A GENETIC ALGORITHM FOR TEST SUITE OPTIMIZATION

CHETAN J. SHINGADIYA\(^1\) & NITESH M. SUREJA\(^2\)

\(^1\)Research Scholar, Rai University, Ahmedabad, Gujarat, India
\(^2\)Director, Om Engineering College, Junagadh, Gujarat, India

ABSTRACT

Software testing is one of the most important parts of the software development process. In software development, developers always rely on software testing to deal with bugs. The problem of software testing in software development is one of the most important and research areas. Here, test set optimization plays an important role in system performance. The genetic algorithm is one of the techniques, widely used for optimization based on problems inspired by nature. In this article, we demonstrate the genetic algorithm with tournament selection techniques. We, evaluate system performance based on a number of test inputs.

KEYWORDS: Genetic Algorithm, Software Testing, Test Suite Optimization

1. INTRODUCTION

Software testing is a process of running the software with the intention of identifying errors. It is an expensive activity and consumes a significant amount of effort and cost. Therefore, reducing the number of test cases or test suites. We are able to significantly reduce test effort and time [1]. For each change, new test cases are recorded and added to existing test suites, but with that size of the test suite increases. This requires optimizing the test suite to cope with time and cost constraints. Optimization is necessary due to the large size of the test suite, limited budget, resource constraints; in this case, it is not possible to run the entire test suite, even with minor changes [2]. In addition, the software consumes 50% of its costs for testing purposes [3]. The goal of optimization is to find a solution that represents the need to request test cases, and that should have maximum code coverage and find the best test suites for software tests. Therefore, our focus is on reducing costs and efforts for software product development.

The first phase of test suite optimization is that, create a random population and then calculate the fitness of each individuals and loop continue till performance is reached or maximum no. of generation is reached. The main aspect of this is that we have to test these test suite and find starting fitness and percentage of target code covered.

The second phase of test suite optimization select two parents based on performance of fitness. Also, generate two children from parents by mixing of genes from each parent and apply chance of random mutation. The third phase repeat child generation until a required new population size is reached and then store new solution and compute end best fitness. For this, we used genetic algorithm because it will search optimal solution from entire search space [4].

In this paper, we solve this problem using genetic algorithm. This GA create a random population using binary encoding and compute fitness of each individuals. Select two parents based on their fitness and then generate
two child solutions from parents by mixing of genes from each parent and apply random mutation. Repeat this child generation process till required population size is reached. Next, it will store new population and compute fitness of it. Rest of the paper is organized as follows: In section 2, existing works are introduced. Section 3 explains GA; in section 4, experiment results and discussion are given. At last in section 5, we come up with conclusions.

2. Genetic Algorithm

A genetic algorithm is randomized based on the principles of development and natural genetics. It combines utilization of past results and exploration of new areas in the search space. Using the survival of the coolest methods and organized but random exchange of information, the genetic algorithm can mimic some innovative abilities in human search. Graphs for graphs are randomized but not simple random. It utilizes historical information efficiently to identify new search sites with potential improvements.

GA maintains a population of representative agents that evolve over time and subsequently converge. Individuals in the population represent chromosomes. Each person has a number that is very helpful in determining the best way to solve this problem. Genetic algorithm consists of three operators. The nominator selects the most appropriate members of the population to be the parents of the next generation. The tour guide selects any pairs of individuals and modifies certain aspects of the information. The mutation operator randomly takes the individual and changes it. Because natural genetics generally have a high probability of crossbreeding, the population evolves iteratively (in the terminology of the genetic algorithm, over a generation) to improve an individual’s fitness. The structure of a genetic algorithm is a loop of selection followed by a series of crosses and mutations. The probability of crossing and mutation is constant and constant at the beginning. Finally, the genetic algorithm is executed until certain completion conditions are reached, such as number of iterations, execution time, execution time, stability of the result, and so on. [4].

3. Our GA Approach

In this approach, we examine the basic genetic algorithm (GA). We will propose new GA that will try to solve the problem selection pressure, based on adjust tournament size. After that, we will measure experiment result of existing Genetic Algorithm (GA) with our approach. However, we will test algorithms based on the solution.

3.1 Initial Population

Genetic Algorithm generate many solutions. For this, we need to generate an initial population of solutions. The population size dependent on type of problem that we examined in our experiment. This initial populations can be represented by chromosomes. Here, we have used binary encoding method to generate chromosomes and that are represented by 0’s and 1’s.

3.2 Fitness Function

Based on chromosome of population, we are going to calculate fitness value of each chromosome. To calculate fitness, we are going to use reference solutions for that. The number of bits are matched with reference solutions based on that we calculate fitness of particular chromosome.
3.3 Reproduction

Reproduction is most important phase of genetic algorithm. As per our approach we have used tournament selection method for select two individuals from our population for generate child.

Crossover

The crossover is a type of genetic operator used to exchange bits from a chromosome based method, we applied. Here, we have used single point crossover method. Two bits are selected randomly to cross the mating pool in order to generate new offspring.

Mutation

The mutation means, when we flip bits from chromosome, we have used bit flipped method. After applying mutation, we get new population for next generation.

Termination Condition

In this approach condition will stop when generation criteria is full filled or we get required solution.

Steps of Genetic Algorithm

Step: 1

Create an initial population of each individual.

Step: 2

Loop is continue till performance is reached or a maximum number of generations is completed.

Step: 3

We select two parents based on performance of fitness – better performing parents are to be selected.

Step: 4

Generate two child solutions from two parents by mixing the genes from each parent.

Step: 5

Repeat child generation until a required new population size is reached.

Step: 6

Store new population and calculate score of latest generated population.

4. RESULTS AND DISCUSSIONS

As we can summarized from the given below table, the GA with tournament selection technique is give better performance than simple GA. While in case of simple GA, we have to test more individuals than GA with tournament selection, so it will take more time to execute. If we increase input test parameters, then our approach behaves well, as compared to simple GA.

Table 1
<table>
<thead>
<tr>
<th>No of Test Inputs</th>
<th>Chromosome Length</th>
<th>Population Size</th>
<th>Maximum Generation</th>
<th>Initial Score</th>
<th>Last Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>500</td>
<td>200</td>
<td>65.3%</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>500</td>
<td>100</td>
<td>70%</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>70%</td>
<td>100%</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>50</td>
<td>50</td>
<td>65.70%</td>
<td>97.10%</td>
</tr>
</tbody>
</table>

### Figure 1

#### 5. CONCLUSIONS AND FUTURE SCOPE

After implementation of genetic algorithm with tournament selection technique, we conclude that GA will perform very well with respect to chromosome length, population size and maximum generation. One important note is that with this approach, GA performs well even if we increase or reduce our parameters as above. While comparing the performance, GA with tournament selection out performs with simple GA. Another important aspect is that, accuracy is better of GA with tournament selection as compared to simple GA. Future scope of this paper is that, here we have to test almost each individual so obviously that it will take more time to execute; so in future, we have to optimized our approach so that it can take less time to execute, which ultimately reduce the cost and efforts of software testing.

### REFERENCES

Impact Factor (JCC): 9.7093

NAAS Rating: 3.76
A Genetic Algorithm for Test Suite Optimization


