ISSN:0975-9646

Regression Testing Using Coupling and Genetic Algorithms

Harsh Bhasin[#], Manoj[*]

#Computergrad.com
Faridabad, Haryana India

**MRKIET*
Reweri, Haryana

Abstract— **Regression testing calls for the execution of all the test cases tested before a change is made in the software. It takes a lot of time and resources and hence a technique is needed to prioritize the test cases so that only important test cases are re-executed thus saving the time and still not compromising with the quality of the software. The proposed technique prioritizes original test suite by assigning fitness value to each of the test cases and then applying Genetic Algorithms so that the new suite will have a superior rate of fault detection when compared to the rates of randomly prioritized test suites. To assign the fitness value, various modules have been given a value based on the system proposed. The fitness value is judged on the basis of coupling. If a module has an undesirable coupling, it is liable to be a source of errors, so it is given a smaller value whereas that having not so undesirable coupling will have greater value.**

Keywords- **Regression Testing, Test case prioritization, Genetic Algorithms, Coupling.**

I. INTRODUCTION

The purpose of regression testing is to ensure that bug and new functionality introduced in a new version of software do not adversely affect the correct functionality inherited from the previous version [1]. There are insufficient resources to allow for the re-execution of all test cases during regression testing. So, test case prioritization techniques are to be used to improve the effectiveness of regression testing by ordering the test cases in such a way so that those having high fitness value are executed first. In this work a new test case prioritization technique using Genetic Algorithms (GAs) has been proposed. The proposed technique prioritizes subsequence of the original test suite so that the new suite, which is run, will have a superior rate of fault detection when compared to rates of randomly prioritized test suites. The various modules have been given a value based on the system proposed. The vale is then converted into a float number between 0 and 1. The value is judged on the basis of coupling and cohesion. The module, if has an undesirable coupling is liable to be a source of errors, so it is given a smaller value whereas that having not so undesirable coupling will have more value. To keep the things simple the factors which are to be multiplied are in powers of 10. Another concept has been incorporated here is the concept of call graph. A program's call graph is an essential underlying structure for performing the various interprocedural analyses used in software development tools for object-oriented software systems [2]. The call graph shows the calling relationships between methods during the execution of the application, and is focused on a method of choice. If a given method is called in several contexts, it is shown once for each context in the call graph. The base method is shown only once in the call graph, unless it is called recursively. So the module which is altered needs to be checked first followed by the modules which call that particular module, a value based on this has been proposed. The above two values first being the coupling and cohesion factor and the second being the call graph based value are clubbed together with the help of formula proposed. This forms the basis of the fitness value which is scaled down to a number between 0 and 1. The different modules in the software will now have different fitness values. The module having high fitness value needs to be checked first followed by that having low fitness value. The modules will now be prioritized based on these values. The problem of prioritizing the modules is converted into a simple knapsack problem which is most apt for applying GAs. The GAs then finds out the most suitable modules according to the fitness function. A test case has been taken as an experiment to analyse the GAs with regard to effectiveness. The GAs based method will perform crossover, mutation, replication and rollet wheel selection to find out the most important modules that needs to be checked. The above method will be time bound and hence more effective as compared to the existing methods. Moreover the changes if made in one module will affect other modules in what way can be judged by the values of the modules obtained.

II. LITERATURE REVIEW

Regression testing is verifying that previously functioning software remains same after a change. With the goal of finding a basis for further research a systematic review of empirical evaluations of regression test selection techniques was conducted. A study in which out of 29 23 papers analysed, 28 papers were identified reporting on empirical comparative evaluations of regression test selection techniques. They report on 38 unique studies, and in total 32 different techniques for regression test selection has been evaluated. Our study concludes that no clear picture of the evaluated techniques can be provided based on existing empirical evidence, except for a small group of related techniques. Instead, the need for more and better empirical studies was identified and concepts were evaluated [3]. The empirical studies were observed where concepts are evaluated rather than small variations in technical implementations.

A. Regression Test Selection Techniques

A variety of regression test selection techniques have been described in the research literature. A survey by Rothermel and Harrold describes several families of techniques [4]. Here the families and approaches of each have been described, and a representative example has been provided of each of the technique.

1) Minimization Techniques

Minimization-based regression test selection techniques, attempt to select minimal sets of test cases from T that yield coverage of modified or affected portions of P [5].For example, the technique of Fischer et al uses systems of linear equations to express relationships between test cases and basic blocks. The technique uses a 0-1 integer programming algorithm to identify a subset T'of T that ensures that every segment that is statically reachable from a modified segment is exercised by at least one test case in T that also exercises the modified segment.

2) Dataflow Techniques

Dataflow-coverage-based regression test selection techniques select test cases that exercise data interactions that have been affected by modifications. For example, the technique of Harrold and Soffa requires that every definition-use pair that is deleted from P, new in P, or modified for P' be tested. The technique selects every test case in T that, when executed on P, exercised deleted or modified definition-use pairs, or executed a statement containing a modified predicate [5].

3) Safe Techniques

Most regression test selection techniques—Minimization and dataflow techniques among them—are not designed to be safe. Techniques that are not safe can fail to select a test case that would have revealed a fault in the modified program. In contrast, when an explicit set of safety conditions can be satisfied, safe regression test selection techniques guarantee that the selected subset, T', contains all test cases in the original test suite T that can reveal faults in P'. Several safe regression test selection techniques have been proposed. The theory behind safe test selection and the set of conditions required for safety have been detailed in Rothermel and Harrold. For example, the technique of Rothermel and Harrold uses control-flow-graph representations of P and P', and test execution profiles gathered on P, to select every test case in T that, when executed on P, exercised at least one statement that has been deleted from P, or that, when executed on P', will exercise at least one statement that is new or modified in P'.

4) Ad Hoc/Random Techniques

When time constraints prohibit the use of a retest-all approach, but no test selection tool is available, developers often select test cases based on "hunches," or loose associations of test cases with functionality. Another simple approach is to randomly select a predetermined number of test cases from T.

5) Retest-All Technique

The retest-all technique simply reuses all existing test cases. It effectively "selects" all test cases in the suit.

B. Previous Work

Unless test selection, program execution with the selected test cases, and validation of the results take less time than rerunning all test cases, test selection will be impractical. Therefore, cost-effectiveness is one of the first questions researchers in this area have studied. Rosenblum and Weyuker and Rothermel and Harrold have conducted empirical studies to investigate whether certain regression test selection techniques are cost-effective relative to retest-all. Rosenblum and Weyuker applied their regression test selection algorithm, implemented in a tool called TestTube, to 31 versions of the KornShell and its associated test suites. For 80% of the versions, their algorithm required 100% of the test cases. The authors note, however, that the test suite for KornShell contained a relatively small number of test cases, many of which caused all components of the system to be exercised.

In contrast, Rothermel and Harrold [4] applied their regression test selection algorithm, implemented in a tool called DejaVu, to a variety of programs. For a set of 100–500 line programs DejaVu was able to discard an average of 45% of the test cases, while for a larger software system (50,000 lines) it was able to discard an average of 95%. Thus, although our understanding of the issue is incomplete, there is some evidence to suggest that test selection can provide savings. Therefore, further empirical investigation of test selection is warranted.

The only comparative study of regression test selection techniques that is known in the literature to date was performed by Rosenblum and Rothermel and compared the test selection results of TestTube and DejaVu. Their study showed that TestTube was frequently competitive with DejaVu in terms of its ability to reduce the number of test cases selected, but that DejaVu sometimes substantially outperformed TestTube. The study did not consider relative fault detection abilities, or compare techniques other than safe techniques.

C. USE OF GENETIC ALGORITHMS IN REGRESSION TESTING

The use of genetic Algorithms in regression testing has been studied in the paper "Prioritizing Regression Test Suites for Time-Constrained Execution Using a Genetic Algorithm by Kristen Walcott, Department of Computer Science, Allegheny College in May 2005. In the paper it has been discussed that complete testing is too expensive. The main aim is to detect whether new errors have been introduced into previously tested code and to provide confidence that modifications are correct. By increasing the overall rate of fault detection, a greater number of errors can be found more rapidly in the code. This research proposes a new test case prioritization technique using GAs. The GAs prioritizes subsequence of the original test suite so that the new suite, which is run within a time constrained execution environment, will have a superior rate of fault detection when compared to rates of naively prioritized test suites. The experiment analyses the genetic algorithm with regard to effectiveness and time/space overhead by utilizing structurally-based criterion to prioritize test cases. An Average Percentage of Faults Detected (APFD) metric was used determine the effectiveness of the new test case orderings.

D. COUPLING AND ITS TYPES

Coupling is the degree to which each program module relies on each one of the other module. Low coupling often correlates with high cohesion, and vice versa. The software quality metrics of coupling and cohesion were invented by Larry Constantine, an original developer of Structured Design who was also an early proponent of these concepts Low coupling is often a sign of a well-structured computer system and a good design, and when combined with high cohesion, supports the general goals of high readability and maintainability [7].

III. Genetic Algorithms

Genetic Algorithms are adaptive heuristic search algorithms which are based on Charles Darwin theory of the survival of the fittest. The main idea behind these algorithms was to replicate the randomness of the nature. This required that the algorithm proposed should behave like a natural system. GAs emulates the nature to large extent. GAs produce a population in such a way that the trait which is popular, that is, has higher fitness value is replicated more, as is done by the nature. This is also the fundamental concept behind evolution. So, these algorithms are also referred as the evolutionary algorithms [9].

A. Steps in Genetic Algorithms

A brief overview of the steps involved in GAs is as follows.

Step 1: A population having P individuals are randomly generated by pseudo random generators whose individuals may represent a feasible solution. This is a representation of solution vector in a solution space and is called initial solution. This ensures the search to be unbiased, as it starts from wide range of points in the solution space.

Step 2: Individual members of the population are evaluated to find the objective function value.

Step 3: In the third step, the objective function is mapped into a fitness function that computes a fitness value for each member of the population. This is followed by the application of GA operators.

B. Genetic Algorithm Operators

1) Reproduction Operator: Reproduction is done on the basis of Rowlett Wheel selection. It selects chromosomes from the initial population and enters them into the mating procedure.

2) Crossover Operator: Crossover Rate (0 to 1) determines the probability of producing a new chromosome form the parents. For example, the strings 10000100 to 11111111 could be crossed over after the third locus in each to produce the two offspring 10011111 to 11100100. The crossover operator roughly mimics biological recombination between two single-chromosomes (haploid) organisms.

3) Mutation Operator: It randomly changes its genetic makeup. This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001) [8].

IV. PROPOSED WORK

Regression Testing calls for the testing of the modules when one of the modules has been changed. This can be done by the retest all method which is pretty expensive and time consuming. Moreover all the tests cannot be executed as the time factor is too important in any project. So prioritization of the test cases is needed and that too in a way which takes care of the type of module and takes in to account the coupling effect as discussed in the previous sections. Considering the above stated reasons a technique has been presented that prioritizes regression test suites on the bases of

1. Coupling
2. Reducing the prioritization problem into 0/1 knapsack problem and hence finding out the most important test cases which needs to be checked when changes are made in one of the modules.

In summary, the work aim to implement the following;

1. Giving values to the modules on the basis of coupling.
2. A GAs based technique to prioritize a regression test suite.
3. The above technique helps us giving values to the modules on the basis of coupling and also reduce the prioritization problem into 0/1 knapsack problem so that GAs can be applied to find out the modules to be tested. Moreover the time required will be much less as compared to the conventional methods of regression testing.

C. COUPLING NUMBER CALCULATOR

Identify the type of module if it’s an undesirable coupling then we multiply it by 0.0001 followed by 0.001,0.01,0.1 in that order. Now the module that has low number needs to be checked first and that having a greater number should be tested later. We call this number cop Number: CNO. The type of coupling as follows Interdependence between modules level names: (from worse to better, high coupling is bad).

1) Content/Pathological coupling (Worse)

When a module uses/alters data in another module thane it is called content coupling. Suppose we make a variable result in C language and we intend to calculate (a+b)/ (a-b) a and b are in the FirstModule () and c and d are in the SecondModule () respectively. Now if we calculate result in FirstModule() as a+b and pass its value to int SecondModule(int result) which changes result to result/(c+d) then the coupling will be called Content Coupling.

2) Control Coupling

Two modules communicating with a control flag; first tells second what to do via flag.

3) Common/Global-data Coupling

If two modules communicate via global data then such coupling is called common coupling. For example

```
int i;
Void FirstMethod ()
{
i=5;

printf (“\n%d”,i);
}
Void SecondMethod ()
{
i++;
printf (“\n%d”,i);
}
```

Both FirstMethod and SecondMethod make use of global data i.

4) Stamp/Data-structure Coupling

If two modules communicate via a data structure passed as a parameter and the data structure holds more information than the recipient needs, such a coupling is called Stamp Coupling.

```
Void FirstModule ()
{
Int array [20];
// input
SecondModule (array);
}
Void SecondModule (int *arr)
{
Printf (“%d”,arr[0]);
}
```

The second module needed only the first element of the array but was provided with the whole array so the above is an example of Stamp Coupling.

5) Data Coupling

The parameters passed are only those that the recipient needs. For example consider a Tax Calculator if Rebate() is a method that calculates the rebate and it passes the information to the best method by just passing the rebate calculated, then it’s the best type of coupling.

6) No data coupling

If two modules are independent then there exists no coupling at all and is the best case scenario.

D. FITNESS VALUE CALCULATOR

In the previous step the type of module was identified if it’s an undesirable coupling then we multiply it by the factor described earlier. Now the module that has low number needs to be checked first and that having a higher number should be tested later. We call it Coupling Number: CNO. The high numbered will be desirable and lower number coupling will not be. The modules having lower number (MNO) will be more important for regression testing then having higher one. With the help of value we find out the fitness value which is $1/(1+e^{\text{factor}})$.

E. APPLYING GA

Now software may have thousands of modules having the above numbers. A table having one field as the module id and second field as the value of 1/1+efactor is to be stored in a file. The Genetic Algorithm program will read this file and apply the following steps to identify the most important modules. The steps of Genetic Algorithm that will be applied are shown in the following flow chart. The steps have been explained in previous section.

The algorithm for the above has been presented below

Algorithm:

Step1: find type of coupling from amongst the list

Step2: assign value to variable factor 1 according to the table below

1.	Content coupling :	T1=100
2.	Common coupling :	T2=10
3.	Control coupling:	T3=1
4.	Stamp coupling:	T4=0.1
5.	Data coupling:	T5=0.01

Step 3: find the value of factor = α*factor 1 + β*factor2

Step 4: The value of factor gives us an idea of how well the two modules are related.

Step 5: calculated fitness= 1/1+efactor

Step 6: when all modules have been given values knapsack is applied.

The process has been summarized in the diagram given below. The Coupling Number Calculator calculates the coupling Coefficient. This is followed by Fitness function calculator which calculates the fitness. The problem then reduces to knapsack after which Genetic Algorithms can be applied.

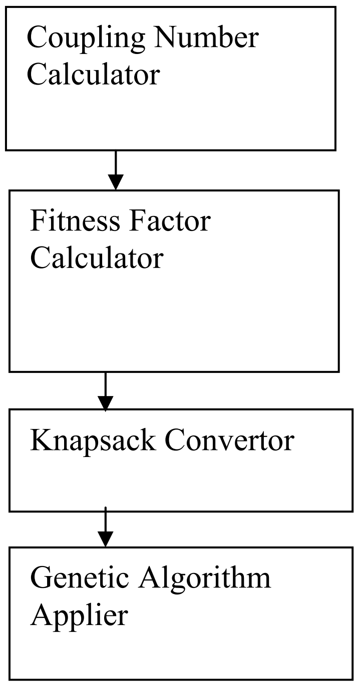
Coupling Number Calculator
Fitness Factor Calculator
Knapsack Convertor
Genetic Algorithm Applier

Fig 1: Test Case Prioritization using Genetic Algorithms

V. CONCLUSIONS

The research outlined in this work a new test case prioritization technique using GAs has been proposed. The proposed technique prioritizes subsequences of the original test suite so that the new suite, which is run, will have a superior rate of fault detection when compared to the rates of randomly prioritized test suites. Reducing the prioritization problem into 0/1 knapsack problem and hence finding out the most important test cases which needs to be checked when changes are made in one of the modules.

In future work, we prioritize the value on the basis of weight based value. This system implements a number of different selection, crossover, mutation, and fitness transformation operators. The system relies on fitness function based on coupling, although any fitness function that conforms to the fitness interface could be used as well. In order to prioritize a test suite, this system requires that coverage and execution time data of a suite's test cases known beforehand and provided in a standard plaintext format even this can be automated.

REFERENCES

[1] Hiralal Agrawal. Joseph R. Horgan. Edward W. Krauser, J. 1993 Incremental Regression Testing. Proceeding ICSM '93 Proceedings of the Conference on Software Maintenance IEEE Computer Society Washington, DC, USA ISBN:0-8186-4600-4

[2] Souter, A. L. and Pollock, L. L., "Incremental Call Graph Reanalysis for Object-Oriented Software Maintenance", in Proceedings International Conference on Software Maintenance, 2001, pp. 682 - 691.

[3] ACMIEEE international symposium on Empirical software engineering and measurement (2008) Publisher: ACM, Pages: 22-31

[4] Gregg Rothermel, Mary Jean Harrold. Analyzing Regression Test Selection Techniques, IEEE Transactions on Software Engineering

[5] W. Eric Wong, J. R. Horgan, Saul London, Hira Agrawal; A Study of Effective Regression Testing in Practice, Proceedings of the Eighth International Symposium on Software Reliability Engineering 1997.

[6] Dr. Arvinder Kaur and Shubhra Goyal. Article: A Genetic Algorithm for Fault based Regression Test Case Prioritization. International Journal of Computer Applications 32(8):30-37, October 2011. Published by Foundation of Computer Science, New York, USA.

[7] Parul Gandhi and Pradeep Kumar Bhatia. Article: Optimization of Object-Oriented Design using Coupling Metrics. International Journal of Computer Applications 27(10):41-44, August 2011. Published by Foundation of Computer Science, New York, USA.

[8] Harsh Bhasin, Surbhi Bhatia: Use of Genetic Algorithms for Finding Roots of Algebraic Equations. International Journal of Computer Science and Information Technology, 2011. Volume 2, Issue 4, pages 693-696.

[9] Harsh Bhasin, Surbhi Bhatia. Application of Genetic Algorithms in Machine learning, 2011. IJCSIT Volume 2 Issue 5, pages : 2412-2415