**Genetic Algorithm**

**Introduction to Genetic Algorithm:-**

Genetic Algorithm (GA) is a search-based optimization technique(exploring a solution space to find the best solution to a problem) based on the principles of Genetics and Natural Selection. Natural selection is a fundamental concept in evolutionary biology. It refers to the process by which organisms with traits that are better adapted to their environment tend to survive and reproduce more successfully than those with less favorable traits. 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.A genetic algorithm is an adaptive heuristic search algorithm.

**Adaptive heuristic search algorithm:**

Adaptive Heuristic Search (AHS) is a search technique that dynamically adjusts the heuristics(rules) used during the search process to improve the efficiency of the search. The main idea of AHS is to use the information gathered during the search process to guide the search towards a better solution.

AHS uses the Euclidean and Manhattan distance in order to get optimize solution for the problem.

**Terminology:**

**Population:** Population is the subset of all possible or probable solutions, which can solve the given problem.

**Chromosomes:** A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.

**Gene:** A chromosome is divided into a different gene, or it is an element of the chromosome.

**Allele:** Allele is the value provided to the gene within a particular chromosome.

**Fitness Function:** The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.

**Genetic Operators:** In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.

**Selection:** Selection process is used to determine which of the individualities in the population will get to reproduce and produce the seed that will form the coming generation.

**Working Principle:**

1. Evaluation: Evaluate the fitness of each chromosome in the population, typically by assigning a score that represents how well the chromosome solves the problem.
2. Selection: Select the fittest chromosomes from the current population to form the mating pool for the next generation.
3. Crossover: Generate new offspring from the mating pool by combining genetic information from parent chromosomes through a process called crossover.
4. Mutation: Introduce small, random changes to the offspring's genetic information through a process called mutation.
5. Repeat: Evaluate the fitness of the new offspring and repeat the selection, crossover, and mutation processes to form the next generation of the population.
6. Termination: Continue the process until a satisfactory solution is found, or until a pre-defined stopping criteria, such as a maximum number of generations, is reached.**Bottom of Form**

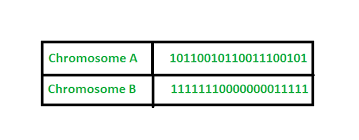
**Encoding :**

Encoding of chromosomes is the first step in solving the problem and it depends entirely on the problem heavily. The process of representing the solution in the form of a string of bits that conveys the necessary information. just as in a chromosome, each gene controls particular characteristics of the individual, similarly, each bit in the string represents characteristics of the solution.

**Encoding Methods or Schemes:**

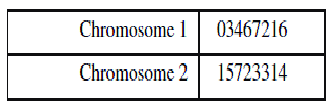
**Binary Encoding:**

This is the most common form of encoding. In this encoding each chromosome is represented using a binary string. In binary encoding every chromosome is a string of bits, 0 or 1 . In this encoding each bit show some characteristic of solution. On the other side each binary string represents a value. With smaller number of alleles, a number of chromosomes can be represented. Crossover operations possible in binary encoding are 1-point crossover, Npoint crossover, Uniform crossover and Arithmetic crossover. The Mutation operator possible is Flip. In Flip mutation, bits changes from 0 to 1 and 1 to 0 based on generated mutation chromosome. This is generally use in Knapsack problem where binary encoding is used to show the presence of items say 1 to denote the presence of a item and 0 to the absence of a item.

****

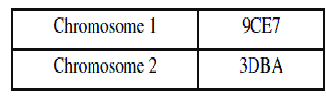
**Octal Encoding:**

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



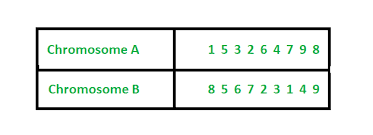
**Hexadecimal Encoding**:

In this encoding chromosome is represented using Hexadecimal numbers (0-9, A-F).



**Permutation Encoding:**

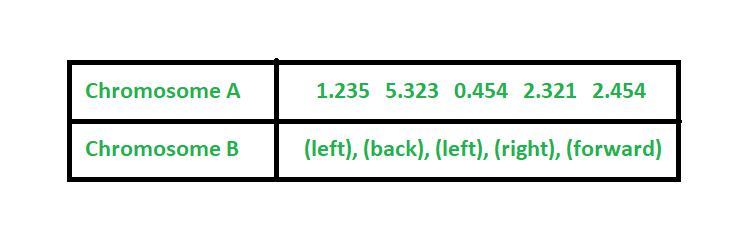
Permutation Encoding is used in ordering problems. In this, each chromosome represents position in a sequence e.g. in travelling salesman problem, the string of numbers represent the sequence of cities visited by salesman. Sometimes corrections have to be done after genetic operation is completed.

****

Permutation encoding is only useful for problems that have specific order. Some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e., have real sequence in it) for such kind of problems. Crossover operators performed on permutation encoding are Partially mapped crossover (PMX), Cycle crossover (OCX) and Order crossover (OX). Inversion is the most commonly used mutation operator applied on ordered chromosomes. It changes the location of characters.

**Value Encoding:**

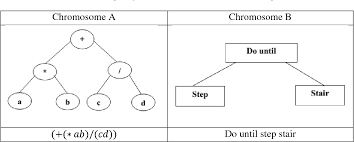
In value encoding, each chromosome is represented as the string of some value. Value can be integer, real number, character or some object. In case of Integer values, the crossover operator applied are same as that applied on binary encoding. Values can be anything connected to problem, form numbers,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



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 mainly used 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. Tree encoding is good for evolving programs. LISP is useful in this encoding as it helps in constructing tree for parsing and hence the crossover and mutation can be performed easily. Chromosomes are functions represented in a tree.



**Fitness Function:**

A fitness function, also known as an objective function, is a mathematical evaluation used to determine the quality of a solution in optimization and machine learning problems. It is a function that takes one or more parameters as input and outputs a score indicating how well the input meets the desired criteria. The goal of an optimization problem is to find the inputs that maximize (or minimize) the fitness function. The fitness function is the core component of genetic algorithms, evolutionary algorithms, and other optimization techniques.

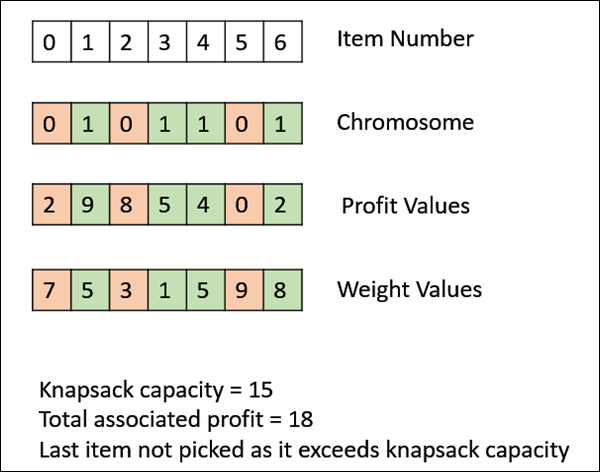
Fitness Function evaluates how close a given solution is to the optimum solution of the desired problem. It determines how fit a solution is.

Example:-

1. Traveling Salesman Problem: Given a set of cities and distances between each city, a fitness function could be the total distance traveled by the salesman to visit all cities and return to the starting city. The goal is to find the shortest possible route, so the fitness function would be minimized.

**Example-2:-**

**0/1 Knapsack Problem**



**Reproduction:-**

Reproduction is a key component of genetic algorithms, which is a type of optimization algorithm inspired by the process of natural selection and evolution.

In a genetic algorithm, reproduction refers to the process of creating a new generation of candidate solutions (also known as individuals or chromosomes) based on the current generation of solutions. The new generation is created by combining the genetic information (also known as genes) of the existing solutions in a way that simulates the process of sexual reproduction in biological organisms.

There are two main steps in the reproduction process: selection and crossover.

1. Selection: In this step, the existing solutions are selected based on their fitness score, which is a measure of how well the solution meets the desired criteria. The solutions with the highest fitness scores are more likely to be selected for reproduction.
2. Crossover: In this step, the selected solutions are combined to form a new generation of solutions. This is done by selecting a random point in each solution and swapping the genetic information between the two solutions. The result is a new solution that contains a combination of the genetic information from both parent solutions.

This process of selection and crossover is repeated for each new generation, and the goal is to produce solutions with higher fitness scores over time. In addition to crossover, other genetic operators such as mutation (randomly changing a small portion of the genetic information) and elitism (preserving the best solution from one generation to the next) can also be used to help drive the evolution of the solutions.

**Genetic Modelling:**

Genetic modelling in soft computing refers to the use of genetic algorithms and other computational methods inspired by biological evolution to solve complex problems in computer science, engineering, and other fields. Soft computing refers to a family of computational methods that are based on the principles of fuzzy logic, neural networks, evolutionary computation, and probabilistic reasoning, and are used to solve problems that are too complex or ill-defined for conventional algorithms.

GAs have been used in a wide range of pattern recognition problems, including image classification, speech recognition, and character recognition. The key advantages of using GAs in pattern recognition include their ability to handle complex and noisy data, their ability to find global solutions, and their ability to handle multiple conflicting objectives.

**Inheritance operator:**

The inheritance operator is a fundamental component of a genetic algorithm (GA). It is responsible for transferring information from one generation of solutions to the next. The inheritance operator helps to preserve good solutions from one generation to the next and to gradually improve the quality of the solutions over time.

There are two main types of inheritance operators in GAs:

Selection: The selection operator is responsible for selecting the best solutions from the current population to form the next generation. This is typically done by evaluating each solution's fitness and selecting the solutions with the highest fitness to be passed on to the next generation. There are several selection methods that can be used, including roulette wheel selection, tournament selection, and rank-based selection.

Crossover: The crossover operator is responsible for generating new solutions by combining the genetic information of two or more parent solutions. The crossover operator helps to create new, potentially better solutions by combining the best features of different solutions. There are several crossover methods that can be used, including single-point crossover, multi-point crossover, and uniform crossover.

In addition to these two main operators, the inheritance operator may also include mutation, which is responsible for introducing random changes into the genetic information of a solution. Mutation helps to avoid getting stuck in local optima and to explore the solution space more effectively.

The choice of inheritance operators and their parameters can have a significant impact on the performance of a GA. The best set of inheritance operators for a particular problem can be determined through experimentation and analysis.

**Mutation Operator:**

The mutation operator is a fundamental component of a genetic algorithm (GA). It is responsible for introducing random changes into the genetic information of a solution. The mutation operator helps to maintain genetic diversity in the population and to explore the solution space more effectively.

Bit Flip Mutation

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

Bit Flip Mutation

Random Resetting

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

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.

Swap Mutation

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.

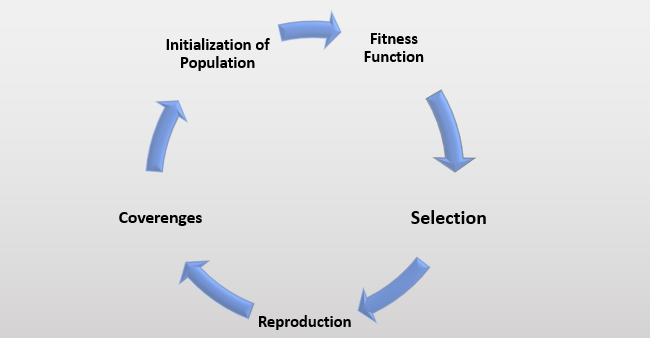
Scramble Mutation

Inversion Mutation

In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

Inversion Mutation

**Generational Cycle:**



The generational cycle is a key aspect of a genetic algorithm (GA) that refers to the sequence of steps that are performed in each generation of the algorithm. The generational cycle is repeated until a stopping criterion is met, such as finding a solution of sufficient quality or reaching a specified number of generations.

The generational cycle typically includes the following steps:

Evaluation: Evaluate the fitness of each individual in the population. The fitness function is used to measure the quality of each individual and determine its relative importance in the population.

Selection: Select individuals from the population to form the next generation. Selection operators, such as tournament selection or roulette wheel selection, are used to preserve high-quality individuals and gradually improve the quality of the population over time.

Crossover: Generate new individuals by combining the genetic information of two or more parent individuals. Crossover operators, such as single-point crossover or uniform crossover, are used to create new, potentially better individuals by combining the best features of different individuals.

Mutation: Introduce random changes into the genetic information of an individual. Mutation helps to maintain genetic diversity in the population and to explore the solution space more effectively.

Replacement: Replace individuals in the population with the new individuals generated through selection, crossover, and mutation. This updates the population of individuals and prepares it for the next generation.

The generational cycle is repeated iteratively to generate new solutions and update the population of solutions. The efficiency and effectiveness of a GA depend on the design of the fitness function, the choice of inheritance operators, and the choice of other key parameters such as the population size and mutation rate.

**Convergance of GA:**

Convergence in a genetic algorithm (GA) refers to the process where the population of solutions becomes more homogeneous and stable over time, as the quality of the solutions improves. When convergence occurs, the GA reaches a state where the best solution found so far remains unchanged for several generations.

The convergence of a GA can be influenced by several factors, including the design of the fitness function, the choice of inheritance operators, the size of the population, and the mutation rate. A well-designed GA should converge towards a high-quality solution that is close to the optimal solution for the problem being solved.

There are two types of convergence in a GA: weak convergence and strong convergence. Weak convergence refers to the convergence of the average fitness of the population over time. Strong convergence refers to the convergence of the best solution found so far over time. In general, it is more difficult to achieve strong convergence, as it requires the GA to find a globally optimal solution for the problem being solved.

Convergence is an important property of a GA, as it allows the algorithm to avoid getting stuck in a suboptimal solution and to find high-quality solutions in a reasonable amount of time. However, it is also important to avoid premature convergence, where the GA converges to a suboptimal solution before reaching the globally optimal solution. This can be achieved by using appropriate parameter settings, such as a larger population size or a higher mutation rate.