Genetic Algorithms - Introduction

- Genetic Algorithm (GA) is a search-based optimization technique.
- It based on the principles of **Genetics and Natural Selection**.
- It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.
- It is frequently used to solve optimization problems, in research, and in machine learning.

Introduction to Optimization

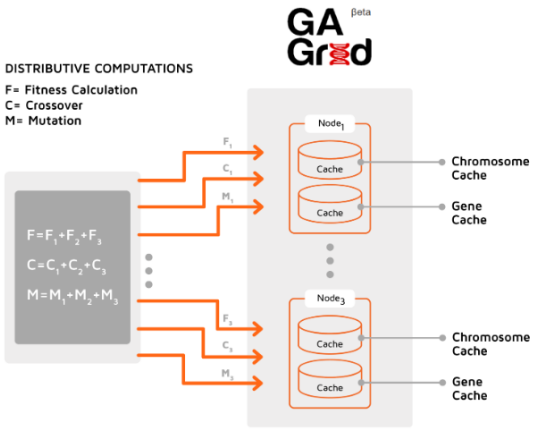
- Optimization is the process of **making something better**.
- In any process, we have a set of inputs and a set of outputs as shown in the following figure.
- it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.
- The aim of optimization is to find that point or set of points in the search space.

What are Genetic Algorithms?

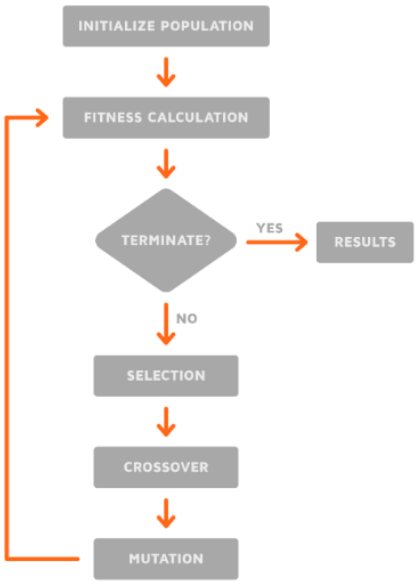
- Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics.
- GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.
- GAs were developed by John Holland and his students and colleagues at the University of Michigan,
- Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search.
- GA is a metaheuristic search and optimization technique based on principles present in natural evolution.
- It belongs to a larger class of evolutionary algorithms.
- GA maintains a **population of chromosomes**—a set of potential solutions for the problem.

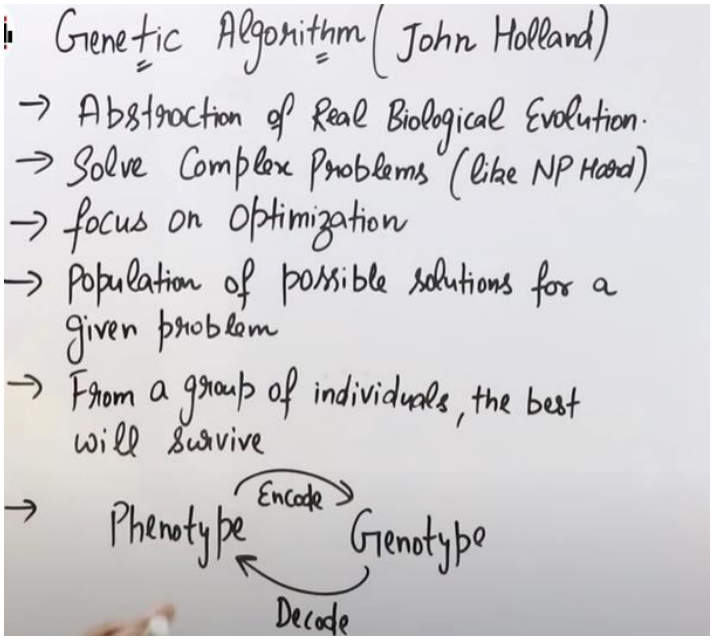
- GA mimics three evolutionary processes: selection, gene crossover, and mutation.
- Genetic Algorithm (GA) represents a subset of Ignite Machine Learning APIs.
- GA is a method of solving optimization problems by simulating the process of biological evolution.
- GAs are excellent for searching through large and complex data sets for an optimal solution.
- Real world applications of GAs include automotive design, computer gaming, robotics, investments, traffic/shipment routing and more.
- All genetic operations such as Fitness Calculation, Crossover, and Mutation are modeled as a ComputeTask for distributive behavior.

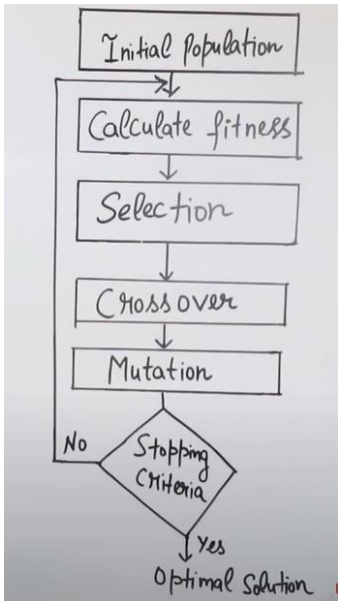
The following diagram depicts the architecture of Genetic Algorithms:



The following diagram depicts the steps performed by Genetic Algorithm:







Benefits of using GA :

1. It is easy to understand.
2. It is modular and separate from application.
3. It supports multi-objective optimization.
4. It is good for noisy environment.

Limitations of genetic algorithm are :

1. The problem of identifying fitness function.
2. Definition of representation for the problem.
3. Premature convergence occurs.

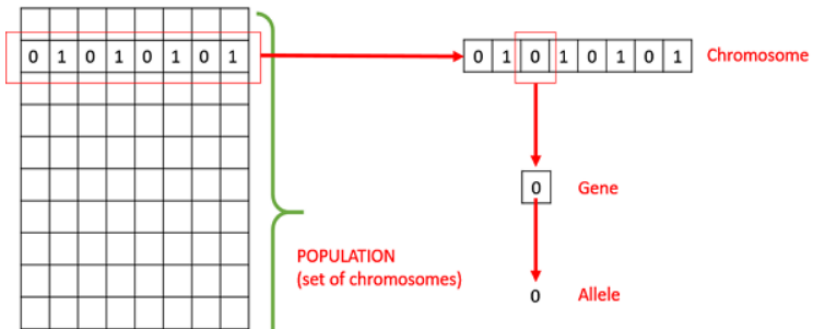
5. Cannot use gradients.
6. Cannot easily incorporate problem specific information.
7. Not good at identifying local optima.
8. No effective terminator.
9. Not effective for smooth unimodal functions.
10. Needs to be coupled with a local search technique.

Application areas

- **Neural Networks** − GAs are also used to train neural networks, particularly recurrent neural networks.
- **Parallelization** − GAs also have very good parallel capabilities, and prove to be very effective means in solving certain problems, and also provide a good area for research.
- **Image Processing** − GAs are used for various digital image processing (DIP) tasks as well like dense pixel matching.
- **Vehicle routing problems** − With multiple soft time windows, multiple depots and a heterogeneous fleet.
- **Scheduling applications** − GAs are used to solve various scheduling problems as well, particularly the time tabling problem.
- **Machine Learning** − as already discussed, genetics based machine learning (GBML) is a niche area in machine learning.
- **Robot Trajectory Generation** − GAs have been used to plan the path which a robot arm takes by moving from one point to another.
- **Parametric Design of Aircraft** − GAs have been used to design aircrafts by varying the parameters and evolving better solutions.
- **DNA Analysis** − GAs have been used to determine the structure of DNA using spectrometric data about the sample.
- **Multimodal Optimization** − GAs are obviously very good approaches for multimodal optimization in which we have to find multiple optimum solutions.
- **Traveling salesman problem and its applications** − GAs have been used to solve the TSP, which is a well-known combinatorial problem using novel crossover and packing strategies.

basic terminology which will be used throughout this tutorial.

- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** − A chromosome is one such solution to the given problem.
- **Gene** − A gene is one element position of a chromosome.
- **Allele** − It is the value a gene takes for a particular chromosome.



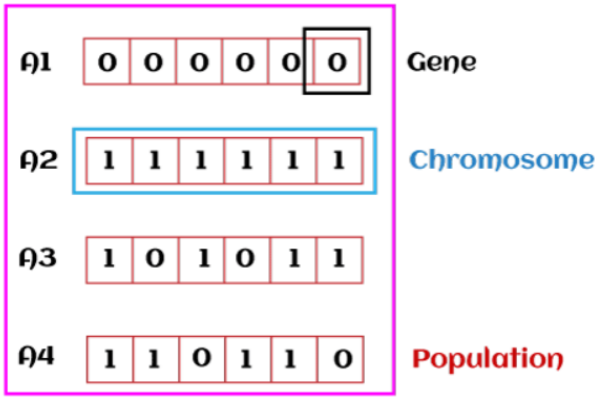
How Genetic Algorithm Work?

- The genetic algorithm works on the evolutionary generational cycle to generate high-quality solutions.
- These algorithms use different operations that either enhance or replace the population to give an improved fit solution.
- It basically involves five phases to solve the complex optimization problems,

- **Initialization**
- **Fitness Assignment**
- **Selection**
- **Reproduction**
- **Termination**

1. Initialization

- The process of a genetic algorithm starts by generating the set of individuals, which is called population.
- An individual contains or is characterized by a set of parameters called Genes.
- Genes are combined into a string and generate chromosomes, which is the solution to the problem.
- One of the most popular techniques for initialization is the use of random binary strings.



2. Fitness Assignment

- Fitness function is used to determine how fit an individual is.
- It means the ability of an individual to compete with other individuals.
- In every iteration, individuals are evaluated based on their fitness function.
- The fitness function provides a fitness score to each individual.
- This score further determines the probability of being selected for reproduction.

- The high the fitness score, the more chances of getting selected for reproduction.

3. Selection

- The selection phase involves the selection of individuals for the reproduction of offspring.
- All the selected individuals are then arranged in a pair of two to increase reproduction.
- Then these individuals transfer their genes to the next generation.

There are three types of Selection methods available, which are:

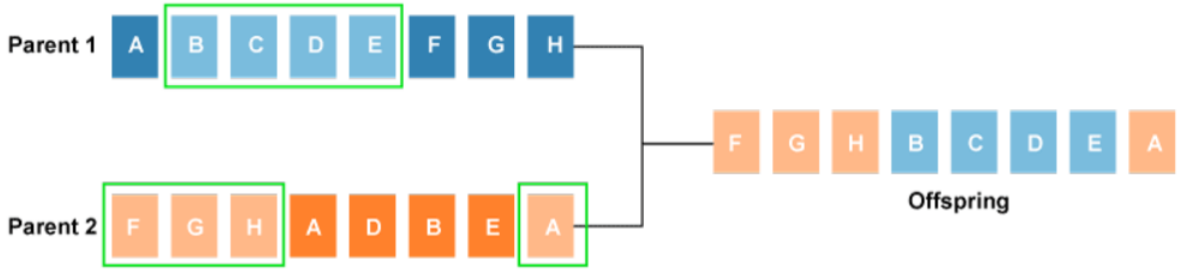
- Roulette wheel selection
- Tournament selection
- Rank-based selection

4. Reproduction

- After the selection process, the creation of a child occurs in the reproduction step.
- In this step, the genetic algorithm uses two variation operators that are applied to the parent population.
- The two operators involved in the reproduction phase are given below:

Crossover:

- The crossover plays a most significant role in the reproduction phase of the genetic algorithm.
- In this process, a crossover point is selected at random within the genes.
- Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing the offspring.



Types of crossover styles available:

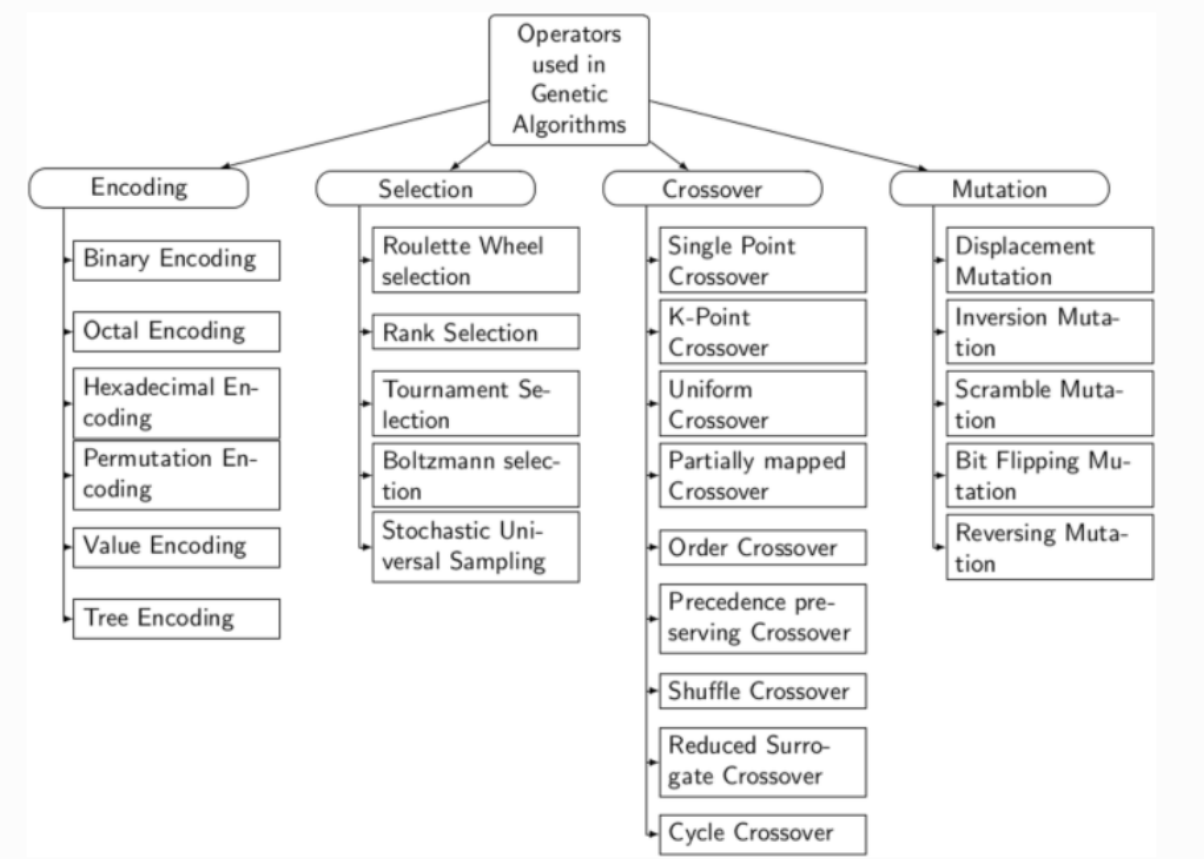
- One point crossover
- Two-point crossover
- Livery crossover
- Inheritable Algorithms crossover

- **Mutation**
  The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population.
- It can be done by flipping some bits in the chromosomes.
- Mutation helps in solving the issue of premature convergence and enhances diversification.
- The below image shows the mutation process:

  Types of mutation styles available
  - **Flip bit mutation**
  - **Gaussian mutation**
  - **Exchange/Swap mutation**

5. Termination

- After the reproduction phase, a stopping criterion is applied as a base for termination.
- The algorithm terminates after the threshold fitness solution is reached.
- It will identify the final solution as the best solution in the population.

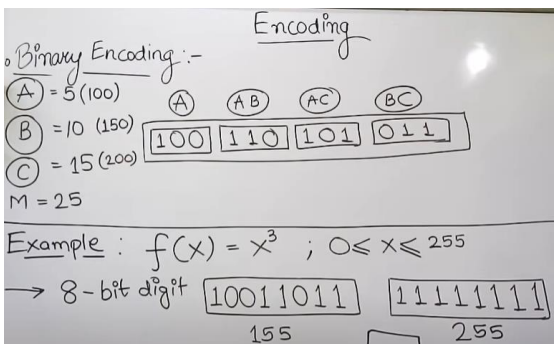


Binary encoding

- Most common method of encoding.
- Represent a gene in terms of bits (0s and 1s).
- Most suitable for optimization in a discrete search space.

Example of Problem: Knapsack problem

Chromosome A	10110010110011100101
Chromosome B	11111110000000011111



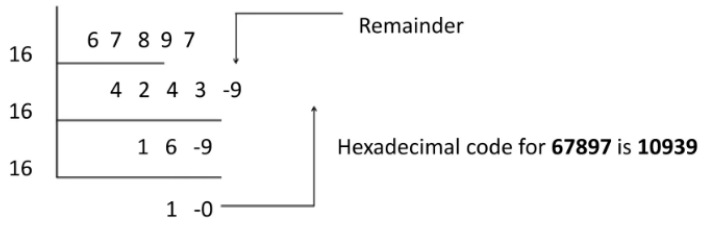
Octal Encoding:

In this encoding chromosome is represented using octal numbers (0-7).

Chromosome1	06254524
Chromosome2	63726425

Hexadecimal Encoding:

- Convert any numbers to hexadecimal form , we go on dividing the numbers by 16.
- The hexadecimal code for 67897 is shown to 10939.
- We get equivalent integer for the hexadecimal code by decoding. The decoded value for the hexadecimal number BO79E6 is 11565542



Chromosome1	97AE
Chromosome2	A2C6

Permutation Encoding:

1.Permutation Encoding is used in ordering problems. In this, each chromosome represents position in a sequence e.g. in travelling salesman problem, the string of numbers represent the sequence of cities visited by salesman.

2. Sometimes corrections have to be done after genetic operation is completed.

Chromosome1	1 5 2 3 5 2 6 4 6 9 8
Chromosome2	8 6 3 6 3 9 6 3 1 5 8

Value Encoding:

- In value encoding, each chromosome is represented as the string of some value. Value can be integer, real number, character or some object.
- In case of Integer values, the crossover operator applied are same as that applied on binary encoding.
- Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.

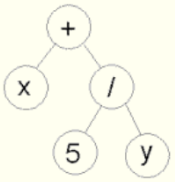
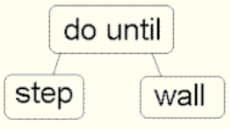
Chromosome1	1.23, 2.12, 3.14, 0.34, 4.62
Chromosome2	ABDJEIFJDHDDLDFLFEGT

- Value Encoding can be used in neural networks. This encoding is generally use in finding weights for neural network. Chromosome's value represents corresponding weights for inputs.

Tree Encoding

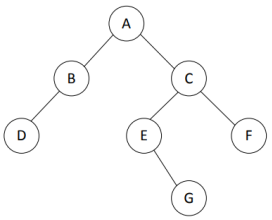
Tree encoding is used mainly for evolving programs or expressions, for **genetic programming**.

In **tree encoding** every chromosome is a tree of some objects, such as functions or commands in programming language.

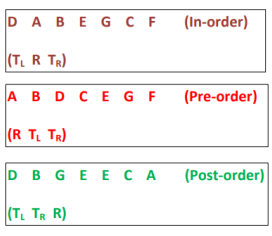
Chromosome A	Chromosome B
	
(+ x (/ 5 y))	(do_until step wall)

Tree encoding

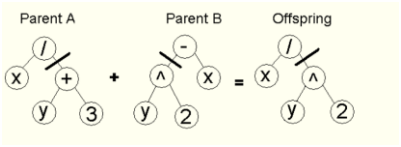
In this encoding scheme, a solution is encoded in the form of a binary tree.



A binary tree

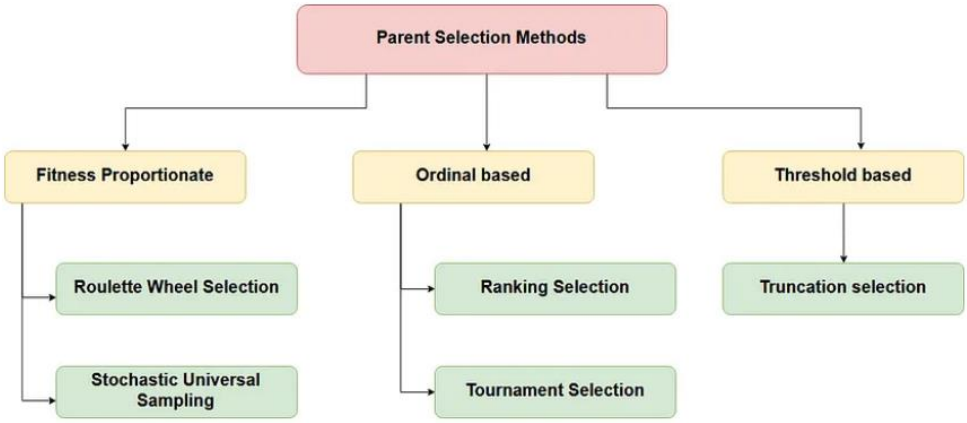


Three compact representation



Selection

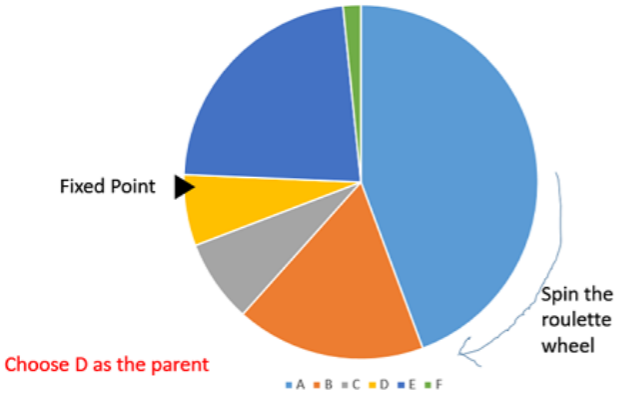
- Parent Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.
- Parent selection is crucial for the convergence rate of GAs.
- Good parents lead to fitter solutions.
- Diversity is essential for the success of GAs.
- Premature convergence occurs when one extremely fit solution dominates the population.
- Premature convergence is undesirable in GAs



Classification of Parent Selection Methods

Roulette Wheel Selection

- In a roulette wheel selection, the circular wheel is divided as described before.
- A fixed point is chosen on the wheel circumference as shown and the wheel is rotated.
- The region of the wheel which comes in front of the fixed point is chosen as the parent.
- For the second parent, the same process is repeated.



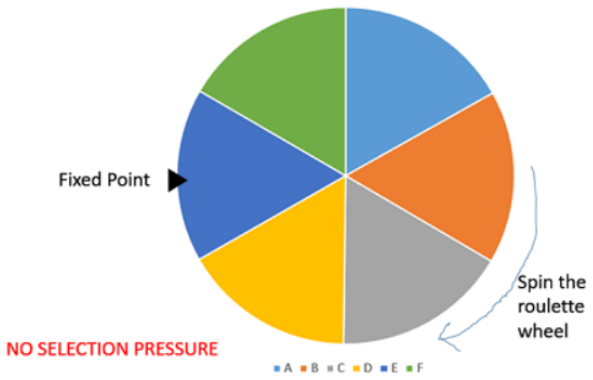
Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Rank Selection

Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values.

This leads to each individual having an almost equal share of the pie as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent.

This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

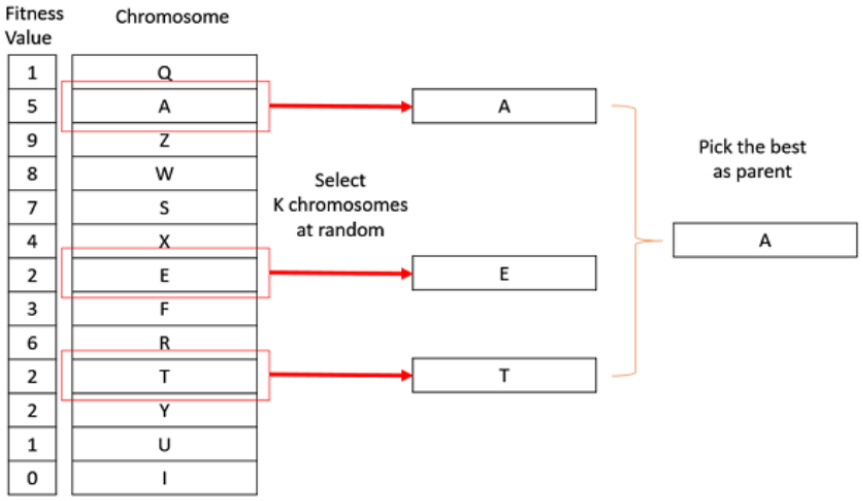
- In this, we remove the concept of a fitness value while selecting a parent.
- However, every individual in the population is ranked according to their fitness.
- The selection of the parents depends on the rank of each individual and not the fitness.

- The higher ranked individuals are preferred more than the lower ranked ones.

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

Tournament Selection

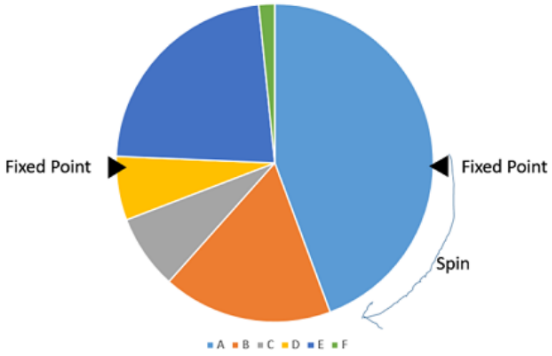
- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent.
- The same process is repeated for selecting the next parent.
- Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.



Stochastic Universal Sampling (SUS)

- Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points as shown in the following image.
- Therefore, all the parents are chosen in just one spin of the wheel.

- Also, such a setup encourages the highly fit individuals to be chosen at least once.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.

Crossover

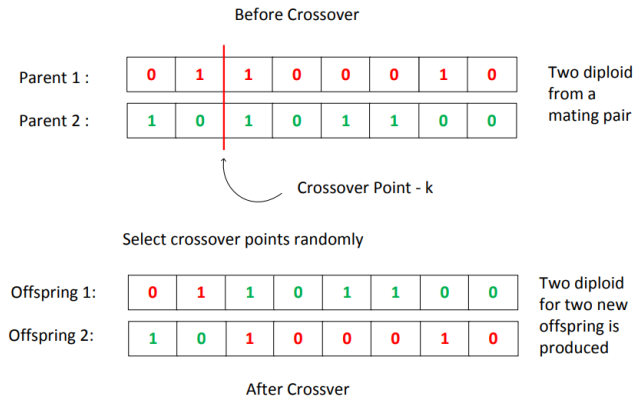
- The crossover operator is analogous to reproduction and biological crossover.
- Crossover is usually applied in a GA with a high probability – p_c .

Crossover operations in Binary-coded GAs

- There exists a large number of crossover schemes, few important of them are listed in the following.
  1. Single point crossover
  2. Two-point crossover
  3. Multi-point crossover (also called n-point crossover)
  4. Uniform crossover (UX)
  5. Half-uniform crossover (HUX)
  6. Shuffle crossover
  7. Matrix crossover (Tow-dimensional crossover)
  8. Three parent crossover

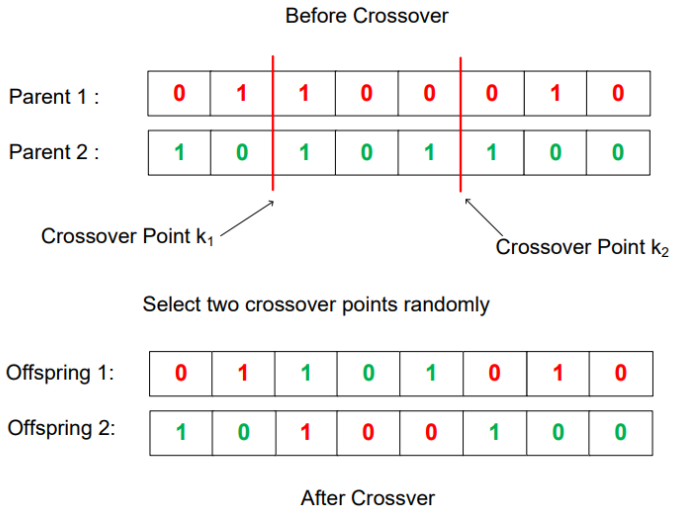
Single point crossover

1. Here, we select the K-point lying between 1 and L. Let it be k.
2. A single crossover point at k on both parent's strings is selected.
3. All data beyond that point in either string is swapped between the two parents.
4. The resulting strings are the chromosomes of the offsprings produced.



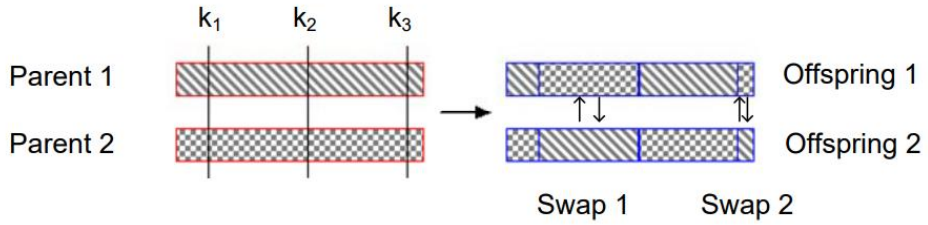
Two-point crossover

1. In this scheme, we select two different crossover points k_1 and k_2 lying between 1 and L at random such that $k_1 \neq k_2$.
2. The middle parts are swapped between the two strings.
3. Alternatively, left and right parts also can be swapped.



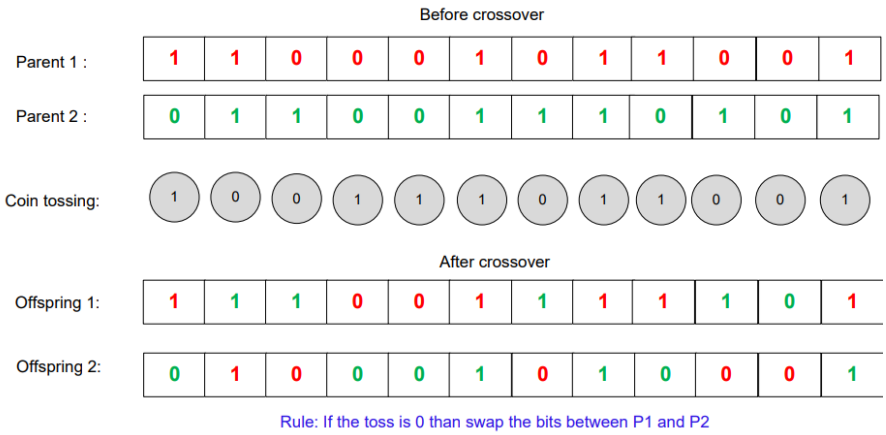
Multi-point crossover

1. In case of multi-point crossover, a number of crossover points are selected along the length of the string, at random.
2. The bits lying between alternate pairs of sites are then swapped.



Uniform Crossover (UX)

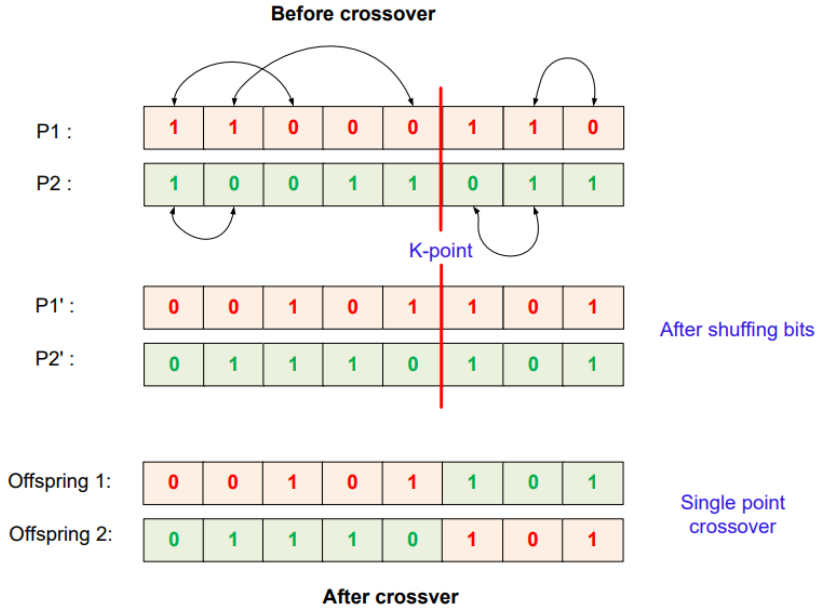
- Uniform crossover is a more general version of the multi-point crossover.
- In this scheme, at each bit position of the parent string, we toss a coin (with a certain probability p_s) to determine whether there will be swap of the bits or not.
- The two bits are then swapped or remain unaltered, accordingly.



Rule: If the toss is 0 than swap the bits between P1 and P2

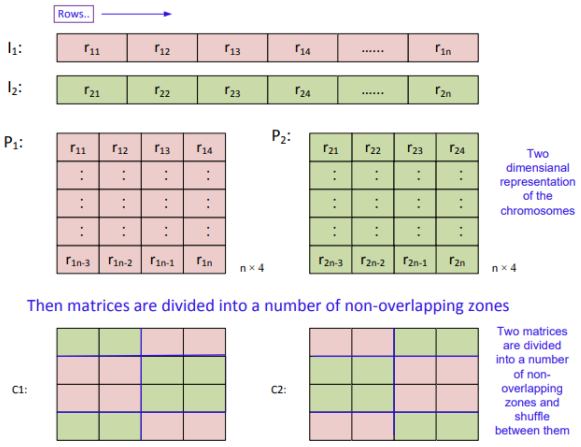
Shuffle crossover

- A single crossover point is selected. It divides a chromosome into two parts called schema.
- In both parents, genes are shuffled in each schema. Follow some strategy for shuflling bits
- Schemas are exchanged to create offspring (as in single crossover)



Matrix crossover

The matrix crossover strategy is expained with the following illustration.



Mutation Algorithm

Bit Flip Mutation —

Point	Description
Mutation operator	Bit flip mutation
Application	Mainly used for bit string manipulation
Representation	Chromosome represented as an array, each index representing a gene
String representation	Strings represented as an array of characters, which is an array of ASCII or numeric values
Process	1. Select one or more genes (array indices)

In bit flip mutation, we select one or more genes (array indices) and flip their values i.e. we change 1s to 0s and vice versa.

Swap Mutation —

In Swap Mutation we select two genes from our chromosome and interchange their values.

Scramble Mutation —

In Scramble Mutation we select a subset of our genes and scramble their value. The selected genes may not be contiguous (see the second diagram).

Inversion Mutation —

In Inversion Mutation we select a subset of our genes and reverse their order. The genes have to be contiguous in this case (see the diagram).

Random Resetting Mutation —

In random resetting mutation, we select one or more genes (array indices) and replace their values with another random value from their given ranges. Let's say a[i] (an array index / gene) ranges from [1, 6] then random resetting mutation will select one value from [1, 6] and replace a[i]'s value with it.