ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

An Intelligent Framework for Test Case Prioritization Using Evolutionary Algorithm☆

Mojtaba Raeisi Nejad Dobuneh[1*] Dayang N.A. Jawawi[1]

ABSTRACT

In a software testing domain, test case prioritization techniques improve the performance of regression testing, and arrange test cases in such a way that maximum available faults be detected in a shorter time. User-sessions and cookies are unique features of web applications that are useful in regression testing because they have precious information about the application state before and after making changes to software code. This approach is in fact a user-session based technique. The user session will collect from the database on the server side, and test cases are released by the small change configuration of a user session data. The main challenges are the effectiveness of Average Percentage Fault Detection rate (APFD) and time constraint in the existing techniques, so in this paper developed an intelligent framework which has three new techniques use to manage and put test cases in group by applying useful criteria for test case prioritization in web application regression testing. In dynamic weighting approach the hybrid criteria which set the initial weight to each criterion determines optimal weight of combination criteria by evolutionary algorithms. The weight of each criterion is based on the effectiveness of finding faults in the application. In this research the priority is given to test cases that are performed based on most common http requests in pages, the length of http request chains, and the dependency of http requests. To verify the new technique some fault has been seeded in subject application, then applying the prioritization criteria on test cases for comparing the effectiveness of APFD rate with existing techniques..

☞ keyword : Regression Testing, Prioritization, Evolutionary Algorithm, Web Application, User Session.

1. Introduction

The purpose of regression testing is to provide confidence that the newly introduced changes do not obstruct the behavior of the existing, unchanged part of the software. Regression testing is one of the largest maintenance costs during the software development cycle. It is a complicated process for web applications based on modern architectures and technologies. User session based testing has the benefit that tests can be automatically constructed from web logs for use in regression testing and they contain sequences of actions that real users have performed.

User session based test cases also have the benefit that testers do not need to specify input for test cases. For instance, web applications are accessible through the Internet and each http POST and GET request that a user makes is written to a log file. The logs can then be passed into test cases by using the IP addresses, cookies, and time stamp for each POST and GET request in order to identify the steps of each user and to create the respective test cases [1].

In general, for each version of a given application a large number of reusable test cases are generated. Thus, there might be a huge pool of test cases which are accumulated from different application versions. Many of these test cases could be applied to test newer versions of the application without need to generate new test cases [2].

However, running all the test cases may take a significant amount of time. The three major approaches for regression testing are test suite minimization, test case selection and test case prioritization [3]. Test suite minimization is a process that seeks to identify and then eliminate the obsolete or redundant test cases from the test suite. Test case selection deals with the problem of selecting a subset of test cases that will be used to test the changed parts of the software. Finally, Test case prioritization (TCP) helps us to find out more faults by using different orders of test cases. In the prioritization process, it is assumed that all test cases must be executed. But, it tries

[1] Software Engineering Department, Faculty of Computing, Universiti Teknologi Malaysia (UTM), Skudai 81310, Johor, Malaysia.
[Received 9 May 2016, Reviewed 24 May 2016, Accepted 15 June 2016]
* Corresponding author: (rmojtaba2@live.utm.my)
☆ A preliminary version of this paper was presented at ICONI 2015 and was selected as an outstanding paper.

http://dx.doi.org/10.7472/jksii.2016.17.5.89

to get the best schedule running of test cases in a way that if the test process is interrupted or early halted at an arbitrary point, the best result is achieved in which more faults are detected [4].

The paper is organized as follows: In section two, the basic flow of the test case prioritization is briefly reviewed. A detailed description of the framework what we have proposed is given in section three. A result brings in section four. Finally, our conclusions and some possible paths of future research are given in section five.

2. TEST CASE PRIORITISION

We formally define the test case prioritization problem as follows:

Definition 1: The Test Case Prioritization Problem: T, a test suite, PT, the set of permutations of T, and f, a function from PT to the real numbers.

Problem: Find T´∈ PT such that (∀T´´) (T´´∈ PT) (T´´≠ T´) [f (T´) ≥ f (T´´)].

Permutation of test cases in a way that leads to faster detection of maximum available faults in a modified version of web application needs to find good criteria. The goal of this research is to propose a new approach which improves the test case prioritization techniques by dynamic weighting criteria in web application regression testing

Regression testing takes place during the maintenance phase of the software development life cycle. Common challenges that are encountered during regression testing are (a) managing the size and quality of an evolving test suite and (b) creating new test cases or reusing old test cases for testing new or changed parts [5].

To address the first challenge, researchers have proposed test case selection strategies, such as reduction and prioritization [6], [7]. For the latter challenge, identifying the changed parts of the code for test case generation and test case repair of older test cases are often the strategies that are used [8], [9]. This section describes the advances in regression testing for the domain of web application testing.

2.1 MOTIVATION

In general, the slight modifications on web applications increase the number of test cases, considerably. As a result, the effectiveness of the conventional testing process would be decreased. In real world scenario the issue of scalability is challenging. For example, suppose there are 100,000 test cases need to be executed. Obviously it is unrealistic to expect a human tester to provide reliable responses for such a large number of test cases.

The most appropriate test cases for a web application are session-based because session's best reflect real user patterns, making the testing process quite realistic [10]. The User-Session based techniques are new, useful lightweight mechanisms of testing. Automating the test process is more feasible and simpler in user sessions when applied on web applications.

In user-session approaches, the total interactions of users with the server are collected and the test cases are generated by using a suitable policy. The client's request, transported as URLs and composed of page addresses and name value pairs, are the data to be captured. These data that can be found in the log files are stored in servers the data collected from user sessions can be used to generate a set of http requests and converted into a real test case. The benefit of the approach is to generate the test cases without any awareness web application's internal structure.

The log file format example is shown in Table 1, the test cases for web application comes from a log file which has some parameters such as date & time %t, http version %H, request method %m, resource address %U, session ID %S, response code %s.

3. PROBLEM FORMULATIONS

In this section, the proposed model by applying Imperialist Competitive Algorithm (ICA) tries to run the test case in the discovery of error is less and the more influence we will explain.

This algorithm proposed by atashpaz et al. [11] which is started by generating a set of candidate random solutions in the search space of the optimization problem. The generated random points are called the initial Countries. Countries in this algorithm are the counterpart of Chromosomes in Genetic Algorithm (GA) and Particles in Particle Swarm Optimization (PSO) and it is an array of values of a candidate solution of optimization problem [12]. The cost function of the optimization problem determines the power of each country. Based on their

(Table 1) SERVER LOG FILE SAMPLE

%t- Date & Time	%H- Request Protocol	%m- Request Method	%U- Requested Url Path	%S- User Session ID	%s- Status Code
[27/Aug/2013:06:23:28]	HTTP/1.1	GET	favicon.ico	[3A3D0AA2D1F19603B07241DC409C]	[404]
[27/Aug/2013:06:23:30]	HTTP/1.1	GET	/AdminMenu.jsp	[3A3D0AA2D1F19603B07241DC409C]	[302]
[27/Aug/2013:06:23:34]	HTTP/1.1	POST	/Login.jsp	[3A3D0AA2D1F19603B07241DC409C]	[200]
[27/Aug/2013:06:23:39]	HTTP/1.1	GET	/Default.jsp	[3A3D0AA2D1F19603B07241DC409C]	[200]
[27/Aug/2013:06:24:26]	HTTP/1.1	GET	/ShoppingCart.jsp	[3A3D0AA2D1F19603B07241DC409C]	[200]
[27/Aug/2013:06:24:34]	HTTP/1.1	GET	/Default.jsp	[3A3D0AA2D1F19603B07241DC409C]	[200]
[27/Aug/2013:06:24:45]	HTTP/1.1	GET	/BookDetail.jsp	[3A3D0AA2D1F19603B07241DC409C]	[200]
[27/Aug/2013:06:24:58]	HTTP/1.1	GET	/images/Logo _ bookstore.gif	[B75CFC481E6F59A7BC1169D8705D]	[200]

power, some of the best initial countries (the countries with the least cost function value), become Imperialists and start taking control of other countries (called colonies) and form the initial Empires.

Two main operators of this algorithm are assimilation and revolution. Assimilation makes the colonies of each empire get closer to the imperialist state in the space of socio-political characteristics (optimization search space). Revolution brings about sudden random changes in the position of some of the countries in the search space. During assimilation and revolution, a colony might reach a better position and has the chance to take the control of the entire empire and replace the current imperialist state of the empire.

Imperialistic Competition is another part of this algorithm. All the empires try to win this game and take possession of colonies of other empires. In each step of the algorithm, based on their power, all the empires have a chance to take control of one or more of the colonies of the weakest empire.

The algorithm continues with the mentioned steps (Assimilation, Revolution, Competition) until a stop condition is satisfied.

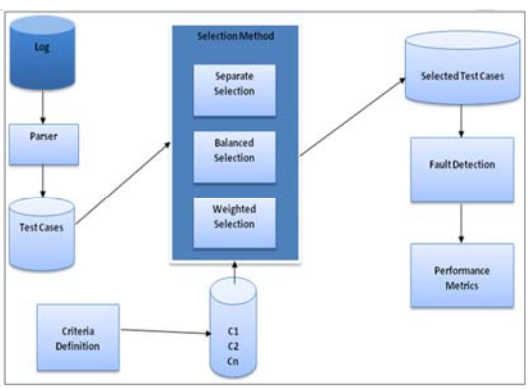
3.1 PROPOSED APPROACH

This section describes the proposed solution, based on dynamic weighting criteria, for the problem of test case prioritization with precedence.

Figure 1 illustrates the intelligent framework. In this framework log files are collected from server side, and then the parser is used to generate test cases. After that the test cases are stored in the database. Selection methods utilize generated test cases and prioritize them based on the defined criteria. Three different techniques have been applied in this framework. All techniques use the same criteria to prioritize the test cases. The criteria are defined as following:

1. Count-based criteria (number of most common HTTP requests in pages)
2. Length-based criteria (length of HTTP request chains)
3. Parameter-value (ascending or descending numbers of parameters)
4. Frequency-based criteria (dependency of HTTP requests).

The Separate Selection technique applies the defined criteria one by one to prioritize the test cases.



(Figure 1) INITIAL INTELLIGENT FRAMEWORK

The Balanced Selection technique uses a combination of the criteria to prioritize test cases. In this technique two or more

balanced criteria are combined together [13]. For instance, if two criteria are combined, the weightage of both will be 50%. In case of the proposed framework, there are four criteria introduced, so weightage assigned to each criterion will be 25%.

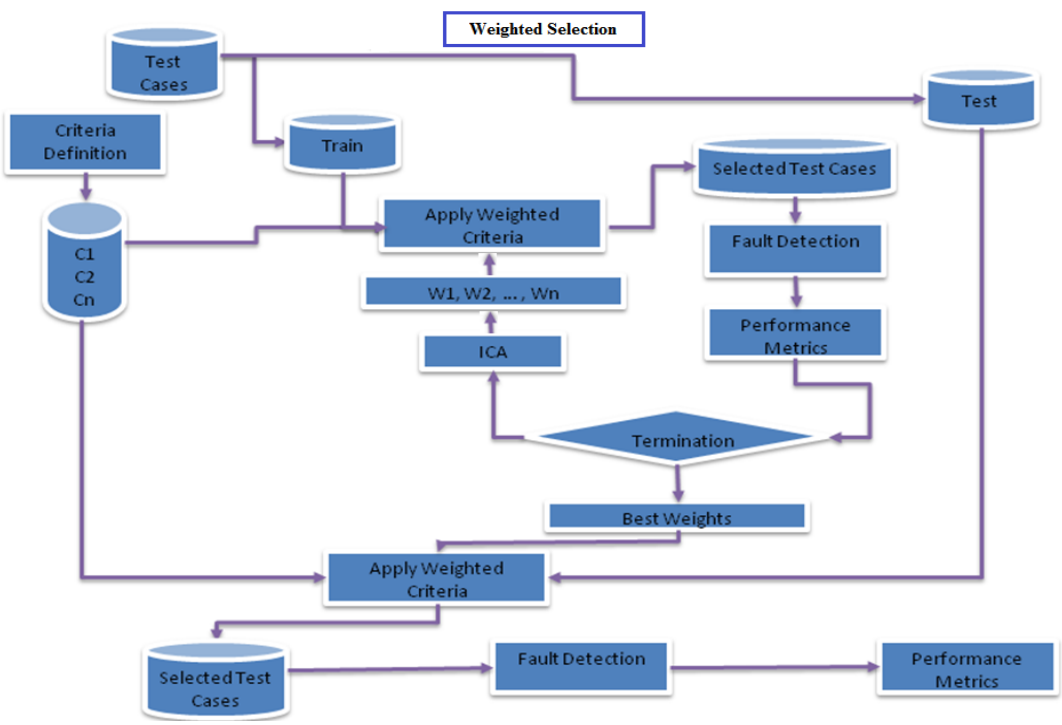
In the Weighted Selection technique, dynamic weighting criteria are applied to prioritize the test cases based on effectiveness of the criteria for finding the faults. A detailed explanation of weighed selection is discussed later. After applying the selection method, the selected test cases by each technique is executed for finding faults in the application. Then, the performance of each technique is evaluated by Average Percentage Fault Detection (APFD) rate.

In this framework we proposed three techniques in order to choose the best type of prioritization to sort the test cases for testing web-based applications which is used to detect the maximum amount of faults by executing less number of test cases. In addition, defining a new criterion is possible which could be utilized by the proposed techniques for test case prioritization. The testers can flexibly add new criteria according to the application types.

In Figure 2, the weighted selection for prioritization of test cases is shown in detail which is the part of Selection Method Module. The ICA initiates the weight optimization of criteria based on effectiveness, and then the calculated weight is applied on the combination of criteria. This leads to prioritize the test cases in the database on the best weights. After the test cases are prioritized these test cases are executed finding the maximum faults.

Following are the algorithmic steps of dynamic weighting criteria model:

1. Apply the initial weightage to the defined criteria for prioritizing the test cases.
2. Execute the prioritized test cases to find faults in the application.
3. Count the number faults detected.
4. Calculate the APFD rate to check performance of the executed test cases in each iteration.
5. If the optimized performance is achieved, means best weight of criteria is found.



(Figure 2) DYNAMIC WEIGHTING CRITERIA APPROACH

6. Else run the ICA to find the best weightage of each criterion by achieving the most optimized performance.

To evaluate the proposed technique train and test method was used. The ICA was applied to the training set data for generating the best weightage. The calculated best weightage was applied to the selected test cases for validation.

3.2 Evaluation of Fault Detection Rate

For evaluation of FDR each criterion takes a weight based on importance. The aim of optimizing of test case execution to find more faults that have detected by the composition of weighted criteria. This can express as:

$$FDR= w_1C1 + w2C_2 + w3C3 + \cdots + wnCn$$

Where C1, C2, Cn are the constant coefficient and, w1, w2, wn are the predictive variable, FDR is the response variable. The FDR function for evaluating rate of fault detection is defined as Table 2.

(Table 2) SYMBOLS and DEFINITIONS

Symbol	Definition
FDR	Represent of Fault Detection Rate
C	Refer to the Criteria
W	Is Weight of each criteria

If we know the value of several explanatory variables for an individual, but do not know the value of that individual's dependent variable, we can use the prediction equation (based on a model using the known variables as its explanatory variables) to estimate the value of the dependent variable for that individual. Typically, the algorithm is done for one of two purposes: In order to predict the value of the dependent variable for individuals for whom some information concerning the explanatory variables is available, or in order to estimate the effect of some explanatory variable on the dependent variable [14].

Function for calculating cost of imperial competition for test case optimization is detailed as follows:

Input: Test suite T, number of faults detected by a test case f, and cost to run each test case T.

Output: prioritized test suite T'.

Cost Function

Function Y = Test (x)

For i= 1 to size (x, 1)

Si= x (i, 1) × criteria (n)

Y (i, 1) = T – E

End

4. RESULTS

To quantify the goal of increasing a subset of the test suite's rate of fault detection, we use a metric called APFD developed by Elbaum et al. [15]. That measures the average rate of fault detection per percentage of test suite execution. The APFD is calculated by taking the weighted average of the number of faults detected during the run of the test suite.

$$APFD = 1 - \{(TF1 + TF2+... + TFm) / mn\} + (1 / 2n)$$

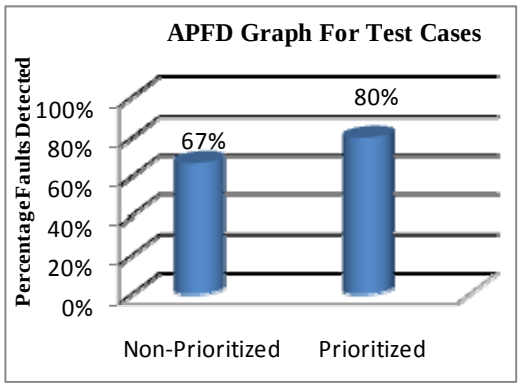
Where n is the number of test cases and m being the number of faults. (TF1 + TF2+⋯ + TFm) are the position of first test T that exposes the fault.

(Table 3) FAULT DETECTION RATE CALCULATION

Faults	Test Cases									
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1	×					×				
F2				×				×		
F3		×								×
F4							×		×	
F5		×				×				
F6	×									
F7	×							×		
F8							×		×	
F9				×						×
F10		×					×			

To illustrate this measure, consider the program with 10 faults and a suite of 10 test cases as shown in Table 3. The prioritized order according to Fi is:

{T4, T2, T1, T7, T6, T9, T10, T5, T8, T3}



(Figure 3) FAULT DETECTION RATE GRAPH

No. of test cases (n) = 10, No. of faults (m) = 10, the position of the first test in T that exposes fault I = TFi.

APFD= 1- {(3+1+2+4+2+3+3+4+1+2)/100} + (1/20) = 80%.

In our paper, we can use the APFD metric for the performance based evaluation and the proposed test sequence is {T4, T2, T1, T7, T6, T9, T10, T5, T8, T3}. Then the APFD metric after prioritization is 80% and the APFD metric before prioritization is 67% as per our formula. Figure 3 shows that the APFD metric comparison for both prioritized and non-prioritized test cases. The APFD is a measure that the average number of faults identified in a given test suite.

5. CONCLUSION

This approach significantly reduces the effort and cost of a rigorous cycle of software development and testing process, as regression test cases can be efficiently generated while maintaining high standards of software quality. A web application system has been tested as the research object by applying several new prioritization criteria to these test suites for identifying whether they can be used to increase the rate of fault detection. An optimized dynamic weightage is calculated and applied on all the criteria for prioritizing the test cases to obtain higher fault detection rate. This study shows that when frequency metrics and systematic coverage parameter-value use ICA as evolutionary computing algorithm it increases the fault detection rate of web applications.

Reference

[1] Yuan-Fang Li, Paramjit K. Das, David L. Dowe, "Two Decades of Web Application Testing: A Survey of Recent Advances," Information Systems, pp. 20 – 54, 2014. http://dx.doi.org/10.1016/j.is.2014.02.001

[2] Serdar Dogan, Aysu Betin-Can, Vahid Garousi, "Web Application Testing: A Systematic Literature Review," *The Journal of Systems and Software*, pp. 174-201, 2014. http://dx.doi.org/10.1016/j.jss.2014.01.010

[3] S. Sampath, and S. Sprenkle, "Advances in Web Application Testing, 2010 – 2014," Advances in Computers, Elsevier Inc. ISSN 0065-2458, 2016. http://dx.doi.org/10.1016/B978-0-12-396535-6.00008-9

[4] S. Sampath, R. Bryce, and A. M. Memon, "A Uniform Representation of Hybrid Criteria for Regression Testing," *IEEE Transactions on Software Engineering, Vol. 39, No. 10*, Oct. 2013. http://dx.doi.org/10.1109/TSE.2013.16

[5] Wei Zheng, Robert M. Hierons, Miqing Li, XiaoHui Liu, Veronica Vinciotti, "Multi-Objective Optimization for Regression Testing," *Information Sciences*, pp. 1-16, 2016. http://dx.doi.org/10.1016/j.ins.2015.11.027

[6] H. Srikanth, L. Williams, and J. Osborne, "test case prioritization of new and regression test cases," *Proceedings of the Empirical Software Engineering, International Symposium on: IEEE*, Oct. 2005. http://dx.doi.org/10.1109/ISESE.2005.1541815

[7] P. Dhareula, and A, Ganpati, "Prevalent Criteria's in Regression Test Case Selection Techniques: An Exploratory Study," *IEEE International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 871-876, 2015. http://dx.doi.org/10.1109/ICGCIoT.2015.7380585

[8] T. Muthusamy, K. Seetharaman, "A Test Case Prioritization Method with Weight Factors in Regression Testing Based on Measurement Metrics," *International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 12,* Dec. 2013. http://dx.doi.org/ 10.3923/jse.2010.193.214

[9] D. Mondal, H. Hemmati, and S. Durocher "Exploring Test Suite Diversification and Code Coverage in

Multi-Objective Test Case Selection," *IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2016.
http://dx.doi.org/10.1109/ICST.2015.7102588

[10] Amanda Schwartz, Hyunsook Do, "Cost-Effective Regression Testing Through Adaptive Test Prioritization Strategies" *The Journal of Systems and Software*, pp. 61-81, 2016.
http://dx.doi.org/10.1016/j.jss.2016.01.018

[11] E. Atashpaz-Gargari, C. Lucase, "Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition," *IEEE Congress on Evolutionary Computation (CEC), Singapore*, pp. 4661-4667, Sept. 2007.
http://dx.doi.org/10.1109/CEC.2007.4425083

[12] C. Catal, "On the Application of Genetic Algorithms for Test Case Prioritization: A Systematic Literature Review," *ACM*, 2012.
http://dx.doi.org/10.1145/2372233.2372238

[13] M. R. N. Dobuneh, D. N. A. Jawawi, M. Ghazali, M.V. Malakooti, "Development Test Case Prioritization Technique in Regression Testing Based on Hybrid Criteria," *8'th Malaysian Software Engineering Conference (MySEC)*, pp. 301-305, Sept. 2007.
http://dx.doi.org/10.1109/MySec.2014.6986033

[14] A. Mor, "Evaluate the Effectiveness of Test Suite Prioritization Techniques Using APFD Metric," *IOSR Journal of Computer* Engineering (IOSR-JCE), Volume 16, Issue 4, PP. 47-51, Aug. 2014.
http://dx.doi.org/10.9790/0661-16414751

[15] S. Elbaum, A. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," *IEEE Transactions on Software Engineering*, Feb. 2002.
http://dx.doi.org/10.1109/32.988497

◐ Authors ◑

Mojtaba Raeisi Nejad Dobuneh
received the B.E. degree from the Kerman University, Iran, in 2008, and received the M.S. degrees from the Azad University, Tehran, Iran, in 2011. From 2011 to 2016, he joined to the Software Engineering Group of Universiti Teknologi Malaysia as a PhD candidate and research member. His research interests include web application testing and communication systems and internet based system.

Dayang N. A. Jawawi
received her B.Sc. in Software Engineering from Sheffield Hallam University, UK, and her M.Sc. and Ph.D. in the field of Software Engineering from Universiti Teknologi Malaysia. She has been an academic staff at Software Engineering Department since 1997 and she has served as the Head of Department from November 2009 till January 2015. She is a member of the Software Engineering Research Group (SERG), K-Economy Research Alliance UTM.