GENETIC ALGORITHM FOR SOLVING SIMPLE MATHEMATICAL EQUALITY PROBLEM

Himani Panwar [1], Pragya[2], Dharamveer Singh [3], Abha Singh [4]

[1]Student, Department of IT, IMS Engineering College, Ghaziabad, India
[2]Assistant Professor, Department of IT, IMS Engineering College, Ghaziabad, India
[3]Student, Department of IT, IMS Engineering College, Ghaziabad, India
[4]Student, Department of IT, IMS Engineering College, Ghaziabad, India

---***---

Abstract - *This paper explain genetic algorithm in brief with the help of flowchart and solve the simple mathematical equality problem with the help of genetic algorithm. Genetic algorithms are used to generate high quality solutions for optimizations problems and search problems. Genetic Algorithm is a popular optimization tool in the field of natural science, finance and economics, mathematics , earth science, industry, management , biological science, earth science and computer science. Genetic Algorithms are based on the ideas of natural selection and genetics. Genetic algorithms follow the process of evolution and natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation and those who are not able to adapt changes they will be discarded.*

***Key Words*: Genetic Algorithm, selection, population, fitness function.**

1. INTRODUCTION

Genetic algorithm became famous because of the work of John Holland in the early 1970s, and with the help of his book "Adaptation in Natural and Artificial Systems (1975)". He was the founder of genetic algorithm. A genetic algorithm is a search heuristic and optimization algorithm which is inspired by Charles Darwin's theory of natural evolution. Darwin's concept of evolution is then applied to computer science to solve computational problems which takes a lot of time if solved manually .This algorithm shows the process of natural selection from the generation where the fittest individuals are selected from the generation for the reproduction in order to produce offspring of the next generation, the fittest ones will survive in the next generation. Genetic algorithms imitate the process of natural selection. In simple words, they choose the individuals from the generation and then find their objective function and follow rest of the steps to find out the best solution. Genetic algorithms are based on an analogy with genetic structure and behaviour of chromosome of the population. Genetic algorithms are used to solve the complex problems in the field of searching and optimization problems. The process of natural selection starts with the evolution selection is very important step in this it selects the fittest individuals from a population and they produce offspring which inherit the characteristics of the parents and will be added to the next generation the process is repeated till we found the optimal answer to the problem. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving, the whole genetic algorithm revolve around this only the fittest one will have more chance of surviving. This process keeps on repeating and at the end, a generation with the fittest individuals will be found. This notion can be applied for a search problem and also for optimization problem.

2. PHASES

There are 5 phases in genetic algorithm which are as follows:

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

2.1 Initial Population

This process starts with a set of individuals which is called a **Population**. An individual have it's own characteristics and theses individuals are known as **Genes**. Genes are combined into a string to form a **Chromosome**.

2.2 Fitness Function

The fitness function determines how fit an individual is it will survive in next generation or not. The fitness function plays a vital role in genetic algorithm. The fitness function gives score to each individual. The probability that an individual will be selected for next generation is based on its fitness score. The fittest ones will survive in next generation.

2.3 Selection

The idea of selection phase is to select the fittest individuals from the population and give approval to them for the next generation. Individuals are selected based on their fitness scores. Individuals with high fitness score have more chance to be selected for reproduction.

2.4 Crossover

Crossover is the most important phase in a genetic algorithm. A crossover point is randomly selected from the generation and Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population and new population will be generated.

2.5 Mutation

This process is used to maintain the diversity in the generation and it prevents premature convergence. In mutation genes are randomly replaced on a position with a new value.

2.6 Termination

The algorithm terminates if the population has converged. On termination algorithm provides the optimal answer.

3. GENETIC ALGORITHM

Process of genetic algorithm is as follows:

STEP 1: Determine the number of chromosomes, generation, and mutation rate and crossover rate value for the population.

STEP 2: Generate chromosomes and initialization of values to the chromosomes.

STEP 3: Repeat steps 4-7 until the number of generations is met.

STEP 4: Calculation of fitness values of chromosomes by calculating the objective function.

STEP 5: Chromosomes selection

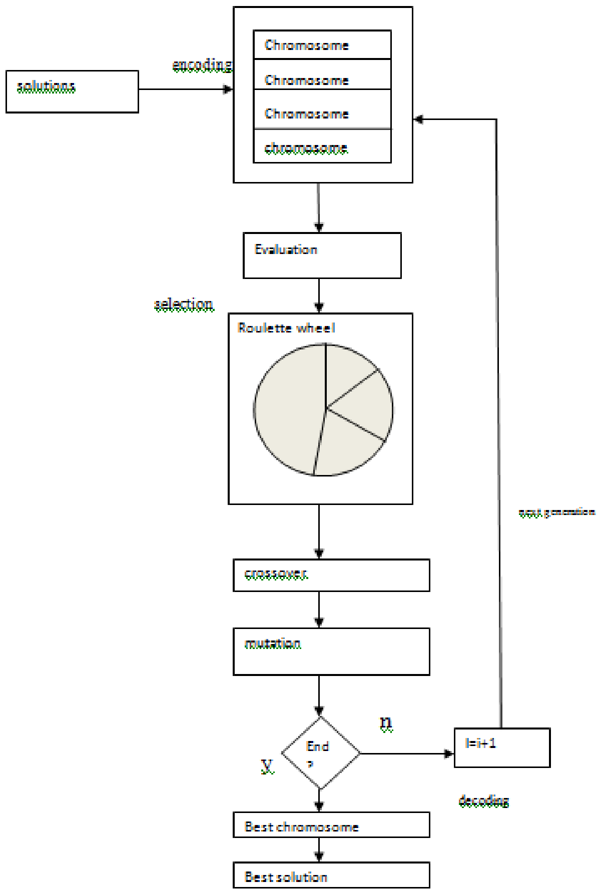
STEP 6: Crossover

STEP 7: Mutation

STEP 8: Solution (Best Chromosomes)

3.1 Flowchart

Flowchart of genetic algorithm is as follows:



4. Linear equality problem

So here is the example of applications of genetic algorithm to solve the simple mathematical linear equality problem. Suppose there is equality a + 2b + 3c + 4d+5e = 20, genetic algorithm will be used to find the value of a, b, c, d and e that satisfy the above equation for this problem the objective is minimizing the value of function f(x) where f(x) = ((a + 2b + 3c + 4d+5e) - 20). Since there are five variables in the equation, namely a, b, c, d and e 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 20.

4.1 Initialization

For example we define the number of chromosomes in population are 6, then we generate random value of gene a, b, c ,d and e for 6 chromosomes

Chromosome[1] = [a;b;c;d;e] = [11;05;04;00;08]

Chromosome[2] = [a;b;c;d;e] = [09;15;03;01;03]

Chromosome[3] = [a;b;c;d;e] = [04;00;01;08;02]

Chromosome[4] = [a;b;c;d;e] = [01;05;13;04;06]

Chromosome[5] = [a;b;c;d;e] = [02;09;12;09;02]

Chromosome[6] = [a;b;c;d;e] = [08;14;05;11;00]

4.2 Evaluation

We calculate the objective function value for each chromosome produced in the initialization step:

F_obj[1] = Abs((11 + 2*5 + 3*4 + 4*0 + 5*8) - 20) =53

F_obj[2] = Abs((9 + 2*15 + 3*3 + 4*1 + 5*3) - 20) = 47

F_obj[3] = Abs((4 + 2*0 + 3*1 + 4*8 + 5*2) - 20) = 29

F_obj[4] = Abs((1 + 2*5 + 3*13 + 4*4 + 5*6) - 20) = 76

F_obj[5] = Abs((2 + 2*9 + 3*12 + 4*9 + 5*2) - 20) = 82

F_obj[6] = Abs((8 + 2*14 + 3*5 + 4*11 +5*0) – 20) = 75

4.3 Selection

The chromosomes which higher fitness will have more probability to be selected for the next generation. To calculate fitness probability we have to calculate the fitness of each chromosome. To avoid the divide by zero problem, the value of F_obj is added by 1 so that fitness will be calculated.

Fitness[1] = 1 / (1+**F_obj[1]**) = 1/(1+53) = 0.0185

Fitness[2] = 1 / (1+**F_obj[2]**) = 1/(1+47) = 0.0208

Fitness[3] = 1 / (1+**F_obj[3]**) = 1/(1+29)= 0.0333

Fitness[4] = 1 / (1+**F_obj[4]**) = 1/(1+76) = 0.013

Fitness[5] = 1 / (1+**F_obj[5]**) = 1/(1+82) = 0.0120

Fitness[6] = 1 / (1+**F_obj[6]**) = 1/(1+75) = 0.0132

Total=0.0185 + 0.0208 + 0.0333 + 0.013 + 0.0120+ 0.0132

=0.1108

The probability for each chromosomes is formulated by this formula**:**

P[i] = Fitness[i] / Total

P[1] = 0.0185 / 0.1108= 0.167

P[2] = 0.0208 / 0.1108 = 0.188

P[3] = 0.0333 / 0.1108 = 0.301

P[4] = 0.013 / 0.1108= 0.1173

P[5] = 0.0120 / 0.1108 = 0.1083

P[6] = 0.0132 / 0.1108 =0.1191

From the probabilities above we find out that the Chromosome 3 has the highest fitness, so Chromosome 3 has the highest probability to be selected for the next generation. For the selection process we use roulette wheel method, for that we should compute the cumulative probability values and their sum should be equal to 1 if their sum is not equal to 1 then there is some error in above computations:

C[1]=0.0167

C[2]= 0.0167 + 0.188=0.2606

C[3]= 0.0167 + 0.188 + 0.301=0.3518

C[4]= 0.0167 + 0.188 + 0.301+0.1173 =0.7167

C[5]= 0.0167 + 0.188 + 0.301+0.1173 + 0.1083 =0.7811

C[6]= 00.0167 + 0.188 + 0.301+0.1173 + 0.1083 + 0.1191=1.00

This process is used to generate random number R in the range 0-1 as follows.

R[1] = 0.209

R[2] = 0.482

R[3] = 0.111

R[4] = 0.842

R[5] = 0.589

R[6] = 0.801

So here we do the comparison and on the basis of that new chromosomes will be formed. 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[1][2]:

NewChromosome[1] = Chromosome[2]

NewChromosome[2] = Chromosome[3]

NewChromosome[3] = Chromosome[1]

NewChromosome[4] = Chromosome[6]

NewChromosome[5] = Chromosome[3]

NewChromosome[6] = Chromosome[5]

Chromosomes in the population thus became:

Chromosome[1] = [09;15;03;01;03]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;00;08]

Chromosome[4] = [08;14;05;11;00]

Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;12;09;01]

4.4 Crossover

In this example, we use one-cut point, i.e. we randomly choose a position in the parent chromosome and then exchange sub-chromosome till we reach the one cut point. Parent chromosome which will mate is randomly choose 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;s
```

Chromosome k will be selected as a parent if R[k]. 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] = 0.117

R[2] = 0.482

R[3] = 0.185

R[4] = 0.199

R[5] = 0.356

R[6] = 0.892

For random number **R** above, parents are **Chromosome[1], Chromosome[3]** and **Chromosome[4]** will be selected for crossover.

Chromosome[1] >< Chromosome[3]

Chromosome[3] >< Chromosome[4]

Chromosome[4] >< Chromosome[1]

After chromosome selection, the next step is to determine the position of the crossover point. This is accomplished by generating random numbers between 1 to length of Chromosome–1. Position of crossover point = length of chromosome-1(5-1) = 4.

In this case, generated random numbers should be between 1 and 4. When we get the crossover point, parents Chromosome will be cut at crossover point and genes will be interchanged. For example we will generate 3 random number and we get:

C[1] = 2

C[2] = 3

C[3] = 1

Then for first crossover, second crossover and third crossover, parent's gens will be cut at gen number 2, gen number 3 and gen number 1 respectively, e.g.

Chromosome[1] = Chromosome[1] >< Chromosome[3]

= [09;15;03;01;03] >< [11;05;04;00;08]

= [09;15;04;00;08]

Chromosome[3] = Chromosome[3] >< Chromosome[4]

= [11;05;04;00;08] >< [08;14;05;11;00]

= [11;05;04;11;00]

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

= [08;14;05;11;00] >< [04;00;01;08;02]

= [08;00;01;08;02]

Thus Chromosome population after experiencing a crossover process:

Chromosome[1] = [09;15;04;00;08]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;11;00]

Chromosome[4] = [08;00;01;08;02]

Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;12;09;01]

4.5. Mutation

M**utation_rate** parameter determines the number of chromosomes that have mutations in a population. Mutation process is done by replacing the generation at random position with a new value. The process is as follows:

So, first we have to calculate the total length of generation in the population. The total length of the generation is

total_generation = number_of_generation_in_Chromosome * number of population

= 5 * 6 = 30

Mutation process is done by generating a random integer between 1 and total gen (1 to 30). . If the generated random number is smaller than the 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 * 30 = 1.8 ≈ 3 Suppose generation of random number yield 2, 14 and 28 then the chromosome which have mutation are Chromosome number 1 gen number 2 , Chromosome 3 gen number 4 and chromosome number 6 gen number 3. The value of mutated gets at mutation point is replaced by random number between 0-20. Suppose generated random number are 2 , 0 and 5 then Chromosome composition after mutation are:

Chromosome[1] = [09;02;04;00;08]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;00;00]

Chromosome[4] = [08;00;01;08;02]

Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;05;09;01]

After finishing the mutation process then we have one iteration or one generation of the genetic algorithm. Now, we can evaluate the objective function after one generation:

Chromosome[1] = [09;02;04;00;08]

F_obj[1] = Abs((9+ 2*2 + 3*4 + 4*0 + 5*8)-20)

= 45

F_obj[2] = Abs((4+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 29

F_obj[3] = Abs((11+ 2*5 + 3*4 + 4*0 +5*0)-20)

= 13

F_obj[4] = Abs((8+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 33

F_obj[5] = Abs((4+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 29

F_obj[6] = Abs((2+ 2*9 + 3*5 + 4*9 +5*1)-20)

= 56

So, these new Chromosomes will repeat the same process of genetic algorithm such as evaluation, selection, crossover and mutation as the previous generation of Chromosomes did and at the end a new generation of Chromosome for the next iteration is produced. This process will be repeated until a predetermined number of generations.

5. EXPERIMENTAL RESULTS

By executing the simple genetic algorithm code (SGA) written in Java programming language. The primary work of the SGA is performed in three routines selection, crossover, and mutation we get the best values of the variables a, b, c, d and e.

SGA Parameters

Population size = 10

Chromosome length = 5

Maximum of generation = 20

Crossover probability = 0.25

Mutation probability = 0.01

Linear equation problem

A value	*B value*	*C value*	*D value*	*E value*
1	*1*	*3*	*2*	*0*
2	*6*	*4*	*0*	*0*
1	*0*	*2*	*2*	*1*
1	*1*	*0*	*3*	*1*
2	*0*	*1*	*0*	*3*
6	*0*	*3*	*0*	*1*
4	*4*	*0*	*2*	*0*
0	*3*	*2*	*2*	*0*
3	*2*	*2*	*0*	*1*
4	*4*	*0*	*2*	*0*

6. CONCLUSION

- GA have an objective function which determines the quality of the solution.
- GA finds the solution near to optimal answer.
- Outcomes shows that the genetic algorithm have the ability to find optimal solution for solving linear equation.
- GA is easy to understand and implement.
- GA can be used for solving the complex problems.
- Most fittest will survive and choose for next generation.

7. REFERNCES

[1] Denny Hermawanto, “Genetic Algorithm for Solving Simple Mathematical Equality Problem”, Indonesian Institute of Sciences (LIPI), INDONESIA[1]

[2] Lubna Zaghlul Bashir, “Solve Simple Linear Equation using Evolutionary Algorithm”, World Scientific News 19 (2015) 148-167[2]