An Optimization Technique for Generating Minimal Pairwise Test cases for Testing WEB Links

1M. Lakshmi Prasad, 2M. Nikhitha, 3G. Divya, 4Y. Kulasekhar Reddy, 5S. Shandhya, 6P. Haneesha

1Dept. of Computer Science & Engineering, NBKRIX, Nellore, AP, India.
23456Dept. of Computer Science & Engineering, NBKRIX, Nellore, AP, India.

Abstract: Many types of web technologies are in existence, which are being used for the development of websites. The web technologies span across the clients and many of the servers. The web sites developed using the web technologies must be of high quality in nature and should be secured, fault-tolerant, transactional, fail-safe and recoverable systems. Every web site must be tested thoroughly so that all the features that are needed to be built in to websites can be guaranteed and assured. Many websites are being tested extensively considering the content, navigation, easy access, data persistence, response time, client server interactions, etc.

Different types of testing’s must be carried which include unit testing, interface testing, integration testing, system testing, module testing, etc. Audit trails also are necessary to ensure proper completion of testing of web sites. Many tools are being used in the market for undertaking the testing of web sites in which bank on testing through user interface. Thus, the methods being used are not full proof and therefore do not guaranteed the proper running of the business, which are launched on to the websites.

Testing of a website requires identification of test requirements, test case generation, test prioritization, test dependency, setting test environment, undertaking the testing and carrying quality certification. The entire process of testing must be completed as fast as possible to ensure the reduction in the cost of testing the web based application.

Generation of the test cases for testing the web sites is one of the most critical phases of the test process. Pair-wise testing is a combinatorial test criterion technique based on specification, which requires each pair of input variables and every combination of their valid values should be covered by at least one test case. Some of pairwise test techniques like AETG (Automatic Efficient Test Generator), IPO (In Parameter Order), SBC (Simulated Bee Colony) etc. generate test cases by pairing the input variables. The methods generates the test cases which are less in number compared to total test cases that can be used to test the system exhaustively. These methods also take lesser time to generate the test cases.

This paper proposes a new strategy called M-PTCG (Minimal-Pairwise Test Case Generator) to generate optimized pair wise test cases for testing web links which are used in many web based applications like railway reservation systems, etc. The method presented in this paper is a deterministic based strategy for generating the optimized pairwise test cases to test WEB links in a web based applications. Finally the results are compared with other methods like Genetic algorithms (GA), Bee Colony Optimization (BCO) and ACO (Ant colony optimization) etc., the experimental results demonstrate that results generated are optimal when compare to other methods.

Keywords: Pairwise testing, Genetic algorithms (GA), Bee Colony Optimization (BCO) and Ant colony optimization (ACO).

I. INTRODUCTION

A web application or web app is any software that runs in a web browser. It is created in a browser-supported programming language (such as the combination of JavaScript, HTML and CSS) and relies on a web browser to render the application. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions.

Combinatorial testing of software analyzes interactions among variables using a very small number of tests. Combinatorial methods have been shown to significantly reduce cost and increase quality for software and system testing. The basis for combinatorial testing is the interaction rule, which is based on analysis of thousands of software failures. The rule states that most failures are induced by single factor faults or by the joint combinatorial effect (interaction) of two factors, with progressively fewer failures induced by interactions between three or more factors. Therefore
if all faults in a system can be induced by a combination of or fewer parameters, then testing all t-way combinations of parameter values is pseudo-exhaustive and provides a high rate of fault detection. Two primary tools can be used: ACTS, to generate tests, and CCM (Combinatorial Coverage Measurement), to measure combinatorial aspects of test quality.

A combinatorial testing approach is a particular kind of functional testing technique consisting in exhaustively validating all combinations of size t of a system’s inputs values. This is equivalent to exhaustively testing t-strength interaction between its input parameters, and requires a formal modeling of just the system features as input variables. In particular, pairwise interaction testing aims at generating a reduced-size test suite which covers all pairs of input values.

Many of the combinatorial testing techniques like IPO, AETG, ACO, GA, and BCO are available to generate the minimal test cases and used in many applications like web services, computer networks, embedded systems etc. Based on the survey we have found that there are no combinatorial techniques are available to test a WEB links in a web based applications.

For example, consider a software system has four input parameters, P1, P2, P3 and P4. Each parameter is treated as a one Web page. Each parameter holds some values that mean each value is treated as a one WEB link. For suppose that parameter A can take 3 possible values, {a0, a1, a2} means that a web page contains three internal WEB links. Similarly, the values for the parameters B, C and D are {b0, b1}, {c0, c1, c2, c3}, and {d0, d1} respectively which means that the web page B contains 2 internal links, parameter C holds 4 internal web links and the web page D contains again 2 internal links.

For this case there are a total of 3 * 2 * 4 * 2 = 48 combinations of input values. For example, one arbitrary test vector is {a0, b1, c3, d0}. In this simplistic scenario there are a total of 44 pairs of input values:

\[
\begin{align*}
3: & \quad a1 \ b0 \ c1 \ d0 \\
4: & \quad a0 \ b1 \ c3 \ d1 \\
5: & \quad a1 \ b0 \ c0 \ d0 \\
6: & \quad a2 \ b0 \ c3 \ d0 \\
7: & \quad a0 \ b1 \ c0 \ d1 \\
8: & \quad a1 \ b0 \ c3 \ d0 \\
9: & \quad a2 \ b1 \ c2 \ d1 \\
10: & \quad a2 \ b0 \ c0 \ d0 \\
11: & \quad a0 \ b0 \ c2 \ d1 \\
12: & \quad a2 \ b1 \ c1 \ d1
\end{align*}
\]

By applying our new strategy, the final test suite may need only 8 test vectors to capture all 44 possible pairs of input values:

\[
\begin{align*}
0: & \quad a0 \ b0 \ c0 \ d0 \\
1: & \quad a0 \ b1 \ c0 \ d1 \\
2: & \quad a1 \ b1 \ c1 \ d0 \\
3: & \quad a1 \ b0 \ c1 \ d0 \\
4: & \quad a0 \ b1 \ c3 \ d1 \\
5: & \quad a1 \ b0 \ c0 \ d0 \\
6: & \quad a2 \ b0 \ c3 \ d0 \\
7: & \quad a0 \ b1 \ c0 \ d1 \\
8: & \quad a1 \ b0 \ c3 \ d0 \\
9: & \quad a2 \ b1 \ c2 \ d1 \\
10: & \quad a2 \ b0 \ c0 \ d0 \\
11: & \quad a0 \ b0 \ c2 \ d1 \\
12: & \quad a2 \ b1 \ c1 \ d1
\end{align*}
\]

II. LITERATURE REVIEW

Pair wise test case generation method is a widely popular approach to combinatorial testing problems. Numbers of articles have been published exploring the idea of generating the test cases using pairs of input variables. Despite the technique's popularity and its reputation as a best practice, its use to test various types of applications has not been properly explained. Knowledge of the weaknesses of the pairwise testing technique, or of any testing technique, is essential to apply the technique wisely. A wide variety of different strategies and implementations for generating pair wise test sets have been published in the literature.


Pedro Flores et.al.,[11] had proposed a new strategy called PWiseGen for generating test cases for pairwise testing using genetic algorithms. Priti Bansal et.al.,[12]

D.M. Cohen et.al., [16] had introduced an AETG System: an approach to testing based on combinatorial design. Chandra Prakash Vudatha et.al.[17] had Automated Generation of Test Cases from Output Domain of an Embedded System using Genetic Algorithms. Chandra Prakash Vudatha et.al.,[18], had automated generation of test cases from output domain and critical regions of embedded systems using genetic algorithms. C. Anderson et.al.,[19] had proposed a method on the use of neural networks to guide software testing activities. Meenakshi Vannali et.al.,[20], had designed and developed an Algorithm for prioritizing the test cases using neural network as classifier.

Lilan Wu et.al.[21] had used back-propagation neural networks for functional software testing.

Yogesh Singh et.al.,[22], had predicted testing effort using artificial neural network. Kristina Smilgute et.al.,[23] had proposed an artificial neural networks application in software testing selection method.R. Raju and P.Subhapriya [24] had presented a neural network approach for randomized unit testing based on genetic algorithm

R. Krishnan, et.al, [25] had introduced the concept of combinatorial testing. Lei Xu, et.al, [26] had presented a testing and fault diagnosis for web application compatibility based on combinatorial method. S. Sampath, et.al [27] had introduced the concept of prioritizing user-session-based test cases for web applications testing. Xun Yuan, et.al, [28] had presented a GUI Interaction Testing which incorporated event context. Malte Lochau, et.al, [29] had proposed a model-based pairwise testing for feature interaction coverage in software product line engineering. Martin Fagereng Johansen, et.al, [30] had proposed an algorithm for generating t-wise covering arrays from large feature models

S.R. Dalal et.al. [31] had introduced an AETGSM web: a web based service for automatic efficient test generation from functional requirements. Jinfu Chen et.al, [32] had proposed a web services vulnerability testing approach based on combinatorial mutation and soap message mutation. K.Z Watkins [33] had introduced a fault-based combinatorial testing to web services. Bellanov s. Apilli, [34] presented a fault-based combinatorial testing of web services. Chengying Mao, [35] performed a combinatorial testing on web service-based software. Li Li et al. [36] proposed a combinatorial approach to multi-session testing of state full web services. S. Noikajana et al. [37] introduced an improved test case generation method for web service testing from WSDL-S and OCL with pair-wise testing technique. Qing li et.al, [38] had presented a combinatorial mutation approach to web service vulnerability testing based on soap message mutations.

### III. PROPOSED STRATEGY

**Step-1:** Read the number of parameters and its domain values, Let the input variable set is considered as shown in the table 2. For example, consider the parameters A,B,C,D which can take the domain values for A is a0, a1 and B is b0, b1, b2 and C is c0, c1 and D is do, d1,d2,d3. Total combination of domain values considering the input variables is $3 * 2 * 4 * 2 = 48$

**Step-2:** Generate all the order pairs considering all adjacent domain values of all the variables taken together. The following are the order pairs that can be generated.

Combination of AB parameters : {a0,b0}, {a0,b1}, {a0,b2}, {a1,b0}, {a1,b1}, {a1,b2}
Combination of BC parameters : {b0, c0}, {b0, c1}, {b1, c0}, {b1, c1}, {b2, c0}, {b2, c1}
Combination of CD parameters : {c0, d0}, {c0, d1}, {c0, d2}, {c0, d3}, {c1, d0}, {c1, d1}, {c1, d2}, {c1, d3}
Combination of AC parameters : {a0,c0}, {a0,c1}, {a1,c0}, {a1,c1}
Combination of AD parameters : {a0, d0}, {a0, d1}, {a0, d2}, {a0, d3}, {a1, d0}, {a1, d1}, {a1, d2}, {a1, d3}
Combination of BD parameters : {b0, d0}, {b0, d1}, {b0, d2}, {b0, d3}, {b1, d0}, {b1, d1}, {b1, d2}, {b1, d3}, {b2, d0}, {b2, d1}, {b2, d2}, {b2, d3}

For testing the Web links in a web based application the combination of AB, BC, and CD is sufficient for generating the pairwise test cases. For the remaining combinations i.e combination of AC, AD, BC is not needed to test. So out of 44 pairs, 22 pair is enough to test the web links of an entire web based application.

Out of those 8 test vectors, our technique may cover 15 input pairs which may not needed to test the input pairs i.e {a0, c0}, {a0, c3}, {a1, c1}, {a1, c3}, {a2, c2}, {a0, d0}, {a0, d1}, {a1, d0}, {a1, d1}, {a2, d0}, {a2, d1} {b0, d0}, {b0, d1}, {b1, d0}, {b1, d1}.

**Step-3:** The optimized pairwise test cases generated by using our strategy are as shown in the table 1.

<table>
<thead>
<tr>
<th>Test vector Number</th>
<th>Test Vector elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Element-A</td>
</tr>
<tr>
<td>t1.</td>
<td>a0</td>
</tr>
<tr>
<td>t2.</td>
<td>a0</td>
</tr>
<tr>
<td>t3.</td>
<td>a1</td>
</tr>
<tr>
<td>t4.</td>
<td>a1</td>
</tr>
<tr>
<td>t5.</td>
<td>a2</td>
</tr>
<tr>
<td>t6.</td>
<td>a2</td>
</tr>
<tr>
<td>t7.</td>
<td>a0</td>
</tr>
<tr>
<td>t8.</td>
<td>a1</td>
</tr>
</tbody>
</table>

---

**Table 1 : Test Case Vectors**
The algorithm that was proposed is as shown below.

**M-PTCG Algorithm**

Step 0: Begin

Step 1: Examine the number of parameters

Step 2: Read the number of values for each parameter

Step 3: Read the values for those parameters

Step 4: See the initial parameter and domain values matrix as shown in table 2

Step 5: Construct the Minimal Test suite matrix

Step 5.1: (a) Consider the initial combination of parameter input pairs ie. AB

(b) Then calculate step value = AB pair values /Number of domain values in parameter A

(c) Fill the initial column in such way that by considering step value

Step 5.2 For the second column repeat the second parameter values consecutively until the all the pair is to be filled.

Step 6: Repeat the Step 6 until final test suite matrix is gets filled.

Step 7: Print the Optimized test suite Matrix

Step 8: Stop

**IV. COMPARATIVE ANALYSIS**

The results generated by other techniques which were published were used and thereby the new technique is compared with other deterministic algorithms which are shown in Table1. Experimental results show that the results generated by our method have provided optimal and best solution. The parametric values considered for different systems have been shown in the table 2.

**Table 2 Parametric configurations of systems considered for testing**

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>System</th>
<th>Number of input variable</th>
<th>Number of selected input variable</th>
<th>Number of domain values considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>S2</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>S3</td>
<td>13</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>S4</td>
<td>61</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>S5</td>
<td>75</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>S6</td>
<td>100</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>S7</td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

The numbers of test cases generated for different systems which can be defined using various parametric combinations are shown in the table 3.

**Table 3 : Comparison with Other Strategies**

<table>
<thead>
<tr>
<th>System</th>
<th>AETG</th>
<th>Pair-test (IPO)</th>
<th>All Pairs</th>
<th>SBC</th>
<th>M-PTCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>S2</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>S3</td>
<td>15</td>
<td>17</td>
<td>22</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>S4</td>
<td>41</td>
<td>34</td>
<td>41</td>
<td>35</td>
<td>12</td>
</tr>
<tr>
<td>S5</td>
<td>28</td>
<td>26</td>
<td>30</td>
<td>27</td>
<td>12</td>
</tr>
<tr>
<td>S6</td>
<td>10</td>
<td>15</td>
<td>16</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>S7</td>
<td>194</td>
<td>212</td>
<td>664</td>
<td>198</td>
<td>100</td>
</tr>
</tbody>
</table>

We have shown the compared results generated by other algorithms with the new technique in the form of bar graphs also.

From the above table, it is seen that the number of test cases generated by our technique is quiet minimal for considering any of the system configurations.
V. CONCLUSION AND FUTURE RESEARCH WORK

In this paper, we have showed a new technique i.e. M-PTCG for generating the pairwise test cases for testing the web links in a web based application. The proposed strategy yields best results and possess less computational time when compared to other deterministic algorithms. Hence we concluded that our proposed strategy will be helpful to generate the optimized pairwise test to test Web links in a WEB based application.

REFERENCES


Science (NCETACS), 2011, pp.no.1-6, 4-5 March 2011.


