Project plan for degree project

Version 1.0 – March 16, 2023

PA2534: Master thesis in Software Engineering

January 16, 2024

Thesis	Tentative title	Analysis of Test Case Prioritization using Search Based Software Testing techniques
	Classification	Search Based Software Testing, Genetic Algorithm, Manual Testing, Artificial Fishing School Algorithm, Ant Colony Optimization, Simulated Annealing
Student 1	Name	Naga Jyothi Devarapalli
	e-Mail	nade22@student.bth.se
	Social security nr	20030822-T043
Student 2	Name	Nikitha Medeshetty
	e-Mail	nime22@student.bth.se
	Social security nr	20020604-T220
Student 3	Name	Siri Visnagari
	e-Mail	sivs22@student.bth.se
	Social security nr	20010815-T182
Student 4	Name	Sri Sai Preetham Poola
	e-Mail	srpl22@student.bth.se
	Social security nr	20010918-T172
Supervisor	Name and title	Nauman Ghazi
	e-Mail	nauman.ghazi@bth.se
	Department	Department of Software Engineering

[**] *Co-advisor from industry or a higher education institution (HEI).*

1 Introduction

A software test is an essential component of software development since it ensures the quality and reliability of software products. Even though testing is performed to identify bugs, reduce the development cost, and improve performance; it consumes a lot of resources such as time, effort, and cost.

The challenging aspects of testing are risk management, cost-benefit analysis, test planning, and being logical about which tests to perform for a particular project. The test package expands along with the software's size, which makes it difficult to maintain. It is not feasible to evaluate every potential scenario in depth for complex applications.

After modifications, the software is evaluated to make sure it is still dependable. To ascertain which areas might be impacted by proposed changes, an effective study of software modifications is conducted. Program updates, configuration adjustments, and earlier problem patches are a few examples of alterations. Regression testing involves doing this. Regression testing ensures that problem-solving adjustments and software upgrades continue to function as intended after being changed.

Test case prioritization is the technique of regression testing that improves the effectiveness and efficiency of software testing. It is a technique that determines the order in which test cases should be executed. The number of test cases needed to test an application can be effectively decreased by prioritizing test cases according to potential risks and customer requirements. Along with ensuring that customer requirements and specifications are met, prioritizing test cases aids in achieving project goals. A test case prioritization technique prioritizes a subset of the entire test suite and optimizes the order in which the test cases are executed, which saves a lot of time and money in the development process [1]. There is a direct correlation between the size of the test suite and that of the software [1]. If the software size increases, then test suite sizes increase as well. The test case prioritization approach generally considers the element of covering the entire number of assertions while prioritizing test cases [1]. The key points while determining the priority of test cases include the measurement of the risk failure of test cases, suspicion of the area of complication in coding and the previous bugs reports, considering the perspective of the system architecture and test cases, and frequency measurement of a function [1].

Search-based prioritization helps in finding the optimum ordering of test cases by searching over the global space for single or multiple objectives [2]. An optimal execution order is determined based on a wide range of criteria such as code coverage or fault detection rate and involves generating and evaluating a variety of possible test case orders. The effectiveness of search-based software testing for test case prioritization depends on the choice of search algorithms.

The research papers [11],[2],[5] tells us about the prioritization of test cases using Genetic algorithm, Ant colony optimization algorithm[3],Artificial fish school algorithm [4], simulated annealing [6],[13]. Based on research paper [1],[12],[7] these papers are conducting an experiment to compare algorithms and we found that there are very few papers which compare these algorithms.
This paper aims to identify the effective search-based software testing technique by comparing different techniques for test case prioritization. In this paper, we are considering the Genetic

Algorithm, Ant Colony Optimization, Artificial Fish School Algorithm, and Simulated Annealing and their effectiveness on test case prioritization by evaluating the possible test case orders based on criteria such as code coverage, fault detection rate, and selecting the optimal order for execution.

2 Related Work

This section consists of all the existing studies that are related to the proposed research. An approach to testing known as "Search based Software Testing" creates test suites, test data and prioritizes test cases to ensure maximum coverage. There are different search based software testing techniques but a few commonly used techniques are Genetic algorithms, Ant Colony Optimization, Simulated Annealing, Artificial Fishing School Algorithm. There are few researches which focus on test case prioritization using different search based software testing techniques. In all these researches they have focused on the application of different search based software techniques for test case prioritization. In most of the research, an experimental analysis was done and conclusions were drawn from the results.

Search-based software testing techniques are used to automate the Test Case Prioritization (TCP) process by locating the best or near-best solution. Search-based software testing employs meta-heuristic optimization search techniques such as the Genetic algorithm, which automates or partially automates test data generation through the use of the fitness function [8]. The TCP is mapped onto genetic information as a series of test cases. By locating the best cases in the search space, Genetic Algorithm(GA) assists in determining the optimal ordering of test cases. The two broad classes of GA are simple-objective GA and multi-objective GA, which are used for enhanced/improved version GA and considering two or more objectives at the same time for prioritisation, respectively [2].

The TCP for regression testing also includes Ant colony optimization (ACO), which takes three factors into account: the number of faults detected, the execution time, and the fault severity, and is used to find severe flaws during the early stages of regression testing. By validating test case prioritization results, the ACO algorithm computes an APFD value indicating a better solution in a shorter testing time [9]. Hill climbing for test case prioritization creates a test suite permutation at random to create the current state, then evaluates the neighbours of the current state, which is another test suite permutation created by exchanging the position of the first test with the test case in the current permutation. It iterates to the state with the greatest difference in fitness function [11].

Stimulated Annealing is another test case prioritization algorithm that works on the analogy of accepting candidates with higher costs to escape from the local minimum [12]. In comparison to the Genetic algorithm, the Markov process can be used to perform algorithm convergence analysis [13]. Two search algorithms, the Additional Greedy Algorithm and the 2-Optimal Greedy Algorithm, outperformed Hill climbing and Genetic algorithms based on factors such as problem representation, fitness function quality, operator choice, and search space characteristics. Another reason could be the problem's structure [14].

Artificial Fish School Algorithm (AFSA) is a cutting-edge swarm intelligence algorithm that solves TCP problems by stimulating foraging behaviour by combining a test case set, the av-

erage percentage of test point coverage, and effective execution time to optimize the design cluster. The effectiveness of AFSA in test case prioritization is determined by the design of the fitness function [4].

There are several search-based software testing techniques and we have found many studies which have applied different techniques to perform test case prioritization and concluded different results. Few studies have compared 2 different techniques and drawn results from the conclusions.
From all the above studies we have concluded that there is little research that compares multiple search based software techniques on test case prioritization and all the studies focus on the effectiveness of test case prioritization when a particular algorithm is used.
The focus of our research is to compare different search based software testing techniques applied during test case prioritization and identify which technique is more effective while prioritizing test cases.

3 Research Methodology

3.1 Aim and objectives:

The main goal of this research is to better understand various search-based software testing techniques and how they are used to prioritize test cases. It also aims to determine which search-based software testing technique is more efficient at prioritizing test cases when testing a software application.

1.To Understand the search based software testing techniques and how they are applied to prioritize test cases.

2. To understand the relationship between size and complexity of the software application and prioritization of test cases.

3. To examine the impact of search-based software testing methods on test case prioritization.

3.2 Research Questions

1. How Search based software testing techniques affect test case prioritization?

2. Which search-based software testing technique(Genetic algorithm, Ant colony optimization, Simulated annealing, Artificial fishing algorithm) is more effective in prioritizing the test cases?

3. Does the choice of search-based software testing technique during test case prioritization depend on the size and complexity of the software?

3.3 Justification

To know how search-based software testing techniques affect test case prioritization, we need to conduct an experiment. Experiments are best way to compare the search based software testing algorithms [15][16]. The research paper [11] has conducted a validation experiment to know the input-based local-beam-search adaptive-randomized techniques for test case prioritization

and compare the effectiveness and efficiency of the code. The research paper [7] has conducted the experiment to compare the four different algorithms.

Justification for RQ1 : Through experimentation, we can know how SBST techniques help in optimizing the test case prioritization. During the experiment, the environment can be controlled to ensure that results are not affected by external factors which are not possible in case studies or surveys. An experiment can be an appropriate quantitative research method to learn how the size and complexity of the software affect the choice of search-based software testing methods during test case prioritization.

Justification for RQ2: There are many studies which suggest different search based software testing techniques for test case prioritization.Through experiment we can identify which search based software testing technique is more effective in prioritizing test cases.

Justification for RQ3: The size and complexity of the software system may be independent variables, and the efficiency of search-based software methods in prioritizing test cases may be dependent variables. On experimenting we can know how variables affect the decision to use search-based software testing methods, such as the size and complexity of the software system, the type of faults, the testing goals, and the resources available. An alternate method to answer this question is by conducting a survey.Through survey we can collect views from software testing team and developers

4 Study Design

4.1 Hypothesis

Null Hypothesis: The prioritization of test cases is affected differently by various search-based software testing techniques.
Alternate Hypothesis: Test case prioritization is unaffected by search-based software testing techniques.

4.2 Subjects

Testers who are testing the software application, Researchers who are involved in identifying the effectiveness of SBST techniques in test case prioritizing.

4.3 Objects

Test case prioritization techniques(Genetic algorithm, Ant colony optimization, Simulated annealing and Artificial fishing school algorithm), test cases and metrics for evaluating the effectiveness of prioritizing of test cases.

4.4 Variables

Independent Variable: Search-based software testing techniques(Genetic algorithm, Ant colony Optimization algorithm, Simulated Annealing, Artificial fishing school algorithm) and traditional methods used to prioritize test cases.

Dependent Variable: Test case prioritization effectiveness.
Control variables: Size of the code, Number of test cases, Experience of the testers.

4.5 Experiment

1. The participants are divided into two groups and each group consists of four software testers.

2. A software application is chosen and a particular module of the application is taken into consideration for testing for group 1.

3. Modules of different sizes and complexities are assigned to group 2 for testing. In order to test the modules of the software application the respective test suites are given to the participants.

4. Each tester in a group is assigned a search-based software testing technique randomly to test the software component.

5. Each tester in the group tests the software module with the given test suits according to their assigned algorithm.

6. Test results are evaluated based on the following metrics: Fault detection, code coverage, time, and redundancy.

4.6 Data collection

A quantitative analysis is performed based on the number of faults detected, code coverage, and time spent in prioritizing test cases. The qualitative data is collected from participant feedback. In order to know how effective search-based software testing techniques are, data is being collected from the results of group 1. We analyze the data from group 2 to understand how size and complexity affect test case prioritization using search-based software testing.

4.7 Data Analysis

The data is analyzed by conducting ANOVA tests to know whether the search- based on software testing or traditional prioritizing techniques are efficient in prioritizing test cases
ANOVA(Analysis of variance) : It is mainly applied to know how independent variables have an effect on dependent variables. In this experiment, independent variables are Search based software techniques and conventional techniques. Dependent variables are test case prioritization. The significant effect on test case prioritization will be examined by F-statistic and its associated p-value.

$$f - statistic = VarianceBetweenSampleMeans/VarianceWithinSampleMeans$$

After calculating the f-statistic value, the associate p value is determined. By considering the P value and significance level , we can either accept or reject the null hypothesis

4.8 Threats to validity

1. Internal Validity: Internal validity is verifying cause-and-effect relationships between your testing situation and study result.

Threat: As the software applications and test suits are examined over time, the efficacy of priority may alter.

Mitigations: The analysis of software applications and test scenarios should be done in a random sequence

2. Conclusion Validity: Whether the hypothesis derived from the analysis is consistent with what actually occurs between variables

Threat: Interpreting data using the wrong statistical method can result in inaccurate conclusions.

Mitigations: The data should be analyzed using proper statistical tests, and the outcome should be interpreted properly. Choosing a proper statistical method, correct conclusions can be drawn.

3. Construct Validity: The purpose of construct validity is to ensure that the research objective is appropriately reflected in the planning and implementation of the research

*Threat:*In order to achieve the optimal results, search-based testing techniques will be selected and implemented according to the preference and familiarity of experimenters

Mitigations: Evaluation can be done using multiple measures, like evaluating multiple software projects of different sizes and complexity, using different metrics for test case prioritization.
Randomly assigning the techniques to the participants can also reduce the risk of bias

4. External Validity: An external validity is measured by the extent to which results of research study can be applied to other studies.

Threat: It is possible that external factors during the research may influence results, limiting their generalizability.
Considering that the experiment took place over a specific time period, there may be no generalization to other periods of time.

Mitigations: In order to ensure that results are not affected by the external conditions, experiments must be conducted in a real world environment with software. To prevent results from being generalized to specific time periods of time, experiments must be conducted over a long period of time.

5 Expected Outcomes

The purpose of our study is to determine which search-based software testing technique is the most effective for prioritizing test cases. The data that is being collected from the tests

conducted by group1 and group2 are analyzed. The test results are evaluated on the basis of fault detection, code coverage, time, redundancy. Despite the fact that our results are not 100 percent generalizable, it can be concluded that genetic algorithms are effective at optimizing multiple objectives as well as performing well in the prioritization of test cases. Ant colony optimization is also effective in the prioritization of large test cases, but it is computationally more expensive than other methods. When optimizing problems with rugged search spaces, simulated annealing can be effective, but parameter tuning is necessary for good performance which needs to be done carefully. When it comes to problems with many variables, the artificial fish school algorithm is effective, but may be sensitive to parameter settings as well as requiring many iterations.

6 Time and Activity Plan

S.no	Activity	Start date	End date
1	Started looking for research publications and identify a good research gap	29-01-2023	09-02-2023
2	Gathering data on research gap and the area of study	10-02-2023	27-02-2023
3	Meeting supervisor for the feedback on research gap and research topic	28-02-2023	28-02-2023
4	Research question formulation based on research gaps	01-03-2023	03-03-2023
5	Gathering information related to research topic and documenting the research proposal	04-03-2023	24-03-2023
6	Submission of final research proposal	15-03-2023	15-03-2023
7	Performing an experiment	16-03-2023	29-03-2023
8	Gathering and analyzing data	30-03-2023	09-04-2023
9	Reviewing the results with the supervisor and getting feedback	10-04-2023	10-04-2023
10	Making necessary changes based on the feedback	11-04-2023	25-04-2023
11	Preparation of the final thesis document	10-03-2023	13-05-2023
12	The final presentation of the thesis	14-05-2023	16-05-2023
13	Final submission of the thesis	18-05-2023	23-05-2023

7 Risk Management

Risk 1: Search space constraint
To find the best test cases, search-based techniques rely on exploring a search space. When the search space becomes too large, the technique may become computationally expensive, if not impossible.

Mitigations: By defining specific objectives and constraints for the prioritization process, you can narrow the search space. Consider using techniques like genetic algorithms or simulated annealing to efficiently explore large search spaces.

Risk 2: Heuristic reliance
Heuristics are frequently used to guide the search process in search-based techniques. The resulting prioritization may be suboptimal if the heuristics are poorly designed or inappropriate for the specific application.

Mitigations: Select heuristics that are relevant to the application and testing objectives. Validate the effectiveness of the chosen heuristics through thorough testing.

Risk 3: Inability to deal with complex systems
Search-based techniques may not be appropriate for complex systems with multiple interacting components or that require specific environmental conditions to function properly.

Mitigations: Consider combining search-based testing techniques with other types of testing, such as model-based testing or static analysis. Also, make certain that the test environment is well-controlled in order to reduce the impact of environmental factors.

Risk 4: Certain types of defects have limited effectiveness.
In general, search-based techniques are effective for detecting defects that can be traced back to specific code locations. They may, however, be less effective at detecting defects caused by system-level interactions or environmental factors.

Mitigations: Use a variety of testing techniques, such as exploratory testing or boundary value analysis, to supplement search-based testing. Consider using mutation testing to assess the quality of prioritization.

Risk 5: Risk of introducing new defects.
The order in which test cases are executed may be altered by test case prioritization techniques, which may introduce new defects if the execution order affects the behaviour of the system under test.

Mitigations: After implementing the new prioritization, perform regression testing. To reduce the risk of introducing new defects, ensure that the test suite is well-designed and comprehensive.

8 References

References

[1] M. A et al., "Comparative Analysis of Ant Colony Optimization and Particle Swarm Optimization for Test Case Prioritization," 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 2022, pp. 680-686, doi: 10.1109/3ICT56508.2022.9990713.

[2] A. Bajaj and O. P. Sangwan, "A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms," in IEEE Access, vol. 7, pp. 126355-126375, 2019, doi: 10.1109/ACCESS.2019.2938260.

[3] W. Zhang, Y. Qi, X. Zhang, B. Wei, M. Zhang and Z. Dou, "On Test Case Prioritization Using Ant Colony Optimization Algorithm," 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 2019, pp. 2767-2773, doi: 10.1109/HPCC/SmartCity/DSS.2019.00388.

[4] Xing, Ying, Xingde Wang, and Qianpeng Shen. "Test case prioritization based on Artificial Fish School Algorithm." Computer Communications 180 (2021): 295- 302.

[5] Getachew Mekuria Habtemariam and Sudhir Kumar Mohapatra, A genetic algorithm-based approach for test case prioritization. In International Conference on Information and Communication Technology for Development for Africa, pages 24-37. Springer, 2019.

[6] Weixiang Zhang, Rui Dong, Bo Wei, Huiying Zhang, Sihong Wang, and Fengju Liu. 2022. Test Case Prioritization Based on Simulation Annealing Algorithm. In Proceedings of the 8th International Conference on Computing and Artificial Intelligence (ICCAI '22). Association for Computing Machinery,New York, NY, USA, 235–240.

[7] Qasim. "Test case prioritization techniques in software regression testing: An overview." International Journal of ADVANCED AND APPLIED SCIENCES (2021): n. pag.

[8] P. McMinn, "Search-Based Software Testing: Past, Present and Future," 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, Berlin, Germany, 2011, pp. 153-163, doi: 10.1109/ICSTW.2011.100.

[9] D. Gao, X. Guo and L. Zhao, "Test case prioritization for regression testing based on ant colony optimization," 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2015, pp. 275-279, doi:

10.1109/ICSESS.2015.7339054.

[10] .W. Jun, Z. Yan and J. Chen, "Test Case Prioritization Technique Based on Genetic Algorithm," 2011 International Conference on Internet Computing and Information Services, Hong Kong, China, 2011, pp. 173-175, doi: 10.1109/ICICIS.2011.50.

[11] Bo Jiang, W.K. Chan,Input-based adaptive randomized test case prioritization: A local beam search approach,Journal of Systems and Software,Volume 105,2015,Pages 91-106,ISSN 0164-1212, https://doi.org/10.1016/j.jss.2015.03.066.

[12] Maheswari, R. Uma, and D. Jeya Mala. "Combined genetic and simulated annealingapproach for test case prioritization." Indian Journal of Science and Technology 8.35 (2015): 1.

[13] Zhang, Weixiang, et al. "Test Case Prioritization Based on Simulation Annealing Algorithm." Proceedings of the 8th International Conference on Computing and Artificial Intelligence. 2022.

[14] S. Li, N. Bian, Z. Chen, D. You and Y. He, "A Simulation Study on Some Search Algorithms for Regression Test Case Prioritization," 2010 10th International Conference on Quality Software, Zhangjiajie, China, 2010, pp. 72-81, doi: 10.1109/QSIC.2010.15.

[15] Jo E Hannay, Dag IK Sjoberg, and Tore Dyba. A systematic review of theory use in software engineering experiments. IEEE transactions on Software Engineering, 33(2):87–107, 2007

[16] Dag IK Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. Guide to advanced empirical software engineering, pages 312–336, 2008