Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.184 – 189

RESEARCH ARTICLE

Using Genetic Algorithm for Whole Test suite Generation of Object Oriented Programs

Mrs.R.Gowri
Associate Professor of IT
SMVEC
Pondicherry, India
gowrithirugnanam@gmail.com

R.DhanBhagya Preety
IT
SMVEC
Pondicherry, India
preetyrajendran@gmail.com

B.Durga Devi
IT
SMVEC
Pondicherry, India
durgadev04@gmail.com

G.Aruna
IT
SMVEC
Pondicherry, India
mail2arunasai@gmail.com

***Abstract*— In Software testing there is a great demand for the automation of test cases. The test cases are either generated before coding using software specifications or after coding using program execution traces. Many genetic algorithms have been proposed for procedural programming but they do not suit well for object oriented programming. This is because in object oriented programs the object relationship exists and considering them is important for the generation of best test suite. We propose a method to generate test suites for object oriented programs using genetic algorithm by considering the object-relationships and their dependencies such as polymorphism, message passing and inheritance that exists among them. The key feature of our proposed technique is that the test suites are evolved as a whole instead of generating one test case for each coverage goal. To investigate the effectiveness of our approach we have used a java program of Chocolate Vending machine as the source code and few test cases have been relatively developed to evaluate the fittest test suite with maximum coverage. We have applied our technique in the EVOSUITE tool to determine the efficiency of our approach. Our results indicate that the use of genetic algorithm in Test case Generation is beneficial than the traditional approach of targeting single branches.**

Keywords—genetic algorithm; coverage goal; object oriented testing; chormose; mutation

1. INTRODUCTION

Testing is considered to be the most critical part in the software development cycle. In order to test software, the test data is generated. A good test data can uncover even the most critical errors in the software. Nowadays various testing tools [11],[12] are available that can automatically generate the test data through a specific coverage criterion. The testing tools should be general, robust and always generate the right test data based on the coverage criteria. So for this purpose, the search algorithm of the tool must find where the best values (test data) lie and concentrate its search there. One such search algorithm that is much efficient to find the best solution to a problem is the Genetic Algorithm (GA) [7].The Genetic algorithms are a part of evolutionary computing, inspired by the Darwin's theory of Evolution. It is a heuristic search method based on the evolutionary ideas of the genetics and the natural selections. The algorithm uses techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover. Genetic Algorithm search mechanism starts with an initial set of population. Any one solution in the population is called as the chromosome. The search mechanism [14] is steered by the fitness principle and it proceeds for a number of generations as a cycle. In each generation the fitter solutions will be selected based on the fitness function, to form a new population. Three main operators [20] namely reproduction, crossover and mutation are used in this cycle. The Cycle will repeat for a number of generations until certain criteria are met

Thus, in this work we have used genetic algorithm in the generation of test suites for object-oriented program. Firstly,

the Original source code of Chocolate Vending Machine is mutated and then the chromosome operations such as crossover and mutation are applied on the mutated code. The test cases are written for the computed Source Code. Finally, the fitness function determines the best test suite that has the maximum coverage and minimum length among the other test suites.

The paper is organized as follows: Following this introductory section, the Section 2 provides the background work. The related work is elaborated in Section 3 and the proposed work is described in section 4. The conclusion and future work are described in the Section 5.

2. BACKGROUND

2.1 Population and Generation

A population can be considered as a list of several guesses, which consist of information about the individuals. Each individual in the population represent a solution. For a healthy Population, the GA must have several members. But a larger population can consume longer time to find a good solution. Small population can find a good solution quickly since only minimum evaluations are required per generation. However premature Convergence generally occurs in small population, which will result in loss of the power of searching or generating better solutions. This can be overcome by an increased mutation probability, which is quite similar to the random testing method. However, the samples of small population do not produce results with higher variance. Therefore, it is necessary to set up a method that will find a strategy, which will avoid converging towards a non optimum solution. In our approach, we have used the fitness function as the main criteria that select the best chromosome out of the entire population and helps in acquiring an optimum solution.

2.2 Genetic Algorithm Operators

A new population is generated from the existing population by means of three operators that GA uses on its population

(a) Selection

In GA, the selection of the starting generation impact a lot on the performance of the next generation. In the selection phase, the chromosomes of the population is chosen for reproduction based on various criteria such as choosing a chromosome at random or giving higher preference to fitter members. Then the evolutionary process is based on the crossover and the mutation operators.

(b) Cross over or Recombination

After Selection, the cross over or Recombination operator is applied to the selected Chromosomes. The crossover operator combines the genes of the two parent chromosome to generate new fitter offspring. The process continues until the next generation has enough individuals.

(c) Mutation

The Mutation operator is applied to alter the genes of the selected chromosomes. It is done in order to create diversity among the population, avoiding stagnation or getting stuck at local optimal solution.

2.3 Fitness Function

The Central Objective of the Fitness Function is to select the best test suite. The Fitness function is directly proportional to the selection of a test case. The Fitness Function in our proposed work is determined by using three factors stated as, likelihood, and close to boundary value and branch coverage. These are the factors that determine whether is test suite is good or bad when compared to other test suites. So the fitness function determines where the test suite is suitable to be chosen or not. The fitness of a test suite in our proposed system is determined as,

$$F(T) = L(T) + B(T) + R(T) / 3$$

3. RELATED WORK

Software testing is a major important concept but many of the software developers may miss the software testing phase due to the lack of time to complete the project. The software developer doesn't get time to test the project fully. The other difficulty in software testing is generating the test case. The works of Antony and jeevarathinam [1], deals about the mutation testing which is used to generate the test case. The formalization technique [8] is used that has two advantages.
Firstly, it analyzes the test set whether it satisfies the specific coverage. Secondly, it allows the tester to focus on less number of formalization and more number of interesting tests. The efficiency of mutation testing depends on the type of faults which are represented. These genetic operators are used for the concepts which are used in object oriented programs like polymorphism, inheritance, message passing, and dynamic binding. The faults are not generated; it may be a programmer specific and an application specific. The characteristics of the program should be selected to execute the mutation operators in testing. The traditional mutation testing does not handle the object oriented programs.

The Genetic algorithm can be used for software testing. It is due to the fact to achieve the disadvantages which are faced by the automation testing. The ESIM technique [2] is an environment used for to test the embedded software. The ESIM environment runs the Genetic Algorithm and the tested program separately and later it is communicated using the hardware ports. An example of ESM technique is that, the genetic algorithms generates the test case and sends the input to the tested program. Later the tested program sends the response

signal back. The response time is the major important concept to check the fitness of the genetic algorithm.

The goal-oriented approach is also used for the automatic test data generation [3]. The genetic algorithm uses the new method of generating the test date by using previously discovered test data. The advantage of the technique is that it combines the goal-oriented, intelligent and features of random. It generates the test data quickly and focused with direction. Another advantage is that it has the ability to execute the large software programs. The fitness function is evaluated using the branch information. The targets are identified using the predicate paths with the use of control dependency graph and it does not require all coverage for the test requirements. Also, N.Gupta et al [9] proposed a technique to generate the test cases for the class in the object oriented programs. This method focuses on automatic generation of test cases and it is achieved using the random test generator. Object oriented programs are the complex to test, in which an object tends to interact by passing messages with each others.

In the literature of test case generation there are also methods that uses two algorithms which is genetic and bee colony algorithm [7] with the intend to reduce the effort and the time of execution of the software programs .It involves the reduction of test cases from a large number of test suites. The technique which was followed is that it identifies the faults in minimum execution time and selects the test cases from larger test suites. The bee colony algorithm is used because, here the bee acts as an agent and search the minimum test cases. The bee will now add a test cases on the search path to check whether it increase the fault detection. This technique has been proved to be beneficial on regression testing. It produces the optimal results and efficient in reducing the effort which was required to the larger test suites and time.

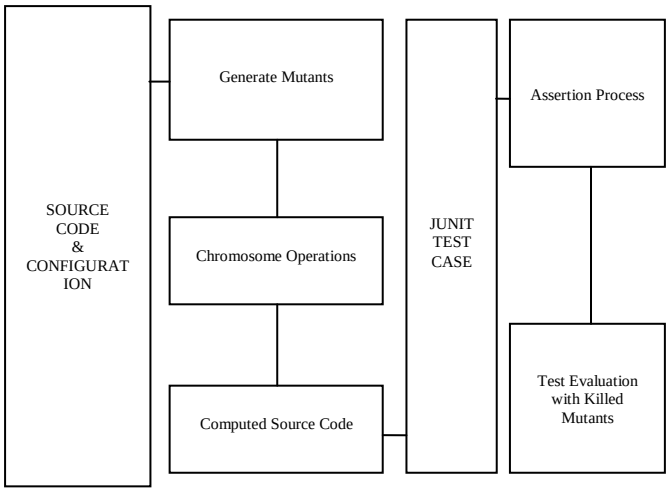
The works of Baudry et al. [10] is towards the optimization of the entire test suites in which all the test suites are considered at the same time. A search algorithm is used to optimize the test cases with respect to mutation analysis. The limitation of this work is that the length of the test cases has to chosen and fixed manually that does not change during the search. There are many automated test case generation tools like DART [11], CUTE [12] that aims to achieve high coverage, but at the cost of assuming the availability of an automated oracle that tests the generated oracles. In general, the test suites generated follow a path coverage criterion that would be too large for a tester to manually evaluate it.

The testing problem that we address in this paper is different from what we previously seen in DART or CUTE or similar other techniques. Our proposed technique targets at difficult faults for which the automated oracles may not be available and hence results in manual evaluation by the tester. So it is necessary for the generated Test Suite to be optimized and of manageable size. The approach we follow can be summarized as: Satisfy mentioned coverage criterion of the fitness function with the smallest available test suite.

4. PROPOSED WORK

Test Case Generation is a tedious process in software testing, even if it automated. A most common scenario is after the test data are generated, the test oracle needs to be added. But there is a least probability for automated oracles to be available for every test data. In such cases, the tester manually adds the test oracles. So it is necessary to produce small test sets to make the oracle problem as easier for the tester.

4.1 SYSTEM ARCHITECTURE



A common approach in literature is to generate a test case for each coverage goal and then combine them all into a test suite. But the size of such a test suite is generally large or sometimes it is difficult to predict. We propose a novel approach for whole test suite generation of object oriented programs, which advance upon the current approach of aiming one goal at a time. Our proposed approach intents to cover all the coverage goals together, at the same time reducing the size of the test suite. There are many benefits of this approach like the infeasible targets in the code don't affect its effectiveness. We have used Genetic algorithm for this proposal. It is shown in Fig 1.

The test cases can be generated from the software specifications during the early stage i.e. before coding or from the program execution traces after the software development i.e. after coding. Heuristic approach helps in creating quality software test cases. A mutation based heuristic approach is proposed here to generate quality test cases from the program execution trace. Initially, the Source code obtained from the tester is configured for inducing mutants. Both the traditional mutation operators and class level mutation operators are implemented for the mutation testing. Then the chromosome operations like cross-over and mutation are applied on the configured source code using the search operators. The resulting computed source code is used for creating JUNIT test case. Finally, the assertion process takes place and the tests are evaluated by determining the live and killed mutants for each search operator.

4.2 GENETIC ALGORITHM for TGEN

We use a Meta heuristic search algorithm, namely, genetic algorithm to evolve the test suites that optimize the chosen coverage criterion. In this section, we describe the applied Genetic algorithm, the chromosome operations and the fitness function.

The GA attempts to imitate the mechanisms of natural adaptation in computer system. The major parameters considered are the chromosome operations and the fitness function. In GA, the population is initially selected as a random sum of test suites. This is taken as the current population. In the current population, the elitism method is applied. This is because when creating a new population through mutation and crossover operations there is a greater chance of losing the best chromosomes of the population. Elitism is the method that first copies the best or some of the best chromosomes to the new population. The remaining population is constructed as described in the algorithm. As elitism prevents the loss of best solution, it increases the performance of the genetic algorithm. Along with the best found solution, it is necessary to select the parent chromosomes through random selection. Then cross over the parents to form the new children. If the crossover was not performed then the children or off springs would be the exact copy of the parent chromosomes. The mutation operation helps in changing one or more parts of the chromosome. If the mutation is 100% applied, then the whole chromosome is changed. The fitness of the parent and the mutated child chromosomes is evaluated by the coverage criteria. And accordingly, either the parent or the child, whichever is fit is added to the new generation along with the elite. Along with the fitness function, the GA also takes into consideration the length of the chromosomes (size of the test suites) for generating a better and best fit chromosomes(Test suites).

The Genetic Algorithm applied is depicted in Algorithm 1. Input given is a random population and the evolution is performed until a solution is found or the specified time and allocated resources have been used up. In each iteration, a new generation is created. The elite of the generation is found out (line 3). The crossover of the chromosome is done (line 6) and then the same is mutated (line 7). Then the fitness function finds the best fit chromosome (line 10)based on the coverage criterion and the length constraints. The best solution yield by the GA is added to the new generation.

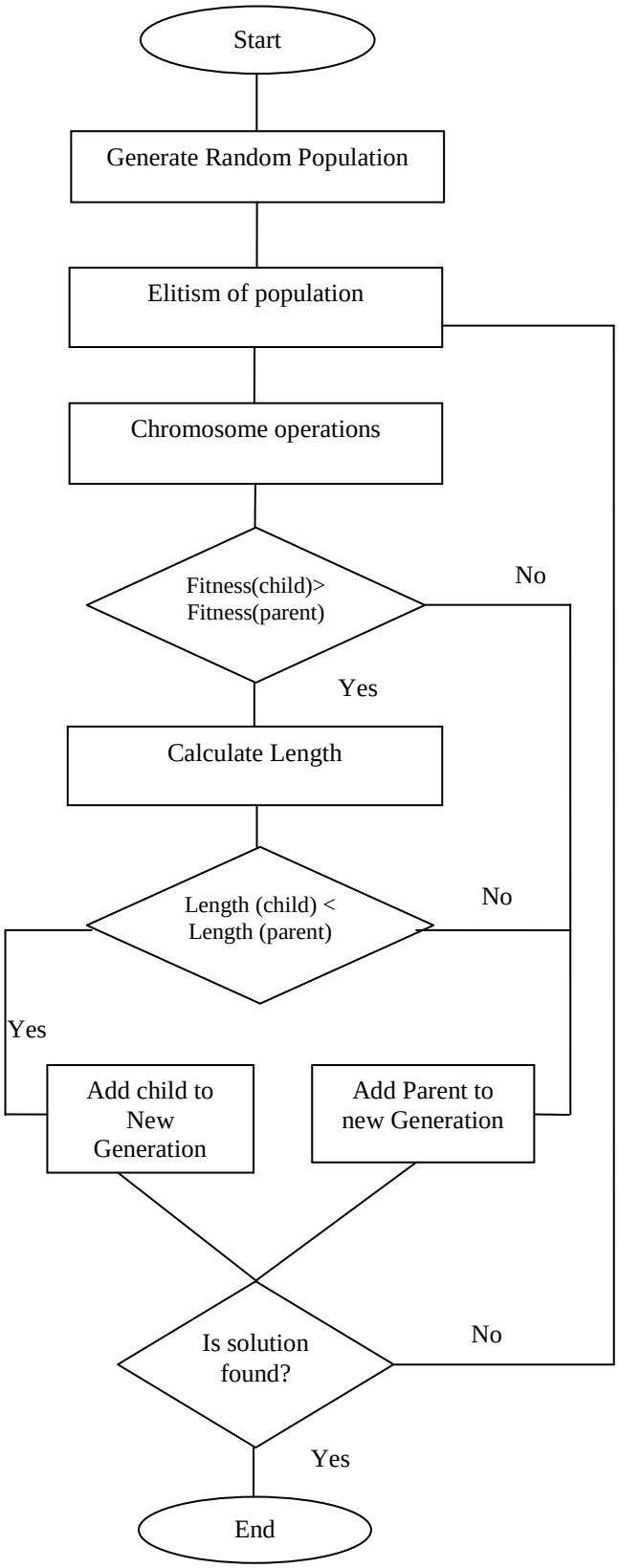
GA ALGORITHM

Input : current population

Output : A new generation with best fit chromosomes (Test suites)

1. Generate random population, $P=\Sigma TS_i$ // ***current population***
2. Repeat
3. $Z \leftarrow$ elite of population P // **elitism**
4. **While** $|Z| \neq P$ **do**
5. $P_1,P_2 \leftarrow$ Any two parent chromosomes in P // ***random selection***
6. $C_1,C_2 \leftarrow CROSSOVER(P_1,P_2)$//***crossover operation***
7. $MC_1,MC_2 \leftarrow MUTATION(C_1,C_2)$//***mutation operation***
8. $F(P_i)$=Fitness of Parent
9. $F(MC_i)$=Fitness of Mutated Child
10. $Fitness=[L(T_i)+B(T_i)+R(T_i)]$ *// **fitness function***
11. $F(P)=MAX(F(P_1),F(P_2))$
12. $F(MC)=MAX(F(MC_1),F(MC_2))$
13. $L(P)=Length\ (P_1)+Length\ (P_2)$
14. $L(MC)=Length\ (MC_1)+Length\ (MC_2)$
15. If $((F(MC)>F(P)) \wedge ((L(P)<L(MC)))$
16. $Z \leftarrow Z\{MC_1,MC_2\}$
17. **else**
18. $Z \leftarrow Z\{P_1,P_2\}$

4.3 WORKING MODEL



The technique begins with an initial population of randomly generated test suites and then uses the GA to optimize towards a coverage criterion and determine the best test suite among the population. To address the generation of test suites, few specialized mutation operators are developed, at both method level and class level. These operators help in achieving the chromosome operations by altering or interesting of one or more parts of the chromosome to generate the children or the off springs. The fitness function determines the coverage criterion of the parent and the child chromosomes. The Coverage criterion focused here is branch coverage, close to boundary value and the likelihood to evaluate the fitness. For an example, a program may contain the control statements such as if or while, with logical predicates. The branch coverage criteria involves in evaluating these predicates to true or false value. Any branch may be considered as infeasible if there is no program input that determines the predicate that this particular branch is executed. After the Fitness of the chromosomes is determined, the secondary objective is to evolve a small test suite that helps the tester to easily add the test oracles. For this purpose, the length of the fit chromosomes is evaluated and best small chromosome is chosen and added to the new generation.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a technique for the whole test suite generation of object oriented programs using genetic algorithm. Though there are many techniques for the test suite generations, they do not concentrate well on object oriented programs. We also discussed that when generating test suites in object oriented program, it is necessary to take into consideration the information related to various dependencies arising due to the object- relationship such as inheritance, polymorphism and encapsulation. So for this purpose, we have used the mutation operators at both method level and class level. The algorithm proposed in this paper, facilitates the generation of test cases for object oriented programs when automatic oracles are not available.

Issues of future work include applying this technique on a real time test generation tool for large and complex programs. We also aim to compare this technique with the bee colony and ant colony algorithm.

REFERENCES

[1] R.Jeevarathinam,Antony Selvadoss Thanamani "Test Case Generation using Mutation Operators and Fault Classification,"International Journal of Computer Science and Information Security,pp.190 -195,2013.

[2] Gordon Fraser,Andrea Arcuri, "EvoSuite : Automatic Test Suite Generation for Objec-Oriented Software, "ACM September 2011.

[3] Juan Pablo Galeotti,Gordon Fraser,Andrea Arcuri, "Improving Search-Based Test Suite Generation With Dynamic Symbolic Execution, "2013.

[4] Gordon Fraser,Andrea Arcuri,Phil McMinn, "Test Suite Generation with Memetic Algorithms,"ACM,2013.

[5] Gordon Fraser,Andrea Arcuri,"EvoSuite:On The Challenges of Test Case Generation in the Real World ," 2012.

[6] N.Sivakumar,K.Vivekanandan,A.Mohan, "Role Oriented Test Case Generation for Agent Based System,"International Journal on Computer Science and Engineering,pp.133-142, 2013

[7] Bharti Suri,Isha Mangal and Varun Srivastava, "Regression Test Suite Reduction using an Hybrid Technique Based on BCO and Genetic Algorithm,"International Journal,pp.165-172.

[8] Ajitha Rajan "Automated Requirements-based test case generation",ACM SIGSOFT, Vol. 31, no. 6, pp. 1-2, 2006

[9] Gupta et al "Using GA for the Unit Testing of Object Orinted softwares", IJSSST,vol.10.no.3,pp. 99-104.

[10] B.Baudray,F.Fleurey,J-M.Jezequel and Y.Le Traon,"Automatic Test Cases Optimization: A Bateriologic Algorithm", IEEE software,vol.22,no.2,pp.76-82,Mar./Apr.2005.

[11] P.Godeford, N.Klarlund, and K.Sen, "DART: Directed Automated Random Testing", Proc.ACM SIGPLAN cofn.programming Language Design and Implementation, pp.213-223,2005

[12] K.Sen,D.Marinov, and G.Agha, "CUTE: A Concolic Unit Testing Engine for C", proc.10th European Software Eng.Conf.Held Jointly with 13th ACM SIGSOFT Int'l symp.Foundations of Software Engineering, pp.263-272,2005.

[13] B.Korel,"Automated Sodtware Test Data Generation", IEEE Trans.Softare Eng., vol.16,no.8,pp.870-879,Aug.1990.

[14] P.McMinn,"Search-Based Software Test Data Generation: A survey", Software Testing, Verification and Reliablity, Vol.2, no.2, pp.226, May 1976.

[15] A.Arcuri and L.Briand, "Adaptive Random Testing: An illusion of Effectiveness?" proc.ACM Int'l Symp.Software Testing and Analysis,2011.

[16] G.Frazer and A.Zeller,"Mutation-Driven Generation of Unit Tests and Oracles", IEEE Trans.Software Eng., vol.38, no.2, pp.278-292,Mar./Apr.2011.

[17] Y.Jia and M.Harman, "An Analysis and Survey of Development of Mutation Testing", Technical Report TR-09-06,CREST Centre, King's College London, Sept.2009.

[18] J.Malburg and G.Fraser, "Combining Search-Based and Constriant -Based Testing", Proc.IEEE/ACM 26th Int'l Conf.Automated Software Eng.,2011.

[19] S.Zhang, D.Saff, Y.Bu , and M.Ernst,"Combined Static and Dynamic Automated Test Generation", Proc.ACM Int'l Symp.Software Testing and Analysis,2011.

[20] Izzat Alsmadi, Faizal Alkhateeb, Eslam Al Maghayreh, Samer Samarah and Iyad Abu Doush, "Effective Generation of Test cases using Genetic Algorithm and Optimization Theory", JCC, ISSN, Jan 1 2010.

