METAHEURISTICS

Genetic Algorithms

Ilya Bychkov
ibychkov@hse.ru

October 15th , 2018
NRU HSE NN

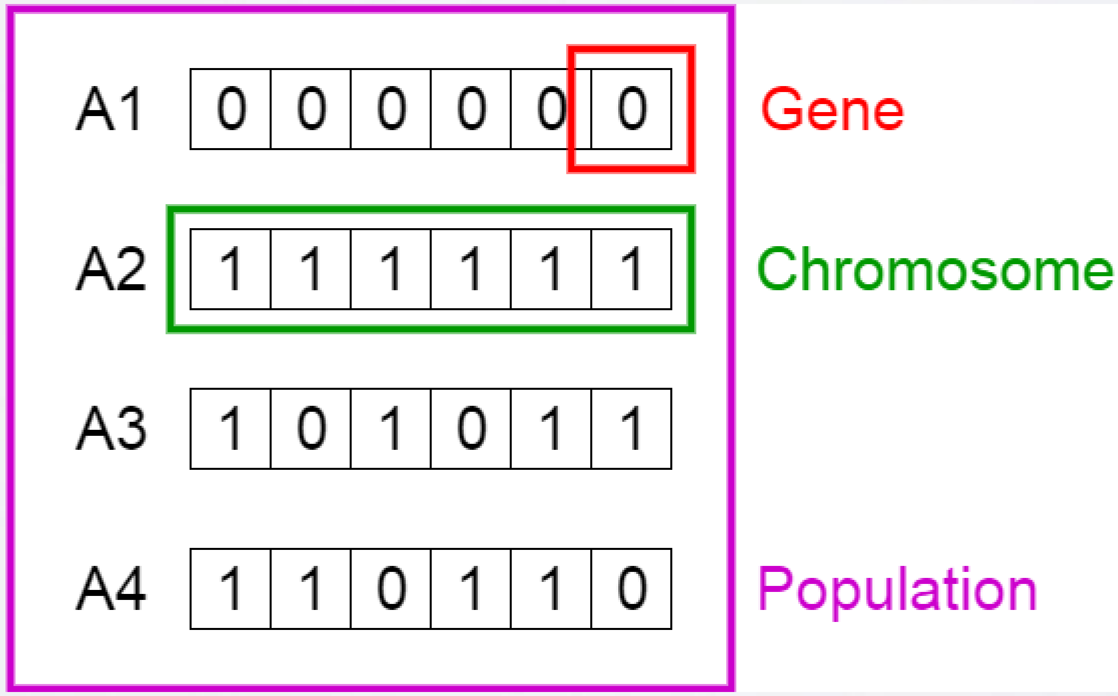
Genetic Algorithms

- **Population-based algorithms.** Algorithms that maintain an entire set of candidate solutions, each solution corresponding to a unique point in the search space of the problem

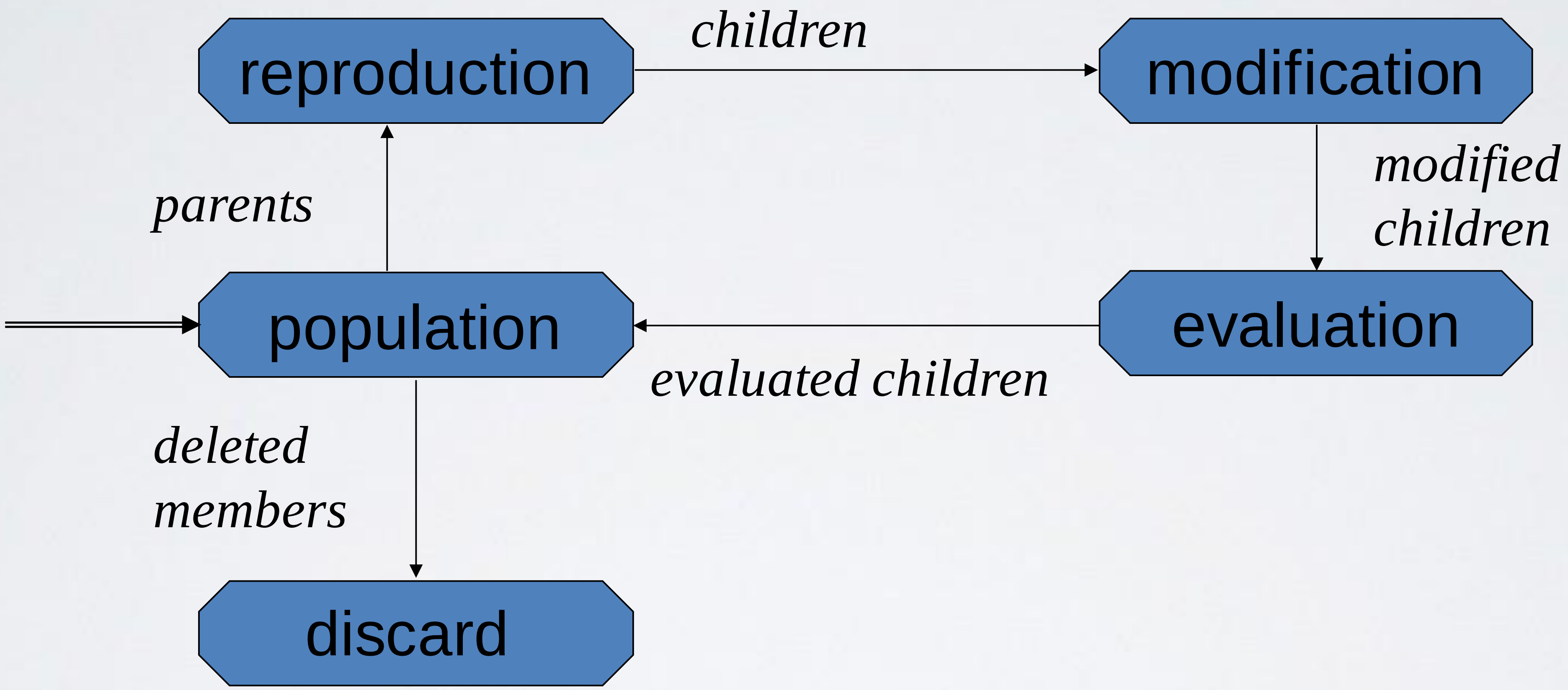
- **Evolutionary algorithms.** Use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions.

GA - Representation

- Solutions representation is important
- **Chromosome** is a single solution to an optimization problem
- **Gene** is an element of a solution
- **Population** is a set of chromosomes which we will operate during the algorithm



GA Reproduction



GA - Representation

- Binary strings (e.g. Knapsack problems)

0	0	1	0	1	1	1	0	0	1

- Integer vectors (e.g. clustering problems)

1	2	3	4	3	2	4	1	2	1

- Permutations (e.g. TSP, QAP)

1	5	9	8	7	4	2	3	6	0

GA – Fitness function

- The fitness function simply defined is a function which takes a candidate solution to the problem as input and produces as output how “fit” our how “good” the solution is with respect to the problem in consideration.

- The fitness function should be sufficiently fast to compute (calculation of fitness value is done repeatedly in a GA)

- In most cases the fitness function and the objective function are the same as the objective is to either maximize or minimize the given objective function.

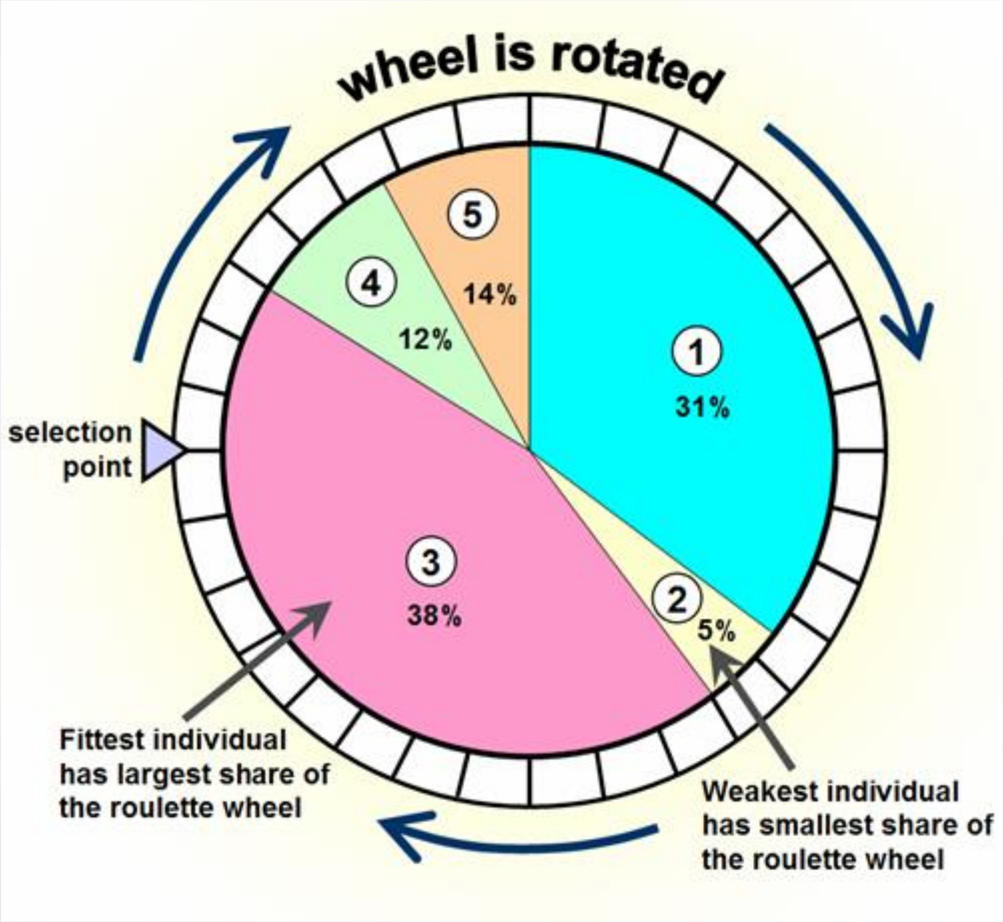
GA – Selection

- Selection is the stage of a genetic algorithm in which individual genomes are chosen from a population for later breeding (using the crossover operator)

- Key idea: better individuals get higher chance

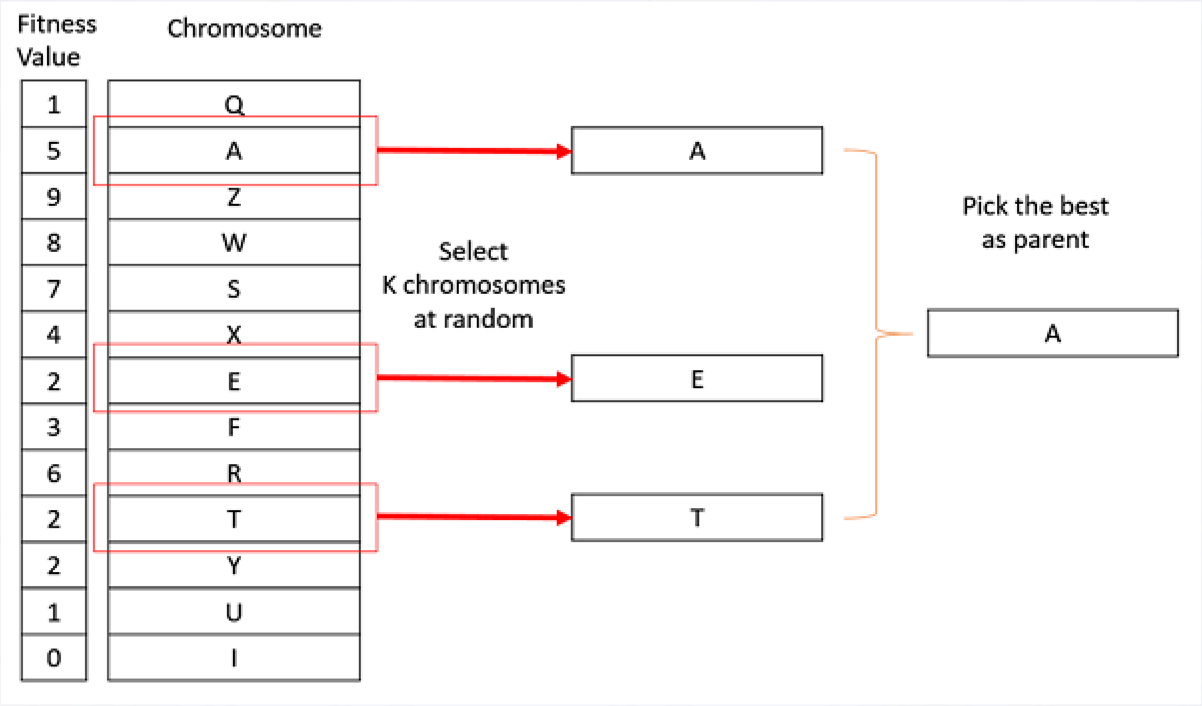
GA – Selection

- **Roulette-wheel selection**
  - Chances to be chosen are proportional to fitness
  - Assign to each individual (chromosome) a part of roulette wheel
  - Spin the wheel *n* times to select *n* individuals



GA – Selection

- **Tournament selection**
  - Select K individuals from the population at random and select the best out of these to become a parent (K-way tournament)
  - We can control the probability of selecting the fittest by changing the size of tournament (1-way is a random search)



GA – Genetic Operators

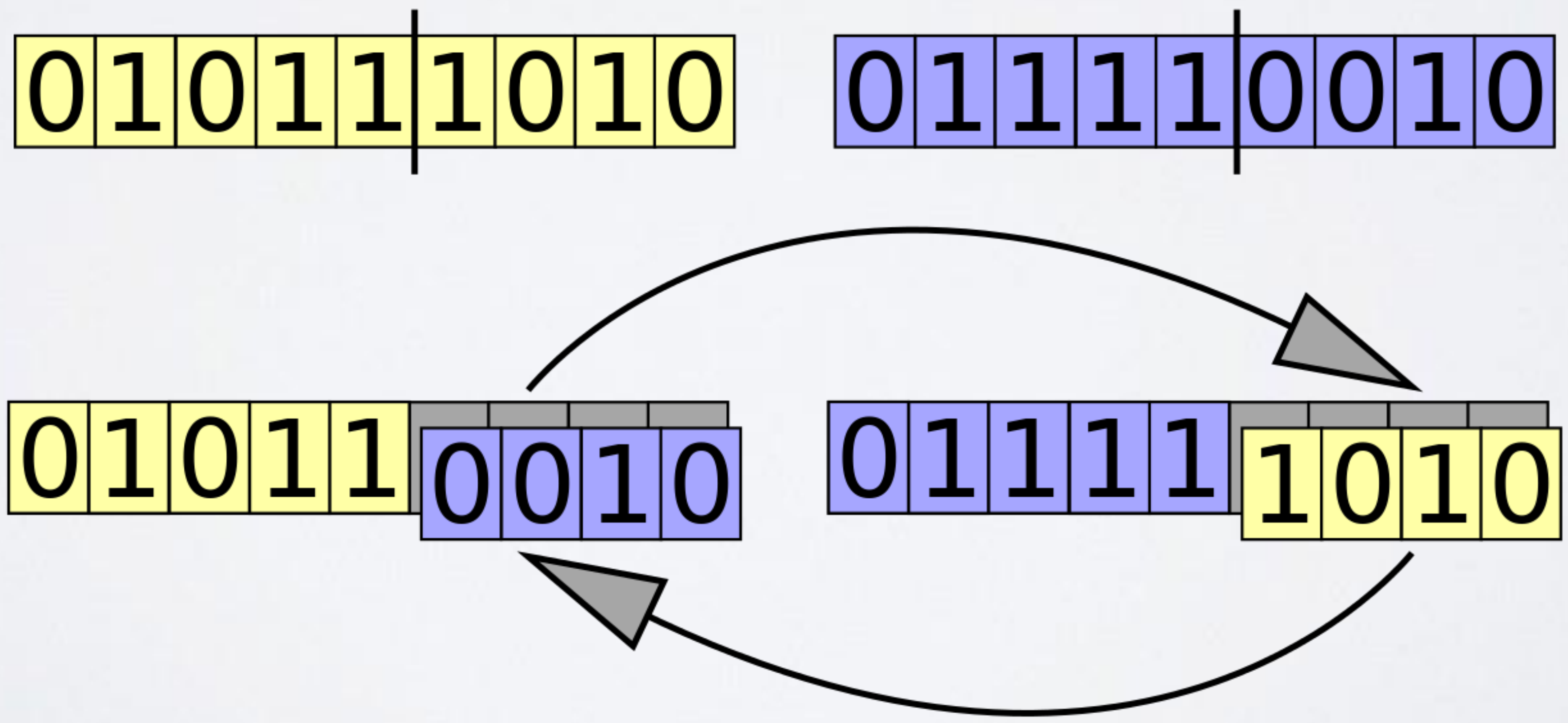
- Manipulates chromosomes/solutions
- Mutation: Unary operator
- Crossover: Binary operator

GA – Crossover operator

- After selection, individuals from the mating pool are recombined (or crossed over) to create new, hopefully better offsprings
- The crossover operator is analogous to reproduction and biological crossover
- In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents.

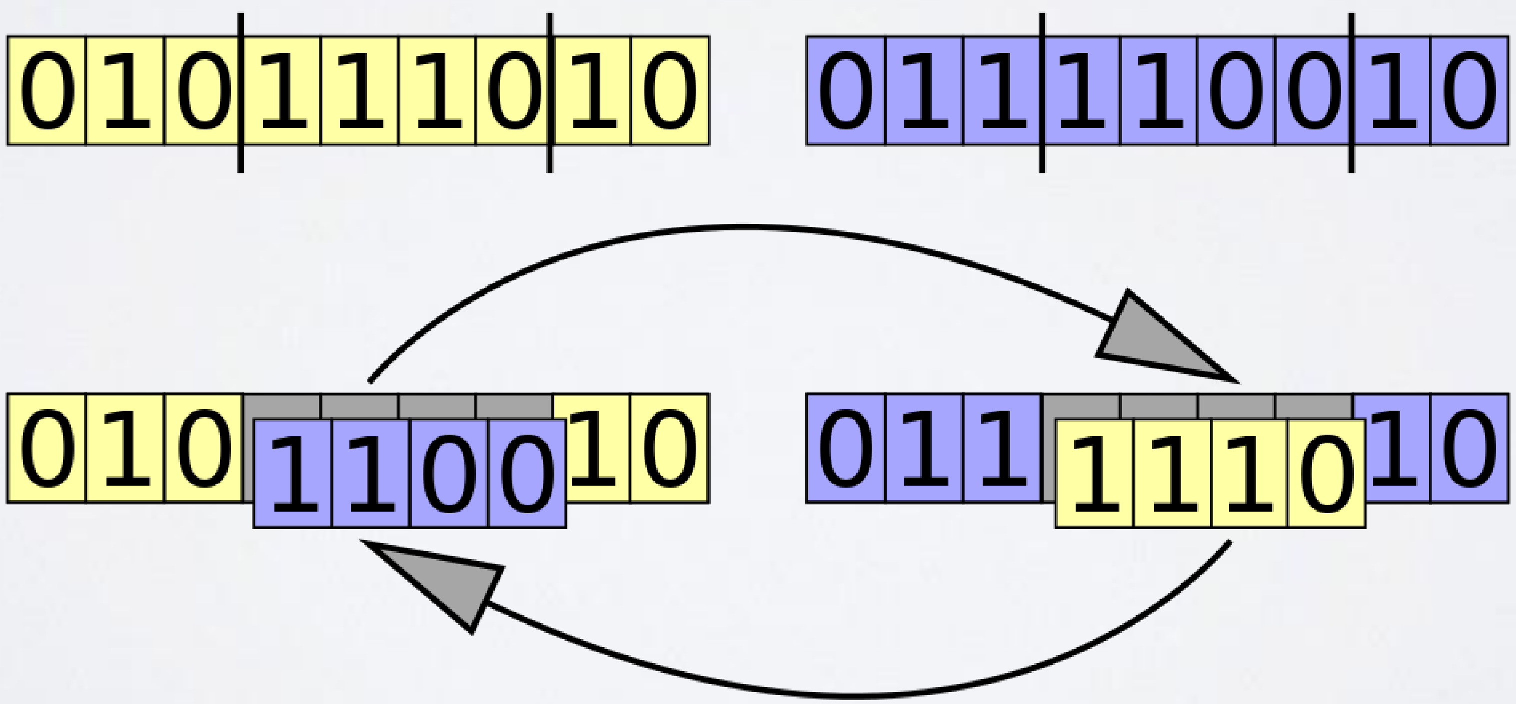
GA – Crossover operator

- One-point crossover
  - In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



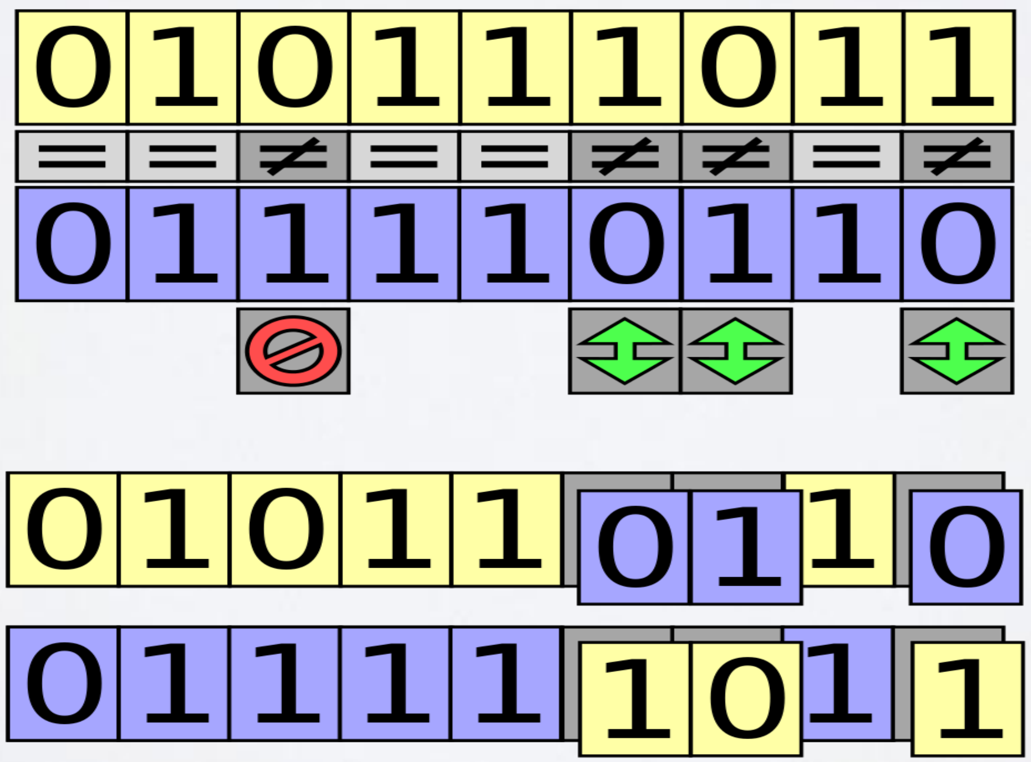
GA – Crossover operator

- Two-point crossover
  - In this two-point crossover, a two random crossover points are selected. Everything between the two points is swapped between the parent chromosomes, rendering two child organisms



GA – Crossover operator

- Uniform crossover
  - In a uniform crossover, we don’t divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it’ll be included in the off-spring.



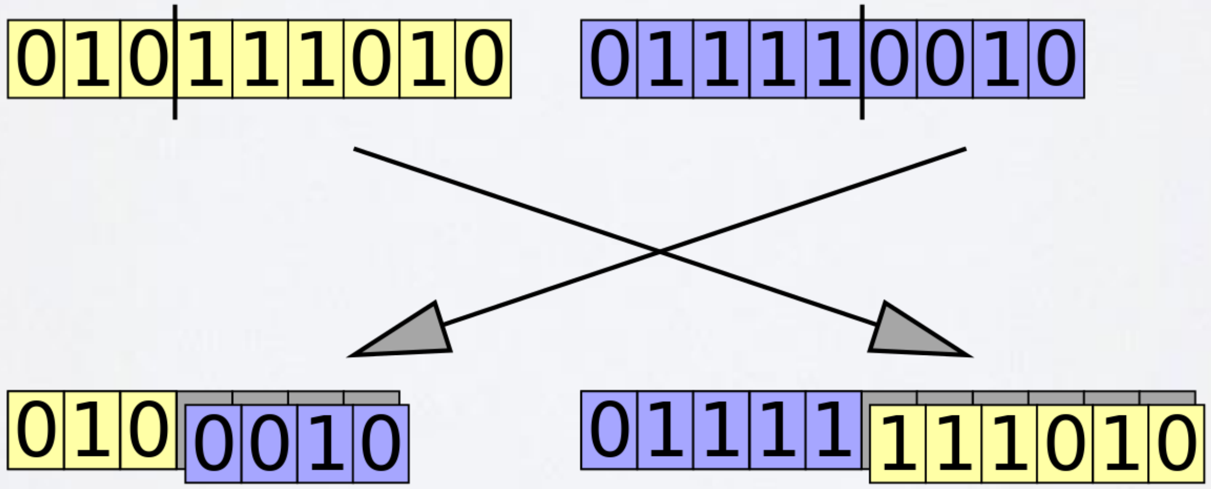
GA – Crossover operator

- Three parents crossover
    - In this technique, the child is derived from three randomly chosen parents. Each bit of the first parent is compared with the same bit of the second parent. When these bits are the same it is used in the offspring, otherwise the bit from the third parent is used in the offspring. For example, the following three parents:

p1	1	1	0	1	0	0	0	1	0
p2	0	1	1	0	0	1	0	0	1
p3	1	1	0	1	1	0	1	0	1
of	1	1	0	1	0	0	0	0	1

GA – Crossover operator

- Cut and Splice
  - In the crossover "cut and splice" approach, cutting points are selected separately for each parent. This may result in a change in length of the child chromosomes.



GA – Mutation operator

- Mutation may be defined as a small random tweak in the chromosome, to get a new solution

- It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – $\boldsymbol{p_m}$

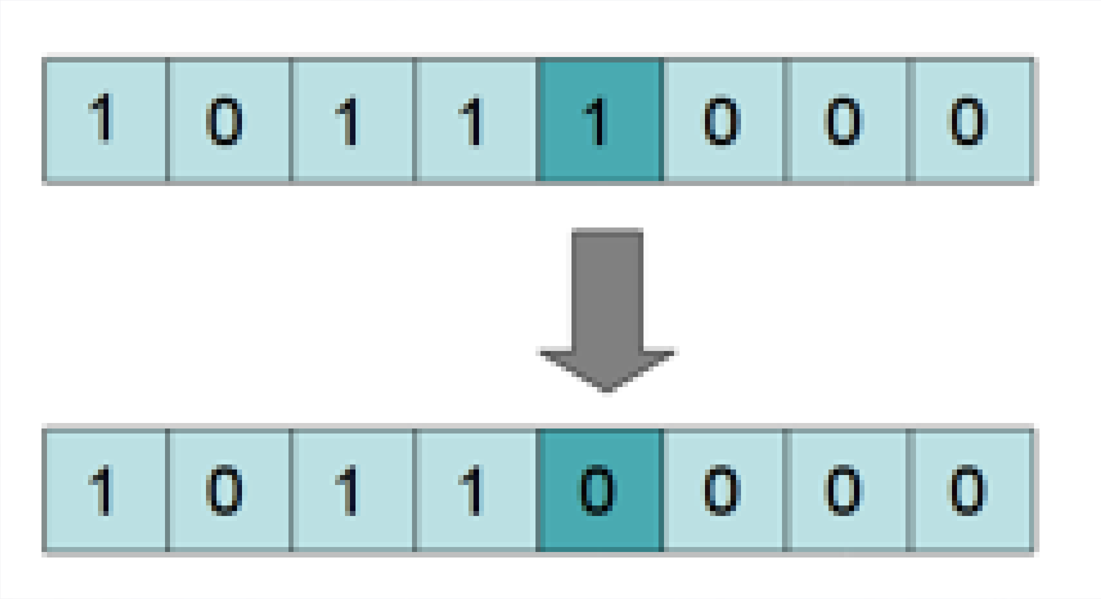
- If the probability is very high, the GA gets reduced to a random search

- Mutation is performed with probability $\boldsymbol{p_m}$ for each gene

  - Typically between 1/pop_size and 1/ chromosome_length

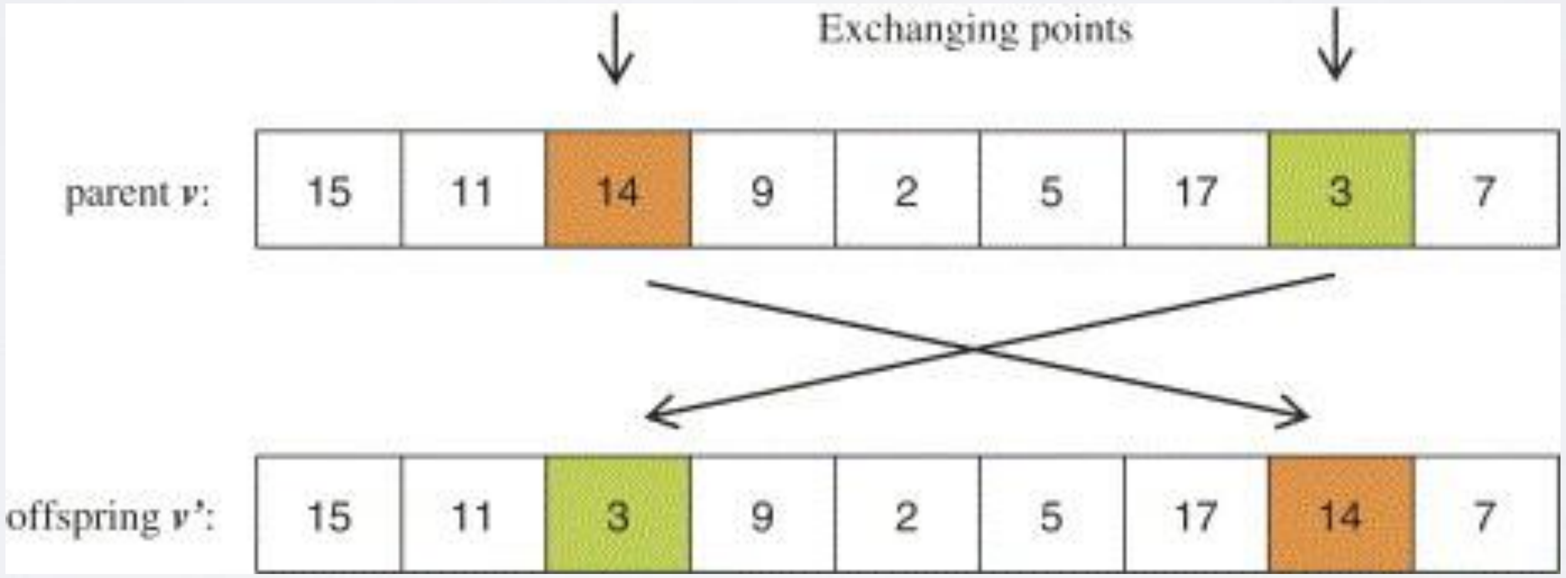
GA – Mutation operator

- Bit flip mutation
  - In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GA.



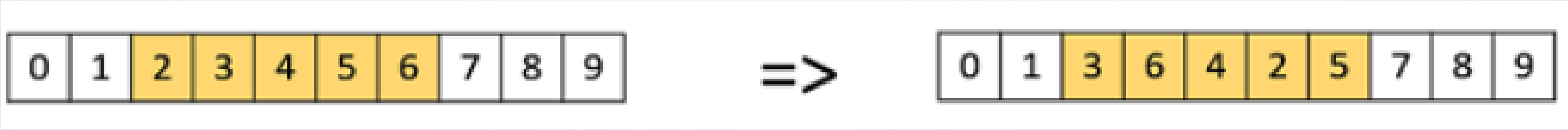
GA – Mutation operator

- Swap mutation
  - In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings

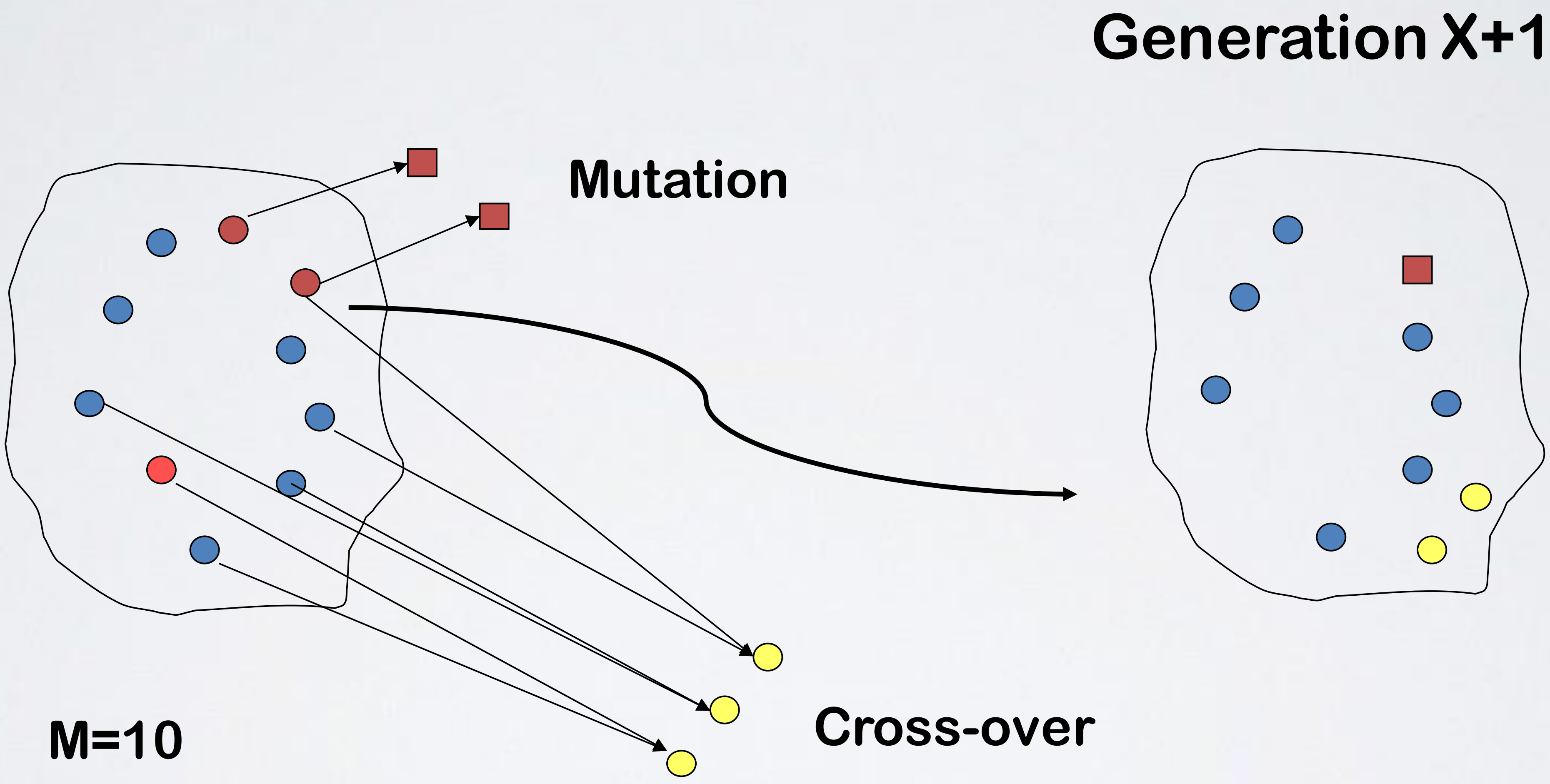


GA – Mutation operator

- Scramble mutation
  - Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



GA – Replacement



GA – Replacement

- Once the new offspring solutions are created using crossover and mutation, we need to introduce them into the parental population

  - **Delete-all**: This technique deletes all the members of the current population and replaces them with the same number of chromosomes that have just been created.

  - **Steady-state**: This technique deletes n old members and replaces them with n new members. The number to delete and replace, n, at any one time is a parameter to this deletion technique. Another consideration for this technique is deciding which members to delete from the current population.

  - **Steady-state-no-duplicates**: This is the same as the steady-state technique but the algorithm checks that no duplicate chromosomes are added to the population. This adds to the computational overhead but can mean that more of the search space is explored.

GA – General Algorithm

Algorithm 3.1.1: $\textsc{GeneticEvolution}(P)$

$t \leftarrow 0;$

$initialize(P(t = 0));$

$evaluate(P(t = 0));$

while $isNotTerminated()$

$$\textbf{do} \begin{cases} P_p(t) \leftarrow P(t).selectParents(); \\ P_c(t) \leftarrow reproduction(P_p); \\ mutate(P_c(t)); \\ evaluate(P_c(t)); \\ P(t+1) \leftarrow buildNextGenerationFrom(P_c(t), P(t)); \\ t \leftarrow t + 1; \end{cases}$$

end

GA – Population Size

- Small populations: undercoverage
- Large population: computationally demanding
- Optimal size increases with the string-length in binary encodings
- Empirically size of 30 can often work, but ok with 10-100