EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Evolutionary Computation Techniques

Genetic Algorithms

Strategic project of TBU in Zlín, reg. no. CZ.02.2.69/0.0/0.0/16_015/0002204

Doc. Ing. Roman Šenkeřík, Ph.D.
FAI, Ústav informatiky a umělé inteligence

Content of the Lecture

- Basic principles of GA
- Selection
- Crossover and mutation operations
- Variants of GA

1. Basic Principles of GA

Principles of GA

- The basic idea of genetic algorithms is a direct analogy with evolutionary processes taking place in biological systems [1].
- **Darwin's theory of natural selection:** only the best adapted individuals survive in the population. The degree of adjustment is the so-called "fitness of the individual". In biology, fitness is understood as the relative ability to survive and reproduce an individual's genotype.
- Biological evolution is the change in the content of a individual's genetic information over many generations towards higher fitness values. Individuals with higher fitness are more likely to survive and more likely to reproduce their genes into a generation of offsprings.
- The so-called mutation is also applied in population development, which is an accidental change in the genetic information of some individuals in a population.

Principles of GA

- In genetic algorithms, fitness is a positive number assigned to an individual's genetic information.
- When using GA, an individual's parameters are called genes (they determine a point in the searched space).
- All genes then give a chain, which we call a chromosome (i.e. the whole individual).
- We call a genotype a set of chromosome parameters.

Principles of GA

- An individual's genetic information (chromosome) is usually expressed by a bit string, usually of constant length *k*:

$$ind = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in \{0,1\}k$$

- A population of size *N* is then a set of such chromosomes:

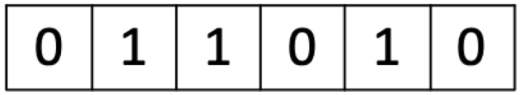
$$P = \{ind_1, ind_2, \ldots, ind_N\}$$

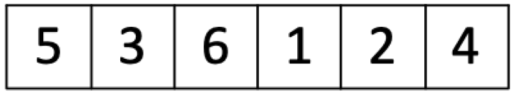
Principles of GA

In Nature	In GAs
Individual	String of symbols, e.g. h = (1001)
Natural selection	Selection according to the fitness function f(h)
Crossover	Combination of two strings e.g. (000 | 1) + (101 |0) (0000)+(1011)
Mutation	Random change 0 to 1 in string e.g. (0010) (1010)

Principles of GA: Representation of individual

For example, an individual may be represented:

- by string of zeros and ones - **binary representation**, 

- by string of numbers – **numerical representation**, 

- by string of characters/symbols – **character representation**,
- **by tree** of objects (e.g. mathematical / logical functions or programming language commands) ➠ genetic programming.

Principles of GA: Representation of individual

- Binary, integer, real, permutation strings are linear structures
- Trees are nonlinear structures - their depth and width changes

Principles of GA: Binary Representation of Individual

The advantages of binary representation are:

- Easy implementation of GA operators (crossover, bit mutation)
- Easy implementation for solving discrete optimization problems (combinatorial problems such as knapsack problem: 0 -> item not selected, 1 -> item selected)

Principles of GA: Binary Representation of Individual

Disadvantages are:

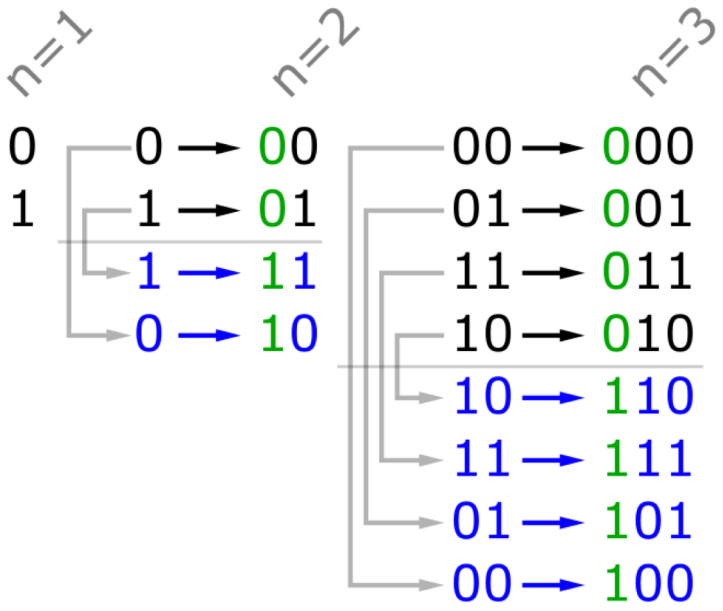
- Finding solutions in the real numbers domain (significant increase in chromosome length).

- Adjacent numeric values are represented by strings that can vary significantly, even in all bits.
  - E.g. the decimal value 7 is in binary (0111), the following numeric value 8 is (1000).
  - The solution is Gray's code, or the representation of individuals by real numbers.

- Variants of genetic algorithms with the representation of an individual as a vector of real numerical values require different crossing and mutation operations than those listed here in this lecture.

Principles of GA: Binary Representation of Individual
Gray's code

- When using Gray's code, strings representing adjacent integer values differ by only one bit
- It is a mirrored binary code.
- The "mirroring" method is to take *n* bit code, extend it with its own mirror copy, and add 0 to the original part, while add 1 to the mirrored part



Principles of GA: Binary Representation of Individual
Gray's code

- The advantage is a more uniform mutation (some studies report better convergence), there can not appear significant changes during mutation and crossover operations.

- BUT: it adds another computational time required for conversion to Gray’s code.

Principles of GA: Integer Representation of Individual

- Integer representation has advantages in permutation optimization problems.
- For example, in the TSP problem: a chromosome (individual) directly represents one of the possible path.

Principles of GA: Fitness

- The suitability (fitness) of an individual is the transformed value of the objective function.
- This value is then used to select individuals for the crossover / elitism operation.
- It is often normalized to the interval [0, 1].
- If we are looking for the global minimum of the objective function f, then fitness F is a representation that satisfies the following condition:

$$f(ind_1) \leq f(ind_2) \Rightarrow F(ind_1) \geq F(ind_2) > 0$$

Principles of GA: Fitness

- The simplest way to satisfy such condition is a linear transformation of functional values to fitness, where for the lowest values of the objective function we obtain the highest fitness values and vice versa, other values are then linearly redistributed in a given interval.

$$F(ind) = \frac{F_{\max} - F_{\min}}{f_{\min} - f_{\max}} f(ind) + \frac{f_{\min} F_{\min} - f_{\max} F_{\max}}{f_{\min} - f_{\max}},$$

where:

f_{min} and f_{max} is the smallest and largest value of the purpose function in the population,

F_{min} and F_{max} is the minimum and maximum value of fitness,

which we usually choose $F_{max} = 1$ a $F_{min} = \varepsilon$, where ε is a small positive number, e.g. $\varepsilon = 0.01$.

Principles of GA: Fitness

Then the equation for fitness is:

$$F(ind) = \frac{1}{f_{\min} - f_{\max}}[(1-\varepsilon)f(ind) + f_{\min}\varepsilon - f_{\max}],$$

Principles of GA: Fitness

- The fitness of individuals in the population can also be re-normalized so that the total sum is always equal to max 1.
- The *i*-th individual's re-normalized fitness is:

$$F'(ind_i) = \frac{F(ind_i)}{\sum_{j=1}^{N} F(ind_j)}$$

- This value then directly indicates the probability of an individual's participation (its selection) in the reproduction of his genetic information, i.e. the creation of an offspring.

Principles of GA: Elitism

- Due to genetic operators, the best individuals may be lost in crossover or mutation.

- **Elitism** solves this problem by choosing the strongest (elite) individuals, and "sends" them directly to the next generation. This will ensure that in each generation, the best individual in the worst case will be of the same quality as in the previous generation.

- Sometimes the relation $k = (1 - P_c) \cdot N$ is used. N is the size of the population, P_c represents the probability of crossover (but this is not always the rule, of course it applies to higher values of P_c).

2. Selection

Selection of Individuals in GA

- Selection process chooses individuals ("parents") for crossover and mutation operations, thus the creation of new offsprings for the next generation.

- Individuals with a higher fitness value should be given preference in the selection, because they have a better "genetic structure" and are therefore more likely to become better individuals (offprings) by crossover and mutation.

- However, this does not mean that worse individuals should not be selected, because also they can carry essential genetic information that better individuals may lack. It is necessary to maintain the population diverse.

- There are a number of methods for selection, the three most used are described here.

Selection of Individuals in GA

- ***Probability selection (Roulette rule)***
- ***Rank selection***
- ***Tournament selection***

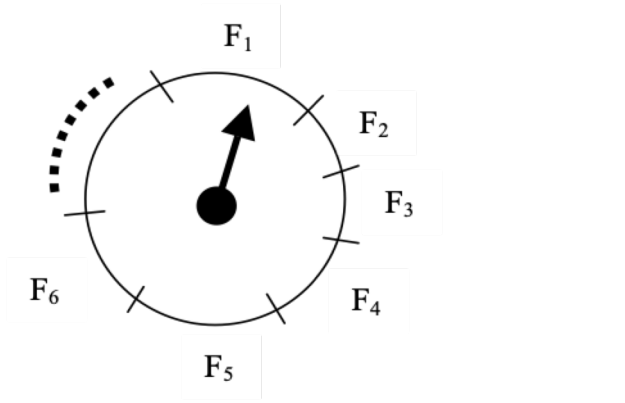
Selection of Individuals in GA
Probability selection (Roulette rule)

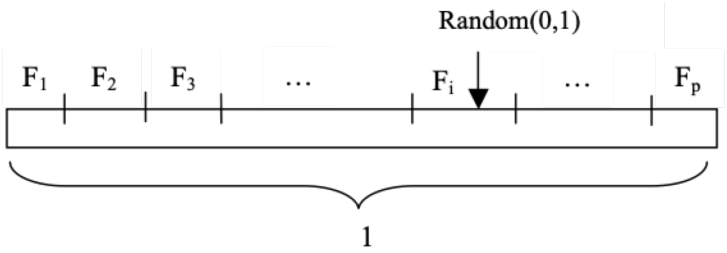
- It is applicable both for re-normalized values of fitness (sum is 1), as well as for unadjusted fitness values.

- Each individual occupies part of roulette depending on its fitness value. The probability of choosing an individual is therefore directly proportional to its quality.

- The imaginary ball on roulette is the generation of a random number that determines which field will be selected.

Selection of Individuals in GA
Probability selection (Roulette rule)





The following "roulette" algorithm applies to both variants (with re-normalization, where S = 1, and without re-normalization):

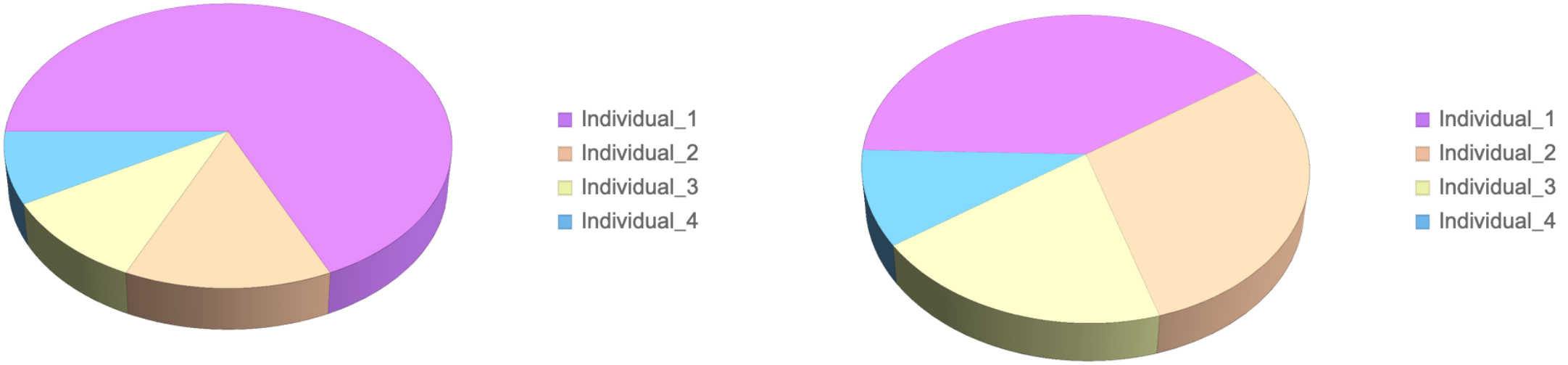
1. the sum of all fitness values in the population - *S*,
2. generating a random number from the interval (0, *S*) - *r*,
3. gradual passage through the population and calculation of the sum of fitness values *S*, from the value 0,
4. as soon as the value of *S* becomes greater than the value of *r*, the individual is selected as the parent.

Selection of Individuals in GA
Rank selection

- Rank selection is similar to roulette selection.
- It assigns “fields” on imaginary roulette according to the order of individuals in the population sorted by their quality.
- The worst individual will occupy 1 “field” and the best *N* fields, where *N* is the size of the population.

Selection of Individuals in GA
Rank selection

This solves the problem of too large differences in fitness values, because the difference between neighboring individuals will always be at most 1 field on roulette.



Roulette selection (left) vs. Rank selection on the right

Selection of Individuals in GA
Tournament Selection

- Tournament selection will initially select *n* individuals to compete for reproduction.
- The tournament is won by the individual with the highest fitness value.
- The size of the tournament can be used to adjust the nature of the tournament. With a small number of selections, even weaker individuals have a better chance of reproduction, while in large tournaments, worse individuals are less likely to win.

Selection of Individuals in GA
Tournament Selection

If n is equal to one, the tournament selection will behave as a random selection.

For $n = 2$, the scheme will be as follows:

- Randomly select two individuals ind_1, ind_2 (with the same probability of selection).
- With the probability P_t , select an individual with a higher F value, otherwise select an individual with a lower F value.

Selection of Individuals in GA
Tournament Selection

For higher n, the procedure will be as follows:

- randomly select *n* (tournament size) individuals from a population
- choose the best individual from the tournament with a probability of P_t
- choose the second best individual with the probability $P_t * (1- P_t)$
- choose the third best individual with probability $P_t * ((1- P_t) ^ 2)$
- and so on…

- Thus, it can be assumed that it is unlikely to get further than the second or third individual in the tournament if P_t is a high value.

3. Crossover and Mutation

Crossover in GA

Crossover is the most important operator because it is the essence of genetic algorithms - a combination of solutions.

Crossover depends on the implemented representation of individuals, but always takes place between two parents.

Crossover in GA

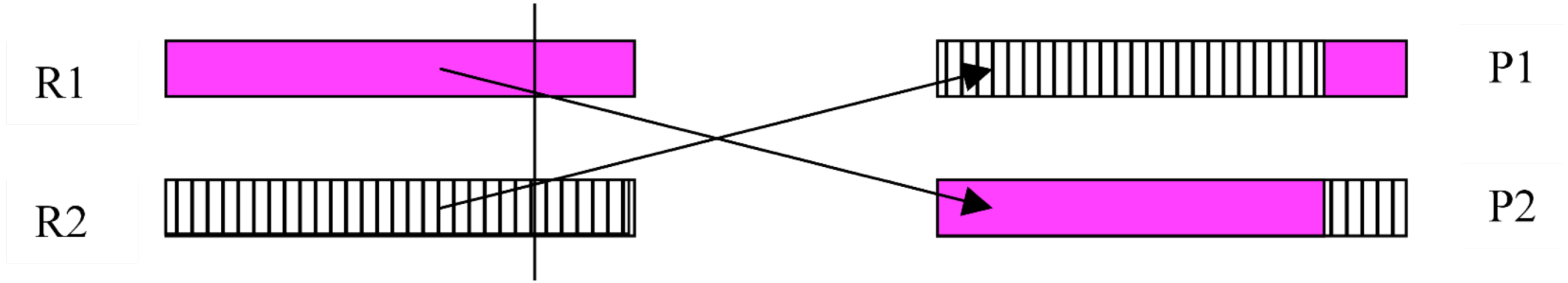
There are several types of crossover with a „classical binary string“ individuals:

- One-point
- Two-point
- Multi-point
- Uniform (not often case)

Crossover in GA
One-point

Random selection of a point in the chain, everything before this point is transferred from the first parent, the rest from the second parent.

Similarly, a second offspring can be created, which will be formed by opposite parts of the parents.

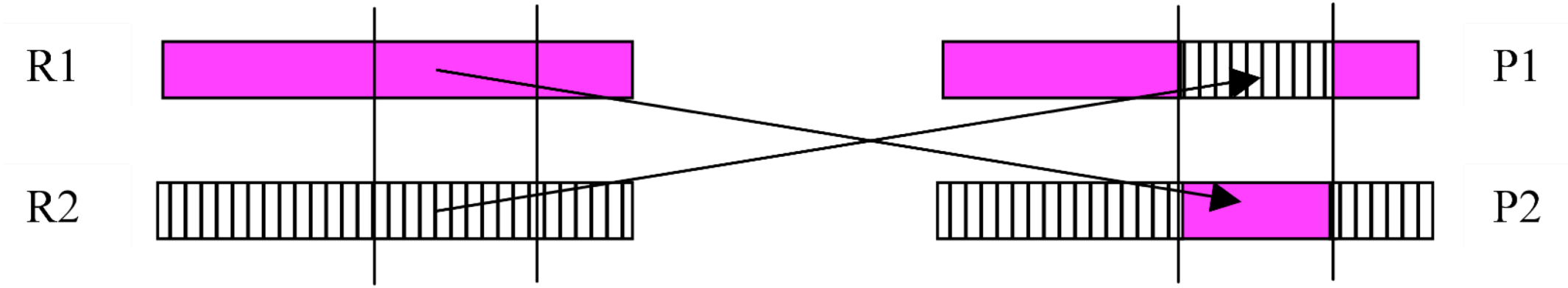


Crossover in GA
Two-point

Instead of one point, two crossover points are generated here.

One offspring is then formed by the center bounded by these points and the remaining parts of the other parent. The second offspring is exactly the opposite.

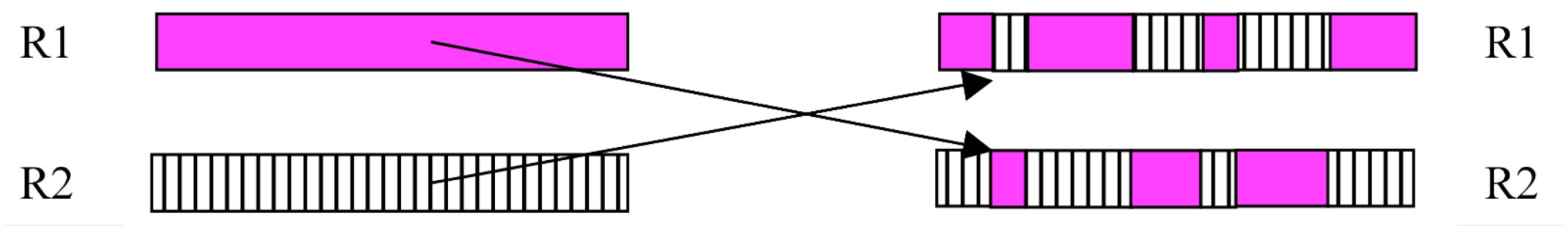
Thus, the “middle” sequence of bits is ”swapped".



Crossover in GA
Multi-point

Generating more than two crossover points.

The odd parts from the first parent are combined with the even parts from the second parent.



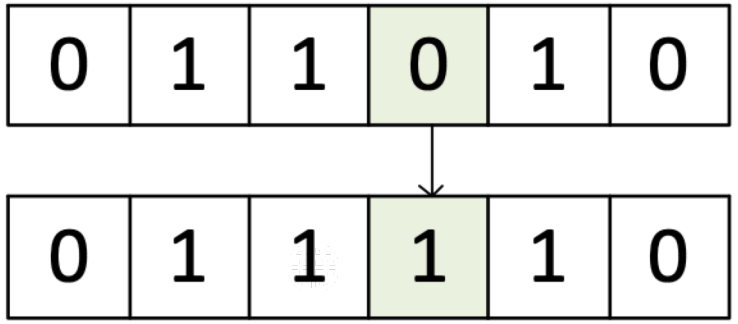
Crossover in GA
Uniform

Here, each gene from the first parent has a probability of 0.5 that will be selected for the offspring.

If not selected, the gene is taken at the same position from the other parent.

Mutation in GA

- The mutation is implemented in genetic algorithms as a change in the value of a randomly selected bit in a chromosome, i.e. the value 0 changes to 1, the value 1 to 0.
- Mutation ensures that genetic information that has not been present in previous generations can be created in the population, or that genetic information lost during previous development can be restored.
- As a result, the algorithm can escape from the area of the local minimum to which it would converge if we only used crossover.
- On the contrary, if we used only mutation, the genetic algorithm would behave similarly to a blind (random) search.



Influence of Control Parameters

- The described evolutionary operations is performed according to the parameters of crossover probability and mutation.
  - The probability of crossover tells whether the crossing will be performed on the selected pair of parents.
  - If there is no crossbreeding, the offspring are exact copies of their parents.

- Even in the extreme case, if the probability of crossover is zero, the offspring are always exact copies of their parents.
  - But that does not mean that the new population is the same as the old one.
  - Still, there is a “selection of parents” who participate in the "production of offspring".

- If the probability is 100%, then (***except for elitist individuals***) all offspring are created by crossover. Crossover assumes that a good part of the chromosome will be passed on to the offspring and together with the new part will lead to finding better solutions.

Influence of Control Parameters

- The probability of mutation has similar functionality as crossing.
- If it is 100%, then the whole chromosome is changed.
- In the case of zero probability, nothing is mutated, so the protection mechanism against stagnation at the local extreme is not active.

Influence of Control Parameters

Population size. It should not be too small. Individuals should cover as much space as possible with their genes. At the same time, it should not be too large, because this increases the computational complexity. The recommendation for GA (but it is necessary to take into account the NFL) is to use large populations and a smaller number of generations, rather than smaller populations and a large number of generations.

Probability of crossing. Usually P_c = 75 - 95%. As already mentioned - often tied to elitism. In some versions of GA, the number of elitist individuals is defined as $N_e = (1 - P_c)^*$ Population size. The rest of the population will undergo crossing operations.

Probability of mutation. It should not be too intense. Usually 1%.

Influence of Control Parameters

Termination condition. Usually a certain number of generations, but there can also be a limit on the number of evaluations of the purpose function (FES), or a time limit, or a threshold value for solving a specific problem (eg business travel route length) from which the solution is considered successful.

Other parameters are, for example, population overlap, or parameters specific to each optimized problem.

GA Pseudocode

One of many versions ...

Algorithm 1. Genetic Algorithm for the NRP

Input: instance Π,
size α of population,
rate β of elitism,
rate γ of mutation,
number δ of iterations

Output: solution X

```
// Initialization
 1 generate α feasible solutions randomly;
 2 save them in the population Pop;
// Loop until the terminal condition
 3 for i = 1 to δ do
// Elitism based selection
 4   number of elitism ne = α · β;
 5   select the best ne solutions in Pop and save them in Pop1;
// Crossover
 6   number of crossover nc = (α − ne)/2 ;
 7   for j = 1 to nc do
 8     randomly select two solutions XA and XB from Pop ;
 9     generate XC and XD by one-point crossover to XA and XB;
10     save XC and XD to Pop2 ;
11   endfor
// Mutation
12   for j = 1 to nc do
13     select a solution Xj from Pop2;
14     mutate each bit of Xj under the rate γ and generate a new solution Xj' ;
15     if Xj' is unfeasible
16       update Xj' with a feasible solution by repairing Xj' ;
17     endif
18     update Xj with Xj' in Pop2;
19   endfor
// Updating
20   update Pop = Pop1 + Pop2 ;
21 endfor
// Returning the best solution
22 return the best solution X in Pop ;
```

4. Other Variants of GA

Other Variants of GA
microGA (μGA)

- μGA ➠ genetic algorithms with very small populations [2].
- They are often used as a local search technique.
- The number of individuals is most often set to 5, with a probability of crossover of 80%, i.e. 1 elite individual.

Other Variants of GA
Steady-State GA

- Unlike generic GA, in "steady state" genetic algorithms, generations overlap [3].

- Thus, new individuals make up only a certain part of the population. In one time (iteration) a whole new generation is not created - what would replace the old one, but only a few individuals are replaced.

- It is necessary to determine the rules as to which older individuals will be replaced by new ones. E.g. replace the worst fitness (but watch out for the loss of diversity), etc.

Other Variants of GA
Messy GA

- Designed to accelerate convergence on selected multimodal problems [4].
- It has a variable chromosome length and genes do not have a fixed position. Each bit value is also determined by its position.
- Messy GAs solve problems by combining relatively short, well-tested building blocks to form longer, more complex strings that increasingly cover all features of a problem.
- The chromosomes in MessyGA can be over-specified or under-specified.

  - **Over-specified**: There are two values in the same position (then the rule of the first used applies).
  - **Under-specified**: Some bits are missing in the chromosome. However, these are needed for evaluation by the objective function. To complete the missing bits, a so-called template is introduced, in which the best solution so far is stored. The values of the arguments, which are then missing in a chromosome, are completed from this template.

Other Variants of GA
Hybrid GA

- Hybrid GA combines the power of standard GA with the speed of local search techniques [5].

- First GA finds a suitable area and the local search technique then (theoretically) finds an extreme.

- Hybrid GA takes various forms:
  - GA runs until it begins to "slow down", then a local search procedure is activated, assuming that GA is very close to the global extreme,
  - every x iterations, the running local search procedure passes its best solution or several solutions, like promising chromosomes, to the population.
  - ….

Other Variants of GA
Parallel GA

- We consider either the redistribution of the processing of genetic operations in the population to multiple processors, or we consider different models of population management / development.

- Therefore, parallel GA (PGA) does not always mean implementation on a multiprocessor machine or using distributed computing [6].

- Population development models similar to those in nature are also considered - developing parallel separate populations, or smaller subpopulations, where individuals may or may not migrate between parallel sub-populations in certain generations.

- More in the presentation about parallel variants of EA.

References

[1] Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning.

[2] Krishnakumar, K. (1990, February). Micro-genetic algorithms for stationary and non-stationary function optimization. In Intelligent control and adaptive systems (Vol. 1196, pp. 289-296). International Society for Optics and Photonics.

[3] Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. In Foundations of genetic algorithms (Vol. 1, pp. 94-101). Elsevier.

[4] Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. Complex systems, 3(5), 493-530.

[5] El-Mihoub, T. A., Hopgood, A. A., Nolle, L., & Battersby, A. (2006). Hybrid Genetic Algorithms: A Review. Engineering Letters, 13(2), 124-137.

[6] Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. Complexity, 4(4), 31-52.

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Thank you for attention

Strategic project of TBU in Zlín, reg. no. CZ.02.2.69/0.0/0.0/16_015/0002204

Assoc. Prof. Roman Šenkeřík
FAI, Department of Informatics and Artificial Intelligence