ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Modified Genetic Algorithm for Efficient Regression Test Cases

Jyoti[1], Mrs. Lekha Bhambhu[2]

M.Tech, CSE, JCDM College of Engineering, Sirsa, India [1]

Asst Professor, CSE, JCDM College of Engineering, Sirsa, India [2]

Abstract: Regression testing is the process of validating modified software to detect errors that have been introduced into previously tested code. As the software is modified, the size of the test suite grows and the cost of regression testing increases. In this situation, test case prioritization aims to improve the effectiveness of regression testing by ordering the test cases so that most beneficial test cases are executed first. In this research paper, a modified genetic algorithm is introduced that will prioritize regression test suite within a time constrained environment on the basis of total fault coverage. The proposed algorithm has been automated and the results are analysed. The results representing the effectiveness of algorithm are presented with the help of Average Percentage of Faults Detected (APFD). Our objective is to achieve more efficiency than existing results.

Keywords: GA (Genetic Algorithm).

I. INTRODUCTION

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities. These modifications in the software may cause the software to work incorrectly and may also affect the other parts of the software, so to prevent this Regression testing is performed.

1.1 Regression Testing

Regression testing is frequently executed maintenance process used to revalidate modified software. Regression testing is a type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes such as enhancements, patches or configuration changes, have been made to them [8]. The intent of regression testing is to ensure that a change such as those mentioned above has not introduced new faults. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software [1]. Common methods of regression testing include rerunning previously completed tests and checking whether program behavior has changed and whether previously fixed faults have re-emerged [3]. Regression testing can be performed to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change. Regression testing can be used not only for testing the correctness of a program, but often also for tracking the quality of its output. For instance, in the design of a compiler, regression testing could track the code size, simulation time and compilation time of the test suite cases.

Regression tests can be broadly categorized as functional tests or unit tests. Functional tests exercise the complete program with various inputs. Unit tests exercise individual functions, subroutines, or object methods. Both functional testing tools and unit testing tools tend to be third-party products that are not part of the compiler suite, and both tend to be automated. A functional test may be a scripted series of program inputs, possibly even involving an automated mechanism for controlling mouse movements and clicks. A unit test may be a set of separate functions within the code itself, or a driver layer that links to the code without altering the code being tested. As the software is modified and new test cases are added to the test suite to test new or changed requirements and need to prioritize [4] or to maintain test-suite adequacy, the size of the test suite grows and the cost of regression testing increases [2]. Thus, Regression testing is an expensive, but important process in software testing.

1.2 Genetic Algorithms

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [3]. Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions [5]. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation [6]. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:
i. a genetic representation of the solution domain,
ii. A fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations.

1.3 Significance of the Research

The main disadvantage in regression testing is that it is an expensive process used to validate modified software. Approximately 50% of the software cost is involved in the maintenance phase so researchers are working hard to come up with best results by developing new Regression Testing techniques so that the expenditure made in this phase can be reduced to some extent. The industrial collaborators reports show that if there are approximately 20,000 lines of code, running the entire test cases requires seven weeks. Test case prioritization methods and processes are required, because: (i) the regression testing phase consumes a lot of time and cost to run, (ii) there is not enough time or resources to run the entire test suite, and, (iii) there is a need to decide which test cases to run first.

The regression test suite is typically large and needs a method to choose those test cases which will detect maximum or all faults at the earliest. The three broad categories for prioritization are Greedy algorithms, non-evolutionary algorithms and evolutionary algorithms. Evolutionary algorithms (EA) are chosen as they are global optimization methods and scale well to higher dimensional problems. They can be easily adjusted to the problem at hand and can be change and customized. Most of the implementations of evolutionary algorithms came from any of these three basic types: Genetic Algorithm (GA), Evolutionary Programming (EP) and Evolutionary Strategies (ES). All these are strongly related but independently developed. Among evolutionary techniques, the Genetic Algorithm, invented by John Holland in the 1960s at the University of Michigan, study the phenomenon of evolution and adaptation as it occurs in nature. They depend on the use of selection, crossover (recombination) and mutation operators. Genetic Algorithm works with a set of prefixed steps and thereby investigates the solution. Studies regarding the performance of metaheuristic algorithms led to several conclusions that have practical ramifications. The results of the empirical study show that the Additional Greedy, 2-Optimal, and Genetic Algorithms always outperform the Greedy Algorithm. The results produced by Hill Climbing show that the nature of the fitness landscape is multimodal. The results produced by the Genetic Algorithm indicate that it is not the best considered in all cases, but that, in most cases, the differences between the performance of Genetic Algorithm and that of the Greedy approach is significant. However, an analysis of the fitness function shows that there are situations in which it is important to consider the entire ordering and, for such cases, Greedy Algorithms are unlikely to be appropriate. Given their generality, the fact that Genetic Algorithms perform so well is cause for encouragement.

Genetic Algorithm is well suited for solving problems where solution space is huge and time taken to search exhaustively is very high. Another advantage of genetic algorithm is that it has ability to solve problems with no previous knowledge. Researchers are investigating on hybrid approaches to Regression Testing which include both regression test selection and test case prioritization. The approach being proposed here would be the use of Genetic Algorithm with another optimization technique to propose a hybrid technique for optimizing test cases and hence further reduce the cost of regression testing.

II. LITERATURE REVIEW

Gregg Rothermel et al [1] explained that the Test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness at meeting some performance goal. Various goals are possible; one involves rate of fault detection. The measure of how quickly faults are detected within the testing process. An improved rate of fault detection during testing can provide faster feedback on the system under test and let software engineers begin correcting faults earlier than might otherwise be possible. One application of prioritization techniques involves regression testing the retesting of software following modifications; in this context, prioritization techniques can take advantage of information gathered about the previous execution of test cases to obtain test case orderings.

In this work, they describe several techniques for using test execution information to prioritize test cases for regression testing, including:

1) Techniques that order test cases based on their total coverage of code components.
2) Techniques that order test cases based on their coverage of code components not previously covered
3) Techniques that order test cases based on their estimated ability to reveal faults in the code components that they cover.

They report the results of several experiments in which we applied these techniques to various test suites for various programs and measured the rates of fault detection achieved by the prioritized test suites, comparing those rates to the rates achieved by untreated, randomly ordered, and optimally ordered suites. Analysis of the data shows that each of the prioritization techniques studied improved the rate of fault detection of test suites, and this

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

improvement occurred even with the least expensive of those techniques. The data also shows, however, that considerable room remains for improvement. The studies highlight several cost-benefit trades-offs among the techniques studied. In this research, they have described several techniques for prioritizing test cases for regression testing and empirically examined their relative abilities to improve how quickly faults can be detected during regression testing. Their results suggest that these techniques can improve the rate of fault detection of test suites.

Zheng Li et al [2] assimilated the knowledge about the concept of Regression testing. Regression testing is an expensive, but important, process. Unfortunately, there may be insufficient resources to allow for the re-execution of all test cases during regression testing. In this situation, test case prioritization techniques aim to improve the effectiveness of regression testing by ordering the test cases so that the most beneficial are executed first. Previous work on Regression test case prioritization has focused on Greedy Algorithms. However, it is known that these algorithms may produce suboptimal results because they may construct results that denote only local minima within the search space. By contrast, metaheuristic and evolutionary search algorithms aim to avoid such problems. Author presents results from an empirical study of the application of several greedy, metaheuristic, and evolutionary search algorithms to six programs, ranging from 374 to 11,148 lines of code for three choices of fitness metric. They explained the problems of choice of fitness metric, characterization of landscape modality, and determination of the most suitable search technique to apply. The empirical results replicate previous results concerning Greedy Algorithms. The results also show that Genetic Algorithms perform well, although Greedy approaches are surprisingly effective, given the multimodal nature of the landscape. They described five algorithms for the sequencing problem in test case prioritization for regression testing. They presented the results of an empirical study that investigated their relative effectiveness.

R.Krishnamoorthi et al [3] explained the importance and effectiveness of Regression testing. Regression testing is an expensive, but important process in software testing. Unfortunately, there may be insufficient resources to allow for the re-execution of all test cases during regression testing. In this situation, test case prioritization techniques aim to improve the effectiveness of regression testing by ordering the test cases so that the most beneficial are executed first. In this paper they proposed a new test case prioritization technique using Genetic Algorithm (GA). The proposed technique prioritizes subsequences of the original test suite so that the new suite, which is run within a time-constrained execution environment, will have a superior rate of fault detection when compared to rates of randomly prioritized test suites. This experiment analyzes the genetic algorithm with regard to effectiveness and time overhead by utilizing structurally-based criterion to prioritize test cases. An Average Percentage of Faults Detected (APFD) metric is used to determine the effectiveness of the new test case orderings.

Arvinder Kaur et al [4] has been assimilated the knowledge about Regression testing and the techniques for implementation. Regression testing is a testing technique which is used to validate the modified software. The regression test suite is typically large and needs an intelligent method to choose those test cases which will detect maximum or all faults at the earliest. Many existing prioritization techniques arrange the test cases on the basis of code coverage with respect to older version of the modified software. In their approach, a new Genetic Algorithm to prioritize the Regression test suite has been introduced that will prioritize test cases on the basis of complete code coverage. The genetic algorithm has also automated the process of test case prioritization. The results representing the effectiveness of algorithms are presented with the help of an Average Percentage of Code Covered (APCC) metric. The algorithm has been proposed to prioritize test cases using Genetic Algorithm. Here, different prioritization approaches have been analyzed, namely: total fault coverage with in time constrained environment and amount of code coverage on different examples and their finite solution obtained, respectively. Through Genetic Algorithm technique, an approach has been identified to pull out suitable population, which was further formulated by Genetic Algorithm operations to make it more flexible and efficient. The elaborations of results are shown with the help of APCC values. The APCC has been calculated for example for code coverage testing to evaluate the usefulness of the proposed algorithm.

Wang Jun et al [5] explained software testing concept. With the rapid development of information technology, software testing, as a software quality assurance, is becoming more and more important. In the software life cycle, each time the code has changed there needs to be regression testing. The huge test case library makes running a full test case library being challenged. To this end, they designed a genetic algorithm-based test case prioritization algorithm and improved the genetic algorithm proposed software test case prioritization algorithm.

Liang You et al [6] explained the cost incur in software testing. After the programmer fixes the bugs and enhances the functionality of the software project, regression testing reruns the regression testing suite to ensure that the new version software projects can run smoothly and correctly. Because the regression testing is the most expensive phase of the software testing, regression testing reduction eliminates the redundant test cases in the regression testing suite and saves the cost of the regression testing.

Author formally defines the time-aware regression testing reduction problem. They also propose a novel genetic algorithm for the time-aware regression testing reduction problem. They define the representation and fitness

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

function of the genetic algorithm; Algorithm also describes the parent selection, crossover and mutation operator of the genetic algorithm. The novel algorithm not only removes redundant test cases in the regression testing suite but also minimizes the total running time of the remaining test cases. Finally, the paper evaluates the genetic algorithm using eight example programs. The experimental result illustrates the efficiency of the proposed genetic algorithm for the time-aware regression testing reduction problem.

III. OBJECTIVES

This section will cover the topic problem formulation, objective, methodology and results of Simulation. The problem definition section explains the existing problem in the test cases prioritization and explains the factor needs to be considered. After analyzing the problems, the objectives and methodology will be explained for implementing the Genetic Algorithm. This algorithm will be implementing in .Net Technology with Framework 4.0 and VC# Programming Language.

In software testing, tests cases need to be generated for detecting the errors in software. But in regression testing, the modified software needs to be tested for identifying the errors and to validate them. Once the software has been modified, the test cases need to be executed again but now the number of test cases has been increased and there is no need to check every test case as in this situation, the cost of regression testing will be increased in terms of time. So as per our literature survey we have analyzed the following Problems

i. To select and prioritize regression test suite within a time constrained environment
ii. To cover the total fault coverage.
iii. Analyse drawbacks in existing technique.
iv. To implement Modified Genetic Algorithm (GA) for test cases that will cover major risks / faults in minimum time.
v. To improve the success rate of Genetic Algorithm (GA).
vi. Compare the efficiency of improved Genetic Algorithm with existing Results..

IV. PROPOSED METHODOLOGY

i. Study of existing Genetic Algorithm.
ii. Research on working of Genetic Algorithm Steps.
iii. Flow Development of new research and its Implementation in C#.net language
iv. Analysis of results.
v. Analysis the benefits of Genetic Algorithm.

REFERENCES

[1] Zheng Li, Mark Harman, and Robert M. Hierons, “Search Algorithms for Regression Test Case Prioritization”, Transactions on Software Engineering, Vol. 33, No.4, 2007.
[2] Harrold, M.J. “Retesting software during development and maintenance”, IEEE, Page(s):99 – 108, 2008.
[3] R.Krishnamoorthi, S.A.Sahaaya, Arul Mary, “Regression Test Suite Prioritization using Genetic Algorithms”, International Journal of Hybrid Information Technology Vol.2, No.3, 2009.
[4] G. Rothermel, “Prioritizing Test Cases For Regression Testing”, IEEE Transactions On Software Engineering, Vol. 27, No. 10, 2011.
[5] Arvinder Kaur, ShubhraGoyal, “A Genetic Algorithm for Regression Test Case Prioritization Using Code Coverage”, International Journal of Computational Science and Engineering, Vol. 3 No. 5, 2011.
[6] Wang Jun, Zhuang Yan, “Test Case Prioritization Technique based on Genetic Algorithm”, IEEE, pg. 173 – 175, 2011.
[7] Liang You Yansheng Lu, “A Genetic Algorithm for the Time-Aware Regression Testing Reduction Problem”, IEEE, Page(s):596 – 599, 2012.
[8] http://en.wikipedia.org/wiki/Regression_testing