A Comparative Study of Five Regression Testing Techniques: A Survey

Jyoti, Kamna Solanki

ABSTRACT: Regression testing is a style of testing that focuses on retesting after changes are made. In traditional regression testing, we reuse the same tests (the regression tests). Over the years, numerous regression test optimization techniques have been described in the literature. Empirical studies of the techniques suggest that they can indeed benefit testers, but so far, few studies have empirically compared different techniques. In this paper, the results of a comparative empirical study of five different regression test optimization techniques are represented. The comparison is based on different qualitative and quantitative criteria such as number of tests selected, execution time of tests, precision, user parameters, global variables handling, type of testing etc. These algorithms are found to be suitable for different requirements of regression testing.

General Terms: Regression Testing Optimization Techniques

Keywords: Regression testing, optimization, minimization, prioritization, minimization

1. INTRODUCTION

Regression testing is a style of testing that focuses on retesting after changes are made. In traditional regression testing, we reuse the same tests (the regression tests). In risk-oriented regression testing, we test the same areas as before, but we use different (increasingly complex) tests. Traditional regression tests are often partially automated. Regression test optimization techniques reduce the cost of regression testing by selecting a subset of an existing test suite to use in retesting a modified program. Over the years, numerous regression test optimization techniques have been described in the literature. Empirical studies of the techniques suggest that they can indeed benefit testers, but so far, few studies have empirically compared different techniques. In this paper, the results of a comparative empirical study of different regression test optimization techniques are represented. Regression testing attempts to mitigate two risks:

- A change that was intended to fix a bug failed.
- Some change had a side effect, unfixing an old bug or introducing a new bug [1].

2. REGRESSION TESTING

Let P be a procedure or program, let P' be a modified version of P and let T be a test suite for P. Regression testing consists of reusing T on P', and determining where the new test cases are needed to effectively test code or functionality added to or changed in producing P'. A typical regression test proceeds as follows [2] [3]:
1. Select T' ⊆ T, a set of test cases to execute on P'.
2. Test P' with T'. Establish P' correctness with respect to T'.
3. If necessary, create T'', a set of new functional or structural test cases for P'.
4. Test P' with T'', establishing P''s correctness with respect to T''.
5. Create T''', a new test suite and test history for P', from T, T', and T''.

When developing T', three problems arise:
- (1) which test cases in T should be used to test P' (regression test selection);
- (2) Which new test cases must be developed (test suite augmentation); and
- (3) Which order should be used to run the test cases (test-suite prioritization) [4].

3. REGRESSION TESTING OPTIMIZATION

3.1 Selection Techniques

RTS attempt to reduce the cost of regression testing by selecting a subset of the test suite that was used during development and using that subset to test the modified program. With this approach only a subset of test cases are selected and rerun. RTS divides the existing test suite into...
• Reusable test cases;
• Re-testable test cases;
• Obsolete test cases [5] [6].

1. **Retest-All Technique**: This conventional method reruns all test cases in T that were previously run during testing phase and not a single test case is left. So the technique is very expensive as regression test suites are costly to execute in full as it require much time and budget and therefore it may be used when test effectiveness is the utmost priority with little regard for cost.

2. **Random/Ad-Hoc Technique**: In this technique, the testers rely on their previous experiences and knowledge to select which test cases need to be rerun. This can include selecting a percentage of test cases randomly.

3. **Dataflow Techniques**: This technique is coverage-based regression test selection technique that selects test cases that exercise data interactions that have been affected by modifications.

4. **Safe Technique**: This technique by definition eliminates only those test cases that are probably not able to reveal faults. A safe technique, routinely selected almost all test cases when more than a few (2 or 3) changes were made to the subject programs. This technique does not focus on criteria of coverage but select all those test cases that produce different output with a modified program as compared to its original version.

5. **Hybrid Technique**: Hybrid Approaches includes techniques of both Regression Test Selection and Test Case Prioritization or Regression Test Minimization and Test Case Prioritization.

### 3.2 Prioritization Technique

In this technique the testers prioritize the regression test cases according to various important factors such as total efforts, cost-benefit tradeoffs, execution time, deploying time etc so as to increase a test suite’s rate of fault detection. The technique reorders test cases, scheduling test cases with the highest priority according to some criterion earlier in the testing process using Least Recently Used or some random technique.

This technique can be classified as:

- General Prioritization: Select an effective order of test case for subsequent versions of software.
- Version specific prioritization: Concerned with particular version of the software [7].

At a high level, test case prioritization works as follows:

1. Apply an RTS technique to test suite T, yielding T ′;
2. Assign a selection probability to each test case in T ′;
3. Draw a test case from T ′ using the probabilities assigned in step 2, and run it, and
4. Repeat step 3 until testing time is exhausted.

### 3.3 Minimization Techniques

In this technique a minimum number of test cases are selected from T, which help the testers to uncover the modified elements of P ′. This technique can include selecting those test cases that are related with those modified elements of the program.

Minimization can be defined as: A test suite, T, a set of test requirements \{r_1, r_2, ..., r_n\}, that must be satisfied to provide the desired ‘adequate’ testing of the program, and subsets of T, T_1, T_2, ..., T_n, one associated with each of the r_s such that any one of the test cases t_i belonging to T_i can be used to achieve requirement r_i [8] find a representative set, T ′, comprised of test cases from T that satisfies all r_i. Under this definition, T ′ is equivalent to a hitting set for all T_i and a minimal T ′ is equivalent to the minimal hitting set [9].

### 4. IMPLEMENTED ALGORITHMS

Different algorithms are described in the paper as follows:

#### 4.1 Slicing Algorithm

A statement level slice based heuristic combining REG (Regular statement/branch) executed by test case, OI (Output influencing) and POI (potential OI) was expressed in an experimental study conducted by Jeffery and Gupta [10]. Aristotle Program Analysis tool was used to compare the technique with total statement and branch coverage. It was interpreted that faults were detected earlier in the testing process from the fact that the information about relevant slicing and modifications traversed by each test case is beneficial when used as a part of test case prioritization process. Jeffrey and Gupta [11] advanced their earlier proposed technique by adding coverage requirements of the relevant slices to the modification information for prioritization. The two techniques derived from the original technique “REG+OI+POI” [12], were named as “GRP_REG+OI+PI” and “MOD*(REG+OI+PI)”. In comparison with the statement and branch coverage techniques, the extended MOD*(REG+OI+POI) proved to be an improvement over the REG approach on the grounds of the fault detection rate of prioritized test suites.

#### 4.2 Incremental Approach

The incremental approach includes different modules that are developed in incremental order. The idea to evolve a system through small incremental releases has many advantages, including:

- It reduces risk
- It allows feedback to guide the development process which helps in building the right system (as opposed to the system as it’s designed up front)
- It opens up the development process and encourages customer involvement
- It can bring real business value earlier.

The approach implements the following:

- Each Module provides a definitive role to play in the project/product structure
- Each Module has clearly defined dependencies some of which can be known only at the runtime.

The incremental integration testing’s greater advantage is that the defects are found early in a smaller assembly when it is relatively easy to detect the root cause of the same. A disadvantage is that it can be time-consuming since stubs and drivers have to be developed for performing these tests.
4.3 Adaptive Firewall Approach
Leung and White [13] proposed the concept of firewall to cover all the affected modules that were modified at the module integration level. The firewall was based on the call graph defined and the effort can be decreased by retesting only those modules and links that were present in the firewall of the modified or affected modules instead of retesting the whole of the system. Later on the authors also introduced data flow based firewalls that were based on the data flow diagram of the affected modules due to coupling effect.

Fig 2: Regression Testing Techniques

4.4 Genetic Algorithm
A GA is a search algorithm that is inspired by the way nature evolves species using natural selection of the fittest individuals. GA is directly analogous to the natural evolution in biological processes presuming that a problem can be solved by a potential solution in the form of an individual capable of being represented as a set of parameters which can be regarded as the genes of a chromosome structured by a binary form of string of values where a positive value i.e. fitness function is used to reflect the degree of usability and goodness of the chromosome that can be closely related to the objective value to solve the problem given. Genetic algorithm is a random searching method that has a better optimization ability and internal implicit parallelism.

4.5 Simulated Annealing Algorithm
Simulated Annealing Algorithm is an optimization algorithm derived from natural and physical phenomena that is adapted to solve the problem of software retesting. The algorithm is augmented with procedures to guarantee feasible solutions to produce optimal or near optimal solutions, solving the integer programming model of the retesting problem without suffering from any kind of intractability. The process involves the current state of a physical system and function to be minimized. It can be thought of as analogous to the internal energy of current state the given system and the objective is to get a state with a minimum possible energy from the initial current state provided.

6. COMPARISON CRITERIA
Here ten quantitative and qualitative criteria are presented that are used for evaluating and comparing the five regression testing algorithms.

The quantitative criteria are:

i. **Execution time** of a regression testing algorithm, denoted as \( t \).

ii. **Number of test cases** selected by an algorithm \( \#R \).

iii. **Precision**, in terms of modification-revealing and modification-traversing tests. If the initial test suite \( T \) contains \( x \), non-modification revealing tests, and a regression testing algorithm selects \( (x_1, y_1) \) from these tests, precision is ratio of \( (y_1/x_1) \).

iv. **Inclusiveness**
If \( T \) contains \( x_2 \) modification-revealing tests, and a regression testing algorithm selects \( y_2 \) from these tests, inclusiveness is the ratio of \( (y_2/x_2) \).

The qualitative criteria are:

v. **User’s parameter setting**: whether user’s intervention is required or not.

vi. **Testing of global variables**

vii. **Type of maintenance**: software maintenance can be [14]:

a) Corrective, due to error reports,

b) Perfective, for enhancing the software’s functionality,

c) Adaptive, due to changes in the external environment, or

d) Preventive, for facilitating future maintenance.

viii. **Type of testing**: functional or structural.

ix. **Level of testing**: algorithm used at module level or integration testing level.

x. **Type of approach**: whether coverage, minimization, or safe.

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

Quantitative criteria:

i. **Execution time and number of test cases**: Adaptive firewall and slicing gave the same results while incremental, genetic and simulated annealing algorithms were better, the latter two taking more execution time in small and medium size modules.

ii. **Precision**: Adapted firewall and slicing recorded the least precision value.

iii. **Inclusiveness**: was found to be high in case of adapted firewall and slicing algorithms while low in genetic and simulated annealing.

Qualitative criteria:

i. **User’s parameter setting**: Two algorithms namely genetic and annealing required different parameters while no other algorithm required any parameters.

ii. **Testing of global variables**: Additional variable matrix is used by all the algorithms.

iii. **Type of maintenance**: All use corrective maintenance.
iv. **Type of testing:** Incremental can be both structure and function based while all others are structure based only.

v. **Level of testing:** Except for the firewall algorithm all are tested at the module level.

i. **Type of approach:** The incremental algorithm is a safe regression testing algorithm, while the slicing and firewall algorithms are coverage algorithms the genetic and simulated algorithms are minimization algorithms.

8. **CONCLUSION**

The use of any of these algorithms depends on the requirements of the user to choose a minimum number of test cases, performing fast regression testing. The incremental approach can be thought of as the best to be chosen according to the user requirements and taking lowest time while simulated annealing or genetic can also be thought of if user have no time constraint.

REFERENCES


<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>SLICING</th>
<th>INCREMENTAL</th>
<th>ADAPTED FIREWALL</th>
<th>GENETIC</th>
<th>SIMULATED ANNEALING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time For Small Size Modules</td>
<td>Fast</td>
<td>Fast</td>
<td>Fast</td>
<td>Acceptable</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Execution Time For Medium Size Modules</td>
<td>Acceptable</td>
<td>Fast</td>
<td>Slow</td>
<td>Acceptable</td>
<td>Slow</td>
</tr>
<tr>
<td>Number Of Selected Retests</td>
<td>Acceptable</td>
<td>Good</td>
<td>Acceptable</td>
<td>Very Good</td>
<td>Very Good</td>
</tr>
<tr>
<td>Precisions</td>
<td>High</td>
<td>Very High</td>
<td>High</td>
<td>Very High</td>
<td>Very High</td>
</tr>
<tr>
<td>Inclusiveness</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>User’s Parameter Setting Global Variables</td>
<td>Not Required</td>
<td>Not Required</td>
<td>Not Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Can be identified</td>
<td>Can be identified</td>
<td>Can be identified</td>
<td>Can be identified</td>
<td>Can be identified</td>
<td>Can be identified</td>
</tr>
<tr>
<td>Type Of Maintenance</td>
<td>CORRECTIVE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Levels Of Testing</td>
<td>MODULE INTEGRATION</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>With modification</td>
<td>No</td>
<td>With modification</td>
<td>With Modification</td>
<td>With modification</td>
<td></td>
</tr>
<tr>
<td>Type Of Approach</td>
<td>Coverage</td>
<td>Safe</td>
<td>Coverage</td>
<td>Minimizatio n</td>
<td>Minimization</td>
</tr>
<tr>
<td>Type Of Testing</td>
<td>Structural</td>
<td>Structural &amp; Functional</td>
<td>Structural</td>
<td>Structural</td>
<td>Structural</td>
</tr>
</tbody>
</table>

Table 1: Table of comparison between different techniques [15]