Test Cases Redundant Elimination on Code Coverage Uses Distance and Correlation Measurement Method

Mochamad Chandra Saputra[*], Tetsuro Katayama[*], Yoshihiro Kita[†], Hisaaki Yamaba[*], Kentaro Aburada[*], and Nanoubu Okazaki[*]

[*]Interdisciplinary Graduate School of Agriculture and Engineering, University of Miyazaki, 1-1 Gakuen-Kibanadai Nishi, Miyazaki, 889-2192 Japan

[†]School of Computer Science, Tokyo University of Technology, Hachioji, Tokyo 192-0982, Japan

Email: chandra@earth.cs.miyazaki-u.ac.jp, kat@cs.miyazaki-u.ac.jp, kitayshr@stf.teu.ac.jp, yamaba@cs.miyazaki-u.ac.jp, aburada@cs.miyazaki-u.ac.jp, oka@cs.miyazaki-u.ac.jp

Abstract

The Euclidean distance and correlation measured by comparing the line executed by the test case to find the similarity of the test cases. The test cases have the lowest value of distance means highly similar and possible executing a similar line or path. The research tries to eliminate redundant test cases based on that similarity. Several redundant test cases are eliminated to get the best test cases. By Euclidean distance, the research can eliminate a similar test case on the test suite.

Keywords: Euclidean distance, redundant test case, test suite, code coverage

1. Introduction

The definition of the test case by IEEE is a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement[1]. Test case repository or test suite may contain redundant, ambiguous, vague, and unfit test cases[2]. Using the test case similarity check on the test suite will decrease the redundant test cases.

The Euclidean distance is used to calculate many cases of similarity. The distance similarity between the test cases with the same length can be calculated by summing the ordered point-to-point distance[3]. The problem with the similarity of the test cases used the distance measurement is not checking the code coverage of the test case. The code coverage needs to check to confirm that each line of code executed on the testing process[4].

The white-box testing consists of analyzing the source code in order to guide the selection of test data uses several methods and concerns the internal logic of source code. Basis path testing guarantees that the test case executing every statement in the program at least one time during testing[5]. Using a given test suite on the source code throughout the software testing process helps to understand the code coverage [6].

This research enhances the previous research on test case redundancy[7] which combines preprocessing on test case similarity. In the previous research, the redundant test cases are investigated by path coverage using all the test cases on the test suite. The duplicate test case must be eliminated before investigating the redundant test case by path coverage to reduce the number of test cases on the test suite. The elimination of redundant test cases is to reduce the number of test cases on the test suite by using the Euclidean distance score. After that, code coverage on test cases is investigated using path coverage to eliminate redundant test cases and the best test cases on the test suite are acquired to achieve 100% code coverage.

2. The Principle of Redundant on Test Case

The consequence of redundant test cases is many test cases with no use. The redundancy will increases the

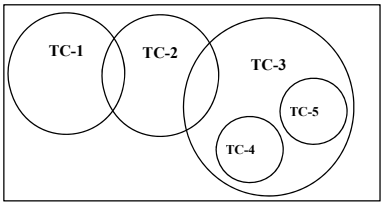

Figure 1. Concept of redundant test cases

the testing effort, cost and time of testing[8]. The fundamental concept of a redundant test case on code coverage will illustrate uses a union of sets on a mathematical approach. The test cases in the test suite may cover one or more code coverage. The research will identify redundant test cases based on coverage information using the line of code executed by the test cases.

The Venn diagram will be used to illustrate the concept of redundancy in Figure 1. The example, there are 5 test cases (TC-1, …, TC-5) that intersect with each other. The intersect means that some or all coverage from the test case containing another test case.

In Figure 1, there are 3 test cases (TC-1, TC-2, TC3) that have an intersection with other test cases. There are 2 redundant test cases because the coverage of the test case is already covered by another test case[9]. Finding and eliminating redundant test cases will increase the effectivity of the testing process.

3. Euclidean Distance

The experiment using the Euclidean distance calculation to find the similarity among the test cases on the test suites. The distance score on Euclidean is fiddling around with distance measures for some time especially with regard to profile comparison methodologies[10].

$$d = \sqrt{\sum_{i=1}^{p} \left(V_{1_i} - V_{2_i}\right)^2} \qquad (1)$$

The equation 1 is the Euclidean formula for calculating the distance between the two variables. Euclidean distance can calculate the similarity of the objects such as two person's profiles, a person and a target profile, in fact basically any two vectors taken across the same variables[10].

Identifying the similarity of a test case on the test suite is critical. Similar test cases in the test suite might repeat the same testing phase. The experiment tries to identify similar test cases by the distance score and then removing

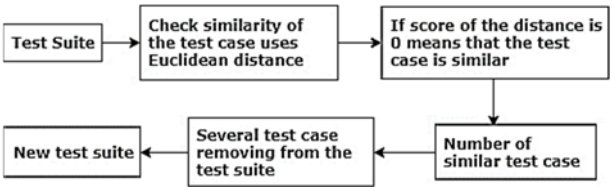

Figure 2. Step on similarity check on the test cases

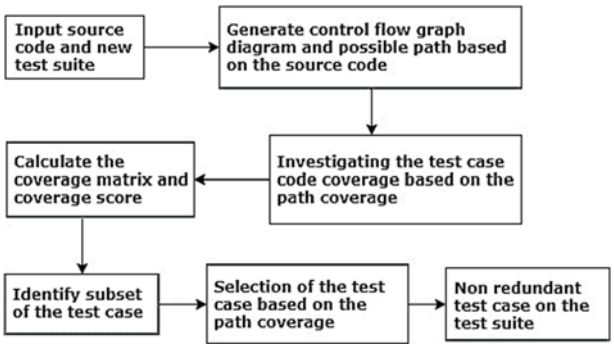

Figure 3. Step by step to find redundant test cases based on the path coverage

the several test cases. The lowest score of the similarity means that the test cases have highly similar.

4. Methodology

The experiment suggests using distance and correlation measurement as preprocessing to minimize the number of test cases on the test suite. The distance and correlation measurement on the pre-processing is to calculate the similarity among the test case that already executed on the source code. The step by step preprocessing describes in Figure 2. The result from preprocessing is a new test suite without the similarities on the line executed by the test case.

The next step is to find the redundant test cases based on path coverage to ensure that the test suite achieves 100% code coverage. The path coverage is used to find the code coverage of the test case which already cover or under coverage by another test case. The coverage matrix and coverage score used to know the relation between the test case and path. The step by step to find redundant test cases based on the path coverage describes in Figure 3.

5. The Experiment

The experiment examines code coverage on the java source code with if-condition as shown in Figure 4.

```
package nestedif;
import java.util.Scanner;
public class IFCond {
    private static Scanner input;
    public static void main(String[] args) {
        int score;
        input = new Scanner(System.in);
          System.out.println(" Please Enter you Score: ");
          score = input.nextInt();
        if(score>=90 && score<=100){
            System.out.println("Your Grade is A");
        } else if (score<90 && score>=80){
            System.out.println("Your Grade is B");
        } else if (score <80 && score >=70){
            System.out.println("Your Grade is C");
        } else if (score <70 && score >=60){
            System.out.println("Your Grade is D");
        } else{
            System.out.println("You Faild on this class, try
next year");
        }
    }
}
```

Figure 4. Java source code with if-condition.

Table 1. Example of the result of the test case executed

Test case name	Result of the test case
TC-1	0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0;
TC-2	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,1,0,1,0;

Table 2. Example of the test case distance measurement by Euclidean distance

Test case distance	Distance score
1 & 2	2.4494897
1 & 3	2
1 & 4	2

The given test suite consists of 10 test cases that already executed on java source code. The result of executing the test cases used in the experiment is line executed.

Table 1 shows the data from the test case, TC-1 is the name of the test case. The value of the test case result consists of the lines executed by the test case. The 0 value means that the line on the code is not inspected by the test case and 1 means that inspected.

The first process of the experiment is to find a similar test case uses Euclidean distance from the test suite to reduce the number of test cases. The example of the result is shown in Table 2. The distance score defines the similarity among the test case. The value of the distance score is 0 means that the test case is definitely similar. The experiment can remove several test cases to eliminate a similar test case.

The new test suite without a similar test case will be used on the next step. Before investigating the redundant test cases based on the code coverage using the path coverage method, the experiment generated

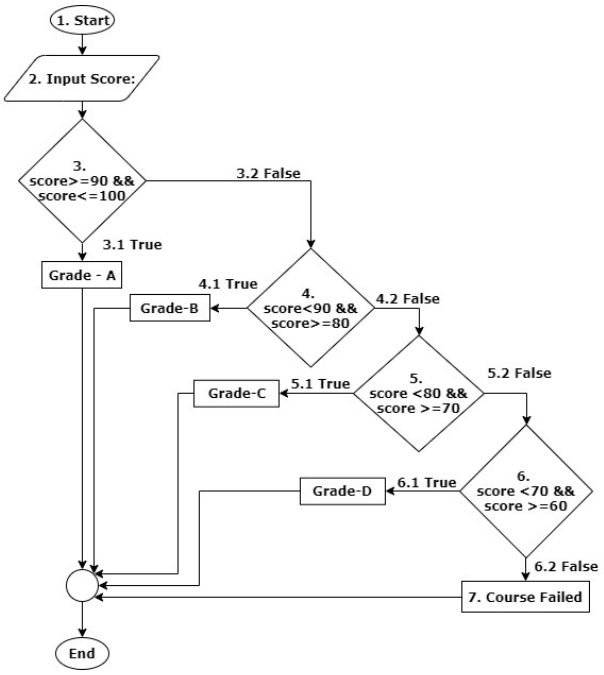

Figure 5. Flow graph of the if-condition java source code

Table 3. Result of coverage matrix

Path	Subset
P1	S1=(TC-1)
P2	S2=(TC-8)
P3	S3=(TC-4)
P4	S4=(TC-5)
P5	S5=(TC-2, TC-6, TC-7)

the control flow graph and the possible path of the if-condition java source code as shown in Figure 5.
Based on the flow graph, the path of the source code is as follow:

- Path-1 (P1): 1-2-3-3.1-End
- Path-2 (P2): 1-2-3-3.2-4-4.1-End
- Path-3 (P3): 1-2-3-4-4.2-5-5.1-End
- Path-4 (P4): 1-2-3-4-5-5.2-6-6.1-End
- Path-5 (P5): 1-2-3-4-5-6-6.2-7-End

The new test suite is investigated to the path and the result using on coverage matrix and coverage score calculation. The coverage matrix describes the possibility of the test case on the path. On the other hand that one path may be investigated by no one or more than one test case such as shown in Table 3.

The coverage score is calculated by the coverage matrix result. The coverage score describes the number of paths executed by the test case divided by all number

of paths. The example result of a coverage score below contains coverage score (CS) on TC-n and then using the coverage score calculation.

1. CS(T1)= 1/5 = 0.2
2. CS(T2)= 1/5 = 0.2

Identifying the subset of the test cases used the coverage matrix and coverage score to guide on selecting the final test suite without a redundant test case.

6. Result and Discussion

The experiment uses the pre-processing to find a similar test case based on the score of the distance. The result of distance calculation using Euclidean distance with distance score 0 are TC-2 & TC-6,TC-2 & TC-7, TC-3 & TC-9, TC-3 & TC-10, TC-6 & TC-7 and TC-9 & TC-10.

Several test cases must be eliminated to find a unique test case. The experiment will select and delete several test cases by random. This is the step to select the test cases.

(i) TC-2==TC-6, TC-6 Deleted, Result: TC-2
(ii) TC-2==TC-7, TC-7 Deleted, Result TC-2
(iii) TC-3==TC-9, TC-9 Deleted, Result TC-3
(iv) TC-3==TC-10, TC-10 Deleted, Result TC-3
(v) TC-6==TC-7, already deleted.
(vi) TC-9==TC-10, already deleted.

The result of the selection of similarity test cases is TC-2, TC-3 and test case deleted is TC-6, TC-7, TC-9, TC-10. The test case elimination by the distance score is not considered the test case code coverage. The result of the selection test case similarity then uses to check the path coverage of the test case to ensure the code coverage. The new test suite consists of TC-1, TC-2, TC-3, TC-4, TC-5, TC-8. The result after the elimination of a similar test case, only 6 test cases need to verify the redundancy and TC-3 is redundant. The redundant on this condition because of the TC-3 is covered by another test case. The result of the new test case can achieve 100% code coverage is TC-1, TC-2, TC-4, TC-5, TC-8.

7. Conclusion

The research confirms the number of redundant test cases on the test suite is reduced because of the similarity. The current research eliminates a similar test case before checking the redundancy of the test case based on path coverage. In the experiment, the new test suite selected by Euclidean distance that must be verified on redundancy by path coverage is reduced by 40% and also satisfy 100% code coverage with 1 test case redundant that analyzes by path coverage.

Elimination of test cases based on coverage information does not guarantee to keep the fault detection capability of a given test suite. Future research is needed to consider the similarity in fault detection capability of test cases for the purpose of redundancy detection.

References

1. The Institute of Electrical and Electronics Engineers - IEEE. *IEEE Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries*. 1990.
2. Mohanty, H., Mohanty, J. R. & Balakrishnan, A. *Trends in Software Testing*. (Springer Singapore, 2017). doi:10.1007/978-981-10-1415-4.
3. Cassisi, C., Montalto, P., Aliotta, M., Cannata, A. & Pulvirenti, A. *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. in Advances in Data Mining Knowledge Discovery and Applications (InTech, 2012). doi:10.5772/49941.
4. Saputra, M. C. & Katayama, T. *Code Coverage Visualization on Web-Based Testing Tool for Java Programs*. J. Robot. Netw. Artif. Life 2, 2015 , pp. 89.
5. Roger S. Pressman, P. D. *Software Engineering: A Practitioner's Approach*. (McGraw-Hill, 2010).
6. Heed, P. & Westrup, A. *Automated platform testing using input generation and code coverage*. Department of Computer Science, Lund University, 2009.
7. Panday, A., Gupta, M., Singh, M. K. & Ali, N. *Test Case Redundancy Detection and Removal Using Code Coverage Analysis*. MIT Int. J. Comput. Sci. Inf. Technol. 3, 2013 , pp. 6–10.
8. Alian, M., Suleiman, D. & Shaout, A. *Test Case Reduction Techniques - Survey*. Int. J. Adv. Comput. Sci. Appl. 7, 2016 , pp. 264–275.
9. Koochakzadeh, N., Garousi, V. & Maurer, F. *Test Redundancy Measurement Based on Coverage Information: Evaluations and Lessons Learned*. in 2009 International Conference on Software Testing Verification and Validation (IEEE, 2009). , pp. 220–229 doi:10.1109/ICST.2009.8.
10. Barrett, P. *Euclidean Distance Raw, Normalized, And Double-Scaled Coefficients*. The Technical Whitepaper Series 6, 2005.