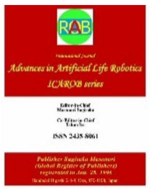
SUGISAKA MASANORI

Journal of Advances in Artificial Life Robotics
Vol. 1(4); March (2021), pp. 187–192
ON LINE ISSN 2435-8061; ISSN-L 2435-8061
https://alife-robotics.org/jallr.html



Research Article

Identification of Redundant Test Cases by Using Similarity Measurement Method for Test Suite Evaluation

Mochamad Chandra Saputra[1], Tetsuro Katayama[1], Yoshihiro Kita[2], Hisaaki Yamaba[1], Kentaro Aburada[1], Naonobu Okazaki[1]

[1]Interdisciplinary Graduate School of Agriculture and Engineering, University of Miyazaki, 1-1 Gakuen-Kibanadai Nishi, Miyazaki, 889-2192 Japan
[2]Department of Information Security, Faculty of Information Systems, Siebold Campus, University of Nagasaki, 1-1-1 Manabino, Nagayo-cho, Nishi-Sonogi-gun, Nagasaki, 851-2195 Japan

ARTICLE INFO

Article History

Received 10 November 2019
Accepted 13 November 2020

Keywords

Euclidean distance
Redundant test cases
Test suite evaluation

ABSTRACT

Evaluating the test suite that contains redundant test cases is necessary to ensure the efficiency of the testing and reducing the cost of testing. The principle of redundant test cases on this research is any test cases on a similar path executed with a similar high value of normalized Euclidean distance. The first, the similarity calculated between the test cases and path coverage uses Euclidean distance. The value of the Euclidean distance on the test case with the lowest value of distance indicating highly similar and possibly executing similar lines of code. The normalized Euclidean distance is using to normalize the value from Euclidean distance result. The experiment uses two java programs. Each redundancy score is 0.37 and 0.67, respectively. It means 37% and 67% of the test cases should be avoided because there are inefficiency test cases on the test suite. The research confirms redundant test cases can be identified by Euclidean distance and normalized Euclidean distance to evaluate the test suite.

© 2022 *The Author*. Published by Sugisaka Masanori at ALife Robotics Corporation Ltd.
This is an open access article distributed under the CC BY-NC 4.0 license
(http://creativecommons.org/licenses/by-nc/4.0/).

1. Introduction

The software contains the series of instruction which execute several tasks to achieve the objective. The several representations of the instruction on the view of the software developer are the action to objects able to work on several tasks and applied to the programming language as the method or class. The testing is important for assessing the method or class run as well as required. Testing helps to increase the source code stability and to ensure that the release version of the software is stable and no impacted user by the mistake on code development.

One of testing technique is structural testing or called white box testing. The white-box testing uses the internal structure of the software under test. The white-box testing is to analyze the source code which concerns the internal structure of source code. The basic principle on white-box testing is to test every method or class by applying the test cases on the unit testing. Using unit testing which applied test cases helps to find the code coverage information as one of the valuable information in the white-box testing [1].

The oldest and one of the most popular approaches in the structural testing technique is basis path testing. Basis path testing is based on the control structure of a program. Basis path testing steps are drawing flow graph then finding all possible paths (independent path) covered and the last, due to the testing all those paths are must be executed [2]. This research uses the basis path testing approach especially to generate the flow graph for considering the code coverage. IEEE has the definition of the test case 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

Corresponding author's E-mail: chandra@earth.cs.miyazaki-u.ac.jp, kat@cs.miyazaki-u.ac.jp, kita@sun.ac.jp, yamaba@cs.miyazaki-u.ac.jp, aburada@cs.miyazaki-u.ac.jp, oka@cs.miyazaki-u.ac.jp

requirement [3]. The test case is applied to unit testing considers individual components are tested to ensure they operate correctly [4]. The programmer uses the unit testing to test individual program units, such as procedures, functions, methods, or classes.

Evaluating the test suite is necessary to ensure the efficiency of the testing and reducing the cost of testing. Test cases on the test suite are a strong element to evaluate to increase efficiency [5]. Reducing the cost of testing is associated with a test process that includes the cost of designing, maintaining, and executing test cases [6]. Test suite may contain redundant, ambiguous, vague, and unfit test cases [7]. Therefore, evaluate the test suite is critical to find the redundant test cases that should be avoided to have better efficiency and the cost of testing.

Testing a software system contains a various set of test cases which capable of examining several parts of the source code. The test cases are redundant if the test case executing similar lines of code, the path of source code, or the test case already covering by another at the current testing. This research tries to identify the redundant test case on the test suite by using the test case similarity based on code coverage information and then calculates the redundancy score for the test suite to evaluate it.

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 [8]. This research is comparing the lines of code executed by the test case and the lines must be executed by the basis path testing approach on calculating the similarity. The code coverage needs to check to confirm that each line of code executed on the testing process [9].

2. The Principle of Redundant on Test Case

The consequence of redundant test cases is many test cases with not suitable to use. The redundancy will increase the testing effort, cost, and time of testing [10]. The test cases in the test suite may examine overlapping or similar code coverage. Figure 1 uses to illustrate the concept of redundancy. The example, there are test case A, test case B, and path-1. The redundant test cases exist when either two test cases serve the same purpose related to the same path. The principle of redundant test cases in this research is in case some test cases have a similar path with a similarly high value of normalized Euclidean distance to other test cases.

The redundancy score is needed to know the degree of test suite redundancy. The redundant score is calculated

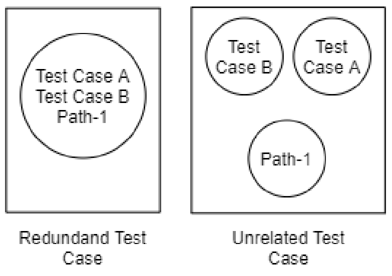


Figure 1. The concept of redundant test cases

by using the percentage approach. The percentage calculation is conducting by dividing the number of redundant test cases on the test suite with the number of test cases. The degree of redundancy represents several redundant test cases in the test suite and the range the score is from 0 to 1. The formula such as follow.

$$Redundancy\ Score = \frac{\sum redundant\ test\ cases}{\sum Test\ cases} \quad (1)$$

where redundant test cases are the number of test cases that have the highest normalized Euclidean distance result and similar to the other test cases at a similar path that should be avoiding. The redundancy score calculated by dividing the number of redundant test cases with the number of test cases on the test suite.

3. Similarity

The similarity measurement is a process to quantified the similarity between two objects. Commonly, the quantification of similarity is using the distance calculation algorithm. One of the distance calculation algorithms is Euclidean distance. This paper uses the Euclidean distance to measure the similarity between test case coverage and path coverage [11]. The coverage is the number of lines of code executed when the test cases are used in testing. The similarity between the test case and path coverage is measured based on the coverage line. The principle of similar in this research is when the result from Euclidean distance between the path and test case coverage has the lowest value of similarity. The lowest Euclidean distance value means that the test cases have highly similar.

Similar test cases in the test suite will through the same path. Using the Euclidean distance, the similarity between the test case and path coverage is described as follows. Let be two object x as path and y test case vectors of the length of index n, and x_i and y_i the i^{th} index of the line of code executed from x and y, respectively. This condition presents distance functions used in Euclidean as follows.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2)$$

The result from the Euclidean distance shows how similar the test case and path. The Euclidean distance result is not scalable. Because of this condition, the result from Euclidean distance is needed to normalize for more scalable on similarity. Normalizing the Euclidean distance result is to change the values to a common scale without distorting differences in the ranges of values. The range of value between 0 to 1. The normalized Euclidean distance is calculated as follows.

$$\text{Normalized Euclidean distance} = \frac{1}{1+d(x,y)} \quad (3)$$

Where $d(x,y)$ is the result of Euclidean distance. The highest normalized Euclidean distance result means that two objects have highly similar.

4. Methodology

The objective of this research is to evaluate the test suite by identified the redundant test cases. *Student grades* and *quadratic function*, which are java programs with each test suite, are used in this research. The test cases and flow graph are generated directly from the java source codes.

This research collects information from the test case executed as the lines of code executed by the test case. The values lines of code are 1 for the executed lines by the test case and 0 for line not executed. The information the lines of code executed for the path coverage also observe based on the independent path.

A. Similarity Calculation

The Euclidean distance formula (2) calculation on this research uses lines executed from path and test case. The result from this calculation is distance value indicating the degree of similarity.

B. Normalized Euclidean Distance

The normalized Euclidean distance on formula (3) uses the result of Euclidean distance to change the Euclidean distance value more scalable for similarity. The value 1 means that the path and test case is similar. The lowest value means that the similarity is low.

C. Identify The Redundant Test Cases

The identification of redundant test cases in this research uses the result from the normalized Euclidean distance. The redundant test cases are when several test cases have a similar path with a similarly high value of normalized Euclidean distance to other test cases.

Table 1. The result of the test case executed on Student Grades

Test case name	Test case coverage
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;
TC-3	0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0;
TC-4	0,0,0,0,0,0,1,1,1,1,0,1,0,1,1,0,0,0,0,0,1,0;
TC-5	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,1,0,0,0,1,0;
TC-6	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,1,0,1,0;
TC-7	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,1,0,1,0;
TC-8	0,0,0,0,0,0,1,1,1,1,0,1,1,0,0,0,0,0,0,0,1,0;

Table 2. The result of the test case executed on Quadratic Function

Test case name	Test case coverage
TC-1	0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,1,0,0;
TC-2	0,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,1,0,0;
TC-3	0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0;
TC-4	0,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,1,0,0;
TC-5	0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0;
TC-6	0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,1,0,0;
TC-7	0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,1,0,0;
TC-8	0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0;
TC-9	0,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,1,0,0;

D. Calculate The Redundancy Score

The number of redundant test cases on the test suite is identified by using the normalized Euclidean distance result, and then the redundant score is calculated by formula (1). The test cases are redundant when the test cases have similar highest value from normalized Euclidean distance result at a similar path. The number of redundant test cases refers to the number of avoiding test cases on the test suite. The redundancy score represents the degree of an inefficiency test case on the test suite.

5. The Experiment

The experiment has two java source codes: s*tudent grades* and *quadratic function* and also test suite for them. The test suite for *student grades* consists of 8 test cases and 9 test cases for a *quadratic function*. The information from executing the test cases uses in the experiment is lines executed as shown in Table 1 and Table 2. The 0 value on test case coverage information means that the line of code is not executed and 1 is executed.

The experiment generated the flow graph based on the java source code as shown in Figure 2 and 3. Based on the flow graph that has been drawing then calculated the value of cyclomatic complexity related to the number of

Table 3. Path coverage

Java Source Code	Path	Path coverage
Student Grades	P1	0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0;
	P2	0,0,0,0,0,0,1,1,1,1,0,1,1,0,0,0,0,0,0,1,0;
	P3	0,0,0,0,0,0,1,1,1,1,0,1,0,1,1,0,0,0,0,1,0;
	P4	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,1,0,0,1,0;
	P5	0,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,1,0,1,0;
Quadratic Function	P1	0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,1,0,0;
	P2	0,0,0,0,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,1,0,0;
	P3	0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0;

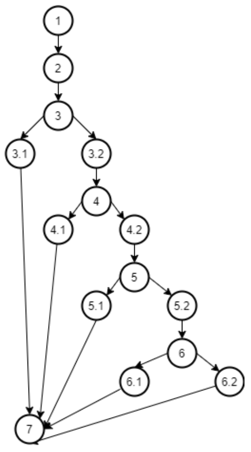


Figure 2. Flow graph of the Student Grades

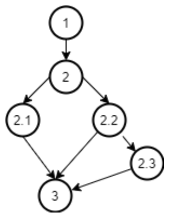


Figure 3. Flow graph of the Quadratic Function

independent path. The principle on basis path testing is that all independent paths are exercised to ensure that all statements in a method have been executed at least once.

One of the famous software quality metrics is cyclomatic complexity which used to find the logical complexity of software by considering the flow graph on pseudocode or source code [2]. The result of cyclomatic complexity from student grades with 4 predicate nodes is 5 and for quadratic function with 2 predicate node is 3. Based on this result the number of the independent path for student grades is 5 and for the quadratic function is 3.

Based on the flow graph, the path coverage related to the source code is as shown in Table 3, the 0 value on path coverage information means that the line of code is not executed and 1 is executed. The information for the lines of code executed that refers to the path coverage gain by

Table 4. The result from Euclidean distance for student grades

	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7	TC-8
Path 1	0.00	2.45	0.00	2.00	2.24	2.45	2.45	1.73
Path 2	1.73	2.24	1.73	1.73	2.00	2.24	2.24	0.00
Path 3	2.00	2.00	2.00	0.00	1.73	2.00	2.00	1.73
Path 4	2.24	1.73	2.24	1.73	0.00	1.73	1.73	2.00
Path 5	2.45	0.00	2.45	2.00	1.73	0.00	0.00	2.24

Table 5. The result from Euclidean distance for quadratic function.

	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7	TC-8	TC-9
Path-1	0.00	2.45	3.00	2.45	3.00	0.00	0.00	3.00	2.45
Path-2	2.45	0.00	2.65	0.00	2.65	2.45	2.45	2.65	0.00
Path-3	3.00	2.65	0.00	2.65	0.00	3.00	3.00	0.00	2.65

Table 6. The result from normalized Euclidean distance for student grades

	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7	TC-8
Path 1	1.00	0.29	1.00	0.33	0.31	0.29	0.29	0.37
Path 2	0.37	0.31	0.37	0.37	0.33	0.31	0.31	1.00
Path 3	0.33	0.33	0.33	1.00	0.37	0.33	0.33	0.37
Path 4	0.31	0.37	0.31	0.37	1.00	0.37	0.37	0.33
Path 5	0.29	1.00	0.29	0.33	0.37	1.00	1.00	0.31

Table 7. The result from normalized Euclidean distance for the quadratic function

	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7	TC-8	TC-9
Path-1	1.00	0.29	0.25	0.29	0.25	1.00	1.00	0.25	0.29
Path-2	0.29	1.00	0.27	1.00	0.27	0.29	0.29	0.27	1.00
Path-3	0.25	0.27	1.00	0.27	1.00	0.25	0.25	1.00	0.27

observing every path and java source code. The next step is calculating the similarity by using Euclidean distance. The result calculation by using Euclidean distance show in Table 4 and Table 5. The result from Euclidean distance is used to calculate the normalized Euclidean distance. The normalized Euclidean distance allows the value of Euclidean distance scalable.

The result of normalized Euclidean distance as shown on Table 6 and 7 uses to identify the redundant test case. The identification of redundant test cases is when the value from normalized Euclidean distance on several test cases has a similar path with a similarly high value to other test cases. The result of redundant test cases shown in Table 8. The number of redundant test cases is used to calculate the redundancy score by formula (1).

6. Result and Discussion

The result from Euclidean distance confirms that several test cases on the student grades and quadratic function have high similarity with path coverage. The lowest value of Euclidean distance indicates that high similarity. The result of Euclidean distance needs to scalable for easy investigation on the redundant test cases.

Table 8. The result of the test case redundancy based on the normalized Euclidean distance.

Sudent Grades	Quadratic Function
Path-1 contains TC-1 and TC-3.	Path-1 contains TC-1,TC-6 and TC-7
Path-2 contains TC-8.	Path-2 contains TC-2, TC-4 and TC-9
Path-3 contains TC-4.	Path-3 contains TC-3, TC-5 and TC-8
Path-4 contains TC-5.	
Path-5 contains TC-2, TC-6, and TC-7.	

The redundant test case is investigated from normalized Euclidean distance calculation on test cases and path. When the result is 1, it means that highly similar. There are several test case which has a high value of normalized Euclidean distance same with other test cases in the same path. This condition is called redundant test cases. The number of redundant test cases is shown in Table 8.

The redundant test cases on student grades found on path 1 and path 5. On quadratic function, there is a redundant test case on path-1, path-2, and path-3. This research uses an independent path in which one test case is required for one path to guarantee the coverage of source codes. The principle of software testing is to ensure one path executed at least once, so one test case adequate to test one path. By using this principle, the redundant test case should be avoided. The number of the redundant test case based on the result normalized Euclidean distance which should avoid on student grades is TC-3, TC-6, TC-7, and for the quadratic function is TC-4, TC-5, TC-6, TC-7, TC-8, TC-9.

This research also calculates the redundancy score by using the formula (1). The redundancy score for student grades is 0.37 and 0.67 for the quadratic function. The redundancy score can represent the percentage of redundancy. The meaning of the percentage is there are 37% of the test case on the student grades test suite is redundant and 67% on the quadratic function that should be avoided because there is an inefficiency test cases on the test suite.

7. Conclusion

This research confirms that the redundant test cases can be identified by Euclidean distance and the normalized Euclidean distance to evaluate the test suite. The Euclidean distance is to measure the similarity between test case coverage and path coverage. The normalization of the Euclidean distance result is necessary for more scalable value for test suite evaluation. The current research identified the redundant test cases on the result from normalized Euclidean distance calculation. The redundancy score for student grades is 0.37 and 0.67 for the quadratic function. The redundancy score can represent the percentage of redundancy. The meaning of the percentage is there are 37% of the test case on the student grades test suite is redundant and 67% on the quadratic function that should be avoided because there are inefficiency test cases on the test suite.

Future research is needed to consider other evaluation methods of test suite, for example the similarity in fault detection capability, reusability of test cases or test suite, and so on.

References

[1] P. Heed and A. Westrup, "*Automated Platform Testing Using Input Generation and Code Coverage*,"Lund, Sweden: Department of Computer Science, Lund University, 2009.

[2] D. Madhavi, "*A White Box Testing Technique in Software Testing: Basis Path Testing*," *J. Res.*, vol. 02, no. 04, pp. 12–17, 2016.

[3] The Institute of Electrical and Electronics Engineers - IEEE, "*IEEE Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries*," 1990.

[4] B. B. Agarwal, S. P. Tayal, and M. Gupta, "*Software Engineering & Testing An Introduction*,"United States of America: Jones and Bartlett Publisher, 2010.

[5] A. Farooq and R. R. Dumke, "*Evaluation Approaches in Software Testing*," *Science (80-.).*, p. Nr.: FIN-05-2008, 2008.

[6] K. Naik and P. Tripathy, "*Software Testing and Quality Assurance*,"vol. 1. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008.

[7] H. Mohanty, J. R. Mohanty, and A. Balakrishnan, "*Trends in Software Testing*,"Singapore: Springer Singapore, 2017.

[8] M. C. Saputra and T. Katayama, "*Code Coverage Visualization on Web-Based Testing Tool for Java Programs,*" J. Robot. Netw. Artif. Life, vol. 2, no. 2, pp. 89–93, 2015

[9] M. C. Saputra and T. Katayama, "*Code Coverage Visualization on Web-Based Testing Tool for Java Programs,*" J. Robot. Netw. Artif. Life, vol. 2, no. 2, pp. 89–93, 2015.

[10] M. Alian, D. Suleiman, and A. Shaout, "T*est Case Reduction Techniques - Survey,*" Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 5, pp. 264–275, 2016.

[11] K. L. Elmore and M. B. Richman, "*Euclidean Distance as a Similarity Metric for Principal Component Analysis,*" Mon. Weather Rev., vol. 129, no. 3, pp. 540–549, Mar. 2001.

==================================

Authors Introduction

Mr. Mochamad Chandra Saputra

He received the Master's Degree from the University of Miyazaki, Japan, and Brawijaya University, Indonesia on Double Degree Program 0n 2014. From 2006 - 2014 also work in Brawijaya University ICT Unit as System Analyst. Since 2015 has been a lecturer on the Faculty of Computer Science, Brawijaya University. Now continue the Doctoral Study at the University of Miyazaki. The research interest includes software testing, software quality, and software project management.

Dr. Tetsuro Katayama

He received the Ph.D. degree in engineering from Kyushu University, Fukuoka, Japan in 1996. From 1996 to 2000 he has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2000 he has been an Associate Professor at Faculty of Engineering, Miyazaki University, Japan. He is currently a Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include software testing and quality. He is a member of the IPSJ, IEICE, and JSSST

Dr. Yoshihiro Kita

He received a PhD degree in systems engineering from the University of Miyazaki, Japan, in 2011. He is currently an Associate Professor with the Faculty of Information Systems, University of Nagasaki, Japan. His research interests include software testing and biometrics authentication.

Dr. Hisaaki Yamaba

He received the B.S. and M.S. degrees in chemical engineering from the Tokyo Institute of Technology, Japan, in 1988 and 1990, respectively, and the Ph D. degree in systems engineering from the University of Miyazaki, Japan, in 2011. He is currently an Assistant Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include network security and user authentication. He is a member of SICE and SCEJ.

Dr. Kentaro Aburada

He received the B.S., M.S and Ph.D. degrees in computer science and system engineering from the University of Miyazaki, Japan, in 2003, 2005 and 2009, respectively. He is currently an Associate Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include computer network and security. He is a member of IPSJ and IEICE.

Dr. Naonobu Okazaki

He received his B.S, M.S., and Ph.D. degrees in electrical and communication engineering from Tohoku University, Japan, in 1986, 1988 and 1992, respectively. He joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation in 1991. He is currently a Professor with the Faculty of Engineering, University of Miyazaki since 2002. His research interests include mobile network and network security. He is a member of IPSJ, IEICE and IEEE.

==================================