MACHINE LEARNING

ASSIGNMENT:2

Aditya Singh Parihar

Roll No.-2001650100009

B-tech

Computer Science

[A]

3 year

Question 1-Define genetic Algorithm With Numerical.

A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature." It is used to solve optimization

problems in machine learning. It is one of the important algorithms as it helps solve complex problems that would take a long time to solve.

Genetic Algorithms are being widely used in different real-world applications, for example, *Designing electronic circuits, code-breaking, image processing, and artificial creativity.*

In this topic, we will explain Genetic algorithm in detail, including basic terminologies used in Genetic algorithm, how it works, advantages and limitations of genetic algorithm, etc.

What is a Genetic Algorithm?

- **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**

Types of selection styles available

:Roulette wheel selection

:Event selection

:Rank- grounded selection

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, which are given as below:

How Genetic Algorithm Work?

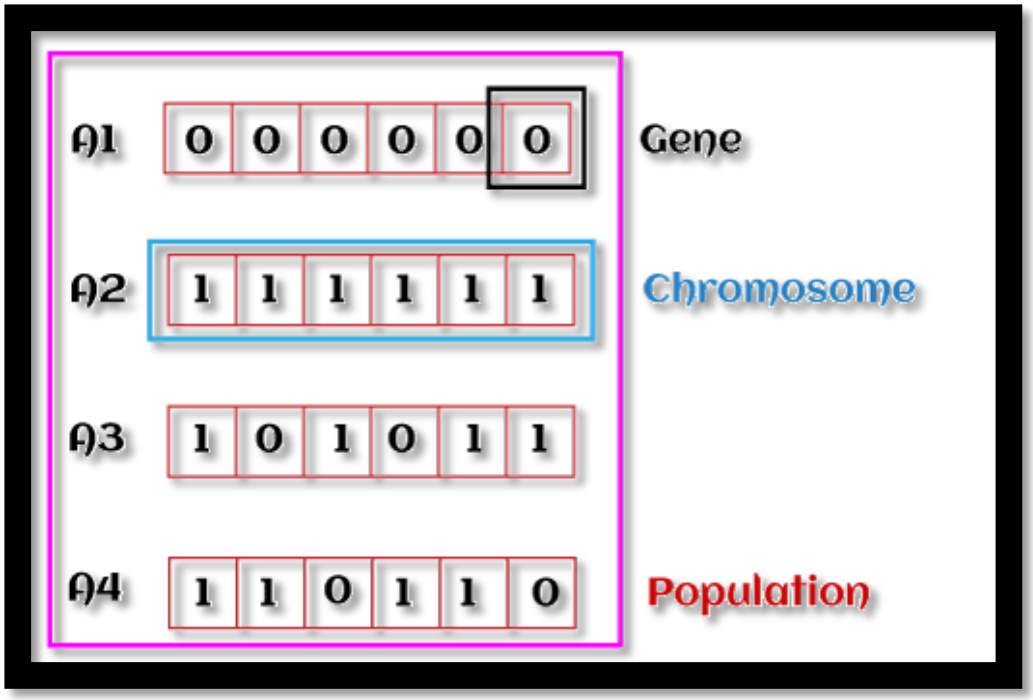
- ***Initialization***
- ***Fitness Assignment***
- ***Selection***

Reproduction 。

Termination

1. Initialization

The process of a genetic algorithm starts by generating the set of individuals, which is called population. Here each individual is the solution for the given problem. 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. 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.

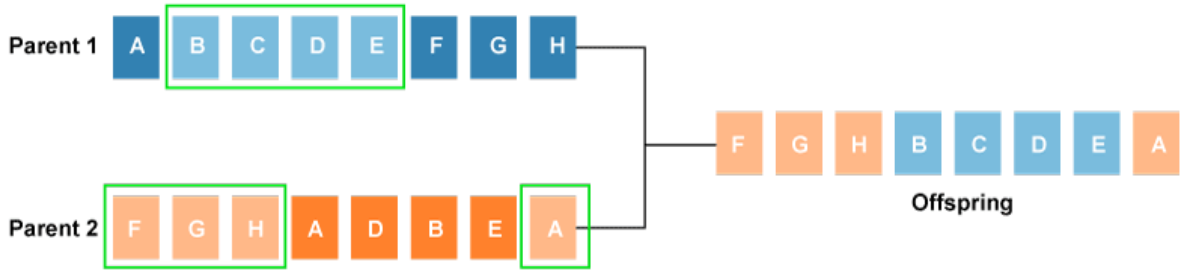
3. Selection

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. 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.



The genes of parents are exchanged among themselves until the crossover point is met. These newly

generated offspring are added to the population. This process is also called or crossover. Types of crossover styles available: ◦ One point crossover ◦ Two-point crossover ◦ Livery crossover ◦ Inheritable Algorithms crossover

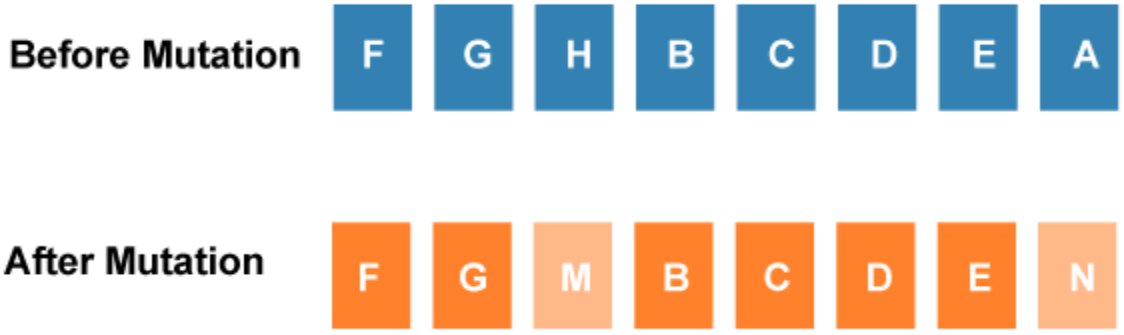
- **Mutation**

  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.

Numerical Example

Here are examples of applications that use genetic algorithms to solve the problem of combination. Suppose there is equality a + 2b + 3c + 4d = 30, genetic algorithm will be used to find the value of a, b, c, and d that satisfy the above equation. First we should formulate the objective function, for this problem the objective is minimizing the value of function f(x) where f(x) = ((a + 2b + 3c + 4d) - 30). Since there are four variables in the equation, namely a, b, c, and d,

we can compose the chromosome as follow: To speed up the computation, we can restrict that the values of variables a, b, c, and d are integers between 0 and 30.

SOLUTION:

Step 1. Initialization

For example we define the number of chromosomes in population are 6, then we

generate random value of gene a, b, c, d
for 6 chromosomes

Chromosome[1] = [a;b;c;d] = [12;05;23;08]

Chromosome[2] = [a;b;c;d] = [02;21;18;03]

Chromosome[3] = [a;b;c;d] = [10;04;13;14]

Chromosome[4] = [a;b;c;d] = [20;01;10;06]

Chromosome[5] = [a;b;c;d] = [01;04;13;19]

Chromosome[6] = [a;b;c;d] = [20;05;17;01]

Step 2. Evaluation

We compute the objective function value for each chromosome produced in initialization

step:

F_obj[1] = Abs((12 + 2*05 + 3*23 + 4*08) - 30)

= Abs((12 + 10 + 69 + 32) - 30)

= Abs(123 - 30)

= 93

F_obj[2] = Abs((02 + 2*21 + 3*18 + 4*03) - 30)

= Abs((02 + 42 + 54 + 12) - 30)

= Abs(110 - 30)

= 80

F_obj[3] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)

= Abs((10 + 08 + 39 + 56) - 30)

= Abs(113 - 30)

= 83

F_obj[4] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)

= Abs((20 + 02 + 30 + 24) - 30)

= Abs(76 - 30)

= 46

F_obj[5] = Abs((01 + 2*04 + 3*13 + 4*19) - 30)

= Abs((01 + 08 + 39 + 76) - 30)

= Abs(124 - 30)

= 94

F_obj[6] = Abs((20 + 2*05 + 3*17 + 4*01) - 30)

= Abs((20 + 10 + 51 + 04) - 30)

a b c d = Abs(85 - 30)

= 55

Step 3. Selection

1. The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. To avoid divide by zero problem, the value of F_obj is added by 1.

Fitness[1] = 1 / (1+F_obj[1])

= 1 / 94

= 0.0106

Fitness[2] = 1 / (1+F_obj[2])

= 1 / 81

= 0.0123

Fitness[3] = 1 / (1+F_obj[3])

= 1 / 84

= 0.0119

Fitness[4] = 1 / (1+F_obj[4])

= 1 / 47

= 0.0213

Fitness[5] = 1 / (1+F_obj[5])

= 1 / 95

= 0.0105

Fitness[6] = 1 / (1+F_obj[6])

= 1 / 56

= 0.0179

Total = 0.0106 + 0.0123 + 0.0119 + 0.0213 + 0.0105 + 0.0179

= 0.0845

The probability for each chromosomes is formulated by: P[i] = Fitness[i] / Total

P[1] = 0.0106 / 0.0845

= 0.1254

P[2] = 0.0123 / 0.0845

= 0.1456

P[3] = 0.0119 / 0.0845

= 0.1408

P[4] = 0.0213 / 0.0845

= 0.2521

P[5] = 0.0105 / 0.0845

= 0.1243

P[6] = 0.0179 / 0.0845

= 0.2118

From the probabilities above we can see that Chromosome 4 that has the highest fitness, this chromosome has highest probability to be selected for next generation chromosomes. For the selection process we use roulette wheel, for that we should compute the cumulative probability values:

C[1] = 0.1254

C[2] = 0.1254 + 0.1456

= 0.2710

C[3] = 0.1254 + 0.1456 + 0.1408

= 0.4118

C[4] = 0.1254 + 0.1456 + 0.1408 + 0.2521

= 0.6639

C[5] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243

= 0.7882

C[6] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243 + 0.2118

= 1.0

Having calculated the cumulative probability of selection process using roulette-wheel can be done. The process is to generate random number R in the range 0-1 as follows.

R[1] = 0.201

R[2] = 0.284

R[3] = 0.099

R[4] = 0.822

R[5] = 0.398

R[6] = 0.501

If random number R[1] is greater than C[1] and smaller than C[2] then select

Chromosome[2] as a chromosome in the new population for next generation:

NewChromosome[1] = Chromosome[2]

NewChromosome[2] = Chromosome[3]

NewChromosome[3] = Chromosome[1]

NewChromosome[4] = Chromosome[6]

NewChromosome[5] = Chromosome[3]

NewChromosome[6] = Chromosome[4]

Chromosomes in the population thus became:

Chromosome[1] = [02;21;18;03]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;08]

Chromosome[4] = [20;05;17;01]

Chromosome[5] = [10;04;13;14]

Chromosome[6] = [20;01;10;06]

In this example, we use one-cut point, i.e. randomly select a position in the parent chromosome then exchanging sub-chromosome. Parent chromosome which will mate is randomly selected and the number of mate Chromosomes is controlled using crossover_rate

(ρc) parameters. Pseudo-code for the crossover process is as follows:

begin

k← 0;

while(k<population) do R[k] =

random(0-1); if(R[k]< ρc) then

select Chromosome[k] as parent;

end; k = k + 1; end; end;

Chromosome k will be selected as a parent if R[k]<ρc. Suppose we set that the crossover rate is 25%, then Chromosome number k will be selected for crossover if random generated value for Chromosome k below 0.25. The process is as follows: First we generate a random number R as the number of population.

R[1] = 0.191

R[2] = 0.259

R[3] = 0.760

R[4] = 0.006

R[5] = 0.159

R[6] = 0.340

For random number R above, parents are Chromosome[1], Chromosome[4] and

Chromosome[5] will be selected for crossover.

Chromosome[1] >< Chromosome[4]

Chromosome[4] >< Chromosome[5]

Chromosome[5] >< Chromosome[1]

After chromosome selection, the next process is determining the position of the crossover

point. This is done by generating random numbers between 1 to (length of Chromosome – 1).

In this case, generated random numbers should be between 1 and 3. After we get the crossover point, parents Chromosome will be cut at crossover point and its gens will be interchanged. For example we generated 3 random number and we get:

C[1] = 1

C[2] = 1

C[3] = 2

Then for first crossover, second crossover and third crossover, parent's gens will be cut at gen number 1, gen number 1 and gen number 3 respectively, e.g.

Chromosome[1] = Chromosome[1] >< Chromosome[4]

= [02;21;18;03] >< [20;05;17;01]

= [02;05;17;01]

Chromosome[4] = Chromosome[4] >< Chromosome[5]

= [20;05;17;01] >< [10;04;13;14]

= [20;04;13;14]

Chromosome[5] = Chromosome[5] >< Chromosome[1]

= [10;04;13;14] >< [02;21;18;03]

= [10;04;18;03]

Thus Chromosome population after experiencing a crossover process:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;08]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;04;18;03]

Chromosome[6] = [20;01;10;06]

Step 5. Mutation

Number of chromosomes that have mutations in a population is determined by the

mutation_rate parameter. Mutation process is done by replacing the gen at random position with a new value. The process is as follows. First we must calculate the total length of gen in the population. In this case the total length of gen is total_gen =

number_of_gen_in_Chromosome * number of population

= 4 * 6

= 24

Mutation process is done by generating a random integer between 1 and total_gen (1 to 24).

If generated random number is smaller than mutation_rate(ρm) variable then marked the position of gen in chromosomes. Suppose we define ρm 10%, it is expected that 10% (0.1) of total_gen in the population that will be mutated:

number of mutations = 0.1 * 24

= 2.4

≈ 2

Suppose generation of random number yield 12 and 18 then the chromosome which have mutation are Chromosome number 3 gen number 4 and Chromosome 5 gen number 2. The value of mutated gens at mutation point is replaced by random number between 0-30.

Suppose generated random number are 2 and 5 then Chromosome composition after mutation are:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;02]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;05;18;03]

Chromosome[6] = [20;01;10;06]

Finishing mutation process then we have one iteration or one generation of the genetic

algorithm. We can now evaluate the objective function after one generation: Chromosome[1] = [02;05;17;01]

F_obj[1] = Abs((02 + 2*05 + 3*17 + 4*01) - 30)

= Abs((2 + 10 + 51 + 4) - 30)

= Abs(67 - 30)

= 37

Chromosome[2] = [10;04;13;14]

F_obj[2] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)

= Abs((10 + 8 + 33 + 56) - 30)

= Abs(107 - 30)

= 77

Chromosome[3] = [12;05;23;02]

F_obj[3] = Abs((12 + 2*05 + 3*23 + 4*02) - 30)

= Abs((12 + 10 + 69 + 8) - 30)

= Abs(87 - 30)

= 47

Chromosome[4] = [20;04;13;14]

F_obj[4] = Abs((20 + 2*04 + 3*13 + 4*14) - 30)

= Abs((20 + 8 + 39 + 56) - 30)

= Abs(123 - 30)

= 93

Chromosome[5] = [10;05;18;03]

F_obj[5] = Abs((10 + 2*05 + 3*18 + 4*03) - 30)

= Abs((10 + 10 + 54 + 12) - 30)

= Abs(86 - 30)

= 56

Chromosome[6] = [20;01;10;06]

F_obj[6] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)

= Abs((20 + 2 + 30 + 24) - 30)

= Abs(76 - 30)

= 46

From the evaluation of new Chromosome we can see that the objective function is decreasing, this means that we have better Chromosome or solution compared with previous

Chromosome generation. New Chromosomes for next iteration are:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;02]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;05;18;03]

Chromosome[6] = [20;01;10;06]

These new Chromosomes will undergo the same process as the previous generation of

Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration. This process will be repeated until a predetermined number of generations. For this example, after running 50 generations, best chromosome is obtained: Chromosome = [07; 05; 03; 01]

This means that: a = 7, b = 5, c = 3, d = 1 If we

use the number in the problem equation:

a + 2b + 3c + 4d = 30

7 + (2 * 5) + (3 * 3) + (4 * 1) = 30

We can see that the value of variable a, b, c and d generated by genetic algorithm can satisfy that equality.

Question 2-Define Markov Decision process.

Reinforcement Learning :

Reinforcement Learning is a type of Machine Learning. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal.

There are many different algorithms that tackle this issue. As a matter of fact, Reinforcement Learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed to decide the best action to select based on his current state. When this step is repeated, the problem is known as a **Markov Decision Process**.

A **Markov Decision Process (MDP)** model contains:

- A set of possible world states S.
- A set of Models.
- A set of possible actions A.

- A real-valued reward function R(s,a).

- What is a State?
- A State is a set of tokens that represent every state that the agent can be in.

- What is a Model?
- A Model (sometimes called Transition Model) gives an action's effect in a state. In particular, T(S, a, S') defines a transition T where being in state S and taking an action 'a' takes us to state S' (S and S' may be the same). For stochastic actions

(noisy, non-deterministic) we also define a probability P(S’|S,a) which represents the probability of reaching a state S’ if action ‘a’ is taken in state S. Note Markov property states that the effects of an action taken in a state depend only on that state and not on the prior history.

-

- What are Actions?
- An Action A is a set of all possible actions. A(s) defines the set of actions that can be taken being in state S.

- What is a Reward?
- A Reward is a real-valued reward function. R(s) indicates the reward for simply being in the state S. R(S,a) indicates the reward for being in a state S and taking an action ‘a’. R(S,a,S’) indicates the reward for being in a state S, taking an action ‘a’ and ending up in a state S’.

- A policy the solution of **Markov Decision Process**.
- What is a Policy?
- A Policy is a solution to the Markov Decision Process. A policy is a mapping from S to a. It indicates the action ‘a’ to be taken while in state

S.

- Let us take the example of a grid world:
- An agent lives in the grid. The above example is a 3*4 grid. The grid has a START state(grid no 1,1). The purpose of the agent is to wander around the grid to finally reach the Blue Diamond (grid no 4,3). Under all circumstances, the agent should avoid the Fire grid (orange color, grid no 4,2). Also the grid no 2,2 is a blocked grid, it acts as a wall hence the agent cannot enter it.

The agent can take any one of these actions: **UP, DOWN, LEFT, RIGHT**

Walls block the agent path, i.e., if there is a wall in the direction the agent would have taken, the agent stays in the same place. So for example, if the agent says LEFT in the START grid he would stay put in the START grid.

First Aim: To find the shortest sequence getting from START to the Diamond. Two such sequences can be found:

- **RIGHT RIGHT UP UPRIGHT**
- **UP UP RIGHT RIGHT RIGHT**

Let us take the second one (UP UP RIGHT RIGHT RIGHT) for the subsequent discussion.

The move is now noisy. 80% of the time the intended action works correctly.

20% of the time the action agent takes causes it to move at right angles. For example, if the agent says UP the probability of going UP is 0.8 whereas the probability of going LEFT is 0.1, and the probability of going RIGHT is 0.1 (since LEFT and RIGHT are right angles to UP).

The agent receives rewards each time step:-

- Small reward each step (can be negative when can also be term as punishment, in the above example entering the Fire can have a reward of -1).
- Big rewards come at the end (good or bad).

- The goal is to Maximize the sum of rewards.