An Adaptive Approach For Test Case Prioritization In Regression Testing Using Improved Genetic Algorithm

[1]Jagpuneet Kaur Bajwa, [2]Ramanjeet Kaur

[1,2]Department of Computer science, Punjabi University, Patiala, Punjab, India
[1]jagpuneetbajwa@gmail.com, [2] ramanjeet.kaur13@gmail.com

Abstract

Test case prioritization technique is used to prioritize and schedule test cases. The technique is developed in order to run test cases of higher priority in order to minimize time, cost and effort during software testing phase. Prioritizing test cases can be done on the basis of requirements, cost of bug fixing, history of the parent device etc. In this paper we have proposed the hybrid technique. This hybrid technique is a combination of adaptive approach and genetic algorithm. The speedup is achieved by using an adaptive approach which schedules the test cases simultaneously during the execution of test cases.

Keywords: *Test case prioritization, Regression testing, Genetic algorithm, Software testing*

1. Introduction

Software testing is one of the most important activities of SDLC. The software testing is the process to evaluating the quality of software. The software testing is done to find as many errors or bugs in the software as possible in order to ensure that the software being developed is of good quality. It can also be defined as a verification and validation process. Verification means that the software meets the required conditions at the starting of the development phase. Validation means that the software meets the required conditions at the end of the development phase i.e. the software is developed as per customer requirements.

Other areas within the application have been affected, when a change in a software application is made. Regression testing is performed to check that affixed bug has not resulted in functionality. The intention behind performing regression is to ensure that a change, such as a bug fix should not result in another fault or error bring uncovered in the application.

However, regression testing can be a costly and time consuming process. This may be the case when the test suite required to test the system is large, there is some manual element in testing or test execution takes up significant resources. So it becomes essential to reduce the cost and time expenses involved in regression testing.

One way to reduce the cost of regression testing is to select a subset of original test suite such that the selected subset achieves total code coverage in minimum time. Such methods that assist in this selection are known as regression test suite optimization methods. Common methods include test case prioritization, test case selection and test suite minimization. Test case prioritization attempts to arrange the test cases so as to improve the rate of fault detection. Test case selection selects a subset of test cases for execution. Finally, Test suite minimization aims to compress a test suite that still maintains the coverage. Among these techniques, test case prioritization is known to be most efficient since it works on all test cases in a test suite and identifies the best test execution sequence that meets a certain testing criteria. On the other hand, test case selection and test suite minimization are comparatively inefficient since they do not cover all the test case due to which, there is a risk that the software contains undetected errors. Therefore, most of the work has been done in the field of test case prioritization.

2. Literature review

Yuejian Li*et al.* (1999) in their paper presented a summary of regression testing in the following areas: types of regression testing, regression testing of global variables, regression testing of object-oriented software, and comparison of selective regression techniques and cost comparisons of the types of regression testing.

S. Yoo*et al.* (2010) in their paper analyzed the areas of regression test minimization, selection and prioritization. The survey revealed an increasing trend in providing support for these techniques, particularly prioritization. It also described the challenges of providing an automated support for regression testing practices in order to promote practical implementation.

Gregg Rothermel*et al.* (1999)in their paper presented several techniques for prioritization of test cases and reported empirical results that measured the effectiveness of these techniques for improving fault detection rate. The results provided an insight into the tradeoffs among various techniques for test case prioritization.

Hema Srikanth*et al.* (2005) in their paper proposed a Prioritization Of Requirements for Test(PORT) technique which prioritized test cases on the basis of four factors: requirements volatility, customer priority, implementation complexity and fault proneness of the requirements. They also conducted a PORT case study on four student developed projects in advanced graduate software testing class. Results showed that PORT prioritization at the system level improved the rate of severe fault detection. Also, customer priority was found to be one of the most important prioritization factors contributing to the improved rate of fault detection.

Paolo Tonella*et al.* (2006) in their paper proposed a TCP approach that took advantage of user knowledge through a machine learning algorithm, known as Case Based Ranking(CBR). The proposed approach worked by integrating user knowledge with multiple prioritization indexes.

The results indicated that CBR overcome the previous approaches and also provided a near-to-optimal solution for moderate suite size.

Xiaofang Zhang *et al.* (2007) in their paper proposed a new TCP technique The case study defined 'rate of units-of-testing-requirement-priority-satisfied-per-unit-test-case-cost' could be increased, and then the testing quality and customer satisfaction can be improved.

Yu-Chi Huang *et al.* (2010) in their paper proposed a cost-cognizant prioritization technique that ordered test cases according to their history information by using genetic algorithm. The technique prioritized test cases on the basis of their test costs and fault severities, without analyzing the source code. It also improved the prioritization performance by avoiding certain scenarios where the test cases with equivalent ability in the previous regression testing were given the same rank. The efficiency of the same was evaluated by using a UNIX utility program and the results confirmed the usefulness of the proposed technique.

Yogesh Singh *et al*. (2010) in their paper presented a regression test prioritization technique based on ACO for reordering of test suites in a time constrained environment. The proposed approach was then compared with other approaches and the results showed that prioritization achieved using ACO led to an ordering which was comparable to optimum ordering.

Md. Imrul Kayes (2011) in his paper presented a new metric for assessing the rate of fault dependency detection and an algorithm for prioritization of test cases. The effectiveness of this prioritization using the new metric was shown by comparing it with non-prioritized test cases and it was observed that prioritized test cases were more effective in detecting dependency among faults.

Dusica Marijan*et al.* (2013) in their paper presented a case study of a test case prioritization approach ROCKET for improving the efficiency of continuous regression testing of industrial video conferencing software. ROCKET ordered the test cases according to historical failure data, test execution time and domain-specific heuristics. The results showed that the test cases prioritized using ROCKET provided faster fault detection and also increased the regression fault detection rate as compared to manually prioritized test cases.

Chu-Ti Lin*et al.* (2013) in their paper presented a software version-aware approach that considered both source code information and historical fault data. The proposed approach was evaluated using Siemens program (an experimental subject) and the results indicated the proposed approach to be better than existing TCP approaches in terms of fault detection, as given by its APFD value.

Md. Junaid Arafeen*et al.* (2013) in their paper carried out an analysis of requirements-based clustering approach to know whether it could help in improving the effectiveness of TCP techniques. To investigate the same, an empirical study was conducted using two Java programs with multiple versions and requirements documents. The results indicated the use of requirement information to be beneficial during the TCP process.

Daniel Di Nardo *et al*. (2013) in their paper presented an industrial case study of coverage-based prioritization techniques on a real world system with real regression faults. It evaluated four common and different test case prioritization techniques and examined the effects of using various coverage criteria on the fault detection rates of the prioritized test suites. The results indicated that the prioritization techniques based on additional coverage with finer grained coverage criteria perform significantly better in fault detection rates. It also revealed that using modification information does not significantly enhance fault detection rates.

Dan Hao*et al.* (2013) in their paper presented an adaptive TCP approach, which works by determining the test case execution order simultaneously during the execution of test cases on the modified program. The proposed approach selected the test cases on the basis of their fault detection capability, which was further calculated from the output of selected test cases. The effectiveness of the approach was evaluated by conducting an experimental study on 8 C programs and 4 java programs. The results showed that the proposed adaptive approach to be significantly better than the total test case prioritization approach and comparable to additional statement-coverage based test case prioritization approach. In fact, the adaptive approach came out to be better than the additional approach on some subjects.

Md. Saeed Siddik *et al.* (2014) in their paper presented an effective TCP framework which took software requirements spec, design diagrams, source code and test cases as input and provided a prioritized order of test cases using their collaborative information as output. The proposed framework was validated using an academic project and the results indicated that the use of collaborative information during prioritization process was beneficial.

Manika Tyagi *et al.* (2014) in their paper proposed a three-phase approach for TCP. In the first phase, redundant test cases were removed using a simple matrix operation. During the second phase, test cases were selected from the test suite by using Multi objective Particle swarm optimizer technique. Finally in third phase, test cases obtained from the second phase were assigned priority. Priority was obtained by calculating the ratio of fault coverage to the execution time of test cases. The proposed technique was then compared with other approaches and it was found that MOPSO achieved max fault coverage, max APFD value and min. execution time.

Dongdong Gao *et al.* (2015) in their paper proposed an algorithm based on ACO for prioritization of test cases. They considered three factors and used them in ACO algorithm for detecting severe faults in the early stage of regression testing process. The three factors were number of faults detected, exec time and fault severity. The effectiveness of proposed approach was calculated on the basis of APFD value and the results which were obtained clearly proved its success in optimizing the test case orderings in an efficient way.

Tanzeem Bin Noor*et al*. (2015) in their paper defined a set of metrics that estimated the quality of test cases using their similarity to previously failed test cases. The effectiveness of these metrics was evaluated by conducting several experiments on five real world open source software systems with real faults. The results revealed the proposed similarity-based quality measure to be significantly more effective for prioritizing test cases as compared to existing test case quality measures.

Avinash Gupta*et al.* (2015) extended the history-based approach for prioritizing the test cases to modified lines. The modified lines were prioritized first and then subsequently followed by the test cases. The results showed that the proposed approach was able to detect faults faster than the previous approach with less effort as compared to previous approach.

Dario Di Nucci*et al*. (2015) in their paper proposed a hyper volume-based genetic algorithm to solve multi-criteria TCP. The results proved the cost-effectiveness of HGA than additional greedy algorithm on large systems and on average required 36% of the execution time required by the additional greedy algorithm.

Kamna Solanki*et al.* (2015) in their paper proposed a modified version of ant colony optimization for test case prioritization, known as m-ACO. The performance of the proposed algorithm was evaluated by using Average Percentage of Faults Detected.

Hongda Wang*et al.* (2015) in their paper proposed a new regression test selection approach by considering the internal structure and fault propagation behavior of activity in service-oriented workflow applications. The proposed approach was compared with conventional approaches and it was found that the approach achieved higher fault detection rates. However, the proposed approach had a narrow perspective and mainly focused on regression test prioritization problem of service-oriented workflow applications in static testing environment.

Surendra Mahajan et al. (2015) in their paper developed and proved the necessity of Component-Based Software testing prioritization framework which uncovered more extreme bugs at an early stage and enhanced software product deliverable quality utilizing Genetic Algorithm with java decoding technique. For this, they proposed a set of prioritization keys to plan the proposed Component-Based Software java framework. This paper demonstrated how software testing could be efficient with management of data integrity factor to avoid major security issues. One of the main advantages of our approach was that domain specific semantics could be integrated with the data quality test cases prioritization, thus being able to discover test feed data quality problems beyond conventional quality measures.

Fang Yuan *et al*. (2015) in their paper analyzed the application of epistatic domains theory in genetic algorithms for TCP, where Epistatic Test Case Segment (ETS) is defined. In addition, two refined crossover operators were proposed, which focused on the change within ETS. The empirical studies proved the GA with proposed crossover operators to be better in terms of effectiveness and efficiency.

Lachana Ramingwong *et al.* (2015) in their paper proposed an algorithm to prioritize test cases based on total coverage using a modified genetic algorithm. The performance of the proposed algorithm on the percentage of condition covered and execution time was then compared with five other approaches and the results indicated that the proposed algorithm was better than other approaches.

Ahlam Ansaria *et al.* (2016) in their paper proposed an optimized TCP technique using ACO to reduce the cost, effort and time taken to perform regression testing and to uncover maximum faults.

3. EXISTING TECHNIQUES

Genetic Algorithm

Genetic Algorithm is a search-based optimization technique that generates optimal or near-optimal solutions to difficult problems by imitating the process of biological evolution. In order to accomplish this, it repeatedly modifies a random population of individuals, represented by chromosomes towards a better solution. During each step, it chooses individuals from the population in accordance to some fitness function of the problem under consideration. The two best fit individuals selected then act as parents to produce new off-springs. For thispurpose, they undergo modifications in the form of following genetic operators:

Crossover: Crossover operator is used to introduce variation among chromosomes belonging to successive generations in such a manner that the new chromosome obtained after the crossover operation is superior over the original chromosomes. Basically, it mimics the natural selection process by taking two or more chromosomes as parents and then generating a child chromosome from them. Figure 1 below explains the process of one-point crossover, in which a random crossover point is chosen in each parent chromosome. Everything beyond that point is then swapped for obtaining off-springs.

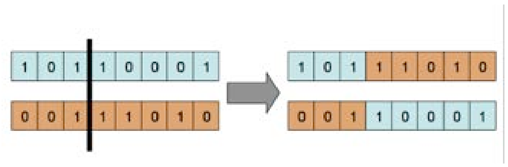

Fig 1: One-point crossover

Mutation: Mutation operator is used to insert a small tweak in the chromosome in order to produce a new solution. The solutions obtained after applying mutation can be completely different from the previous solution. Figure 2 below explains the process of bit-flip mutation, in which one or more bits are randomly selected and then flipped.

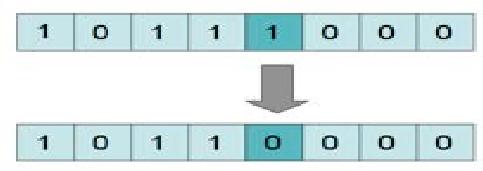

Fig 2: Bit-Flip mutation

4. **PROBLEM FORMULATION**

Regression testing is a type of testing aimed at finding out new errors after the modification of software. In other words, it assures that no additional errors were introduced in the process of fixing other problems and the software still works as it did before. However, executing the entire test suite for this purpose can be expensive in terms of cost and time. Therefore, it is essential to reduce the expense involved in regression testing. One way to achieve this is by test case prioritization. Test Case Prioritization aims to order the test cases so as to maximize the rate of fault detection. This topic has been a major subject of research for many years and many techniques have been proposed for achieving the same. Popular among these are genetic algorithm, ant colony optimization, bee colony optimization and particle swarm optimization.

Although these techniques are able to efficiently prioritize the test cases, but take significant amount of time to do so. For example in case of genetic algorithm, a large population of candidate solutions has to be repeatedly evolved in an iterative manner for reaching the best solution. When the population is large i.e. when there are some 100 or 1000 test cases, then genetic algorithm may consume a large proportion of time to prioritize the test cases. To remove this drawback, a new hybrid technique has been proposed which speeds up the time taken to prioritize the test cases. This hybrid technique is a combination of adaptive approach and genetic algorithm. The speedup is achieved by using an adaptive approach which schedules the test cases simultaneously during the execution of test cases.

5. **RESEARCH METHODOLOGY**

Test Case Prioritization using Genetic Algorithm yields excellent results. This is so because it uses the techniques inspired by the natural selection for generating the solutions. But the main drawback with Genetic Algorithm is that it consumes too much time to complete this activity. The reason behind this is the isolation of prioritization and execution processes i.e. test cases are executed only after they have been prioritized. On the contrary, an adaptive approach carries out both the processes side by side. Since both processes occur simultaneously, the time expenses are minimized to a great deal. But the problem with this approach is that it does not schedule the order of all the test cases contained in the test suite. It only prioritizes those test cases which have attained some amount of statement coverage in the past. On the other hand, the test cases which have been unable to achieve statement coverage are left non-prioritized, which implies that 100% statement coverage has not yet been achieved. As a result, a hybrid approach has been proposed in this paper, which is a combination of Genetic Algorithm and Adaptive approach. This approach works by initially employing the adaptive approach for the prioritization of those test cases which have achieved statement coverage on the previous program. Further, the test cases with no statement coverage i.e. leftover test cases are prioritized using Genetic Algorithm. This is done with the help of four operations: parent selection, crossover, mutation and duplicate elimination.In this manner, the hybrid technique succeeds in overcoming the constraints of both techniques. Thus other than saving time, the proposed approach additionally achieves more statement coverage.

6. CONCLUSION

In this paper, a hybrid approach to test case prioritization has been presented with a view to combat the issues involved in regression testing. A combination of adaptive approach and Genetic Algorithm, the proposed approach firstly utilizes the adaptive approach for the prioritization of test cases. It works by selecting a test case with the largest priority. Then it runs that test case and records its output. On the basis of this output and the execution history of next unselected test case, it prioritizes the next test case. This process terminates when all the test cases that cover code statements have been prioritized and executed. As far as test cases with zero statement coverage are concerned, Genetic Algorithm prioritizes them using four operations: parent selection, crossover, mutation and duplicate elimination. The performance of the proposed approach will be analyzed and compared with the existing algorithm.

7. REFERENCES

[1] Y. Li,Nancy JWahl: “An Overview of Regression Testing.”ACM SIGSOFT Software Engineering Notes, vol. 24, no. 1, (1999)

[2] S. Yoo,M. Harman: “Regression testing minimization, selection and prioritization: a survey.”Software Testing, Verification and Reliability (2010)

[3] G. Rothermel, Roland H. Untch,C.Chu,M. Harrold: “Test Case Prioritization: An Empirical Study.”In: Proceedings of IEEE International Conference on Software Maintenance (ICSM), (1999)

[4] H. Srikanth, L. Williams, J. Osborne: “System test case prioritization of new and regression test cases.” In: Proceedings of IEEE International Symposium on Empirical Software Engineering, (2005)

[5] P. Tonella, P. Avesani,A. Susi: “Using the Case-Based Ranking Methodology for Test Case Prioritization.”In: Proceedings of 22ndIEEE International Conference on Software Maintenance, (2006)

[6] X. Zhang,C.Nie,B. Xu, B. Qu: “Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs.”In: Proceedings of 7thIEEEInternational Conference on Quality Software, (2007)

[7] Y. Huang,C.Huang,J. Chang, T. Chen: “Design and Analysis of Cost-Cognizant Test Case Prioritization Using Genetic Algorithm with Test History.”In: Proceedings of34thIEEE Annual Computer Software and Applications Conference, (2010)

[8] Y. Singh,A.Kaur,B. Suri: “Test Case Prioritization using Ant Colony Optimization.”ACM SIGSOFT Software Engineering Notes, vol. 35, no. 4, (2010)

[9]Md. Kayes: “Test Case Prioritization for Regression Testing Based on Fault Dependency.”In: Proceedings of3rd IEEE International Conference on Electronics Computer Technology (ICECT), (2011)

[10] D. Marijan,A.Gotlieb, S.Sen: “Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study.”In: Proceedings of IEEE International Conference on Software Maintenance, (2013)

[11]C.Lin,C. Chen,C. Tsai,G. Kapfhammer:“History-based Test Case Prioritization with Software Version Awareness.”In: Proceedings of IEEE International Conference on Engineering of Complex Computer Systems, (2013)

[12] Md. J. Arafeen,H. Do: “Test Case Prioritization Using Requirements-Based Clustering.”In: Proceedings of6thIEEE International Conference on Software Testing, Verification and Validation, (2013)

[13] D. D. Nardo, N.Alshahwan,L. Briand,Y. Labiche: “Coverage-Based Test Case Prioritisation: An Industrial Case Study.”In: Proceedings of6thIEEE International Conference on Software Testing, Verification and Validation, (2013)

[14]D.Hao,X. Zhao,L. Zhang: “Adaptive Test-Case Prioritization Guided by Output Inspection.”In: Proceedings of37thIEEEAnnual Computer Software and Applications Conference (COMPSAC), (2013)

[15] Md. S. Siddik,K.Sakib: “RDCC: An Effective Test Case Prioritization Framework using Software Requirements, Design and Source Code Collaboration.”In: Proceedings of 17thIEEEInternational Conference on Computer and Information Technology (ICCIT), (2014)

[16] M. Tyagi, S. Malhotra: “Test Case Prioritization using Multi Objective Particle Swarm Optimizer.”In: Proceedings of IEEE International Conference on Signal Propagation and Computer Technology (ICSPCT), (2014)

[17]D.Gao,X. Guo, L. Zhao: “Test Case Prioritization for Regression Testing Based on Ant Colony Optimization.”In: Proceedings of6thIEEE International Conference on Software Engineering and Service Science (ICSESS),(2015)

[18] T. B. Noor,H.Hemmati: “A similarity-based approach for test case prioritization using historical failure data.”In: Proceedings of 26th IEEE International Symposium on Software Reliability Engineering(ISSRE), (2015)

[19] A. Gupta, N. Mishra, A. Tripathi, M. Vardhan, D. Kushwaha:” An Improved History- Based Test Prioritization Technique Using Code Coverage.” In: Proceedings of 1st Springer International Conference on Communication and Computer Engineering

[20] D. D. Nucci,A.Panichella,A. Zaidman,A. D. Lucia: “Hypervolume-Based Search for Test Case Prioritization.”In: Proceedings of7thSpringer International Symposium, (SSBSE), (2015)

[21] K. Solanki,Y.Singh,S. Dalal: “Test Case Prioritization: An Approach Based on Modified Ant Colony Optimization (m-ACO).”In: Proceedings of IEEE International Conference on Computer, Communication and Control (ICCCC), (2015)

[22] H. Wang,J.Xing, Q. Yang: “Modification Impact Analysis based Test Case Prioritization for Regression Testing of Service-Oriented Workflow Applications.”In: Proceedings of39th IEEE Annual International Computers, Software & Applications Conference, (2015)

[23] S. Mahajan, S. D.Joshi, V. Khanaa: “Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform.” In: Proceedings of IEEE International Conference on Computing Communication Control and Automation, (2015)

[24] F. Yuan,Y.Bian,Z. Li(B), R. Zhao: “Epistatic Genetic Algorithm for Test Case Prioritization.”In: Proceedings of7thSpringer International Symposium,(SSBSE), (2015)

[25] L. Ramingwong, P.Konsaard:“Total Coverage Based Regression Test Case Prioritization using Genetic Algorithm.”In: Proceedings of12thIEEE International Conference onElectrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), (2015)

[26] A. Ansaria,A.Khanb,A. Khanc,K. Mukadamd: “Optimized Regression Test using Test Case Prioritization.”In: Proceedings of 7th Elsevier International Conference on Communication, Computing and Virtualization, (2016)

[27] M. Srinivas, L. Patnaik: “Genetic Algorithms: A Survey.”In: Proceedings ofIEEEComputer, vol.27, issue 6, (1994)

[28] K. F. Man,K.**S**. Tang**,**S. Kwong:“Genetic Algorithms: Concepts and Applications.” IEEE Transactions on Industrial Electronics, vol. 43, no. 5, (1996)