An Adaptive Strategy of Genetic Operators in Genetic Engineering

Manjula S[1]
Lecturer,
DOS in Computer Science
Davangere University,
Davangere-02, INDIA
manjula.shamarao@gmail.com

Shivamurthaiah M[2]
Lecturer,
DOS in Computer Science
Davangere University,
Davangere-02, INDIA
shivamurthaiah@gmail.com

Abstract- **The genetic engineering is also known as "Genetic Modification". The direct manipulation of an organism's genome is done by using Genetic algorithm an organism that is generated through genetic engineering is considered to be a Genetically Modified Organism (GMO). Transgenic organisms are able to express foreign genes because the genetic code .Genetic Engineering is implemented in the form of genetic programming technique which is designed with the help of artificial intelligence concepts and methods which performs a user defined tasks. This genetic programming is a set of instruction and fitness function to measure how well a computer has performed a task. The functions which are used in genetic programming is represented in the form of tree structure; The main purpose of genetic algorithm is to create an offspring from existing population and this task can be performed with the help of genetic operators; the main operators used in evolutionary algorithm such as genetic programming are crossover and mutation. This paper is mainly focusing on the crossover and mutation operators; which has different types and operation systems. The cross over is applied on an individual by simply switching one of its nodes with another individual in the population. Mutation affects an individual in the population, it can replace a whole node in the selected individual or it can replace just node's information. Mutation can happen with any of the individual randomly, while crossing of chromosomes with unexpected resultant values. The selection of individual for crossing is based on the fitness of each individual in the population.**

Keywords- ***Crossover, DNA, Encoding, Fitness, Genetic algorithm, Genetic engineering, Genetic Operators, Genetic Programming, Initialization, Mutation, Offspring, Selection, Transgenic.***

I. INTRODUCTION

The recent trend of computer science is mainly focusing on genetic computing, which is the part of "Soft Computing", it refers to a collection of computational techniques in computer science, Artificial Intelligence, Machine learning and some engineering disciplines. The Genetic Computing is implemented in the year 1975 John Halland first with the help of "Genetic Algorithm" introduced GA in AI System, an effective GA representation and meaningful fitness evaluation are the keys of the successive GA applications[1]. The appeal of GA's comes from their simplicity and elegance as robust search algorithms. The GA population based search and optimization method that mimics the process of natural evaluation. The two main concepts of natural evaluation which are natural selection and genetic dynamics. The genetic algorithm is a part of evolutionary computing which is rapidly growing area of AI. GA's are inspired by Darwin's theory about evaluation-"Survival of the fittest", With respect to biological background Charles Darwin has formulated the fundamental principles of natural selection as a main evolutionary tool. In 1865, George Mendel discovered these hereditary principles by experiments he carried out on persons with the help of experimental results Morgan found that Chromosomes are the carriers of hereditary information and that genes representing the hereditary factor were lined upon chromosomes. Each chromosome is built of DNA. The biological terminology used in evolutionary computation, the gene code; it represents the characteristics of an individual and these gene can take different value or alleles. This set of alleles is represented by gene pool. This gene pool can determine all the different possible variations for further generations. The diversity of the individuals in the population is determined by the size of the gene pool. Most living organisms store their genome (is the set all the genes of specific species) on several chromosomes. But in GA to simplify the representation all the genes are usually stored on the same chromosomes. The word genotype to describe the set of its genes. The phenotype describes the physical aspect of an individual. The process of decoding a genotype to produce the phenotype is known as morpho genesis. The biological terms are also used in GA with representing chromosomes as strings gene as feature/character. Genomes as guesses, solutions, collection of genes. Darwin also stated that the survival of organisms can be maintained to the process of reproduction, crossover and mutation. Darwin's concept of evolution is adapted to computational algorithms to find solution to a problem called objective function in natural fashion. A solution generated by GA is called a chromosome, while collection of chromosome is referred as a population. These chromosomes are composed from genes and its

value can be either in numerical, binary symbols or characters depending on the type of problem which needs to be solved. The selection of a good chromosome is based on fitness function, so that it can generate a suitable form of solution for a given problem. This fitness function is also used to select a best chromosome for reproduction of new offspring through crossover operation and mutation state value. The chromosome which has higher fitness value will have greater probability of being selected for creating next generation [2].

II. STEP-BY STEP IMPLEMENTATION OF GENETIC ALGORITHM

Step 1: Random selection of individual so that we can create a set of initial population for further steps.

Step 2: These selection of individual is base on fitness value, if the chromosome fitness value is above the fixed fitness rate. Then the particular chromosome or individual is considered as valid chromosome for further operation.

Step 3: The randomly selected chromosome need to be determine with this we need to fix the mutation rate and cross over rate value.

Step 4: After determining the chromosome need to determine each chromosome with random values.

Step 5: The initialized random values are used to calculate objective functions. This can be calculated with given genetic equation.

Step 6: The fitness value is derived with the help of mathematical equation, fit[i] =1/ (1+fit_object) by taking object function from previous step.

Step 7: The probability for each chromosomes is formulated by P[i] =fitness[i]/total fitness value of each chromosome.

Step 8: For selection of chromosome we should compute cumulative probability, it is a sum of all probability value that is each chromosome from last step that is ,C[i]=P[1]+P[2]+P[3]+…..+P[i]. After calculating cumulative probability the selection process of chromosomes is done by using roulette-wheel methodology is known as chromosomes selection.

Step 9: Crossover- crossing; i.e. randomly select a position in the parent1 chromosome then exchanging the position value of parent 1 with the same position of parent2 chromosomes. This process of exchanging these position values with the parent chromosome is known as crossing techniques. The parent chromosome which will mate randomly selected and the number mate chromosomes controlled using crossover_rate (pc) parameter

Step 10: Mutation- Number of chromosomes that have mutations in a population is determined by the mutation rate parameter. This mutation process is done by replacing the gene at random position with a new value for this to happen a must calculate. The total length of gene in the population, but this mutation is not mandatory for all offspring generation.

Step 11: The random number R[i] is greater than P[i] (probability value and its chromosomes. And similar than
P [i+1] (probability of i+chromosomes) then select chromosome [i+1] as a chromosome in the new population for the next generation. P[i] <R[i] <P [i+1]
New chromosome [1] =chromosome [i+1]
This represents generation of new chromosome.

Step 12: Extraction of best chromosomes from generated chromosomes.

Figure 1 represents the gene flow with respect to flow chart of GA we are mainly focusing on initialization & evaluation, selection, crossover and mutation operation.

III. GENETIC COMPUTING

A. Encoding:

Process of converting the biological chromosomes into computational chromosomes to perform mathematical operations by initializing random numbers to solve problem with GA. It is purely depend on the problem, here we are focusing on 4 different of encoding system.

1. Binary
2. Permutation
3. Value
4. Tree

A.1 Binary Encoding:

It is most common, it is mainly because in binary encoding every chromosome is a string of 0 and 1 bit configuration. This gives many possible chromosomes even with small number of alleles. This encoding is often not natural for may problem and sometimes correction must be made after crossover and/or mutation.
Chrome A: 1 0 1 1 0 0 0 0 1 1 0 1
Chrome B: 0 1 1 1 1 1 1 0 0 0 0 0

A.2 Permutation Encoding:

This encoding methodology is mainly used in ordering type of problems example TSP(Travelling Salesman Problem)[6]. In permutation encoding, every chromosome is a string of numbers which represents a sequence of numbers as chromosome values. Even in this problem, for some types of cross over and mutation corrections must be made to live chromosome consistently.
Chrome A: 1 5 8 2 9
Chrome B: 9 7 6 4 3

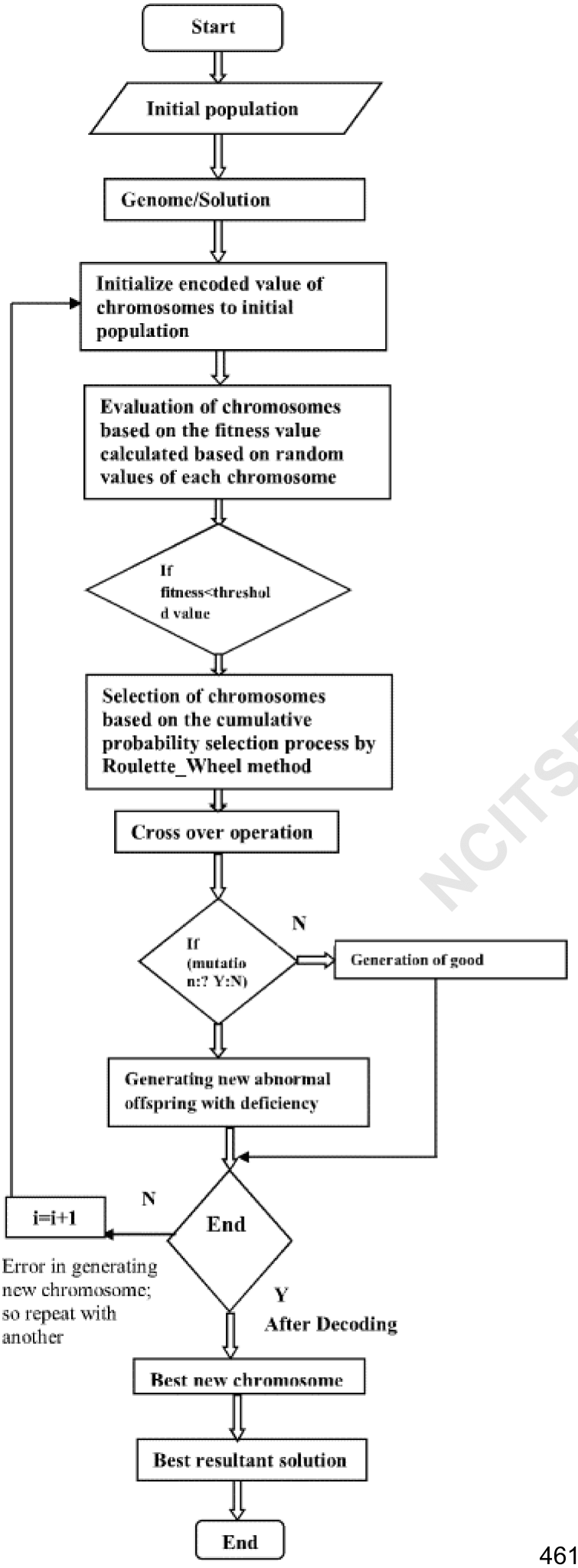


Fig1: Gene Flow

A.3 Value Encoding:

Value encoding technique is usually implemented in complicated problems, where use of binary encoding for this type of problem would be very difficult. this uses correct value encoding technique in problem solving with complicate values such as; real number, form numbers, characters which are connected to problem. This encoding is often necessary to develop some new cross over and mutation specific for the problem.

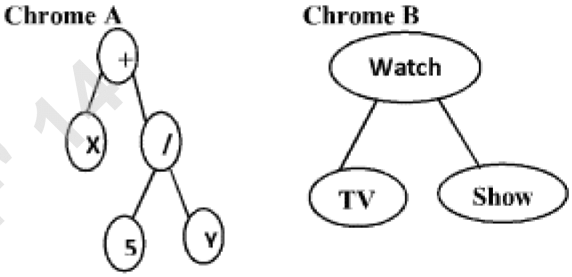
Chrome A (1.258 6.765 8.119

Chrome B (left), (right), (forward)

Chrome C ABCJEFGHIONPQSUV.

A.4 Tree encoding:

A tree encoding is mainly used for valuing programs/expressions for GP. In this tree encoding every chromosome is tree of some object such as commands/functions used in programming language LISP is often used to this, because program in it are represented in the form and can be easily parsed as a tree. So the crossover and mutation can be done relatively easily.



Prefix expression (+x(/5y)). Watch TV show.

Fig 2

B. Initialization and Evaluation:

B.1 Initialization:

To get the knowledge of both, consider of an example of an application that uses GA to solve the problem of combination, consider an equation a+2b+c=10. To find the value of a,b,c we are using GA, the main objective of this problem is minimizing value if function f(x). where f(x)=(a+2b+c)-10. Since there is a 3 variables in the equation, we can compose chromosome as follows. To speed up the computation we can distinct the initialization values to variable as used integer values between 0 and 10. To solve this problem we define 3 different number of chromosomes[3] In population, then we generate random values of gene a,b,c for 3 chromosomes.

Chrome[1]=[a;b;c]=[5;2;3]

Chrome[2]=[a,b,c]=[8,6,4]

Chrome[3]=[a,b,c]=[7,9,1]

B.2Evaluation:

We compute the objective function value for each chromosome produced in initialization step. This can be evaluated by implementing the initialized value into an given equation. a+2b+c=10➔f(x)=(a+2b+c)-10.

F_obj[1]=abs((5+2*2+3)-10)

=abs((5+4+3)-10)
=abs(12-10)
=2

F-obj[2]=abs((5+2*6+4)-10)
=abs((8+12+4)-10)
=14

F-obj[3]=abs((7+2*9+4)-10)
=abs((7+18+1)-10)
=16

The object function value is calculated, the fittest chromosomes have higher probability to be selected for the production for the next generation. To compute the probability of fitness, first we need to compute fitness of each chromosomes. To avoid division by zero problem, the E_obj value will be added by default value 1.

Fitness [1]=1/(F_obj[1]+1)=1/(2+1)=1/3=0.333
Fitness [2]=1/(F_obj[2]+1)=1/(14+1)=1/15=0.0666
Fitness [3]=1/(F_obj[3]+1)=1/(16+1)=1/17=0.058823

Total fitness of all the 3 chromosomes.
Total=0.333+0.0666+0.058=0.457

The probability of each chromosome is calculated by;
P[i] =Fitness[i]/Total.
P[i]=Fitness[i]/Total.
P[1]=0.333/0.457=0.7286
P[1]=0.066/0.457=0.1444
P[1]=0.058/0.457=0.1269

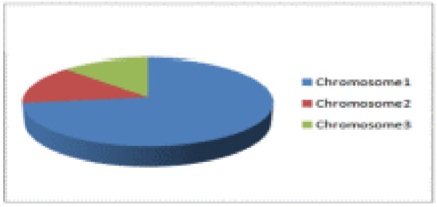


Fig 3: Chromosome Probability value

From the above probability, chromosome 1 has the highest fitness and also highest probability, which is to be selected for next generation chromosome.

C. Roulett Wheel Selection:

The selection process where implementing Roulett_wheel; for that we should compute the cumulative probability values.
C[1]=0.7286
C[2]=0.7286+0.1.444+0.1269=0.873
C[3]=0.7286+0.1444+0.1269=0.9999

After calculating the cumulative of selection process using Roulette_wheel can be done. The process is to generate random number R in range 0-1.
R[1]=0.486
R[2]=0.1199
R[3]=0.586

If the random number R[1] is greater than P[2] and smaller than P[1].

P[2]<R[1]<P[1]➔0.1444<0.486<0.7286

Then select chromosome 1as a chromosome in new population for nest generation/offspring.
i.e. new chromosome[0]=chromosome[1]

D. Crossover:

In GA crossover is a genetic operator used to vary the programming of a chromosomes from 1 generation to next generation. The crossover is a process of taking more than 1 parent solutions and producing child solutions from them the pseudo code for the crossover process is as follows:

Crossover Algorithm:

Begin K←0;
While(K<population) do
R(K)←random[0-1];
If (R[K]<PC) then
Select chromosome[K] as parent
else
discard the chromosome and goto next;
end;
//if loop end
K=K=1
end; //while loop end
end; //end if begin

In this paper we are mainly focusing on one-point crossover, two-point crossover, cut and splice crossover and uniform & half uniform crossover, 3 parent crossover.

D.1: 1-point crossover:
A single crossover point on both parents organism string is selected. All data beyond that point in either organism string is swapped between the two parent organisms. the resulting organisms are the children/offspring.

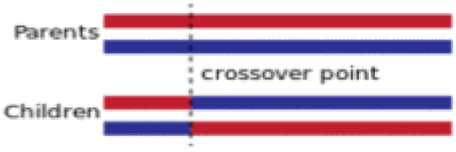


Fig 4: 1-Point Crossover

D.2: 2-point crossover:

In this, 2 points to be selected on the parent organism's string. Everything between the 2-points is swapped between the parent organisms, rendering two child parents.

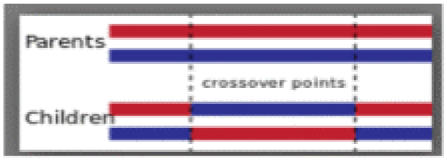


Fig 5: 2-Point Crossover

D.3 Cut & Splice:

Another crossover variant, this approach results in a change in length of the children string. The reason for

this difference is that each parent string has a separate of crossover point.

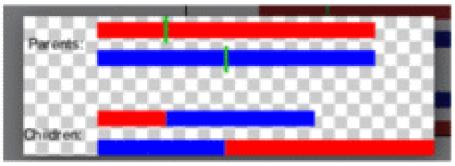


Fig 6: Cut and Splice

D.4 Uniform and half-uniform:

Uniform crossover uses a fixed mixing ratio between two parents; in this crossover it enables the parent chromosome to contribute the gene level rather than the segment level. if the maximum ratio is 0.5; the offspring has approximately half of the genes from the first parent and other half of from second parent. Although crossover point can be randomly chosen.

In half-uniform crossover, scheme exactly half of the non matching bits are swapped. thus the first we must calculate the number of differing bits, the number is divided by two, the resulting number is how many of the bits that do not match between two parents will be swapped.

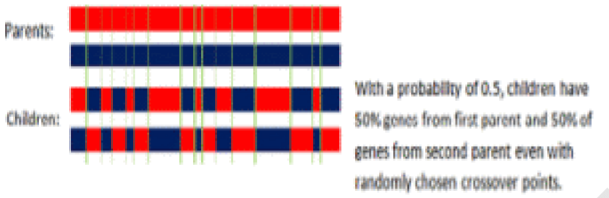


Fig 7: Uniform and Half-Uniform

D.5: 3-parent crossover:

In this technique, a child is derived from 3 parents, the bits are randomly chosen; each bit of first parent is checked with bit of second parent whether they are same, if same then the bit is taken for the offspring, otherwise the bit taken for offspring.

Parent 1: 1 1 0 1 0 0 0 1 0
parent 2: 0 1 1 0 0 1 0 0 1
parent 3: 1 1 0 1 1 0 1 0 1

Resulting offspring= 1 1 0 1 0 0 0 0 1

E. Mutation:

It is a GO used to maintain genetic diversity from 1 generation of the other population of GA chromosome to the next[7]. in biological aspects, mutation alters one or more gene value in a chromosome from its initial state, by this resultant solution is entirely change from previous solution, hence GA can come to better solution by using mutation.

Different types of mutations are:

1. Bit string mutation:
2. Flip bit mutation
3. Boundary
4. Non uniform
5. Uniform
6. Gaussian

E.1 Bit string mutation:

The mutation of bit stream ensure through bit flips at random positions.

ex: 1 0 1 0 0 1 0

⇩

1 0 1 0 1 1 0

The probability of a mutation of bit is 1/L, where l-is a length of binary vector. Thus a mutation state of 1/mutation is reached.

E.2 Flip bit:

This mutation operator takes the chosen genome and inverts the bits.

E.3 Boundary:

It replaces the genome with either lower or upper bound randomly. This can be used for integer and flod genes.

E.4 Non-uniform:

The probability that amount of mutation will go to zero with next generation increased by using non uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. it tunes solution in the later stages of evolution.

E.5 Uniform:

It replaces the value of the chromosome gene with a uniform random value selected between the user specified upper and lower bounds for that gene.

E.6 Gaussian:

This operator adds a unit Gaussian distributed random value to the chosen gene. if it falls outside of the user specified lower or upper bounds for that gene, the new gene value is clipped.

IV. IMPLEMENTATION & RESULT

As per the algorithm code has been designed with the help of MATLAB application and result is represented in the form of graph.

A. The coding represents initialization technique as a sample

```
function [pop]=initialise(popsize, stringlength, fun);
pop=round(rand(popsize, stringlength+2));
pop(:,
stringlength+1)=sum(2.9(size(pop(:,1:stringlength),2)−1
:−1:0).
*pop(:,1:stringlength))*(b−a)/(2.9stringlength−1)+a;
pop(:, stringlength+2)=fun(pop(:, stringlength+1));
end
```

B. The coding represents selection technique as a sample

```
function [newpop]=roulette(oldpop);
totalfit=sum(oldpop(:,stringlength+2));
prob=oldpop(:,stringlength+2) / totalfit;
prob=cumsum(prob);
rns=sort(rand(popsize,1));
```

```
fitin=1; newin=1;
while newin<=popsize
if (rns(newin)<prob(fitin))
newpop(newin,:)=oldpop(fitin,:);
newin=newin+1;
else
fitin=fitin+1;
end
end
```

C. The coding represents crossover technique as a sample

```
function [child1, child2]=crossover(parent1, parent2, pc);
if (rand<pc)
cpoint=round(rand*(stringlength−2))+1;
child1=[parent1(:,1:cpoint)
parent2(:,cpoint1+1:stringlength)];
child2=[parent2(:,1:cpoint)
parent1(:,cpoint1+1:stringlength)];
child1(:,
stringlength+1)=sum(2.9(size(child1(:,1:stringlength),2)
−1:−1:0).
*child1(:,:stringlength))*(b−a)/(2.9stringlength−1)+a;
child2(:,
stringlength+1)=sum(2.9(size(child2(:,1:stringlength),2)
−1:−1:0).
*child2(:,1:stringlength))*(b−a)/(2.9stringlength−1)+a;
child1(:, stringlength+2)=fun(child1(:, stringlength+1));
child2(:, stringlength+2)=fun(child2(:, stringlength+1));
else
child1=parent1;
child2=parent2;
end
end
```

D. The coding represents Mutation technique as a sample

```
function [child]=mutation(parent, pm):
if (rand<pm)
mpoint=round(rand*(stringlength−1))+1;
child=parent;
child[mpoint]=abs(parent[mpoint]−1);
child(:, stringlength+1)=sum(2.9
(size(child(:,1:stringlength),2)−1:−1:0).
*child(:,1:stringlength))*(b−a)/ (2.9stringlength−1)+a;
child(:, stringlength+2)=fun(child(:, stringlength+1));
else
child=parent;
end
end
```

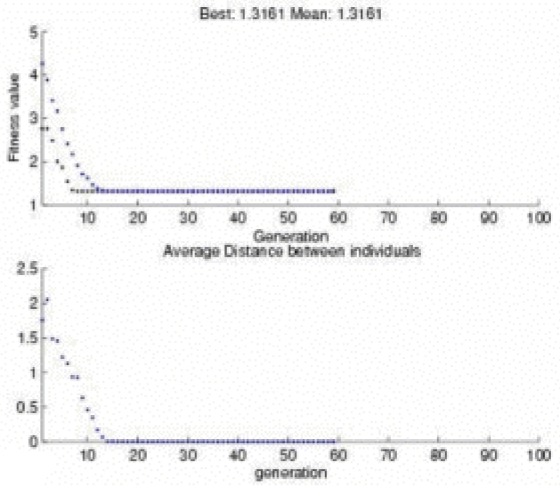


Fig 8: Graphical representation of Fitness Value from Population Group

Figure 8 represents the graph of valid persons is the group of population which has been initialized for the genetic operation, the above implementation code represents the how creation of child will takes place, to create a child first we need to initialize the group of population, This can be evaluated by implementing the initialized value into an given equation. a+2b+c=10➔f(x)=(a+2b+c)-10. By using this equation we need to find the Functional object for each and every person in the population, once it has been calculated we need to compute the probability of fitness, first we need to compute fitness of each chromosomes. To avoid division by zero problem, the E_obj value will be added by default value 1. This detail mathematical implementation is shown in 3.2 Initialization and Evaluation, based on this we have implemented the MATLAB code where it represents the initialization program, selection of population program, crossover program and mutation program, the graph represents the valid persons which has been selected based on fitness value, this graph is also representing best value & mean value of the entire generation. By keeping these valid fellows we are implementing cross over technique.

Mutation can happen randomly with respect to old chromosome and it will create new chromosome with mutation resultants, it is explain by considering an example problem & code has been implemented and resultant is shown below; consider binary population, old chrome with 4 individual each of length 8:

old chrome = [0 0 0 0 0 1 1 1
1 0 0 0 1 0 0 1
0 0 1 0 1 0 0 0
1 1 0 1 1 0 1 1]

By calling mutation(parent, pm) where pm represents the threshold value of mutation parameter and program has been executed with relevant resultant value

New chrome = [0 0 1 0 0 1 1 1
1 1 0 0 0 0 0 1
0 0 0 0 1 0 0 0
1 1 0 1 1 0 1 1]

The complement of binary string is obtained by applying mutation and probability. Here mutation ([1 0 10 1 1 1 0], 1) and resultant value is 0 1 0 1 0 0 0 1.

V. CONCLUSION

Genetic algorithms are a very powerful tool, allowing searching for solutions when there are no other feasible means to do so. The algorithm is easy to produce and simple to understand and enhancements easily introduced. This gives the algorithm much flexibility. This paper presents the optimal solution for a genetic computational problem. It has demonstrated with the help of mathematical equations and accurate results have been extracted. This Genetic Computation can be implemented with the help of MATLAB or any other applications.

REFERENCES

[1] An Introduction to Genetic Algorithms: MELANIE MITCHELL

[2] A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II; Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T.Meyarivan; IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL.6, NO.2, APRIL 2002

[3] Genetic Algorithm for Solving Simple Mathematical Equality Problem: Denny Hermawanto; Indonesian Institute of Sciences (LIPI), INDONESIA.

[4] GENETIC ALGORITHMS: kumara Sastry, David Goldberg; University of Illinois, USA.

[5] ARTIFICIAL INTELLIGENCE: R.Goebel, J.Siekmann and W.Wahlster.

[6] Introduction to Genetic Programming; Matthew Walker.

[7] ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING: Begavioral and Cognitive Modelling of the Human Brain; Amit Konar, Jadavpur University, Calcutta, India.

[8] Soft Computing : S N Sivanandam S N Deepa