Evolutionary algorithms

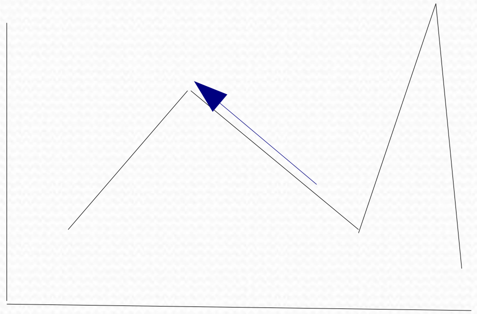
- Simple genetic algorithms
- Evolutionary Strategies
- Genetic Programming

Partially based on slides by Thomas Bäck

Heuristic Search

- SAT solvers, CP solvers, ILP solvers:
  - find exact solutions to constraint optimization problems
  - can be time consuming
- Heuristic solvers:
  - employ “heuristics”: guidelines for finding good solutions quickly
  - don't find exact solutions
  - can be much faster

Hill-Climbing



- Hill-climbing is arguably the simplest heuristic algorithm

1. S = arbitrary candidate solution
2. S' = solutions in the neighborhood of S
3. **if** best solution in S' is not better than S **then** stop
4. let S be the best solution in S'
5. go to 2.

Other Well-known Heuristic Search Strategies

- Simulated annealing
- Tabu search
- Evolutionary algorithms
  - genetic algorithms
  - genetic programming
  - evolutionary strategies
- Artificial ants
- Particle swarms

Genetic algorithms

- Randomized search algorithms based on mechanics of natural selection and genetics

- Principle of natural selection through `survival of the fittest' with randomized search

Advantages of GAs

- Evolution and natural selection has proven to be a robust method
- A "black box" approach that can easily be applied to many optimization problems
- GAs can be easily parallelized and run on multiple machines

Some definitions

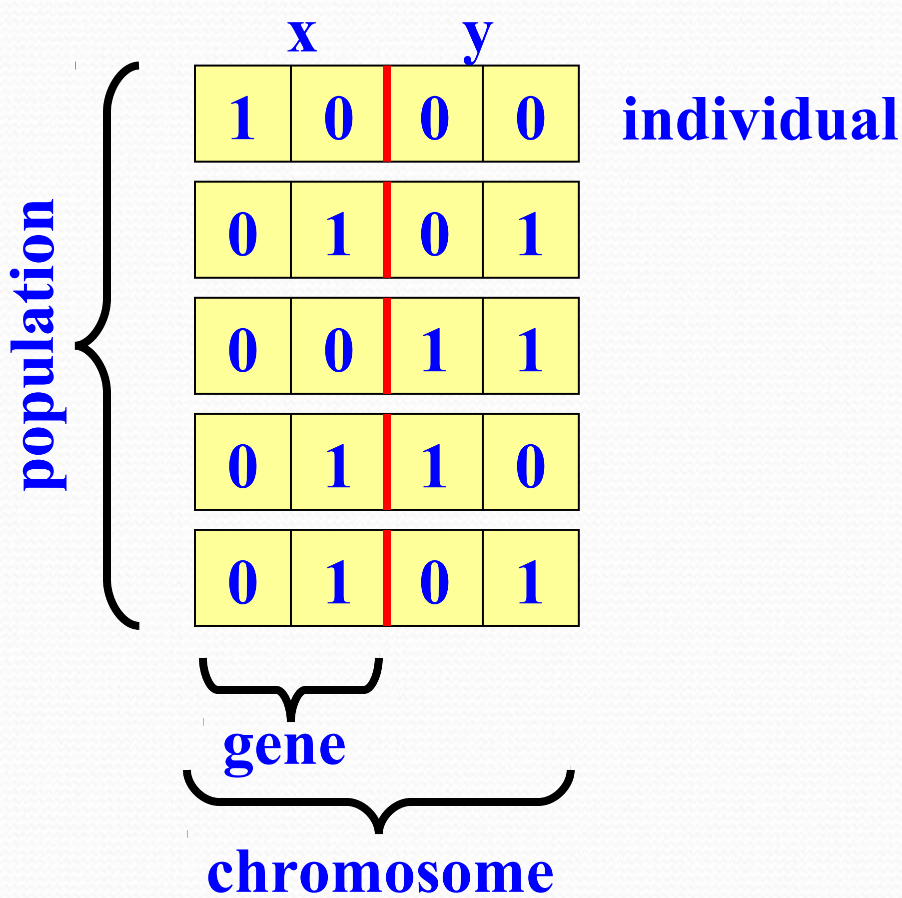
- **Population**: a collection of solutions for the studied (optimization) problem
- **Individual**: a single solution in a GA
- **Chromosome (genotype)**: representation for a single solution
- **Gene**: part of a chromosome, usually representing a variable as part of the solution

Some definitions

- **Encoding**: conversion of a solution to its equivalent representation (chromosome)
- **Decoding**: conversion of a chromosome (**genotype**) to its equivalent solution (phenotype)
- **Fitness**: scalar value denoting the suitability of a solution

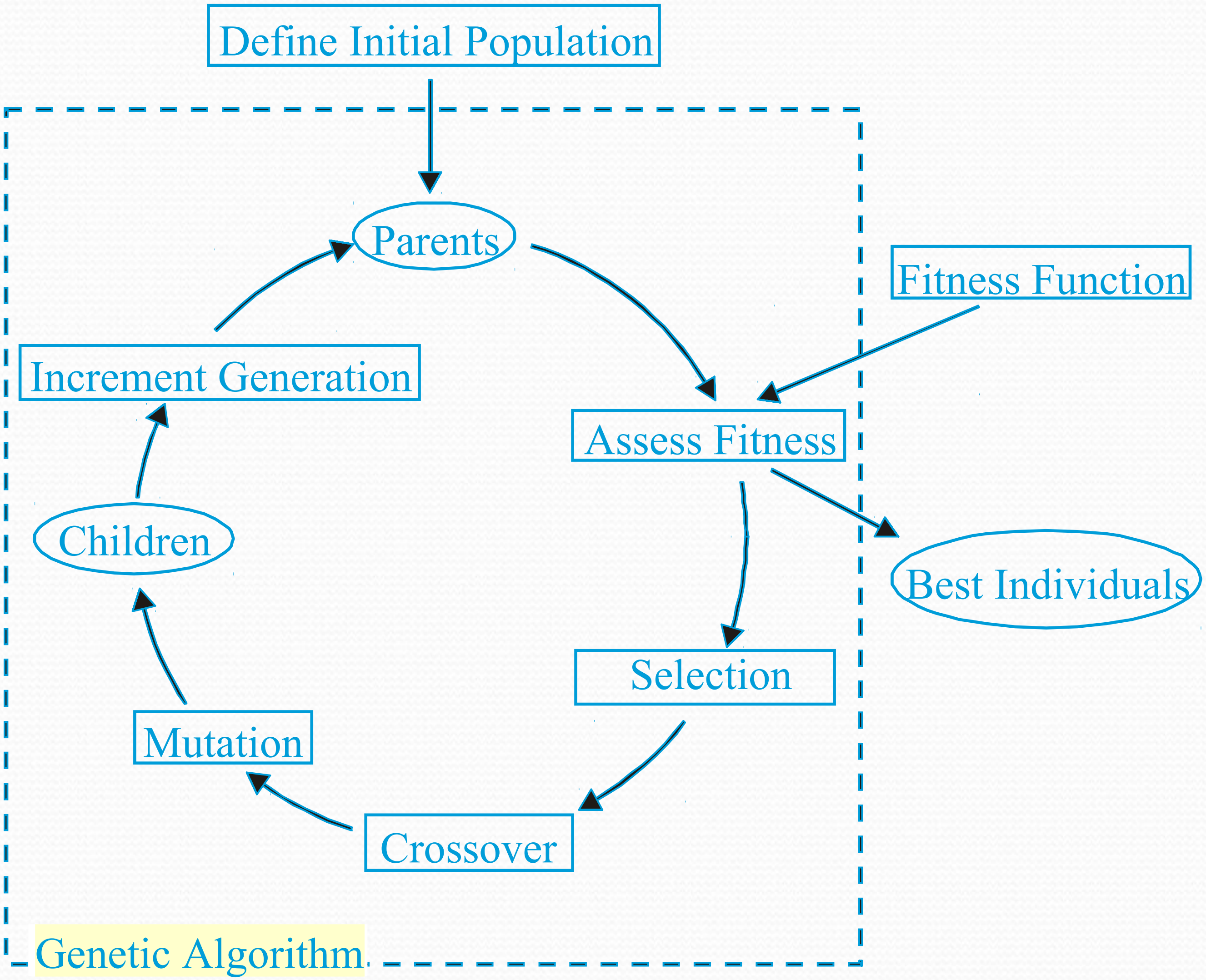
GA terminology

Generation t



solution	fitness
(2,0)	4
(1,1)	2
(0,3)	3
(1,2)	3
(1,1)	2

Genetic algorithm



Pseudo code

- Initialize population P:
  - E.g. generate random p solutions
- Evaluate solutions in P:
  - determine for all $h \in P$, Fitness(h)
- **While** terminate is FALSE
  - Generate new generation P using genetic operators
  - Evaluate solutions in P
- **Return** solution $h \in P$ with the highest Fitness

Termination criteria

- Number of generations
  (restart GA if best solution is not satisfactory)
- Fitness of best individual
- Average fitness of population
- Difference of best fitness (across generations)
- Difference of average fitness (across generations)

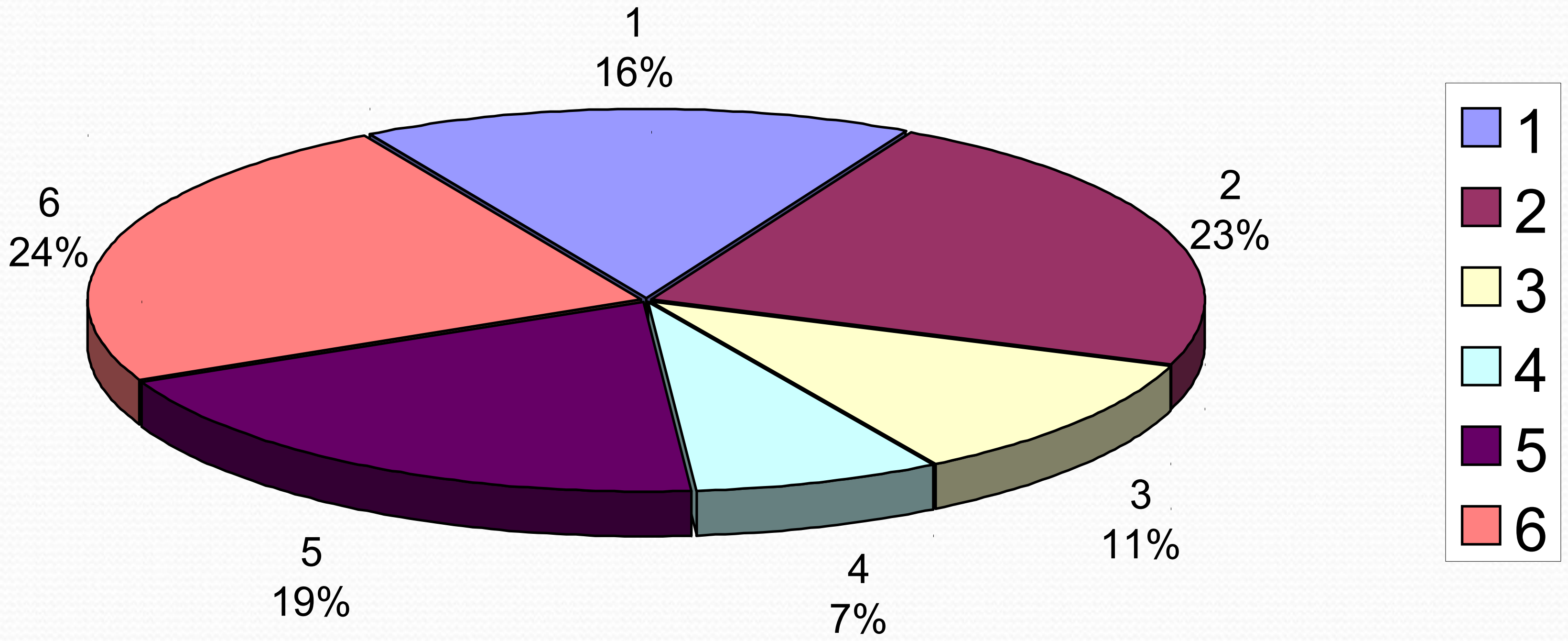
Reproduction

Three steps:

- Selection
- Crossover
- Mutation

In GAs, the population size is often kept constant. User is free to choose which methods to use for all three steps.

Roulette-wheel selection



Roulette-wheel selection

individuals fitness

$$\begin{array}{l} p = 0.16 \\ p = 0.23 \\ p = 0.11 \\ p = 0.07 \\ p = 0.19 \\ p = 0.24 \end{array} \begin{bmatrix} 01100 \\ 10001 \\ 11010 \\ 00111 \\ 11000 \\ 10110 \end{bmatrix} \begin{bmatrix} 34 \\ 48 \\ 23 \\ 15 \\ 41 \\ 50 \end{bmatrix} \xrightarrow{\text{selection}} \begin{bmatrix} 01100 \\ 10001 \\ 10001 \\ 11000 \\ 10110 \\ 10110 \end{bmatrix}$$

Sum = 211

Cumulative probability: 0.16, 0.39, 0.50, 0.57, 0.76, 1.00

Tournament selection

- Select pairs randomly
- Fitter individual wins
  - deterministic
  - probabilistic
    - constant probability of winning
    - probability of winning depends on fitness

It is also possible to combine tournament selection with roulette-wheel

Crossover

- Exchange parts of chromosome with a crossover probability (p_c is usually about 0.8)
- Select crossover points randomly

One-point crossover:

0	1	0	1	1	1	1	1	0	1	1

0	1	1	1	0	1	0	1	1	1	0

crossover point

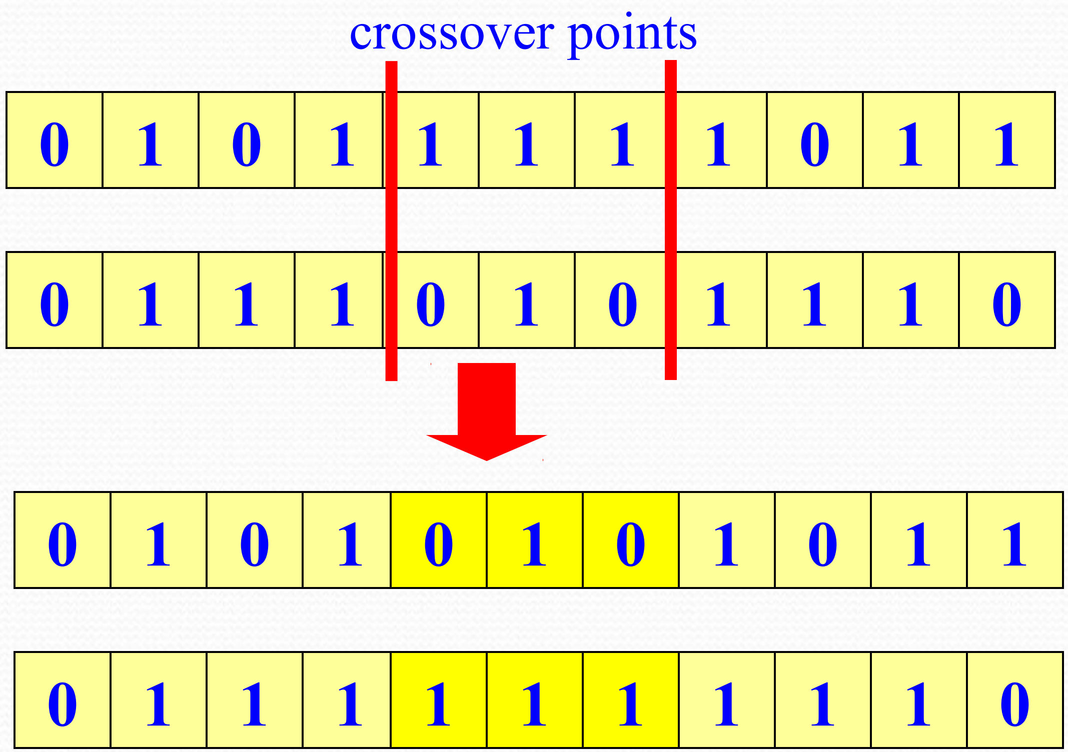
0	1	0	1	1	1	1	1	1	1	0

0	1	1	1	0	1	0	1	0	1	1

N-point crossover

- Select N points for exchanging parts
- Exchange multiple parts

Two-point crossover:



Uniform crossover

- Exchange bits using a randomly generated mask

0	1	0	1	0	0	1	0	0	1	1	mask

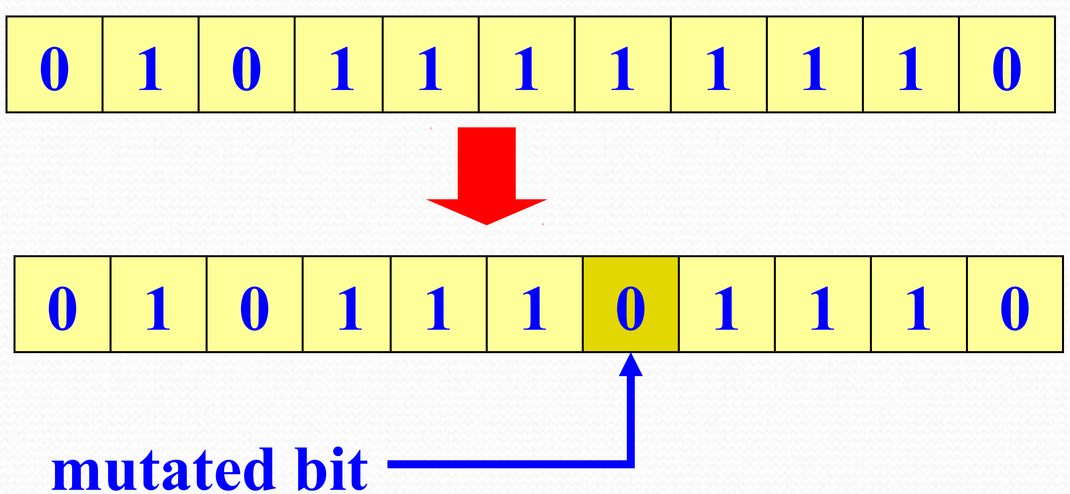
0	1	0	1	1	1	1	1	0	1	1
0	1	1	1	0	1	0	1	1	1	0

0	1	0	1	1	1	0	1	0	1	0
0	1	1	1	0	1	1	1	1	1	1

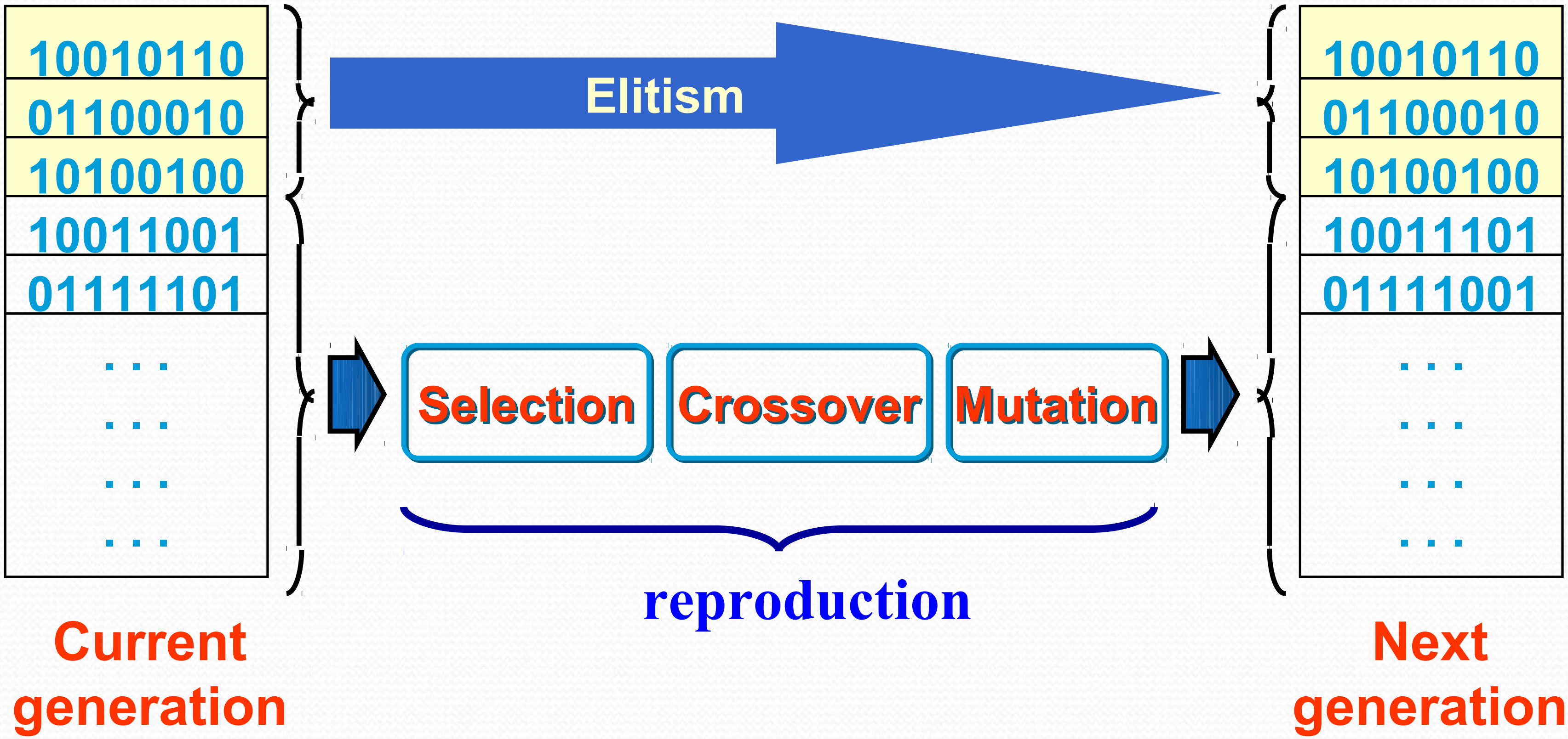
Mutation

- Crossover is used to search the solution space
- Mutation is needed to escape from local optima
- Introduces genetic diversity
- Mutation is rare (p_m is about 0.005)

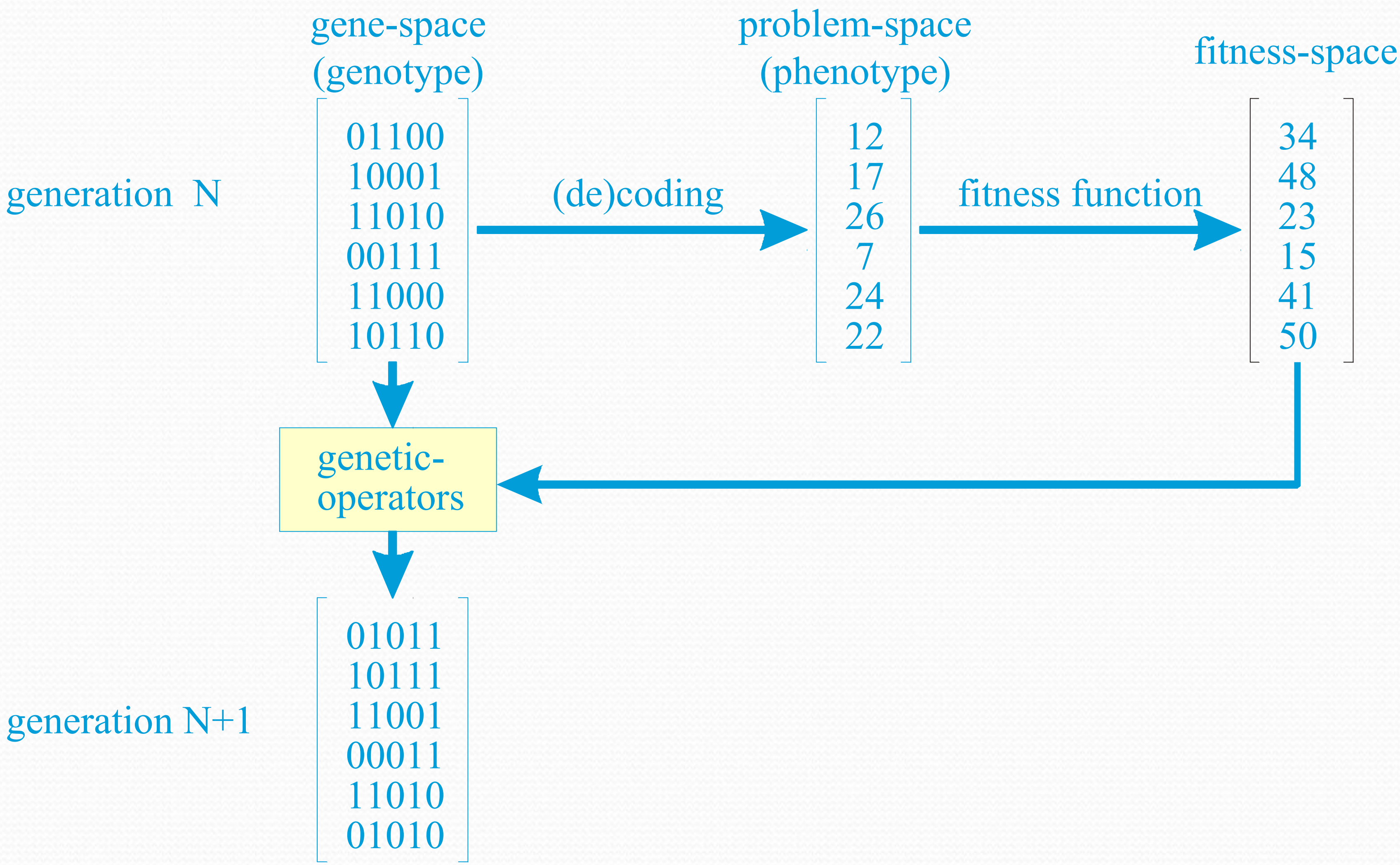
Uniform mutation:



GA iteration



Spaces in GA iteration



Encoding and decoding

- Common coding methods for numbers
  - simple binary coding
  - Gray coding (binary)
  - real valued coding (*evolutionary strategies*)
  - tree structures (*genetic programming*)