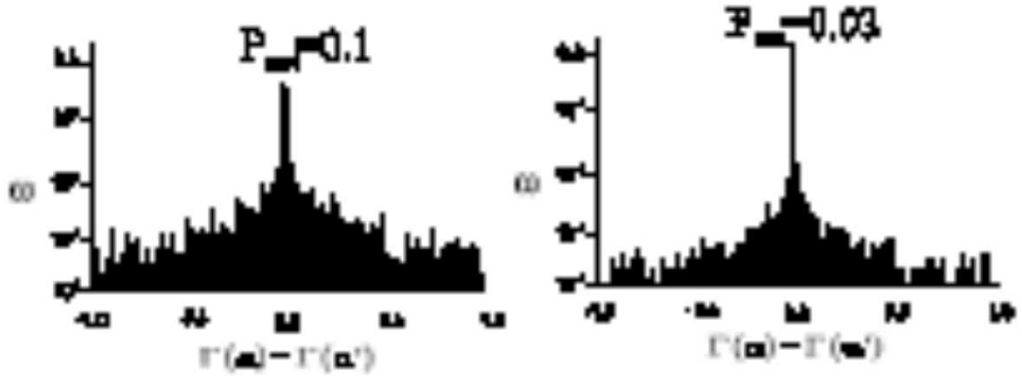
[Kvasnička et al., 2000]

Hamming's barrier: solution

1) Inversion operators

2) Gray coding





Crossover operators

Chromosomes' coding dependent:

- For binary coding, Gray's included:
  - One-point crossover

parents:

111 | 1111111111
000 | 0000000000

children:

111 | 0000000000
000 | 1111111111

  - *k*-point crossover

parents:

111 | 11111 | 11111
000 | 00000 | 00000

children:

111 | 00000 | 11111
000 | 11111 | 00000

Crossover operators

Chromosomes' coding dependent:

- For binary coding, Gray's included:
  - Uniform crossover

parents:

1111111111111
0000000000000

children:

1011000110101
0100111001010

Crossover operators

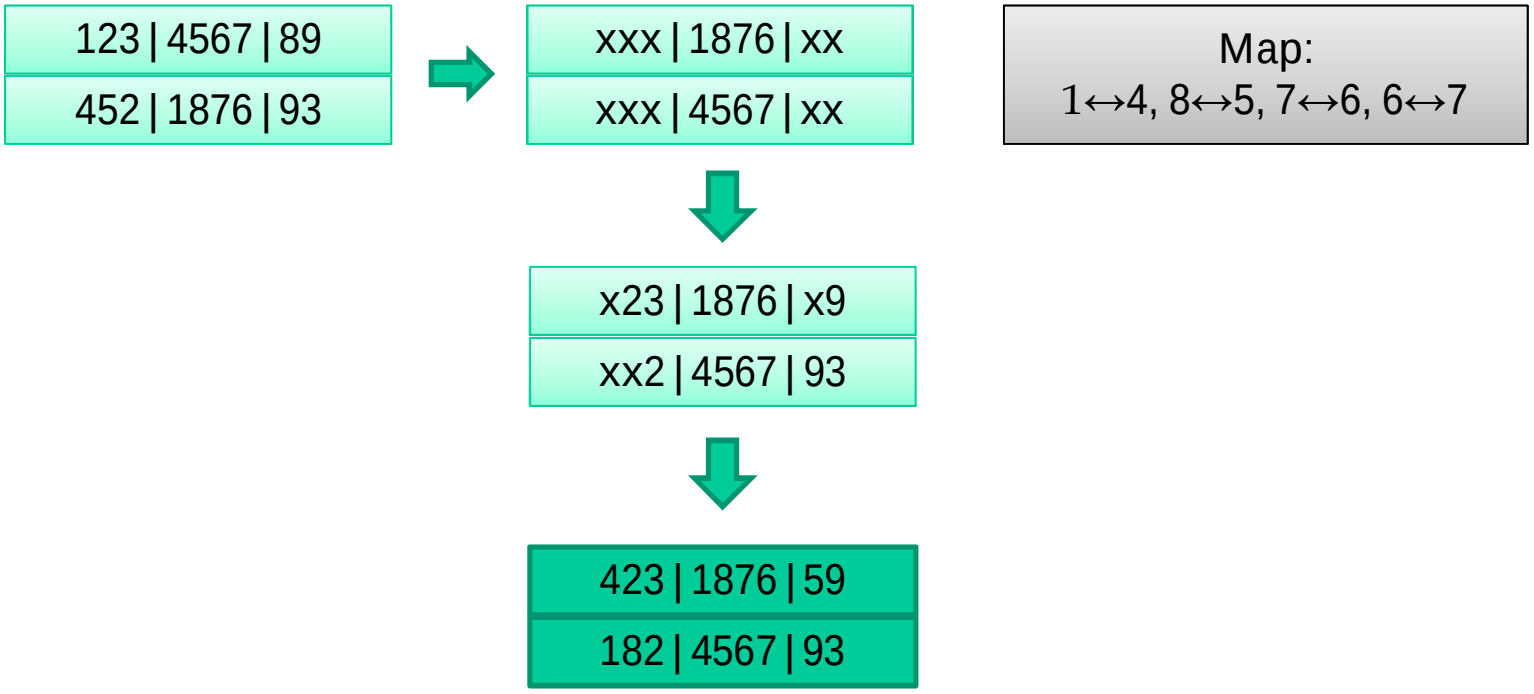
Chromosomes' coding dependent:

- For real coding of chromosomes, e.g.:
  - Arithmetic
    - Child 1: $C_1 = \alpha * P_1 + (1- \alpha) \cdot P_2$
    - Child 2: $C_2 = \alpha * P_2 + (1- \alpha) \cdot P_1$
    - α is a random number between 0 and 1
  - *blend crossover* BLX-α
    - Two parents P_1 and P_2 with given bounds L and U
    - $R_1 = \max (L, P_1 - \alpha(P_2 - P_1))$
    - $R_2 = \min (U, P_2 + \alpha(P_2 - P_1))$
    - Children randomly selected between R_1 and R_2
    - α usually selected at 0,5

Crossover operators

Special operators:

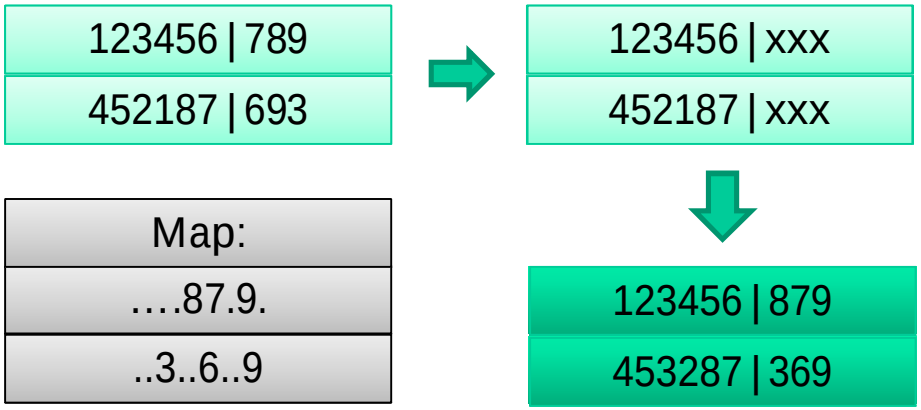
- *Partially matched crossover* (PMX) – for permutation tasks



Crossover operators

Special operators:

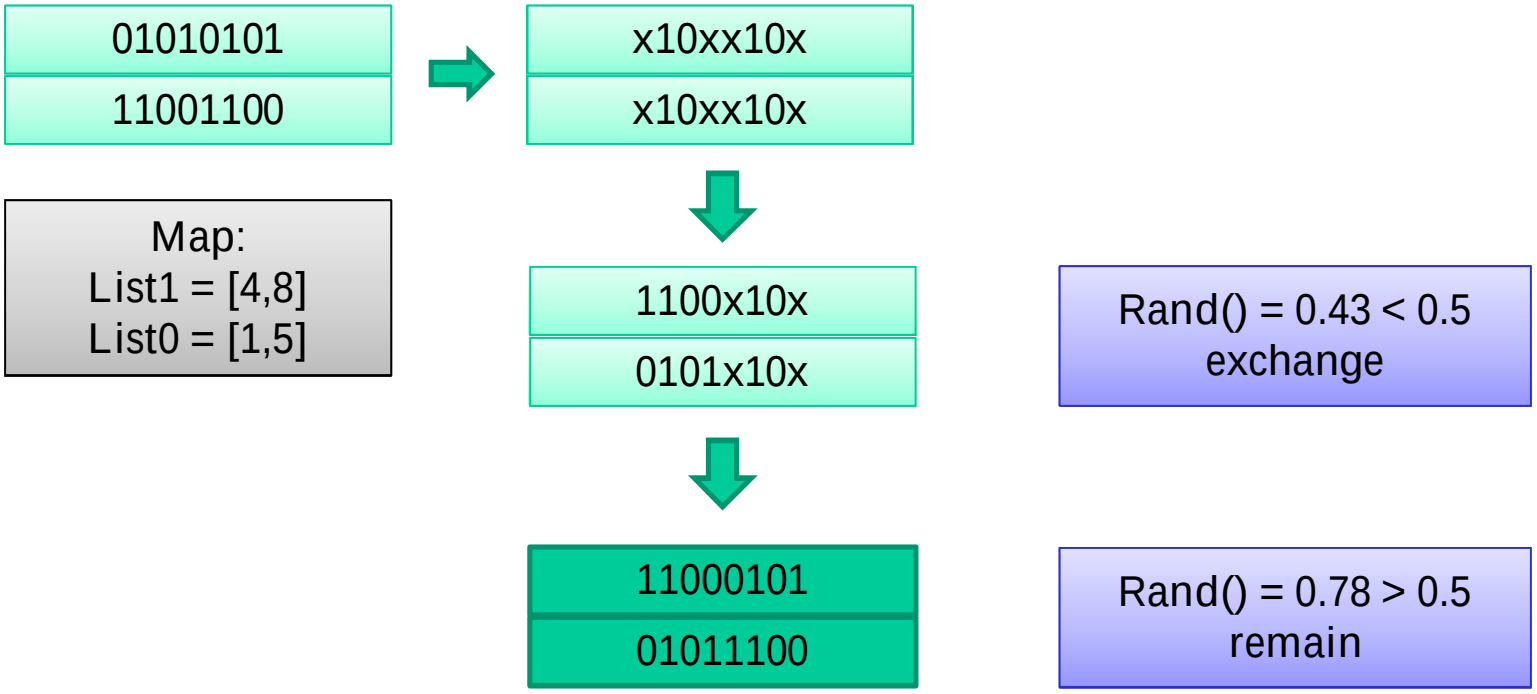
- Uniform order-based crossover – for permutation tasks



Crossover operators

Special operators:

- Count preserving crossover (CPC) – ensures the same amount of 1 and 0



Mutation operators

Chromosomes' coding dependent:

- For binary coding, Gray's included:
  - **One bit mutation**
  - **Uniform mutation**
- For real coding of chromosomes:
  - **Random exchange**
  - **Additive change** – a random number $c \in < -\varepsilon, \varepsilon >$ is added
  - **Gaussian mutation**

Special operators for permutation tasks, count preserving

- **exchange of two numbers**

Selection

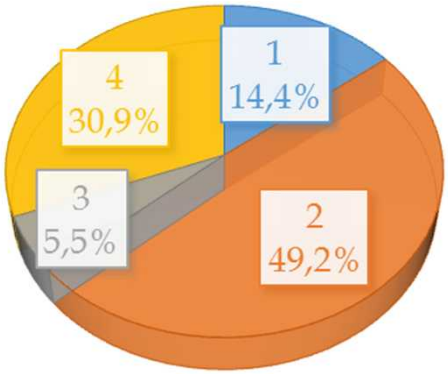
- Selection is needed to create *mating pool*, i.e. so-called *parent selection*, or to create new *generation*, i.e. so-called *survivor selection*

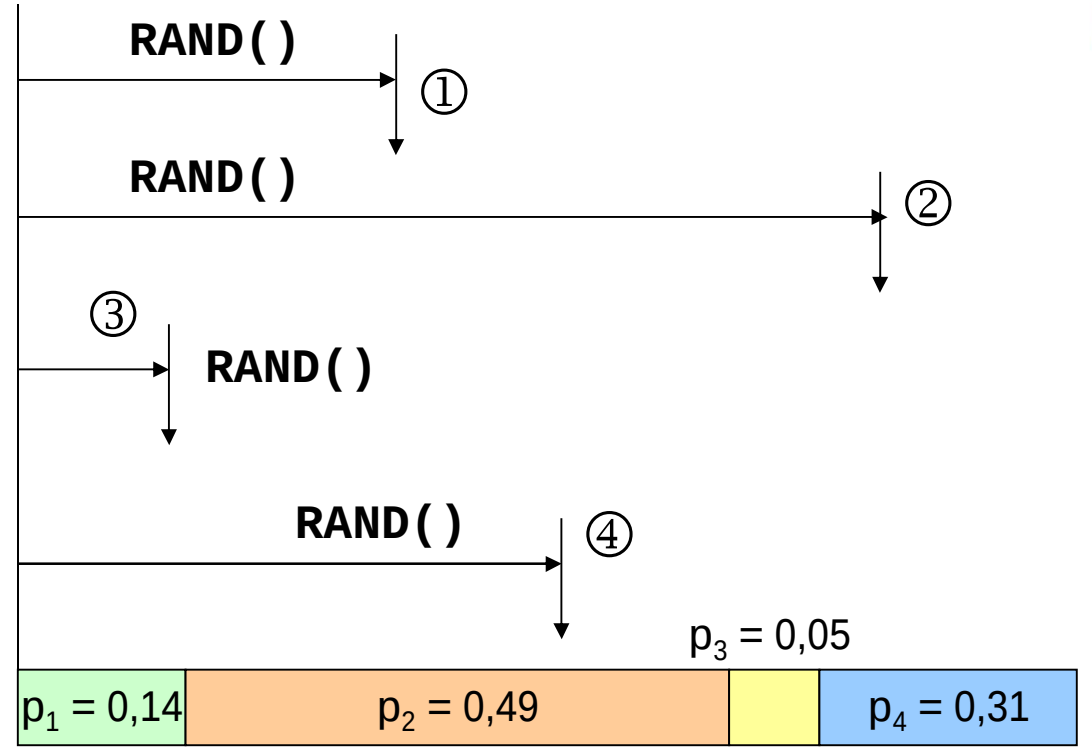
- Dozen of selection methods and tools exist:
  - Roulette wheel
  - SUS – Stochastic Universal Sampling
  - Tournament selection
  - Inverse tournament selection
  - Elitism
  - Ranking a linear scaling

Example

No.	Chromosome	Value x	f(x)	Probability of selection
1	01101	13	169	169 / 1170=0,144
2	11000	24	576	576 / 1170=0,492
3	01000	8	64	64 / 1170=0,055
4	10011	19	361	361 / 1170=0,309
Sum			1170	1

Roulette wheel

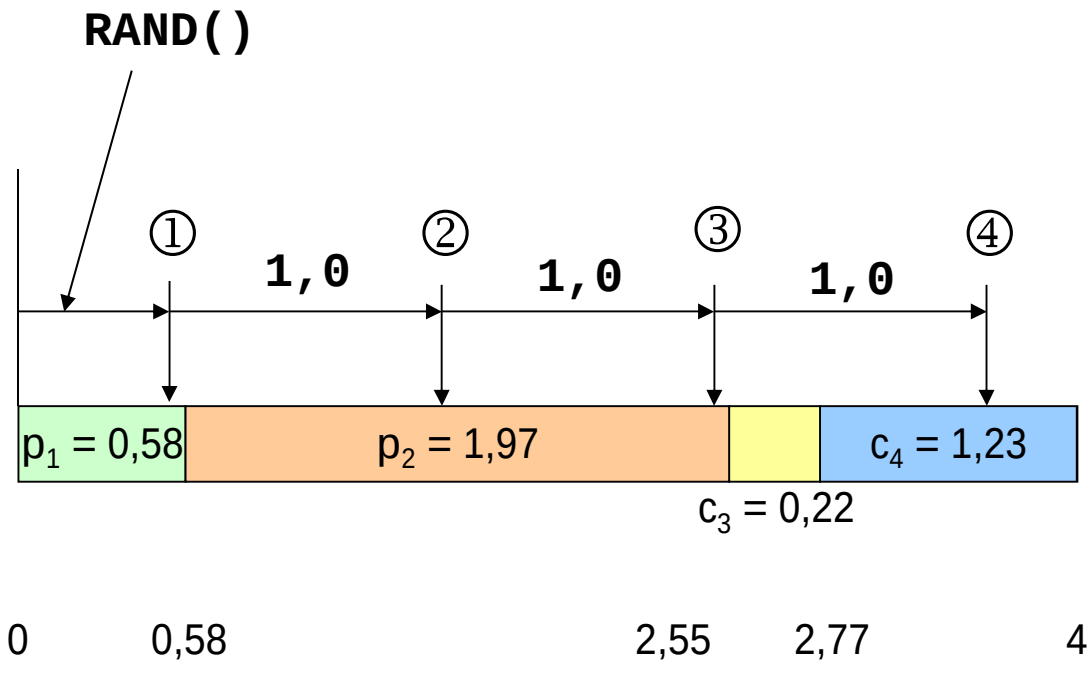


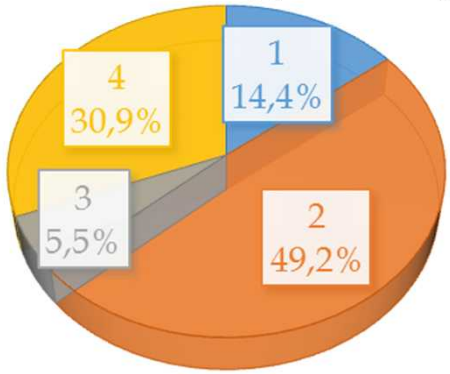


Roulette wheel

- Classical tool
- Does not ensure selection of the best solution especially in the case of large populations
- Computationally expensive method, *n* x **RND( )**

SUS – Stochastic Universal Sampling





Expected count

Cumulative expected count

SUS – Stochastic Universal Sampling

- Does not ensure selection of the best solution especially in the case of large populations, but
- Ensures integer expected counts, e.g.
  solution with c = 2,0001 will be always selected 2x
- Computationally lest demanding method,
  only 1 x **`RND()!`**

Tournament selection

- Randomly selects from population *k* „contestants“ (most often *k* = 2) and select the best of them (tournament) and this procedure is done *n*-times

- Does not ensure selection of the best solution because there is a probability that the best solution would not be selected to the tournament

- Computationally expensive method:

  k x *n* x **RND()**

Inverse tournament selection

- Selection of *p* solutions, where $p < n$
- Randomly selects from population *k* „contestants“ (most often $k = 2$) and the worst one is discarded (inverse tournament) and this procedure is done *n*-times
- This method ensures selection of the best solution!
- The most „democratic“ method – there exists a probability of survival of the worst solution because of not being selected to the tournament
- Computationally expensive method :

  $k \times n \times$ **RND( )** [Hrstka and Kučerová, 2004]

Elitism

- Has two senses:

1) Selection or copy of the best solution to the next generation (or mating pool). Used to correct deficiency of selection methods or to control speed of convergence.

2) Property or procedure which preserves actual "better" solutions. For instance, tournament with $k = 3$ is more *elitist* than tournament with $k = 2$.

Ranking and linear scaling

- Methods for control of fitness, supporting or blocking *elitism*
- Ranking computes fitness not based on objective function value, but on the ranks of solutions within population. E.g. the best solution will have 3 copies, second best 2 copies and reminders to selection one copy. The worst solutions will have 0.

Ranking and linear scaling

- Linear scaling computes fitness to be linearly distributed, e.g. to preserving the average value of objective functions and also ensure the maximal fitness within 1,2-2,0 multiple of average (selected by user)

Proof of convergence: „Schema theorem“

two solutions:

0101001010010101010111010101010101 and
0101101010010111000111011101010111

are covered by this schema:

0101#010100101#10#011101#10101#1

Expected count of appearance of this scheme in time t+1 is given by

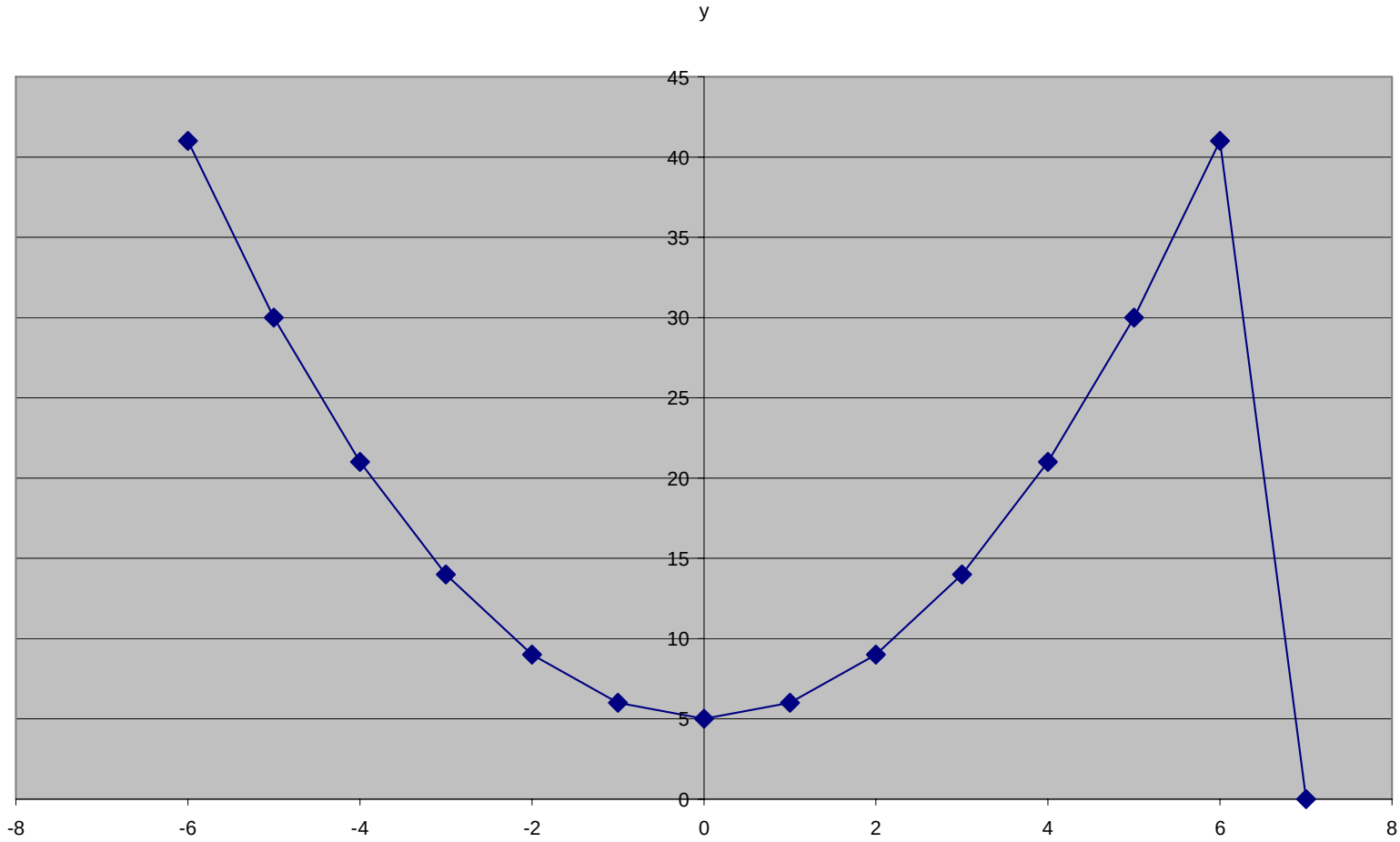
$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - P_c \frac{\delta(H)}{L-1} - P_m o(H) \right]$$

see e.g. [Goldberg, 1989]

Premature Convergence

- If the whole population is covered by one genotype, the population has *converged*
- Convergence is *premature*, if the global optimum has not been found
- Solution:
  - Higher mutation probabilities
  - Preserving diversity within population
  - Multi-modal solutions

Deceptive functions



Look also for “Robust design optimization”

References

[1] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. MIT Press.

[2] Goldberg, D. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

[3] Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. AI Series. Springer-Verlag, New York.

[4] Wolfram, S. (2002). Cellular Automata and Complexity. Perseus Publishing.

[5] A. E. Eiben, J. E. Smith (2003). Introduction to Evolutionary Computing. Springer.

References

[6] Dreo J, Petrowsky A, Siarry P and Taillard E. (2006). Metaheuristics for hard optimization. Methods and case studies. Berlin Heidelberg: Springer

[7] V. Kvasnička, J. Pospíchal, P Tiňo (2000). Evolučné algoritmy. STU Bratislava.

[8] Hrstka, O. and Kučerová, A. (2004). Improvements of real coded genetic algorithms based on differential operators preventing the premature convergence. Advances in Engineering Software, 35(3–4):237–246.

A humble plea. Please feel free to e-mail any suggestions, errors and typos to **matej.leps@fsv.cvut.cz**.

Date of the last version: 12.12.2019
Version: 001