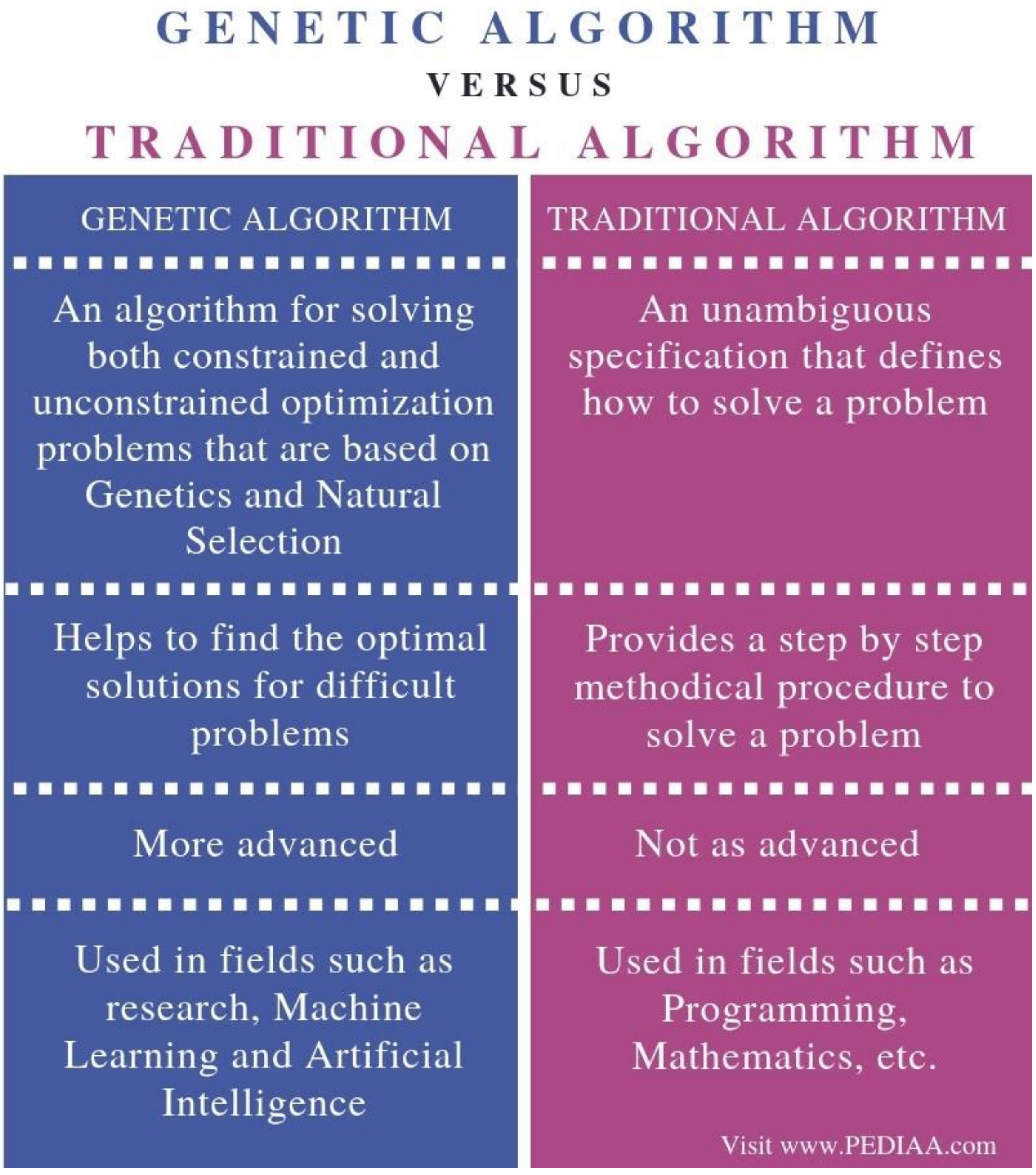
Genetic Algorithm

Biological Background

- **Genetics** is a branch of biology concerned with the study of **genes**, **genetic** variation, and heredity in organisms
- Genetics is the study of heredity.
- Heredity is a biological process where a parent passes certain genes onto their children or offspring.
- Every child inherits genes from both of their biological parents and these genes in turn express specific traits.
- Some of these traits may be physical for example hair and eye color and skin color etc.
- On the other hand some genes may also carry the risk of certain diseases and disorders that may pass on from parents to their offspring.
- Genetics is a field of biology that studies how traits are passed from parents to their offspring.
- The passing of traits from parents to offspring is known as heredity, therefore, genetics is the study of heredity.

Introduction

- Genetic 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**.
- **Genetic Algorithm** (GA) is a search-based optimization technique 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.
- GAs were developed by John Holland and his students and colleagues.
- In GAs, we have a **pool or a population of possible solutions** to the given problem.
- These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations.
- Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals.
- This is in line with the Darwinian Theory of "Survival of the Fittest".



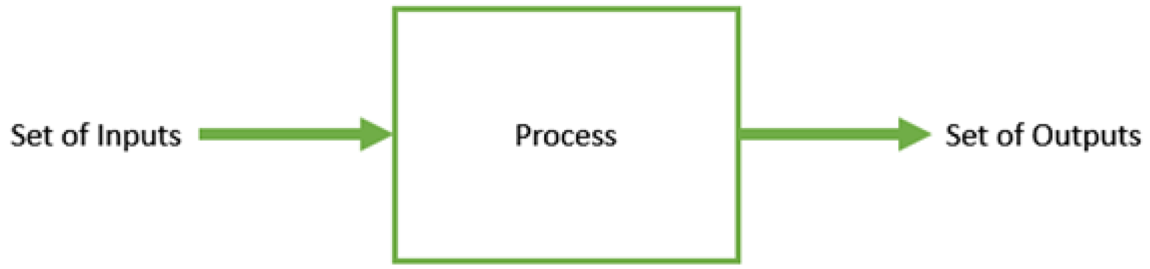
GENETIC ALGORITHM VERSUS TRADITIONAL ALGORITHM

GENETIC ALGORITHM	TRADITIONAL ALGORITHM
An algorithm for solving both constrained and unconstrained optimization problems that are based on Genetics and Natural Selection	An unambiguous specification that defines how to solve a problem
Helps to find the optimal solutions for difficult problems	Provides a step by step methodical procedure to solve a problem
More advanced	Not as advanced
Used in fields such as research, Machine Learning and Artificial Intelligence	Used in fields such as Programming, Mathematics, etc.

Visit www.PEDIAA.com

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.



Optimization refers to finding the values of inputs in such a way that we get the "best" output values.

The definition of "best" varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

Advantages of GAs

- Does not require any derivative information (which may not be available for many real-world problems).
- Is faster and more efficient as compared to the traditional methods.
- Has very good parallel capabilities.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of "good" solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

Limitations of GAs

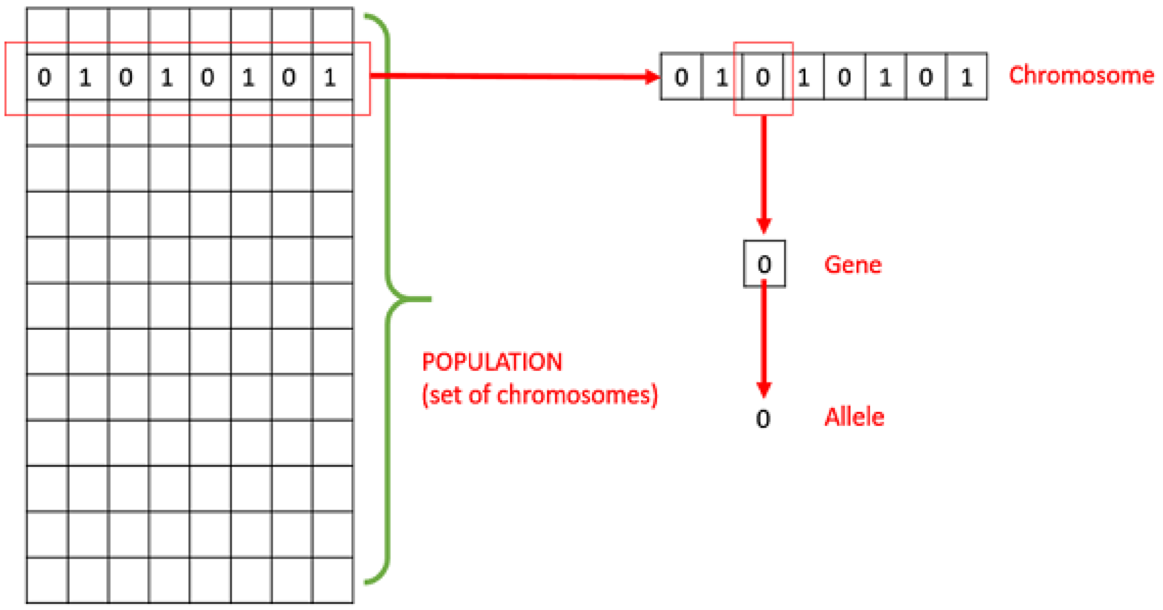
- GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.

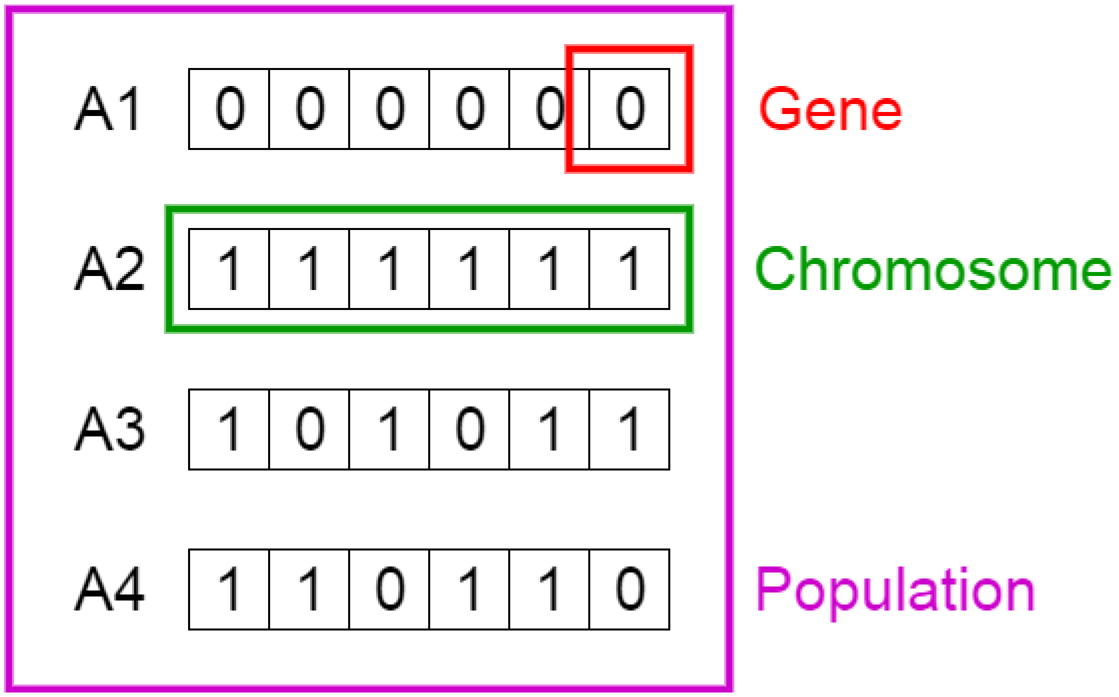
Search Space

- If we are solving some problem, we are usually looking for some solution, which will be the best among others.
- The space of all feasible solutions (it means objects among those the desired solution is) is called search space (also state space).
- Each point in the search space represent one feasible solution.
- Each feasible solution can be "marked" by its value or fitness for the problem.
- We are looking for our solution, which is one point (or more) among feasible solutions - that is one point in the search space.
- The looking for a solution is then equal to a looking for some extreme (minimum or maximum) in the search space.
- The search space can be whole known by the time of solving a problem, but usually we know only a few points from it and we are generating other points as the process of finding solution continues.
- The problem is that the search can be very complicated.
- One does not know where to look for the solution and where to start.
- There are many methods, how to find some suitable solution (ie. not necessarily the best solution), for example hill climbing, tabu search, simulated annealing and genetic algorithm.
- The solution found by this methods is often considered as a good solution, because it is not often possible to prove what is the real optimum.

Basic Terminology

- **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

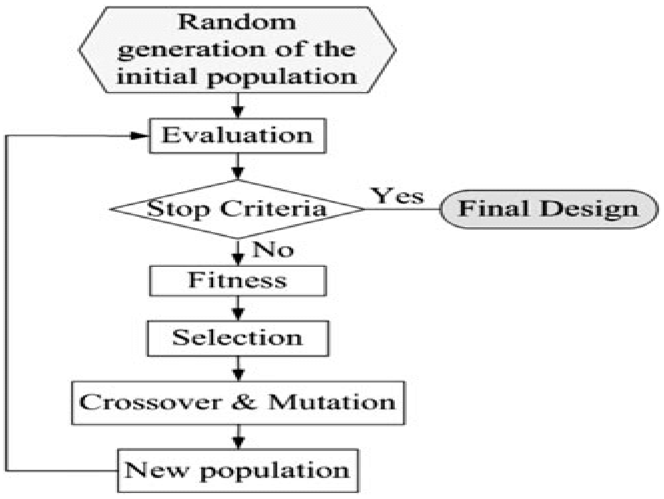




- **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
- **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the

phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

Simple Genetic Algorithm

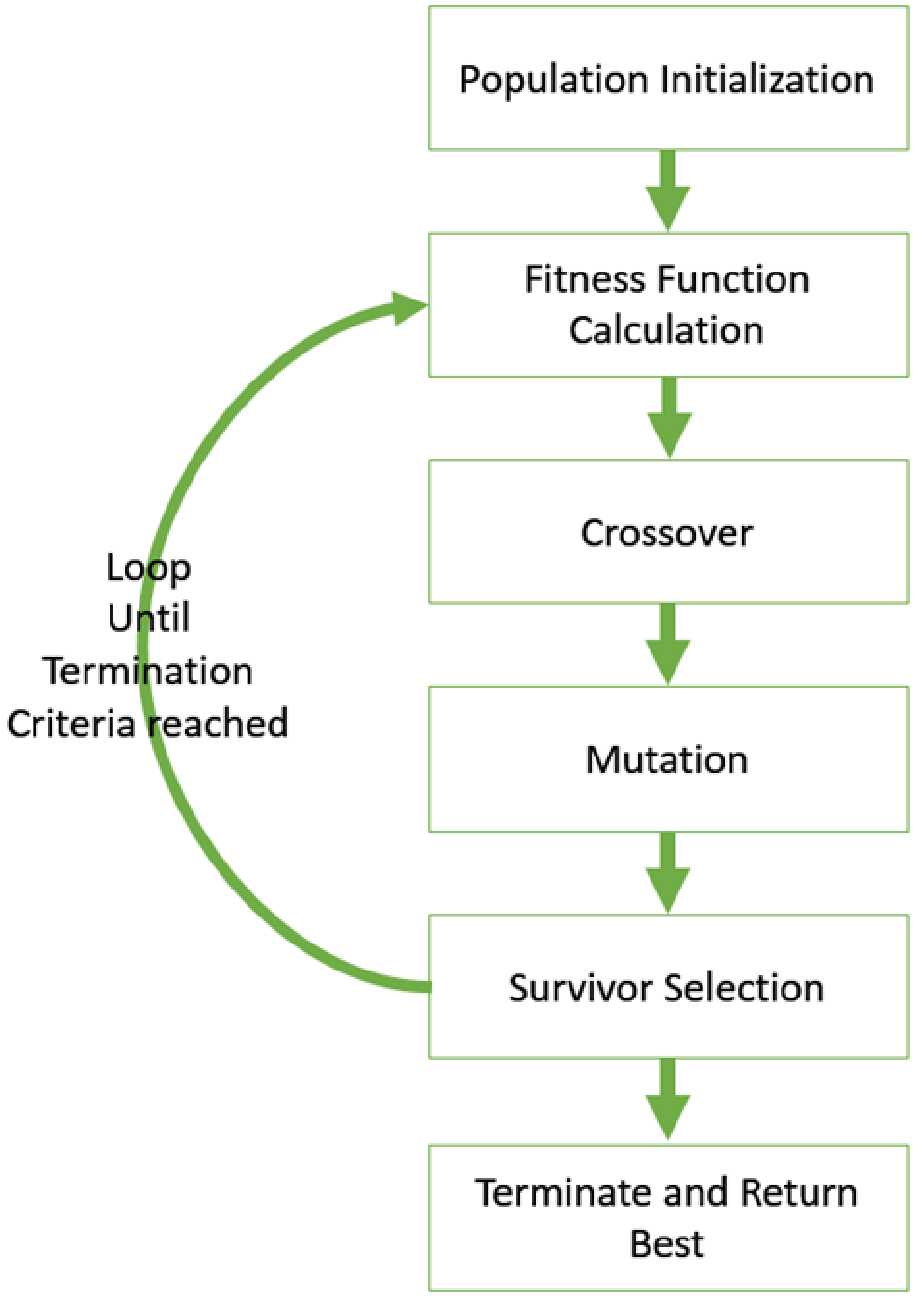


General Genetic Algorithm

- The basic structure of a GA is as follows
- We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating.
- Apply crossover and mutation operators on the parents to generate new off-springs.
- And finally these off-springs replace the existing individuals in the population and the process repeats.
- In this way genetic algorithms actually try to mimic the human evolution to some extent.

Five phases are considered in a genetic algorithm.

1. Initial population.
2. Fitness function.
3. Selection.
4. Crossover.
5. Mutation.


Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

A generalized pseudo-code for a GA is explained in the following program −

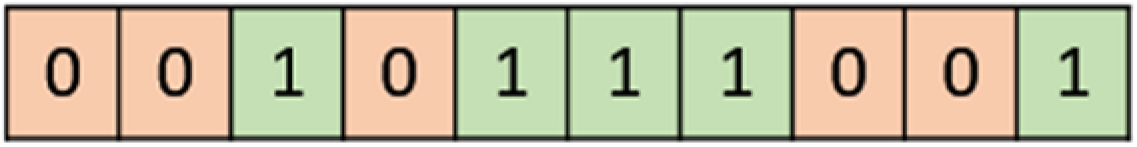
```
GA()
  initialize population
  find fitness of population

  while (termination criteria is reached) do
    parent selection
    crossover with probability pc
    mutation with probability pm
    decode and fitness calculation
    survivor selection
    find best
  return best
```

- One of the most important decisions to make while implementing a genetic algorithm is deciding the representation that we will use to represent our solutions.
- It has been observed that improper representation can lead to poor performance of the GA.
- Therefore, choosing a proper representation, having a proper definition of the mappings between the phenotype and genotype spaces is essential for the success of a GA.
- In this section, we present some of the most commonly used representations for genetic algorithms.
- However, representation is highly problem specific and the reader might find that another representation or a mix of the representations mentioned here might suit his/her problem better.

Binary Representation

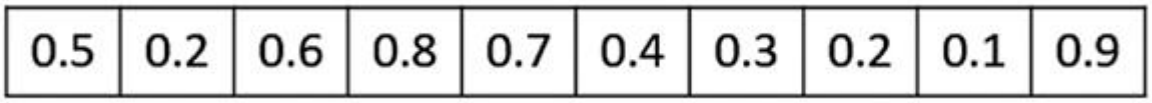
- This is one of the simplest and most widely used representation in GAs. In this type of representation the genotype consists of bit strings.



- For other problems, specifically those dealing with numbers, we can represent the numbers with their binary representation.

Real Valued Representation

- For problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the most natural. The precision of these real valued or floating point numbers is however limited to the computer.



Integer Representation

- For discrete valued genes, we cannot always limit the solution space to binary ‘yes’ or ‘no’. For example, if we want to encode the four distances – North, South, East and West, we can encode them as **{0,1,2,3}**. In such cases, integer representation is desirable.

1	2	3	4	3	2	4	1	2	1

Permutation Representation

- In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited.
- A classic example of this representation is the travelling salesman problem (TSP). In this the salesman has to take a tour of all the cities, visiting each city exactly once and come back to the starting city.
- The total distance of the tour has to be minimized.
- The solution to this TSP is naturally an ordering or permutation of all the cities and therefore using a permutation representation makes sense for this problem.

1	5	9	8	7	4	2	3	6	0

Operators in genetic algorithm

- A **genetic operator** is an **operator** used in **genetic algorithms** to guide the **algorithm** towards a solution to a given problem.
- There are three main types of **operators** (mutation, crossover and selection), which must work in conjunction with one another in order for the **algorithm** to be successful.

Selection Operator

- Selection operators give preference to better solutions (chromosomes), allowing them to pass on their 'genes' to the next generation of the algorithm.
- The best solutions are determined using some form of objective function (also known as a 'fitness function' in genetic algorithms), before being passed to the crossover operator.
- The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.

- The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.
- “Selects the best, discards the rest”

Functions of Selection operator

- Identify the good solutions in a population Make multiple copies of the good solutions
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population
- Now how to identify the good solutions?
- Fitness function
- A fitness value can be assigned to evaluate the solution.
- A fitness function value quantifies the optimality of a solution.
- The value is used to rank a particular solution against all the other solutions
- A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

Crossover Operator

- The crossover operator is used to create new solutions from the existing solutions available in the mating pool after applying selection operator.
- This operator exchanges the gene information between the solutions in the mating pool.
- Crossover is the process of taking more than one parent solutions (chromosomes) and producing a child solution from them.
- By recombining portions of good solutions, the genetic algorithm is more likely to create a better solution.

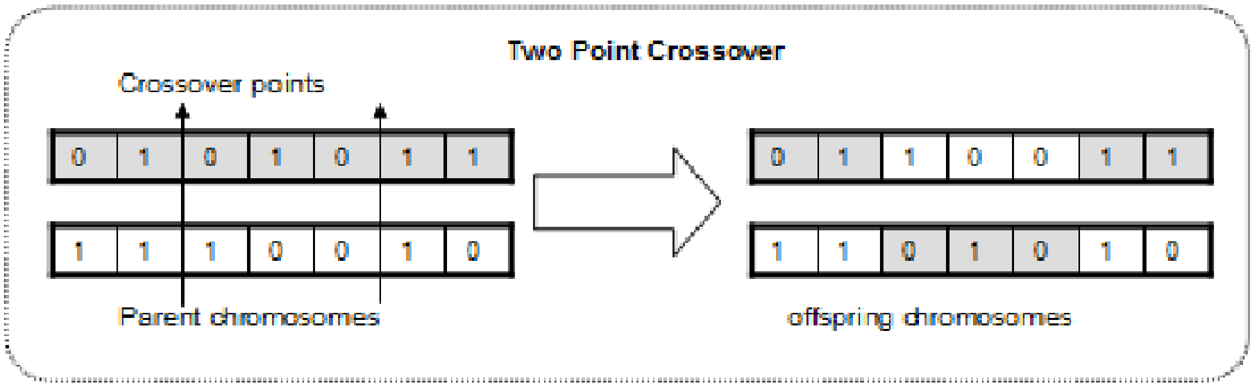
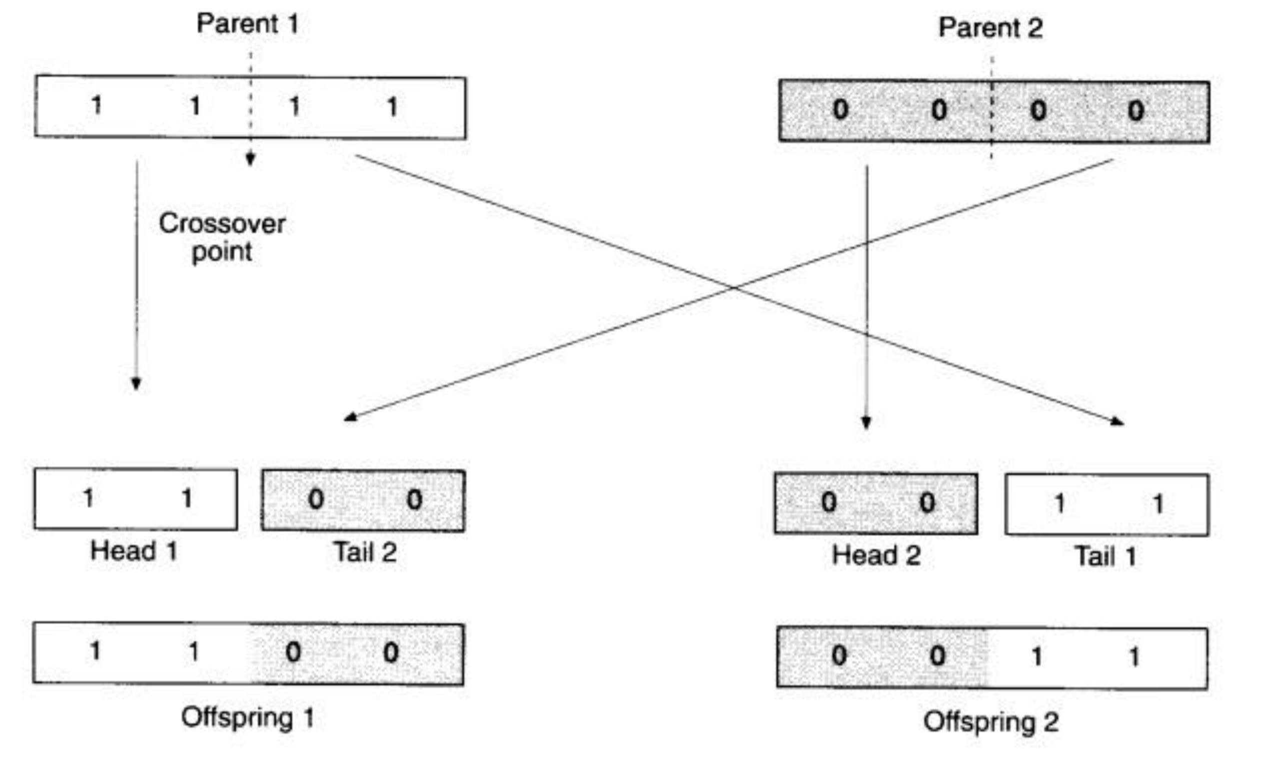
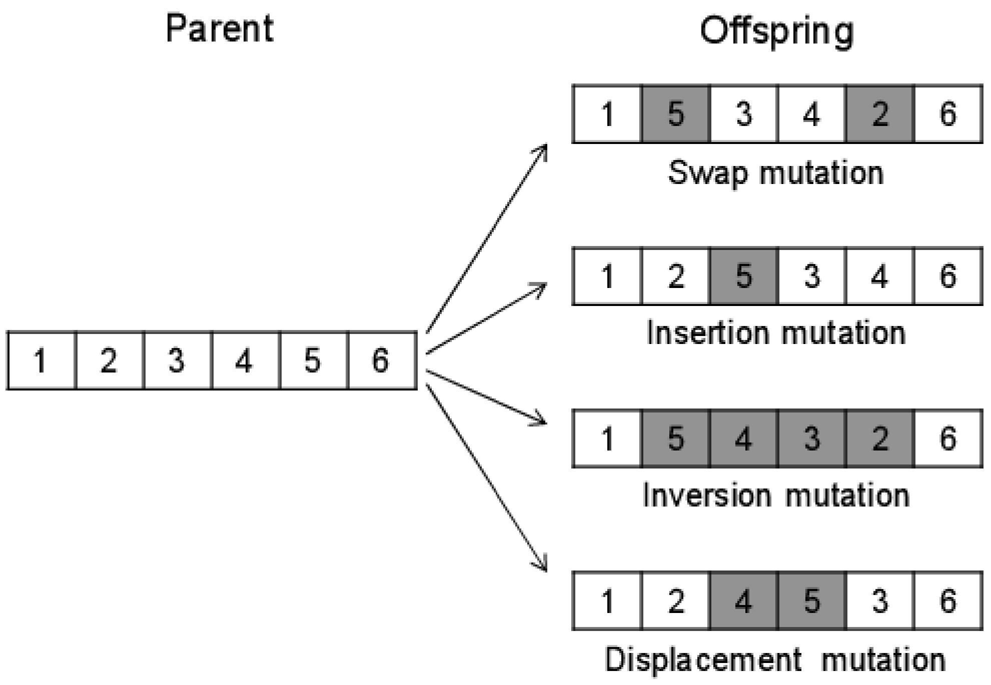


Figure 2. Two point crossover



Mutation Operator

- **Mutation** is a **genetic operator** used to maintain **genetic** diversity from one generation of a population of **genetic algorithm** chromosomes to the next.
- It is analogous to biological **mutation**. **Mutation** alters one or more **gene** values in a chromosome from its initial state
- In mutation, the solution may change entirely from the previous solution. Hence GA can come to a better solution by using mutation.



<u>Stopping condition for genetic algorithm flow</u>

Stopping Conditions for the Algorithm

The genetic algorithm uses the following conditions to determine when to stop:

- **Generations** — The algorithm stops when the number of generations reaches the value of **Generations**.
- **Time limit** — The algorithm stops after running for an amount of time in seconds equal to **Time limit**.
- **Fitness limit** — The algorithm stops when the value of the fitness function for the best point in the current population is less than or equal to **Fitness limit**.