CHAPTER 5

GENETIC ALGORITHM

5.1 INTRODUCTION

Genetic Algorithm (GA) technique, is used in the third module of the integrated system to prioritize test cases based on the objectives like maximization of fault coverage, minimization of execution time, maximization of completeness and minimization of memory usage.

5.2 GENETIC ALGORITHM OVERVIEW

Genetic Algorithm (GA) was proposed by John Holland in the 1960s at the University of Michigan. It is used as a search heuristic for combinatorial optimization problems based upon the principles of evolution observed in nature. Though the ranges of problems to which genetic algorithms have been applied are quite broad, they are mainly used for function optimization. The basic unit in GA is the chromosome. The chromosome is usually represented as a string. The general procedure for GA is

1. Population

   - Initial population is formed by generating random population of \( n \) chromosomes
   - The fitness function is evaluated on each \( n \) chromosomes
New population is created and the genetic operators are applied.

2. **Selection**

   Selects two parent chromosomes from a population according to their fitness.

3. **Crossover**

   With a crossover probability, crossover the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

4. **Mutation**

   With a mutation probability mutate new offspring at each locus (position in chromosome).

5. **Termination**

   Stopping condition for the GA process.

GA provides efficient and effective techniques for optimization, business, scientific and engineering, medical field to develop treatment programs, optimize drug formulas, improve diagnostics and can be applied to a wide variety of optimization problems such as scheduling, adaptive control, transportation, the traveling salesmen problem, used to evolve behaviors of characters and events within computer games and can be used to improve trading systems.

GA is capable of discovering a different set of solutions for complex problems. In genetic algorithm, there are four main steps:
chromosome generation, selection, crossover and mutation. These operators are combined for finding the optimal solution until a specified termination criterion is met.

5.2.1 Population

In a genetic algorithm, using random generation technique, an initial population of chromosomes is formed. The fitness of every chromosome in the population is evaluated, and the chromosomes are selected from the current population based on their fitness to form a new population. The same procedure is repeated with the new population. This repeats till an environment with maximum number of generations being produced or the satisfactory level being reached. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

5.2.2 Selection

Selection is a genetic operator used to select a chromosome from the current population and is included in the next generation’s population. Before the inclusion of these chromosomes application of other operators such as crossover and mutation are done on them.

There are many types of selection available. They are

5.2.2.1 Roulette wheel selection

Roulette wheel selects chromosomes according to their fitness value. The chance of a chromosome getting selected is proportional to its fitness. The better the chromosomes are, the more chances to be selected.
5.2.2.2 Tournament selection

Tournament selection is an additive selection method that uses roulette wheel selection for N times to produce a tournament subset of chromosomes. The best chromosome in the subset will become the selected chromosome.

5.2.2.3 Steady state selection

Elimination of the worst of individuals is done in steady state selection.

5.2.2.4 Top percent selection

Top percent selection uses a selection operator that randomly selects a chromosome according to the specification provided by the user from the top N percent of the population.

5.2.2.5 Best selection

The best chromosome is selected in this process based on the fitness measure. In the case of same fitness, a random selection is made.

5.2.2.6 Random selection

As the name suggests, a chromosome is randomly selected from the given population.

5.2.2.7 Elitism selection

Elitism selection brings in a selection of the best chromosome and includes them into the new population then the others are done in the classical way. A search on the best solution is obtained here.
5.2.3 Crossover

Crossover is a genetic operator that produces a new chromosome (offspring) by combining two chromosomes (parent). The main idea behind crossover is that the new and better chromosome than both of the parents can be produced by taking the best characteristics from each of the parents.

The types of crossover are

5.2.3.1 One point crossover

A single crossover point is selected randomly within the chromosome. All information beyond that single crossover point in either parent is swapped.

For example: here two parents are selected for crossover. The “|” symbol indicates the randomly chosen crossover point.

Parent 1: 11101|010
Parent 2: 10100|110

After interchanging the parent chromosomes at the crossover point, the following offspring are produced

Offspring1: 11101|010
Offspring2: 10100|110

The portion right of the selected point of these two strings is exchanged to form a new pair of strings. The new strings are thus a combination of the old strings. One point crossover is more suitable when string length is small.
5.2.3.2 Two point crossover

Two point crossover is a variation of the one site crossover, except that two crossover points are chosen and the bits between the points are exchanged. Two crossover points are selected within a chromosome and then the two parent chromosomes are exchanged between these points to produce two new offspring.

Consider the following 2 parents which have been selected for crossover. The “|” symbols indicate the randomly chosen crossover points.

Parent 1: 100|010|11
Parent 2: 101|000|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced

Offspring1: 100|000|11
Offspring2: 101|010|11

The two point crossover is suitable for large strings. The underlying objective of crossover is to exchange information between strings to get a string that is possibly better than the parents. One point crossover and two point crossover are the most common ones that are adopted.

5.2.3.3 Uniform crossover

A uniform crossover operator is used to decide the probability of contribution of gene values of each parent in the offspring chromosomes. Here the parent chromosomes are mixed at the gene level.
Parent 1: 10101010
Parent 2: 00101001

If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. Below is a possible set of offspring after uniform crossover.

Offspring1: 10101000
Offspring2: 00101011

5.2.3.4 Arithmetic crossover

In the arithmetic crossover, two parent chromosomes are combined to produce two new offspring according to the arithmetic equations.

5.2.3.5 Heuristic crossover

In the heuristic crossover, the direction of search is based on the fitness value of the two chromosomes and a crossover that is made on it.

5.2.3.6 Three parent crossover

In this crossover, the child is derived from three parents. Each bit of first parent is checked with each bit of second parent if same then the bit is taken for the offspring otherwise the bit from the third parent is taken for the offspring. For example, the following three parents

parent1  0 1 0 0 0 1 0 1 0
parent2  0 1 1 0 0 1 0 0 1
parent3  1 1 0 1 1 0 1 0 1
produce the following offspring

offspring 0 1 0 0 1 0 0 1

5.2.4 Mutation

Altering one or more gene values in a chromosome in its initial state is the mutation technique. The new gene values are added onto the gene pool. The better solution may be achieved with these new gene values. The main advantage of mutation is that it prevents the population from stagnating at any local optima.

To perform mutation a single character at mutation point is changed. And usually the mutation point is chosen randomly. Even multiple mutation points may be chosen on a single string and the output is taken as the next generation population.

The concept of mutation can be simply explained with an example. Here mutation may be

- Single, where the original string is mutated at the single mutation point 4 there by performing single mutation on the original string or

- Multiple, where the original string is mutated at multiple mutation points 2, 4 & 5 there by performing multiple mutation on the original string.

Original string  10001001

Mutated string  10011001

(single mutation on point 4)

Mutated string  11010001

(multiple mutations on point 2,4,5)
There are many types of mutation. They are

5.2.4.1 Flip bit

Flip bit mutation operator is suitable for binary genes and inverts the value of the gene from 1 to 0 and 0 to 1.

5.2.4.2 Boundary

Boundary mutation operator is appropriate for integer and float genes and replaces the value of the chosen gene with either the upper or lower bound for that gene.

5.2.4.3 Non-uniform

The Non-Uniform mutation operator is appropriate for integer and float genes and it increases the probability that the amount of the mutation will be close to 0 as the generation number increases. Though this results in stagnating the population in the early stages, later on this reduces, and this is appropriate for integer and float genes.

5.2.4.4 Uniform

Uniform mutation operator is replacing the value of the chosen gene with the randomly chosen user-specified bounds and it is appropriate for integer and float genes.

5.2.4.5 Gaussian

Gaussian mutation operator is appropriate for integer and float genes that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is trimmed if it ranging beyond the user-specified lower or upper bounds for that gene.
5.2.5 Termination

Having selected the initial population at random, the fitness function validates the fitness of the chromosomes and selects those with higher potential for the production of new offspring. After the application of genetic operators, a new population is created from the current population. These operators are enhancing the present population. The values of the objectives function of each individual are gained by decoding the strings express the fitness of the upcoming generations. These values are being improved at each stage that replaces the position of best solution. The whole process represents the completion of a single cycle by genetic algorithm. After many generations of selection for the fitter chromosomes, the resultant population is considered as fitter than the original. And this process is repeated till the stopping criterion is met.

There are many types of termination. They are

5.2.5.1 Generation number

Generation number termination method stops the evolution when the maximum number of evolutions specified is reached. This termination method is always active.

5.2.5.2 Evolution time

In evolution time, the termination method the evolution gets stopped when the elapsed evolution time exceeds the user-specified maximum evolution time.
5.2.5.3  **Fitness threshold**

The Fitness threshold termination method stops the evolution when the fitness threshold is not met that is the best fitness in the current population becomes less than the user-specified fitness threshold. The main objective here is set to minimize the fitness.

The Fitness threshold termination method also stops the evolution when the fitness threshold is not met that is the best fitness in the current population becomes greater than the user-specified fitness threshold. The main objective here is set to maximize the fitness.

5.2.5.4  **Fitness convergence**

The Fitness convergence termination method stops the evolution as soon as the fitness is seemed to be converged.

5.2.5.5  **Population convergence**

Here, in population convergence termination method, the evolution terminates, when the population is deemed as converged.

5.2.5.6  **Gene convergence**

The Gene convergence termination method stops the evolution when a user-specified percentage of the genes that make up a chromosome are deemed as converged.

5.3  **TEST CASE PRIORITIZATION USING GENETIC ALGORITHM**

The test case prioritization is done using genetic algorithm. To prioritize the test cases, the steps to be followed are
• Chromosome generation
• Chromosome selection
• Chromosome crossover
• Chromosome mutation
• Termination

Here chromosome represents the test cases.

5.3.1 Chromosome Generation

After using the flocking algorithm, a genetic algorithm is used to discover the most desirable test case. The flocking algorithm removes the noise as well as the outliers. It identifies the core points and thus the needed clusters are built. These clusters have a relatively uniform group of points. The birds flocking algorithm outputs the test case clusters. In order to obtain the desirable test case, the output of the flocking is subjected to genetic algorithm. So, here the output from birds flocking forms the chromosome.

5.3.1.1 Chromosome representation

In GA, the solution to the problem is usually represented by a chromosome and is comprising of a string of genes. Usually the chromosomes are represented by the binary alphabet \{0, 1\} and sometimes, depending on the application, integers or real numbers are used. Whatever may be the representation, it can be used to form a solution as a finite length string.

A population \( T = \{t_1, t_2, \ldots, t_n\} \) is formed from a set of chromosomes. Where \( T \) is the test suite containing the test cases \( t_1, t_2, t_3, \ldots, t_n \). The test cases which forms the chromosome are represented as a string of 4 tuple and is given as \((x_1, x_2, x_3, x_4)\). Here \( x_1 \) means the memory usage value of the test case, \( x_2 \) is execution time, \( x_3 \) is coverage and \( x_4 \) is fault coverage.
5.3.2 Chromosome Selection

In the selection stage of genetic algorithm, the test cases are chosen from a population for subsequent procreation. The genetic algorithm solves the optimization problem by creating a population of chromosomes, which is a set of possible solutions for the problem. First of all, several individual solutions are randomly created in order to form initial population. The output from birds flocking algorithm is taken as the population input for genetic algorithm. The population gives better solutions as the search evolves and it eventually converges. To form a new generation, a proportion of the existing population is chosen during each consecutive generation.

A fitness based process is performed to select the fitter solution. The fitter solution is a measure of fitness function. Weighted Sum Approach is used for forming fitness function. The approach used here to solve the multi-objective optimization problem is to allot a weight $w_i$ to each normalized objective function $f_i^*(x)$; hence, the problem is changed into single objective problem with a scalar objective function as Equation (5.1).

$$
\min z = w_1 f_1^*(x) + w_2 f_2^*(x) + \ldots + w_p f_p^*(x)
$$

(5.1)

where, $f_i^*(x)$ is the normalized objective function, $f_i^*(x)$ and $\sum w_i = 1$.

Here, the user is expected to give the weights. Solving a problem with an objective function as in Equation (5.1) for a given weight vector $\mathbf{w} = \{w_1, w_2, \ldots, w_p\}$ produces a single solution.

But in real time multiple solutions are desired. Then the problem needs to be solved multiple times with diverse weight combinations. The main problem in this method is selecting a weight vector.
In the proposed work, in order to calculate the fitness value for each chromosome in the population obtained from flocking algorithm, a number of fitness functions are used such as Equations (5.2), (5.3) and (5.4).

\[
\text{fitness} = \frac{1}{1 + E^i_r} \quad (5.2)
\]

where, \(E^i_r\) is the vector which is given as,

\[
E^i_r = \frac{E_r}{2} \quad (5.3)
\]

where, \(E_r\) is the sum of error

\[
\text{i.e., } E_r = \sum_{i=2}^{n} \sum_{j=1}^{n} \left( \frac{d_d - d_a}{d_d} \right)^2 \quad (5.4)
\]

where, \(d_d\) is the desired distance between the test cases \(i\) and \(j\), \(d_a\) is the actual distance between the test cases \(i\) and \(j\), and \(n\) is the number of test cases.

5.3.2.1 Fitness function evaluation by threshold factor

Fitness evaluation involves defining an objective or fitness function against which each test case is tested for suitability for the environment under consideration. As the genetic algorithm process precedes the individual fitness of the best test case increases as well as the total fitness of the test suite as a whole.

The final fitness function is given by the following Equation (5.5).

\[
F(S) = O(S) + \alpha \cdot P(S) \quad (5.5)
\]
where, $O(S)$ is the objective function of genetic algorithm fitness, $P(S)$ is a non negative penalty function, and $\alpha$ is a penalty time coefficient, which varies adaptively during the GA evolution. The value of $\alpha$ is selected such that if the value is larger than the threshold value $t$, which is calculated as the average of all $x_1$, $x_2$, $x_3$ and $x_4$.

Elitist selection is used as a selection method to select the test cases for recombination. When using genetic algorithm to solve complicated global optimization problems, only the elitist selection genetic algorithm (ESGA) can converge to optimal global solution. The main advantage of this selection is that it produces best solution in every generation. Only the test cases with strong fitness values are selected for next generation.

In this elitist selection method, the fitness value of all the test cases are calculated based on the fitness function. Then the test cases are sorted by descending fitness values. The sum $S$ is calculated which is sum of all test cases fitness in population. The cut off value $r$ is calculated as the average of all $x_1$, $x_2$, $x_3$ and $x_4$. When the fitness value of the test case is greater than $r$, the current test case is declared as selected, or else it is declared as omitted.

The added penalty term is a function of the degree of violation of the constraints, in order to produce a gradient toward valid solutions. The penalty term for any solution that breaches the constraints can be formulated by a quantity $d(S)$, which measures the level of constraint violation of solution $S$. Thus, the penalty function $P$ depends on $d(S)$ is given as Equation (5.6).

$$P(d(S)) = A \cdot d(S) + Bt$$ (5.6)

Where, $A$ is a “severity” factor, which defines the slope of the penalty function and $Bt$ is a penalty threshold factor.
So, by using elitism or elitist selection, the best individuals are retained in a generation unchanged in the next generation.

5.3.3 Chromosome Crossover

Cross over is the salient operator in GA. The two strings participating in the crossover operation are known as parent strings and the resulting strings are known as children strings. In crossover, two parent chromosomes are pooled together to form new chromosomes called offspring. From the existing chromosomes, the parents are chosen with preference towards fitness so that offspring is expected to appear good genes that make the parents fitter. The effect of cross over is usually beneficial.

Assume, \( S_1 = \{s_{11}, s_{12}, \ldots, s_{1n}\} \) and \( S_2 = \{s_{21}, s_{22}, \ldots, s_{2n}\} \) be the two chromosomes. A random number is selected from the integers \( 0 \leq r \leq n \). \( S_1 \) and \( S_4 \) are the offspring of crossover \( S_1 \) and \( S_2 \), where \( S_3 = \{s_i | i \leq r, s_i \in S_2\} \) and \( S_4 = \{s_i | i > r, s_i \in S_1\} \). Here in case of test case optimization it is \( T_1 = \{x_{11}, x_{12}, x_{13}, x_{14}\} \) and \( T_2 = \{x_{21}, x_{22}, x_{23}, x_{24}\} \). By applying the crossover operator, the genes of good chromosomes are expected to emerge more frequently in the population finally leading to an excellent solution.

Many crossover operators exist in the GA. One point crossover and two point crossover are the most common ones adopted. In most crossover operators, two strings are picked from the pool at random and some portion of the strings is exchanged between the strings. Crossover operation is done at string level by randomly selecting two strings for crossover operations.

A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring.
Here the uniform crossover is used. The mixing ratio used is 0.5, so approximately half of the genes in the offspring will come from test case 1 and the other half will come from test case 2. Below is a possible set of offspring after uniform crossover.

Before crossover

Chromosome 1

Feature 1  Feature 2  Feature 3  Feature 4

Chromosome 2

Feature 5  Feature 6  Feature 7  Feature 8

After crossover: (the resulting offspring)

Chromosome 3

Feature 1  Feature 2  Feature 7  Feature 8

Chromosome 4

Feature 5  Feature 6  Feature 3  Feature 4

Each test case has 4 features. In case of uniform cross over, since the mixing ratio used is 0.5, the first two features from the first test case is combined with the last two feature of the second test case to form the first offspring. Similarly, the last two features from the first test case is combined with the first two feature of the second test case to form the second offspring.
Thus approximately half of the genes in the offspring will come from test case 1 and the other half will come from test case 2.

### 5.3.4 Chromosome Mutation

In mutation, the characteristics of chromosomes are changed randomly, thereby changing the structure of a chromosome. Normally, the mutation is applied at the gene level. The mutation operator is used to introduce change into the chromosome population and it is applied to each new structure individually. When the bits are being copied from the current string to the new string, there is probability that each bit may become mutated. A given mutation involves randomly altering each gene with a small probability called mutation probability $P_m$.

With this probability, a random real value is generated which is used to make a random change in the $m$-th element selected randomly of the chromosome. If random number is less than the mutation probability, then the bit is inverted. This is explained below using an example.

Here the test suite contains 16 test cases from T1 to T16 and the mutation probability $P_m = 0.0035$.

Test suite

| T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 | T15 | T16 |

Mutation probability $P_m = 0.0035$

The individual test case random values are given as

\[ R(T1) = 0.567 \]
\[ R(T2) = 0.36 \]
\[ R(T3) = 0.34 \]
\[ \mathbf{R(T4)} = 0.001 \]
\[ R(T5) = 0.67 \]
\[ R(T6) = 0.0035 \]
\[ R(T7) = 0.342 \]
\[ R(T8) = 0.247 \]
\[ \mathbf{R(T9)} = 0.0024 \]
\[ R(T10) = 0.0089 \]
\[ R(T11) = 0.2342 \]
\[ R(T12) = 0.87 \]
\[ R(T13) = 0.2348 \]
\[ \mathbf{R(T14)} = 0.0012 \]
\[ R(T15) = 0.3456 \]
\[ R(T16) = 0.457 \]

Here \( R(T4), R(T9) \) and \( R(T14) \) are having \( P_m \) the mutation probability value less than the specified value 0.0035. So, they are omitted for the next generation.

So, the next generation test suite contains

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T10</th>
<th>T11</th>
<th>T12</th>
<th>T13</th>
<th>T15</th>
<th>T16</th>
</tr>
</thead>
</table>


With this a bit of diversity to the population by scattering the occasional points is introduced resulting in better optima. In this the weak individual that will never be selected for further operations.

In mutation during the local search, a point is created in the neighborhood of the current point around the current solution. The main aim of mutation is to maintain diversity in the population.

The objective function is used to provide a measure of how individuals have performed in the problem domain. In the case of a minimization problem, the most fit individuals will have the lowest value of the associated objective function. The fitness function is normally used to transform the objective function value into a measure of relative fitness. The fitness function is defined as Equation (5.2).

Based on the fitness criterion, poorer performing individuals are gradually taken out, and better individuals have a greater possibility of conveying genetic information to the next generation.

Mutation adds new information in a random way to the genetic search process and mutation may cause the chromosomes of individuals to be different from those of their parent individuals. The mutation rate is usually less than 1%.

5.3.5 Termination

Termination is the criterion used by the genetic algorithm to make a decision about whether to continue the process or to stop the process.

Having selected the initial population at random, the fitness function validates the fitness of the chromosomes and selects those with higher potential for the production of new offspring. After the application of
genetic operators, a new population is created from the current population. These operators are enhancing the present population. The whole process represents the completion of a single cycle by genetic algorithm. After many generations of selection for the fitter chromosomes, the resultant population is considered as fitter than the original. Until the stopping criterion is met the steps are repeated. Here the number of generations is used as a stopping criterion. Finally the genetic algorithm outputs the prioritized test cases.

The Figure 5.1 shows the application of genetic algorithm to the test cases. The values for the operators of genetic algorithm are specified.

![Figure 5.1 Application of genetic algorithm](image)

The output from birds flocking algorithm forms the input to genetic algorithm. Then genetic algorithm is applied on the test cases which are from birds flocking. The output from genetic algorithm is the prioritized test cases as shown in Figure 5.2.
5.4 SUMMARY

Genetic algorithms are used for prioritizing the test cases with multi objectives. The population is formed with multi objective test cases. The fitter solution is found to prioritize the test cases. Only the test cases with strong fitness values are selected and prioritized according to their fitness value. The output from genetic algorithm gives the prioritized set of test cases. These test cases are used for testing the target system. The performance of the developed multi objective test case prioritization system is analyzed in the next chapter.