Software Testing and Test Case Optimization: Concepts and Trends

Vinita Tomar and Mamta Bansal

Abstract Software testing aims at identifying the defects and errors in any developed system. Software testing has been viewed as an optimization problem, indicating that various optimization approaches might be used to address software testing challenges. This research paper discusses the significance of software testing and concepts of test case optimization under regression testing. It describes many forms of regression testing and their approaches and identifies several gaps. A literature survey has been done on test case optimization in regression testing, including test case generation, test case prioritization, and test case minimization. A new methodology is also suggested to overcome the gaps.

Keywords Software testing · Regression testing · Optimization · Test case generation · Minimization · Prioritization

1 Introduction

Software testing is a procedure for thoroughly testing the runtime and software quality. It is also known as the process of verifying and validating an application under test to detect bugs and errors. Basic testing of the software is carried out in the environment in which the software is intended to be used. It checks correct and effective output during operation. The test should systematically discover different types of errors in the shortest time and with the least workload. Testing must prove that the software seems to be running according to specifications. The information gathered during testing can be used to determine the software's reliability and quality. The test case must be made in order to improve software quality.

Software testing can be performed manually or automatically. Manual testing entails manually testing software without using any automated methods, whereas automated testing entails creating scripts and using software to test the system.

V. Tomar · M. Bansal (✉)
Shobhit Institute of Engineering & Technology (Deemed-to-be University), Meerut, U.P, India
e-mail: mamta.bansal@shobhituniversity.ac.in

P. K. Mallick et al. (eds.), *Electronic Systems and Intelligent Computing*, Lecture Notes in Electrical Engineering 860, https://doi.org/10.1007/978-981-16-9488-2_50

Software development and planning are incomplete without testing. It's used at all stages of the software development process. It is usually divided into the following different stages: requirements analysis, test planning, test development, test execution, evaluation exit criteria, and test closure. The software testing method is introduced to aid in the early detection of software defects, as the cost of modifications in the maintenance phase rises if errors are identified later in the SDLC [1].

Functional and non-functional testing are two basic categories of software testing [2]. Functional testing involves testing the functionality of software applications, whereas non-functional testing tests the non-functional features of the application such as performance, reliability, usability, security, and many more.

Software testing is aimed at ensuring the application software matches the users' specifications. Good Test coverage is required to ensure the application software runs as expected. To generate coverage lists, the test cases need to be designed to provide maximum chances of finding various errors or bugs. The effectiveness of the test cases is measured by the number of defects or errors reported. This is done by using heuristic techniques. The automated generation of test data helps to reduce costs and time when developing test cases.

2 Regression Testing

When a modification is made to a software application, it is possible that other portions of the application may be affected. Regression testing is utilized to make sure that fixed bugs did not result in violations of other functions. Regression testing is a form of software testing in which test cases are re-executed to ensure that the previous functions of the application are still functional and that the new changes haven't introduced any mistakes. The most important aspect of the software development life cycle is regression testing [3]. The goal of regression testing is to make sure that changes, such as a bug fix, should not cause another bug to be found in the application. It is carried out by selecting suitable test cases from the test suite. The test cases cover the modified and affected sections of the code. Test cases are usually automated because they need to be executed again and again, and manually running the same test case is a time-consuming task. Prioritization of regression test cases is one of the most essential tasks in the regression testing process [4]. Unit regression, partial regression, and complete regression are three categories of regression testing.

2.1 Regression Testing Techniques

Regression testing can be classified as follows:

1. **Test case selection**: It re-executes a test from the test suite. It will not be necessary to re-execute the complete suite. The test cases are chosen based on the module's code changes.
2. **Test case minimization**: It is used to minimize test costs in terms of execution time, resources time, etc. Test suite minimization is the optimization task of identifying the smallest subset of test cases in the suite that satisfy similar coverage requirements as the original suite.
3. **Test case prioritization**: It prioritizes test cases in the test suite according to many parameters such as code coverage, risk modules, functions, and features, among others.

3 Optimization in Software Testing

The term "optimize of the testing process" means the practice of making the testing process faster while maintaining its correctness. The test optimization process can be carried out by altering how the test cases are executed, such as running the tests in the optimal sequence or executing the tests that cover the changes in the build. Optimization is the process of accomplishing certain goals using minimal resources and better methods than other contemporary methods. To ensure software quality, optimizing test suites from the start is essential. There are several effective ways for optimizing test suites for efficient outcomes. An optimization algorithm is a search method whose goal is to find the best solution to a problem to satisfy one or make an objective function that may be subject to a set of constraints. It is a way to find the maximum or minimum value of a function or process. It can either refer to maximization or minimization.

4 Significance of Software Testing

The significance of software testing are as follows:

1. Early software testing saves both time and cost.
2. It helps to verify that all the software requirements are implemented correctly or not.
3. It helps in determining the functionality, reliability, usability, efficiency, maintainability, and portability of the software.
4. It improves the consistency and performance of a software product or application.
5. The primary goal of testing is to identify and eliminate defects and errors.

6. It helps developers and testers to compare actual results with expected results to improve the quality.
7. It gives confidence in the quality of the final product. High-quality products delivered to customers help win their trust.
8. It ensures that the stored and processed data is protected from unauthorized access and hacker intrusion.
9. The test makes it easier to add new features to the application or products.
10. A good test allows other developers to better understand the purpose of the code.

5 Literature Survey

In the field of software testing optimization, numerous researches have been conducted. Some of them are mentioned below.

Dhareula et al. [3] proposed a flower pollination approach for regression testing test case prioritizing. Only those test cases were included in this study that covered the most defects in the shortest amount of time. The results were validated using the average proportion of fault detected metrics. It was concluded that the flower pollination algorithm's stopping criteria for test case prioritization were found to be largely dependent on the nature and size of the application.

Hourani et al. [6] considered the Artificial Intelligence main features that can be used in software testing. The result showed that AI-driven testing will be the better option for quality assurance work as artificial intelligence can give better results in software testing than other techniques. It was concluded that artificial intelligence software testing will enhance the quality of the software and organizational efficiency will also be improved.

Durelli et al. [7] provide all the information on the research for machine learning and software testing. Machine learning algorithms are mostly utilized for test case generation, refining, and evaluation, according to their research work. Also, software testing operations are automated using machine learning techniques.

Shrivathsan et al. [8] proposed two fuzzy-based clustering techniques for test case prioritization. The novel similarity coefficient and dominancy measure were used by the researchers. They used the arithmetic sum-product method to assess. The results of the testing revealed that the proposed algorithms improve the chances of choosing more related tests. Although it all depends on the cluster size, this strategy increased the efficacy of test case prioritization techniques. It was concluded that the regression testing process can be successfully handled by grouping test cases.

Preethi et al. [9] proposed the automated test case generation system that generates the test cases automatically. It also evaluates the test cases and gives the test reports. Researchers have used a genetic algorithm that generated test cases using the fitness value function. They have also used classification and regression trees algorithm to generate test cases reports. The results showed that this automated system can be used

on all types of testing. They have suggested developing a system for non-functional testing types in future work.

Ahmad et al. [5] proposed a combination of two techniques. Researchers first created test cases based on prioritized test cases based on test variables and then used the ant colony optimization technique to calculate the best sequence with the shortest execution time and highest defect rate. It was concluded that by combining these two methods, they got the best solution for the problem. They have suggested implementing the Ant colony optimization-based algorithm and to compare with other search-based algorithms in the future.

Panwar et al. [10] proposed a Cuckoo search algorithm and an improved ant colony optimization algorithm to summarize test cases in an optimized sequence in a time-constrained environment. Researchers worked on nature-inspired optimization algorithms and searched for better optimization for the test case prioritization. Cuckoo search is more effective and easier to implement than other optimization techniques as it just uses one parameter. It was concluded that the proposed hybrid cuckoo algorithm was shown to be extremely beneficial for prioritizing test cases.

Khari et al. [11] generated a testing tool for automated use that consists of two key automated software testing components. The first is the creation of a test suite, and the second is the optimization of a test suite. For automated defect detection, this optimized test suite is used. The researchers used an artificial bee colony and cuckoo search algorithm. The researcher in this study gave a set of minimal test cases. They also used maximum path coverage.

Manivasagam et al. [1] proposed an optimization technique. This method is used to select potential features. It will make software defect prediction more accurate. Using metaheuristic search, the researchers have utilized a technique based on fuzzy mutual information. The results revealed an improvement in routine prediction for three different classifiers utilized in the study. It was concluded that the proposed optimization technique has greater accuracy in terms of predicting software defects.

Ozturk et al. [12] suggested a bat-inspired test case prioritization algorithm. The researchers adopt several features of the algorithm, which include test execution time and code errors, to the notion of the algorithm while constructing it. The proposed method was then compared to the existing methods and found to be the most effective. And due to test case complexity, the code increases, and the average percentage drop in fault detection in this method is less than the drop produced by the other four compared by four algorithms.

Babbar [2] in this study described the common test cases and test techniques used for error detection. He discussed unit testing, integration testing, and system testing as well as other forms of software testing methodologies. He has also described in detail testing techniques namely black box testing and white box testing along with their advantages.

Luo et al. [4] conducted a thorough investigation and compared the efficacy and a likeness of defects for static as well as dynamic test case prioritization techniques at various test case granularities. The results revealed that at the test case level, a static methodology outperformed a dynamic technique, whereas dynamic outperformed

static at the test method level. It was concluded that that test case prioritization techniques perform differently in different disciplines.

6 Scientific Gaps

In the existing work, the emphasis was given to optimizing test cases for maximum path coverage. Coverage of the critical path is more significant than code coverage percentage in quality testing. The existing test case optimization techniques lack costs and benefits when test suites have many critical paths that need to be prioritized. There is a lack of multi-objective optimization algorithms that can focus on both coverage, criticality, and resources simultaneously. Existing optimization techniques include Genetic Algorithm, Ant Colony Algorithm, Particle swarm Algorithm, and Firefly. However, as there have been a lot of improvements been done in this area, newer algorithms must also be explored. The previous research used code coverage as a criterion, which differs from defect detection criteria. When used to fault detection-based prioritization problems, metaheuristic algorithms may produce a variety of outcomes.

7 Methodology

The methodology for test case optimization for proposed work will be as follows:

Step 1 To search for novel optimization techniques. The extensive study of the metaheuristic algorithms will determine the complexity, challenges, opportunities, and limitations of each algorithm to determine the sustainability of the algorithm for the proposed work.

Step 2 To create the test cases and minimize the dimensions of test suites by disposing of excess test cases. Assess the optimum test coverage requirement and prioritize the test cases.

Step 3 To compute the values of metrics and assess the efficacy of the proposed test case optimization for regression testing with existing approaches.

Figure 1 shows a flow chart for optimizing test case generation, prioritization, and minimization of proposed work:

8 Conclusion

Every software development and planning process is incomplete without software testing. The objective of software testing is to detect defects and errors in a developed system that has already been developed. Various regression testing techniques and

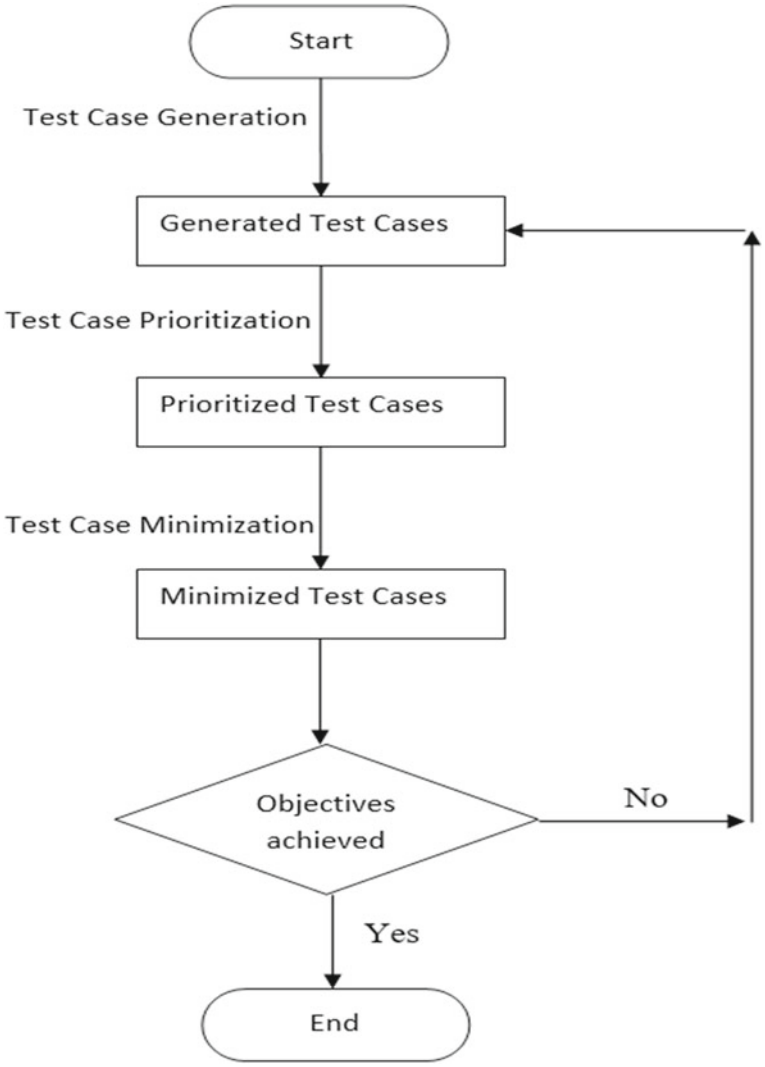

Fig. 1 Optimization of test case generation, prioritization, and minimization

the importance of software testing are explained in this research paper. Several gaps have been identified using a literature survey. The existing test case optimization techniques lack costs and benefits when test suits have many critical paths that need to be prioritized. There is a lack of multi-objective optimization algorithms. This paper describes the proposed methodology for the optimization techniques by extensive study of metaheuristic algorithms. The value of metrics will be computed and the efficacy of the proposed test case optimization for regression testing will be compared with the existing approaches.

References

1. Manivasagam G, Gunasundari R (2017) An optimized feature selection using fuzzy mutual information-based ant colony optimization for software defect prediction. Int J Eng Technol 456
2. Babbar H (2017) Software testing: techniques and test cases. Int J Res Comput Appl Robot 5:44–53
3. Dhareula P, Ganpati A (2020) Flower pollination algorithm for test case prioritization in regression testing. ICT Anal Appl
4. Luo Q, Moran K, Poshyvanyk D (2016) A large-scale empirical comparison of static and dynamic test case prioritization techniques. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundation of software engineering
5. Ahmad SF, Singh DK, Suman P (2018) Prioritization for regression testing using ant colony optimization based on test factors. Intell Commun Control Dev 1353–1360
6. Hourani H, Hammad A, Lafi M (2019) the impact of artificial intelligence on software testing. In: IEEE Jordan international joint conference on electrical engineering and information technology
7. Durelli VHS, Durelli RS, Borges SS, Endo AT, Eler MM, Dias DRC, Guimaraes MP (2019) Machine learning applied to software testing: a systematic mapping study. IEEE Trans Reliabil 1–24
8. Shrivathsan AD, Ravichandran KS, Krishankumar R, Sangeetha V, Ziemba SKP, Jankowski J (2019) Novel fuzzy clustering methods for test case prioritization in software projects. Symmetry 11:1400
9. Preethi MBM, Aishwarya MR, Pradeesh MP (2019) Automated test case generation using data mining. Int Res J Eng Technol (IRJET) 06
10. Panwar D, Tomar P, Singh V (2018) Hybridization of Cuckoo-ACO algorithm for test case prioritization. J Stat Manag Syst 539–546
11. Khari M, Kumar P, Burgos D, Crespo RG (2017) Optimized test suits for automated testing using different optimization techniques. Soft Comput
12. Ozturk MM (2017) A bat-inspired algorithm for prioritizing test cases. Vietnam J Comput Sci 45–57