An survey on Test Case Prioritization using Binary Integer Programming

Sunil
HOD & Assistant Professor, Department of Computer Science & Engineering, SSIET, Dinanagar

ABSTRACT
Test case prioritization techniques provide a way to schedule and run test cases, which have the highest priority in order to provide earlier detect faults. This study presents numerous techniques developed, that can improve a test suite’s rate of fault detection. In this Paper we have surveyed related works in different types Test case prioritization techniques of and mapped them to our taxonomy to guide future design and development efforts.

Keywords
Greedy Algorithms; Traceability matrix; prioritization.

1. INTRODUCTION
In this technique test cases are assigned a priority. Priority is set according to some criterion and test cases with highest priority are scheduled first. For example, criterion may be that the test case which has faster code coverage gets the highest priority. Advantage to previous techniques is that it doesn’t discard or permanently remove the test cases from test suite. Another criterion may be rate at which fault is detected. Test case prioritization schedule test cases in order to increase their ability to meet some performance goal:

- Rate of fault detection
- Rate of code coverage
- Rate of increase of confidence in reliability

2. RELATED WORK
On the basis of literature review, following gaps has been identified. Target of software engineering is to develop high quality software within time and budget. Since software testing is time consuming, error prone, full of uncertainty and expensive process. Industry does not have enough time or resources to run the entire test suite. Test case prioritization and selection is vital activity of software testing. The need of hour is to provide cost effective strategy for software test case optimization. Therefore, there is a strong need to decide which test cases to run first to maximize the fault detection. Quality of software testing is low due to inadequate test case prioritization and selection technique. By applying appropriate test case prioritization and selection technique, cost, efforts, uncertainty of software testing can be reduced considerably. Industry requires cost effective adequate technique for test case selection and prioritization. Fault detection is prime factor for evaluating adequacy and ranking the test cases.

Test cases should prioritize on the basis of fault detection capability. Test cases should be designed, selected and prioritized in such a manner that it will detect the maximum number of faults in software under test within deadline of time and budget. Genetic algorithm to reorder test cases in a test suite using execution time as a constraint had shown that prioritization technique is appropriate for regression testing environment and explains how the baseline approach can be extended to operate in additional time constrained testing circumstances. Literature review is an evidence that several classical and search based techniques have been applied to find out the solutions for test case prioritization problem. Some of these works apply Ant Colony Optimization, Genetic Algorithms, Greedy Algorithms, Linear Programming, Case Base Reasoning, Fuzzy logic and so on.

Ant Colony Optimization, Genetic Algorithms, Greedy Algorithms provide near optimal and local solutions to test case prioritization problem. Also, these approaches have not considered the precedence of test cases. Classical computing techniques such as Linear Programming provide single global 24 solution to test case prioritization and selection problem. Several classical and soft computing techniques have been explored for test cases selection and prioritization using code, requirement coverage, fault coverage, and cost.

Many interesting results have been received but the test case prioritization and selection based on fault detection with time constraint using Binary Integer Programming technique has not been explored. So, there is still space for the researchers to explore and experiment the Binary Integer Programming based approach to find out the order of test cases on the basis of faults detections with time constraint Kim [20] defined the selection probabilities of each test case, TC, at time, t, to be Ptc, t (Htc, α), where Htc is a set of t, time-ordered observations {h1, h2, …hn} drawn from runs of TC and α is a smoothing constant used to weight individual historical observations. The higher values emphasize recent observations, while lower values emphasize older ones. These values are then normalized and
used as probabilities. The general form of:

\[ P = P_0 = h_1 \text{ and } P_k = \alpha h_k + (1-\alpha)P_{k-1}, \ 0 \leq \alpha \leq 1, \ k \geq 1. \]

When testing in a black box environment, source code related information is not available. In such situations, practitioners only have output of test cases and other run-time information available, such as the running time of test cases.

3. TEST CASE PRIORITIZATION TECHNIQUES

In this section various types of Test case prioritization techniques along with related works has been discussed.

3.1 Customer Requirement-Based Prioritization Techniques

Customer requirement-based techniques are methods to prioritize test cases based on requirement documents. Many researchers have researched this area and introduced many weight factors which used in these techniques, including custom-priority, requirement complexity and requirement volatility.

Hema et al. [3] presented the requirements-based test case prioritization approach to prioritize a set of test cases. Amitabh [4] built upon current test case prioritization techniques and proposed to use several factors to weight (or rank) the test cases. Those factors are the customer-assigned priority (CP), requirements complexity (RC) and requirements volatility (RV). Additionally, they assigned value (1 to 10) to each factor for the measurement. They stated that higher factor values indicate a need for prioritization of test case related to that requirement. Weight prioritization (WP) is measured as:

\[ WP = \sum (PF_{value} \times PF_{weight}); \ PF = 1 \text{ to } n \]

where: WP denotes weight prioritization that measures the importance of testing a requirement.

PF value is the value of each factor, like CP, RC and RV.

PF weight is the weight of each factor, like CP, RC and RV.

Test cases are then ordered such that the test cases with high WP are executed before others.

Manish et al. [5] proposed an approach for test case generation for web based applications. They presented a simple approach for test case prioritization through the requirement traceability matrix. The matrix can be produced by mapping from use cases in the use case diagram to functional requirements from users. They also proposed to use weight values assigned to each requirement by developers. Each requirement is assigned a priority weight from 1 to 10, 10 being highest.

3.2 Coverage-Based Prioritization Techniques

Coverage-based techniques are methods to prioritize test cases based on coverage criteria, such as requirement coverage, total requirement coverage, additional requirement coverage and statement coverage.

Test coverage analysis is a measure used in software testing known as code coverage analysis for practitioners. It describes the quantity of source code of a program that has been exercised during testing. It is a form of testing that inspects the code directly and is therefore a form of white box testing.

The following lists a process of coverage-based techniques:

(a) Finding areas of a program not exercised by a set of test cases

(b) Creating additional test cases to increase coverage

(c) Determining a quantitative measure of code coverage, which is an indirect measure of quality.

(d) Identifying redundant test cases that do not increase coverage.

The coverage-based technique is a structural or white-box testing technique. Structural testing compares test program behavior against the apparent intention of the source code. This 17 contrasts with functional or black-box testing, which compares test program behavior against a requirements specification. It examines how the program works, taking into account possible pitfalls in the structure and logic. Functional testing examines what the program accomplishes, without regard to how it works internally. The coverage-based techniques are methods to prioritize test cases based on coverage criteria, such as requirement coverage, total requirement coverage, additional requirement coverage and statement coverage.

3.3 Cost Effective-Based Prioritization Techniques
Cost effective-based techniques are methods of prioritizing test cases based on costs, such as cost of analysis and cost of prioritization. Many researchers have researched this area. The following paragraphs present existing cost effective-based test case prioritization techniques.

Leung and White [17] presented a cost model for regression test selection in. The proposed model incorporates various costs of regression testing, including the costs of executing and validating test cases and the cost of performing analyses to support test selection, and provides a way to compare tests for relative effectiveness. This model can be appropriately applied to an effective regression test selection techniques, which necessarily select all test cases in the existing test suite that may reveal faults. However, Leung’s model does not consider the costs of overlooking faults due to discarded tests.

Alexey Malishevsky et al [18] presented cost models for prioritization that take these costs into account. They defined the following variables to prioritize test cases: cost of analysis, Ca(T) and cost of the prioritization algorithm, Cp(T). Then weight prioritization value (WP) is calculated as:

\[
WP = Ca(T) + Cp(T)
\]

where:
1. Ca(T) includes the cost of source code analysis, analysis of changes between old and new versions, and collection of execution traces.
2. Cp(T) is the actual cost of running a prioritization tool, and, depending on the prioritization algorithm used that can be performed during either the preliminary or critical phase.

3.4. Chronographic history-based techniques

Chronographic history-based techniques are methods to prioritize test cases based on test execution history. The following paragraphs present an overview of existing chronographic history-based test case prioritization techniques. Jung-Min and Adam [20] proposed to use information about each test case’s prior performance to increase or decrease the likelihood that it will be used in the current testing session. Their approach is based on ideas taken from statistical quality control (exponential weighted moving average) and statistical forecasting (exponential smoothing).

4. REFERENCES


