Metaheuristic Optimization: Genetic Algorithm

Adaptive and Cooperative Algorithms (ECE 457A)

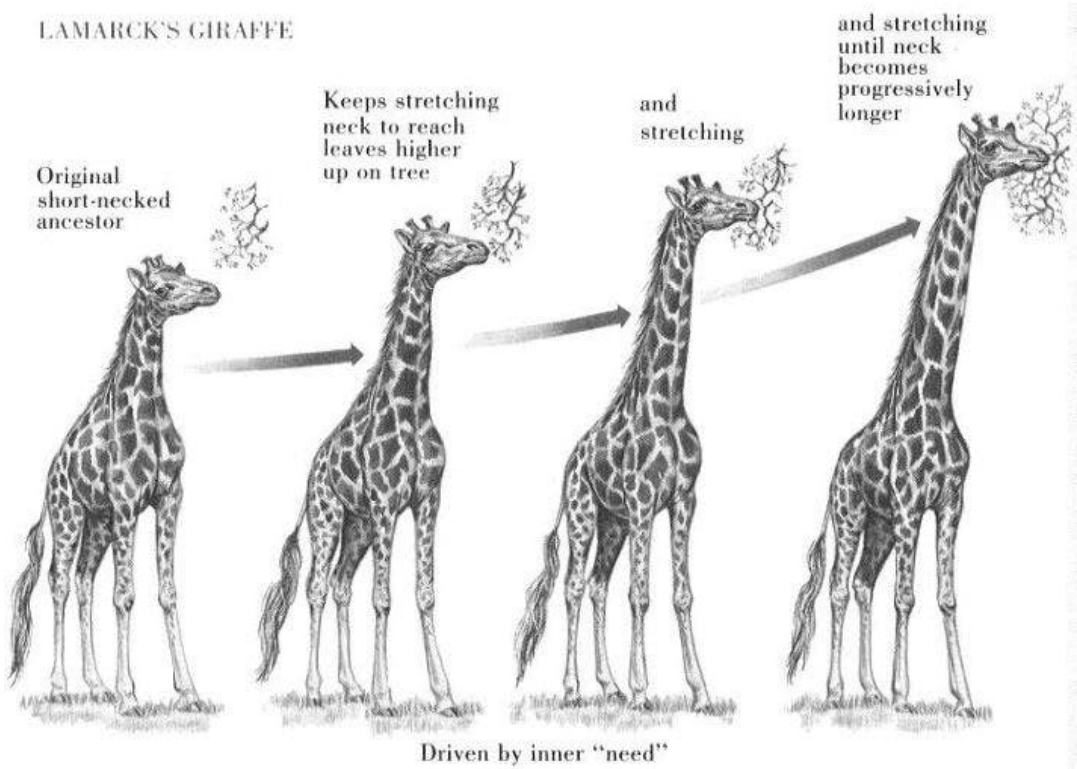
ECE, MME, and MSCI Departments,
University of Waterloo, ON, Canada

Course Instructor: Benyamin Ghojogh
Fall 2023

Evolution, Natural Selection, and Darwinism

Evolution, Natural Selection, and Darwinism

- Biology and geology show that living beings **evolve** during long-term periods.
- According to **natural selection**, every generation tries to adapt better to the environment compared to the previous generation.
- **Genetic Algorithm (GA)**, proposed by John H. Holland in 1975 [1], is inspired by natural selection and Darwinism. It is one of the **evolutionary algorithms**.

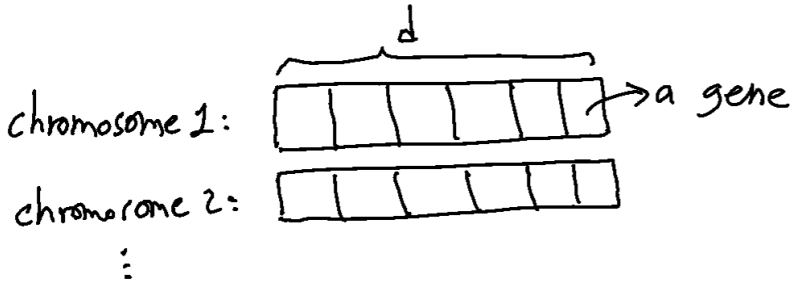


Credit of image: `https://www.zmescience.com/science/what-is-natural-selection/`

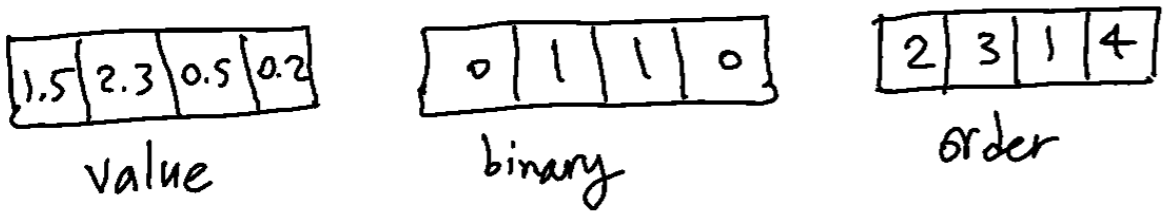
Cost and Optimization Variables

Chromosomes

- The candidate solutions are called the **chromosomes**.
- Every chromosome has several **genes**. The genes are the features/dimensions of the candidate solutions.

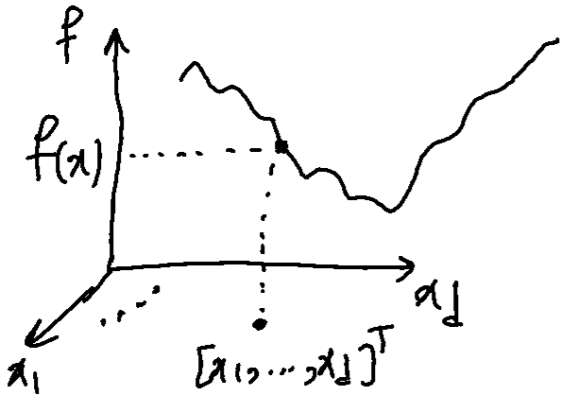


- Possible ways to create/encode the chromosomes:
  - value encoding: the genes are float values of the features/dimensions of vector solutions.
  - binary encoding: every solution is a scalar and the scalar is converted to a binary string. Every gene is binary $\{0, 1\}$.
  - order encoding: the genes are integers if we want to find an optimal order of some integers.



Fitness/Cost function

- The fitness/cost function value is the evaluation of function at a candidate solution.



- Usually we want to maximize the fitness function and minimize the cost function. In other words, fitness function is minus cost function.
- Here, in genetic algorithm, we usually maximize a fitness function (or equivalently maximize the negative cost).
- We do it iteratively.
- **fitness/cost calculation**: at every iteration, we calculate the fitness/cost of chromosomes to see how good they are.

Natural Selection

Natural selection

- **Natural selection**: at every iteration, we **select** some of the chromosomes, from the current generation, to be the parents of the next generation.
- The other chromosomes which are not selected will die without having **offspring (children)**.
- **Selective pressure** (also called the **take-over pressure**): the more the selective pressure is, the more the speed of selecting the elites as parents, so the sooner the elites take over the population, so the sooner the algorithm converges (the more it exploits rather than exploring).

There are various ways for natural selection of chromosomes to be parents of the next generation:

- **Random selection**: selecting k random chromosomes as parents out of the chromosomes at the existing generation. It has the least amount of selection pressure. It has **too much exploration**.
- **Elitism selection**: selecting the top k chromosomes with the best fitness/cost values. It has the risk of getting **stuck in local optimum**. It has **too much exploitation**.

Natural selection

- **Proportional selection**: for every chromosome, we calculate a probability of being good (compatible with the environment / having good fitness value):

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \quad \forall i \in \{1, \ldots, n\}, \tag{1}$$

where f_i is the fitness value for the i-th chromosome and n is the population of chromosomes at the generation. Every chromosome has its probability for being selected as a parent of the next generation.

- For making the differences of probabilities bolder, we can use this formula:

$$p_i = \frac{f_i - f_{\min}}{\sum_{j=1}^{n} (f_j - f_{\min})}, \quad \forall i \in \{1, \ldots, n\}, \tag{2}$$

where $f_{\min}$ denotes the minimum fitness among the fitnesses of the chromosomes in the current generation.

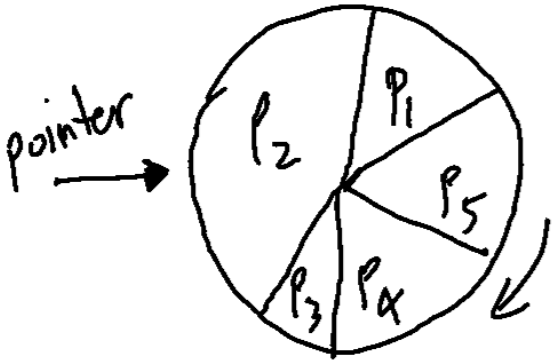
- In the calculation of probabilities, If the some (or all) fitness values are negative, we can add $|f_{\min}|$ to all of them so all of them become nonnegative:

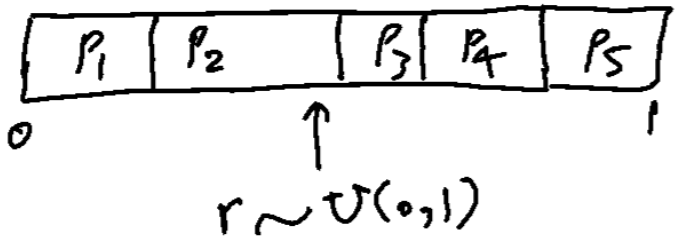
$$f_i := f_i + |f_{\min}|, \quad \forall i \in \{1, \ldots, n\}, \tag{3}$$

where $|.|$ denotes the absolute value.

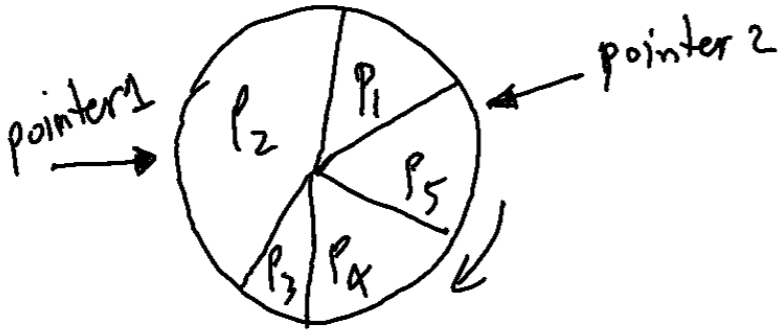
Natural selection

- In computer programming, we can select by probabilities using a **roulette wheel**:



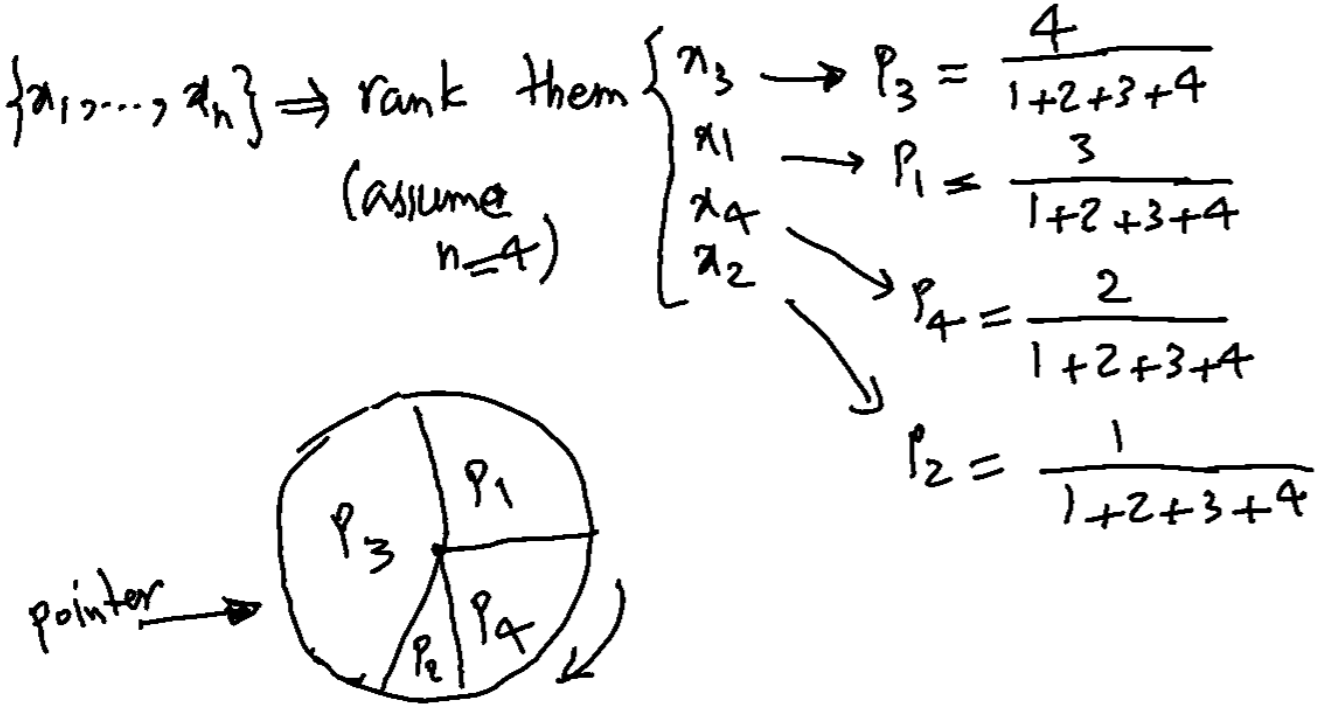


- **Stochastic Universal Sampling (SUS) selection**: it is similar to the roulette wheel but with more than one selection pointer. It gives **more chance** to chromosomes with less fitness value.



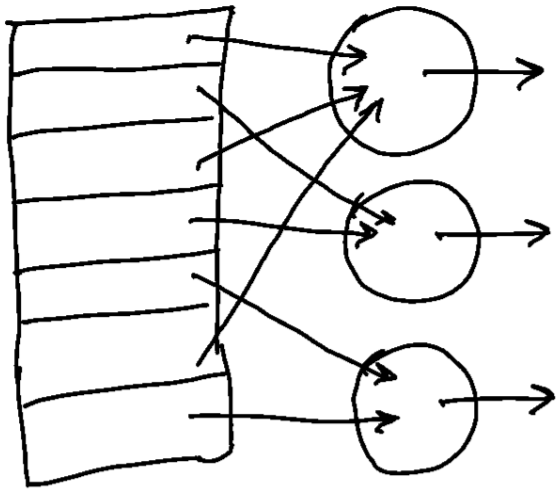
Natural selection

- **Rank-based selection**: Close to the termination, the probabilities in the roulette wheel with probabilities will become very close to each other. So, it will be better to rank the chromosomes by their fitness values in the late iterations. Then, we create the roulette wheel using the probabilities proportional to the ranks. This gives more chance to all chromosomes.



Natural selection

- **Tournament selection**: We randomly group the chromosomes and then, we randomly sample from the groups.
- This method is using **stratified sampling** in which the data are grouped into multiple strata and samples are drawn from the strata. For more information on stratified sampling refer to our tutorial [2].



- A special case of the tournament selection: first we randomly select k_1 chromosomes and then, we select k_2 best chromosomes among them with best fitness values.

$$\{x_1, \ldots, x_{k_1}\} \Longrightarrow \text{rank them} \begin{Bmatrix} x_3 \\ x_1 \\ x_5 \\ \vdots \end{Bmatrix} \rightarrow \text{get top } k_2$$

Cross-over

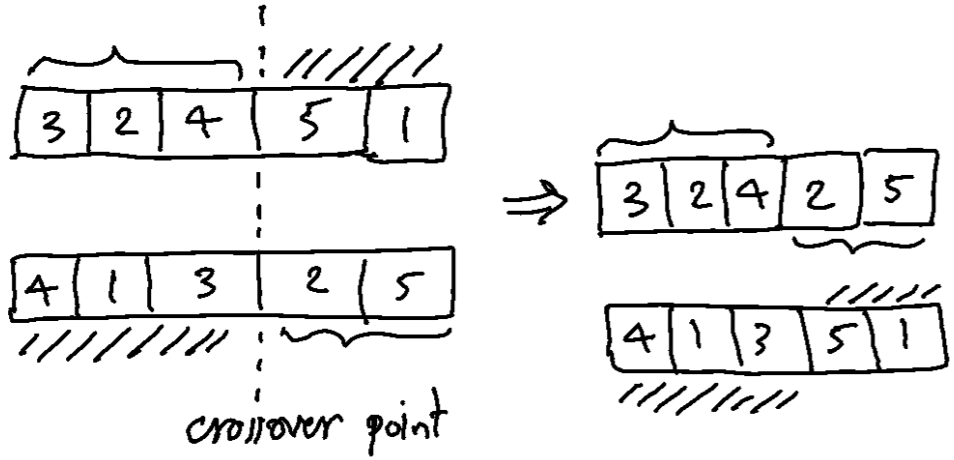
Cross-over

- **Cross-over**: Mating of parent chromosomes to generate offsrpings (children).
- It is useful for **exploration**.
- There can be cross-over either with **two** or **multiple (more than two)** parents.
- From every two or more than two parents, we can generate either **only one** or **multiple** offsprings.

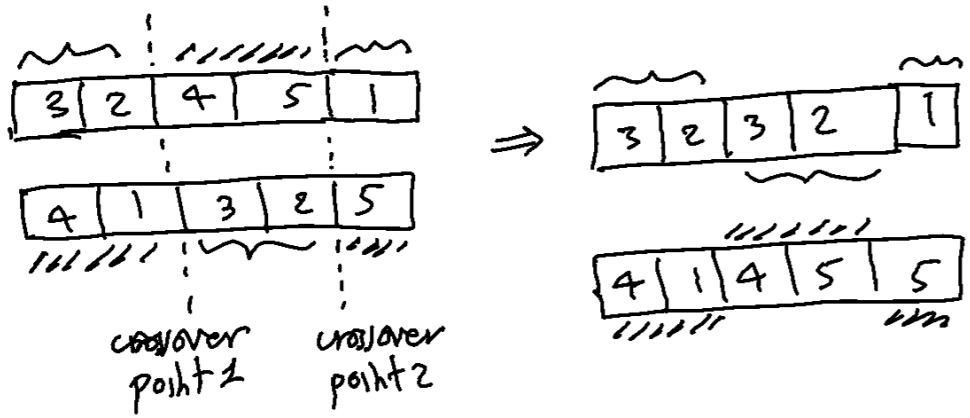
Cross-over

There are various ways for cross-over. Here, we explain the cross-over of two parents. These methods can be generalized to have multiple parents per off-spring.

- **One-point cross-over**:

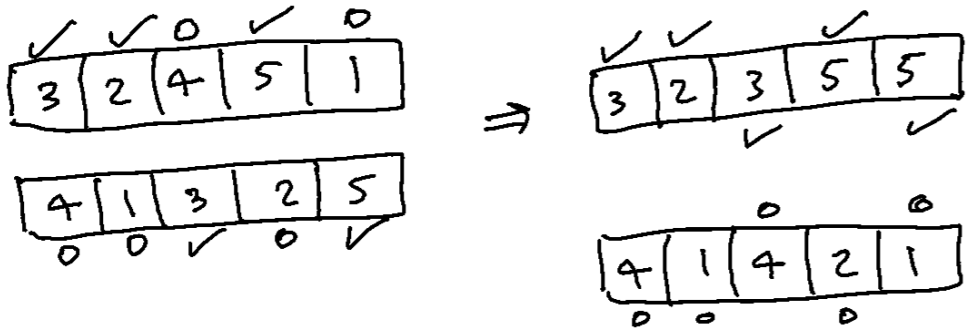


- **Multi-point cross-over**:



Cross-over

- **Uniform cross-over**:
  - We randomly choose the j-th gene from one of the parents for the j-th gene in the offspring.

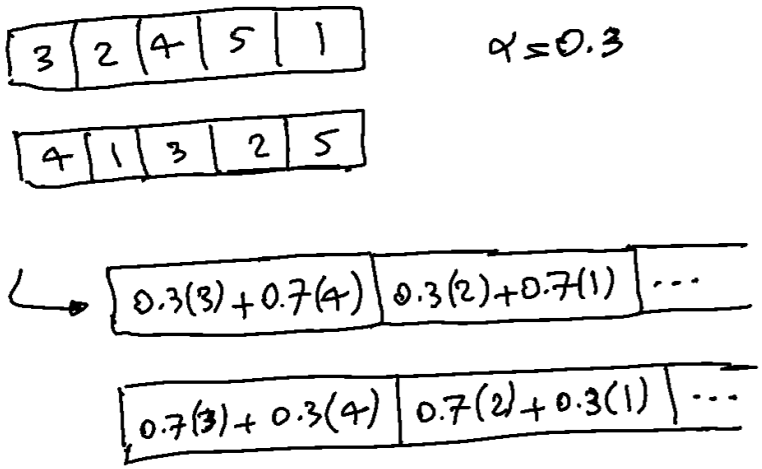


Cross-over

- **Arithmetic mean cross-over**:

$$\text{offspring1} = \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2, \tag{4}$$
$$\text{offspring2} = \alpha \boldsymbol{x}_2 + (1-\alpha)\boldsymbol{x}_1, \tag{5}$$

where $\alpha \in (0, 1)$.



- **Geometric mean cross-over**:

$$\text{offspring1} = \boldsymbol{x}_1^{\alpha} \times \boldsymbol{x}_2^{(1-\alpha)}, \tag{6}$$
$$\text{offspring2} = \boldsymbol{x}_2^{\alpha} \times \boldsymbol{x}_1^{(1-\alpha)}, \tag{7}$$

where $\alpha \in (0, 1)$.

Cross-over

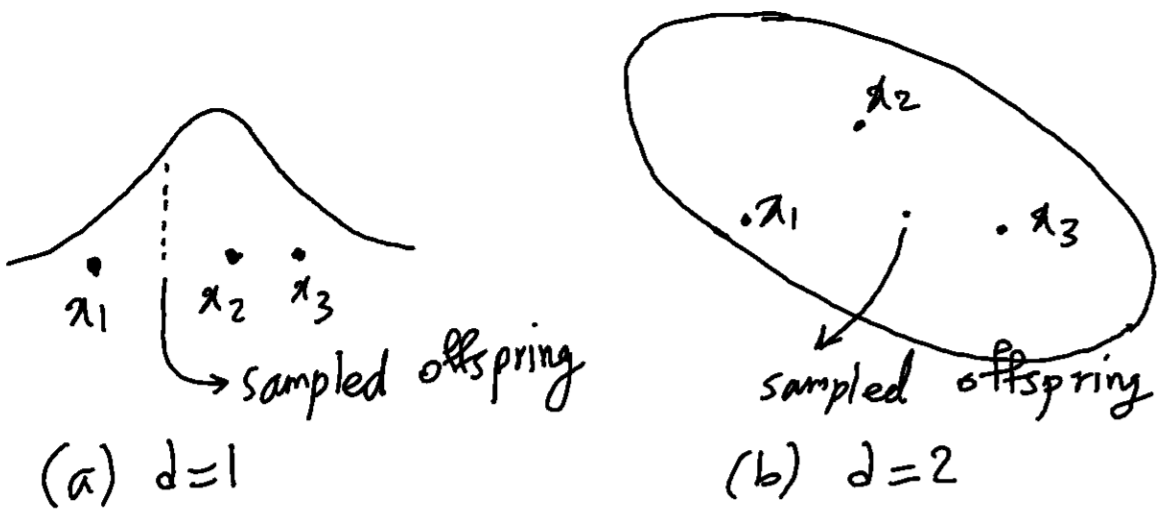
- **UNDX (Unimodal Distribution Crossover)**:
  - It is multi-parent crossover method.
  - We fit a unimodal Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to the m parents, where the mean and covariance of the distribution are calculated as:

$$\mathbb{R}^d \ni \boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_i, \tag{8}$$

$$\mathbb{R}^{d \times d} \ni \boldsymbol{\Sigma} = \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top, \tag{9}$$

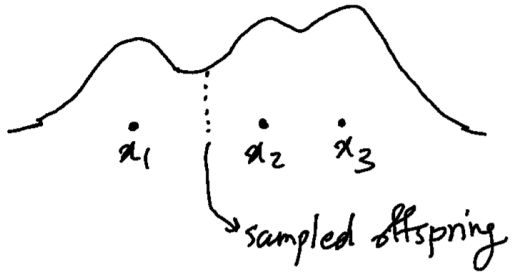
    where m is the number of parents (e.g., $m = 2$).
  - Then, we sample as many offsprings as desired from this Gaussian distribution.

$$\text{offspring} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{10}$$

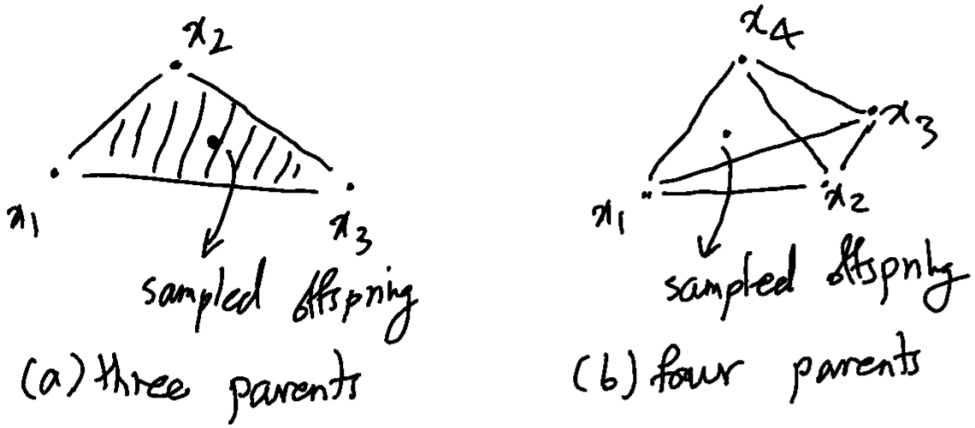


Cross-over

- **PCX (Parent-Centeric Crossover)**:
  - Rather than one Gaussian distribution to the m parents, we fit a Gaussian Mixture Model (GMM) with m modes.
  - For more information on fitting mixture distributions using expectation maximization, refer to our tutorial [3].
  - GMM using sklearn in Python: `https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html`



- **SPX (Simplex Crossover)**:
  - We consider the m parents as m corners of a simplex in the space.
  - Then, we sample the offpring(s) from the region inside the simplex.

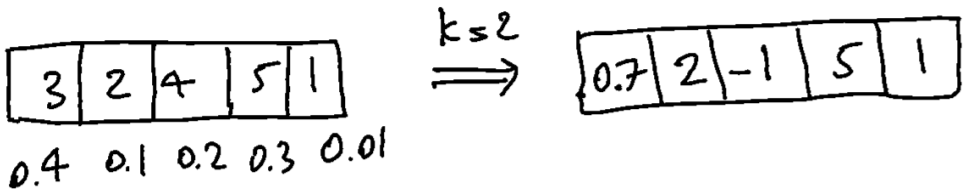


Mutation

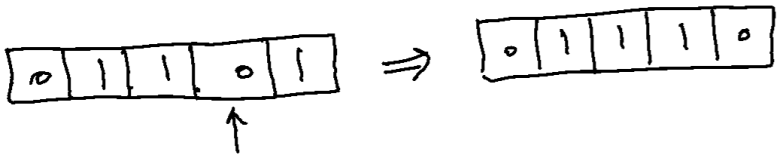
Mutation

Mutation: Random changes to the offsprings after crossover. Useful for **exploitation**.

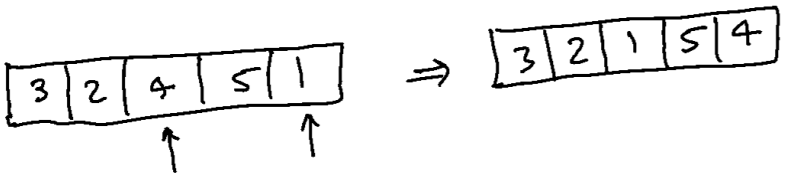
- **Uniform mutation**: We generate random probability for every gene, i.e., $p_j \in [0, 1], \forall j \in \{1, \ldots, d\}$. Then, for the top k genes with highest probabilities (e.g., $k = 1$), we change genes' values to random values in the range of that feature.



- **Bit-flip mutation**:

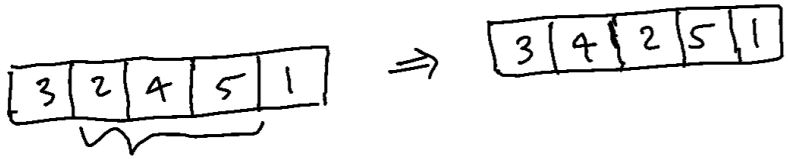


- **Swap mutation**:

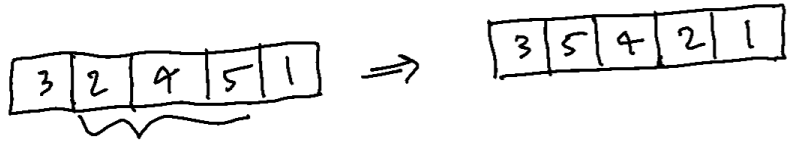


Mutation

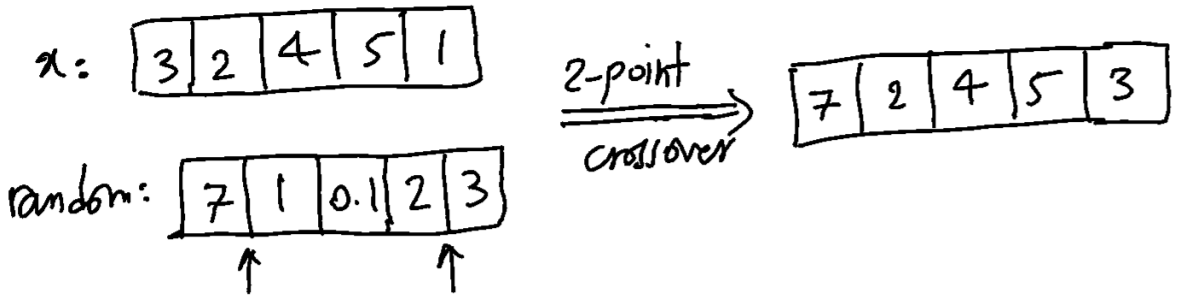
- **Scramble mutation**:



- **Inversion mutation**:



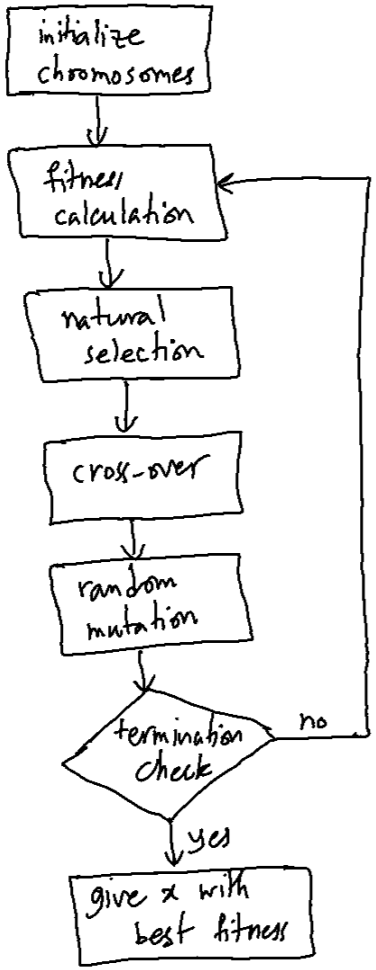
- **Macro mutation (headless chicken)**: this mutation is implemented as a crossover of the chromosome and a random chromosome, using any of the crossover methods.



Flowchart

Termination and Flowchart

- Termination criterion:
  - reaching maximum number of iterations
  - reaching maximum elapsed (passed) time
  - reaching small enough cost (large enough fitness) compared to a known threshold



Acknowledgment

- Some slides of this slide deck are inspired by teachings of Prof. Saeed Sharifian at the Amirkabir University of Technology, Department of Electrical Engineering.
- A good web link about genetic algorithm: `https://medium.com/@AnasBrital98/genetic-algorithm-explained-76dfbc5de85d`
- Surveys on genetic algorithm variants: [4, 5]

References

[1] J. H. Holland, "Adaptation in natural and artificial systems," *Ann Arbor*, vol. 7, pp. 390–401, 1975.

[2] B. Ghojogh, H. Nekoei, A. Ghojogh, F. Karray, and M. Crowley, "Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review," *arXiv preprint arXiv:2011.00901*, 2020.

[3] B. Ghojogh, A. Ghojogh, M. Crowley, and F. Karray, "Fitting a mixture distribution to data: tutorial," *arXiv preprint arXiv:1901.06708*, 2019.

[4] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[5] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia tools and applications*, vol. 80, pp. 8091–8126, 2021.