Generating Test cases for Testing WEB sites through Neural Networks and Input Pairs

Lakshmi Prasad Mudarakola [1] and Dr. JKR Sastry [2],
and Chandra Prakash Vudatha [3]

[1] Dept. of Computer Science & Engineering, NBKRISt, Nellore, AP, India.
prasad.hinduniv@gmail.com.
[2] Dept. of Electronics & Computer Engineering, KL University, Vaddeswaram,
Guntur
drsastry@kluniversity.in
[3] Dept. of Computer Science and Engineering, KL University, Vaddeswaram, Guntur
vchandrap@kluniversity.in

ABSTRACT

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. Pairwise test case generation is the process of producing a subset of all possible test cases that can test the system to a reasonable acceptance. Pairwise testing can be adapted when exhaustive testing is not possible or prohibitively expensive. 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 an artificial neural network (ANN) to generate optimized test cases based on the considerations of the pairs of input variables (Test pair generation). The method presented in this paper demonstrates that the test cases generated through associative mapping of test pairs to the neural network provides optimized solution when compared to other methods especially when test cases are generated to test a WEB applications. A tool called ANN-PTCG (Artificial Neural Network- Pair wise Test Case Generator) has been developed based on the method proposed in this paper. Test cases are generated for testing a WEB based application using ANN-PTCG and other similar pairwise testing methods which include AETG (Automatic Efficient
Test Generator), IPO (In-Parameter Order), and SBC (Simulated Bee Colony). The number of test cases that are generated, the extent of coded covered and the amount of time taken to generate the test cases has been used to compare the ANN-PTCG with other related test case generation methods. It is seen that ANN-PTCG method is quite good compared to other methods due to generation of minimal test cases in less time and the test cases covering entire source code.

**Keywords:** Testing WEB sites, neural networks, Input pairs, AETG (Automatic Efficient Test Generator), IPO (In-Parameter Order), SBC (Simulated Bee Colony),

**I. INTRODUCTION**

Combinatorial testing is a specification based sampling technique that provides a systematic way to select combinations of program inputs or features for testing. It is an effective testing technique to test hardware/software that reveals failures in a given system based on input or output combinations. It has been applied over the years to test system configurations, web forms, protocols, graphical user interfaces, software product lines etc. The pair wise testing can detect possible t-way combinatorial interactions between the input variables when t=2, 3, 4, 5, 6 or more.

A Neural network is an interconnection of several computing stations that are organized into three distinct partitions that include Input layer, output layer and a set of middle processing layers. While there can be one layer in input and output stations there can be many layers in the middle partition. Each of the computing station reacts to an external stimulus and produce output. Input layer passes its output to the middle layer and the middle layer after processing passes the output to output layer.

Inputs are fed to the computing nodes present in the input layer where some kind of normalization is done and the normalized data is passed to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where another level of normalization is taken. The interconnectivity between the computing stations present in each of the layers is shown in the figure 1.

![Figure 1: A Typical Neural Network](image)
The neural network can be recognized for representing a WEB site. The individual WEB site can be recognized as a set of computing nodes and the navigation from one page to other can be implemented by passing output of a processing node as input to another. The neural network thus can represent even huge WEB sites containing even 500 Pages. WEB pages that related to a WEB site can be organized into a set of layers containing one input layer, one output layer and as many as hidden layers. Each hidden computing node can be considered as WEB page and the navigation from one WEB page to another can be achieved keeping the weight of connectivity between the processing nodes as 1 and the others as zero. It is necessary to associate a page in the WEB site to a computing node in the hidden layer of the neural network.

The neural networks must be recognized considering that the same shall be used for undertaking pair wise testing which is proved to generate less number of test cases. There are many pair wise test case generation methods that exist in the literature and all of them slightly differ in terms of number of test cases generated.

In this paper, an algorithm is presented that constructs a neural network which is suitable for representing a WEB site. The neural network is used for generating the test cases using the input pairs. A comparative analysis also is provided that shows that the proposed algorithm produces minimal of the testing cases and therefore is found to be more effective compared to other available pair-wise test case generation methods.

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.

Yu Lei and K.C. Tai [1] have proposed a test generation strategy, called in-parameter-order (or IPO), for pairwise testing. The IPO strategy allows the use of local optimization techniques for test generation and the reuse of existing tests when a system is extended with new parameters or new values of existing parameters. They have presented the implementation of an IPO-based test case generation tool and have shown some empirical results.

Xiang Chen, et.al, [2] have adopted Ant Colony Optimization (ACO) to build prioritized Pairwise Interaction Test Suite (PITS). A biased covering array is proposed and the Weighted Density Algorithm (WDA) is developed. In their research, they have proposed four concrete test generation algorithms based on Ant System, Ant System with Elitist, Ant Colony System and Max-Min Ant System respectively. They
also have implemented these algorithms and applied those considering typical inputs and reported experimental results.

Kewen Li and Zhixia Yang [3] proposed ant colony arithmetic, which is a new way to generate test cases considering the input pairs. It generates fewer test cases which can cover more pair combinations. The method can be used for undertaking regression testing. J.D. McCaffrey [4] has presented the results of generating pairwise test sets using a simulated bee colony algorithm. The bee colony algorithm has generated fewer test cases but has taken long time to generate the test cases.

Xiang Chen, et al. [5] applied particle swarm optimization (PSO), a kind of meta-heuristic search technique that can be used for pairwise testing. The algorithm systematically builds pairwise test suites. The have proposed methods to define test case search space, define the fitness function and set some heuristic settings. Mohammed I. Younis, et.al. [6] proposed an efficient pairwise strategy for generating pairwise combinatorial test set using artificial parameters and values, termed RA and ORA, that can systematically minimize the pairwise test set generated from higher order test parameters to lower order ones.

S.A. Ghazi and M.A. Ahmed [7] proposed a GA-based technique which identifies a set of test configurations that are expected to maximize pair-wise coverage, with the constraint that limit the number of test configurations. S. Khatun, et.al. [8] proposed an effective random search based pairwise test data generation algorithm named R2Way to optimize the number of test cases. The algorithm is developed to support both uniform and non-uniform values effectively.

James D. McCaffrey [9] has presented a method that generates pairwise test sets using a genetic algorithm. The genetic algorithm approach presented by the authors produced test sets which are fewer in number compared to others. J.D. McCaffrey [10] described the results of an investigation of pairwise test set generation using a genetic algorithm.

Pedro Flores and Yoonsik Cheon [11] formulated the problem of finding a pairwise test set as a search problem and applied a genetic algorithm to solve it. They also described an open source tool called PWiseGen for generating pairwise test sets. PWiseGen produces competitive results compared to the existing pairwise testing tools. Besides, it provides a framework and a research platform for generating pairwise test sets using genetic algorithms.

Priti Bansal, et.al. [12] have presented a method to generate initial population using hamming distance and an algorithm to find crossover points for combining individuals selected for reproduction. They have applied a genetic algorithm to find an optimal solution to the pair-wise test set generation problem. They have also described the implementation of the proposed approach by extending an open source tool PWiseGen and evaluated the effectiveness of the proposed approach. The empirical results indicate that their approach can generate test sets with higher fitness level by covering more pairs of input parameter values.

Manisha Patil and P.J. Nikumbh [13] have formulated the problem of finding a pair-wise test set as a search problem and apply a search technique “simulated annealing” to solve it. Their empirical results concluded that the optimal solution for their problem statement is to find the maximum number of different pairs which are
Generating Test cases for Testing WEB sites

the best test set. The key contribution of their work includes simulating annealing algorithm approach for generating the best test case set. A software for web based and semantic based application is developed for determining minimized test cases.

Generating a pair wise test set which has a minimal size for a specified set of parameters and parameter values has been shown to be an NP-complete problem [1]. One of the approaches to generate pair wise tests is the use of orthogonal arrays [15]. Another generation algorithm uses an iterative technique which employs a greedy algorithm to construct one test set vector at a time until all possible pairs are captured [16]. A third approach is to generate a test set for the first two system parameters, and then iteratively extend the test set to account for each remaining parameter [1]. Similarly some of the latest strategies for generating pair wise test case are Ant colony optimization[2][3], Bee colony optimization[4], Genetic algorithms[7][9][10][11][12], simulated annealing[13], and Particle swarm optimization[5].etc.

V. Chandra Prakash et.al,[17] have presented a new approach to automate the generation of test cases from output domain of a pilot project “Temperature Monitoring and Controlling of Nuclear Reactor System” (TMCNRS) which is an embedded system developed using modified Cleanroom Software Engineering methodology. V. Chandra Prakash et.al.[18] have proposed an Automated Test Case Generator (ATCG) that uses Genetic algorithms (GAs) to automate the generation of test cases from output domain and the criticality regions of an embedded System. Exhaustive testing of the embedded systems in the critical regions is important as the embedded systems must be basically fail safe systems. The Critical regions of the input space of the embedded systems can be pre-identified and supplied as seeds.

All these techniques do not completely consider the application framework, architecture and structure for generating the test cases. They all are purely dependent on input domain, output domain and combination of both

C. Anderson, et.al. [19] have proposed a neural network based method for pruning the number of test cases that are generated by an automated tool as innumerous number of test cases are generated by the automated tool and it is not possible to carry the testing using all the generated test cases. Meenakshi Vannali et.al, [20] have proposed a method of generating test cases using artificial neural network to reduce the effort by the tester in generating the test cases manually. They have presented the way a neural network is trained using back propagation method and a set of test cases that have been tested on a system manually. The trained network is used as an artificial oracle for evaluating the correctness of the output produced by new versions of the software

Lilan Wu et.al, [21] have explained how propagation neural networks can be used for functional software testing efficiently and significantly. They have shown how the functional units of software can be used for generation of the test cases through building a neural network representing a system that must be tested.

Yogesh Singh et.al, [22] have proposed a method based on neural networks and object oriented metrics for assessing the quality of a system. The effort needed for testing a system could be predicated by using the method proposed by them.

Kristina Smilgyte et.al, [23] have used an artificial network for selecting a method for undertaking the testing. The authors have proposed a way of determining
the method (automated or manual) that can be used for undertaking the testing. R. Raju and P. Subhapriya [24] have proposed a method of undertaking randomized unit testing using a neural network and genetic algorithms.

It seen that neural networks are being used either for test case generation or for undertaking the testing such as unit testing, integration testing. No author has attempted to find how neural networks can be used to represent specific type of applications and using the neural networks thus constructed to generate the test cases. This paper aims at finding a neural network that best represents an application and then using the same and input pairs of the system, to generate the test cases. A set of algorithms have been presented in this paper using which the neural networks are used for representing the input variables in the hidden domain and generating the test cases using the input pair combinations. The extent of code covered is presented based on the coverage of WEB pages which are placed in different computing nodes.

III. PROPOSED ALGORITHM
The proposed algorithm has two important major steps which include constructing a neural network from the input domain and determining the test vectors which represent minimum number of test cases. Using the test cases generated, code coverage is computed.

Major Step-1:
Algorithm-1

Step-1:
Consider the entire input variable set and their domain values, each value representing the state of an input variable such as the example variables and their domain values shown in the table 1.

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Variable</th>
<th>Domain Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A</td>
<td>a0, a1, a2</td>
</tr>
<tr>
<td>2.</td>
<td>B</td>
<td>b0, b1</td>
</tr>
<tr>
<td>3.</td>
<td>C</td>
<td>c0, c1, c2, c3</td>
</tr>
<tr>
<td>4.</td>
<td>D</td>
<td>do, d1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total combination of domain values considering the input variables =  3 * 2 * 4 * 2 = 48</td>
</tr>
</tbody>
</table>

Step-2:
Generate all the order pairs considering all the domain values of all the variables taken together. The following are the order pairs that can be generated.
Step-3:
In this step a neural network is constructed considering the input pairs which are generated at step-1.

The neural network shall have one input and output layer and there will be as many hidden layers equivalent to number of input variables considered for the system. The number of computing nodes in each of the layer will be equivalent to the number of domain values for each of the variable’s designating and associated to a hidden layer. Each input node is connected to every computing node in the first hidden layer. The connectivity between the processing nodes can be achieved by connecting every computing node in one hidden layer to every computing node in the succeeding hidden layer until the output layer is reached.

The neural network generated by following the above mentioned procedure is shown in the Figure 2.

```
Figure 2: Development of Neural network for a chosen set of input variables.
```

Major Step-2:
In this step Test vectors are generated each representing a test case by using the input pairs and the generated neural networks.

Step-1: Execute the following algorithm:

Algorithm:

```
Do a process equivalent to number of hidden layers that were chosen for a neural network
*

If the hidden layer is the first layer
*

Note the number of elements in the first layer
Do for each of the element in the first layer
*

(a) Generate the value pairs considering chosen element and all the elements that are present in the next hidden layer.
(b) Filter the generated value pairs by comparing each of the value pair with the previously generated value pairs.
(c) The value pairs thus filtered are included into the Test vector matrix as independent test case.

If the hidden layer is not the first layer
*

Note the number of elements in the current layer
Select the last elements in the test vector (Rows) which are previously stored in the test vector matrix
Select the data values in the next hidden layers
For each of the last data element in each of the test vector,
*

(a) Form value pairs with the data elements contained in the next hidden layer, thus generating the value pairs
(b) Filter the generated value pairs by comparing each of the value pair with the previously generated value pairs, thus obtaining filtered value pair set.
(c) Update each of the existing text vectors by adding an element from the matching value pair
(d) Add additional test vectors for each of the test vector presently existing the test matrix using the remaining matching value pairs

Thus the rows left over in the test vector matrix are the test cases that can be used for undertaking the testing. As a result of executing the above mentioned algorithm the test vectors generated are shown in the table 2.
### Table 2 Test Case Vectors

<table>
<thead>
<tr>
<th>Test vector Number</th>
<th>Test Vector elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Element-1</td>
</tr>
<tr>
<td>t1.</td>
<td>a0</td>
</tr>
<tr>
<td>t2.</td>
<td>a0</td>
</tr>
<tr>
<td>t3.</td>
<td>a0</td>
</tr>
<tr>
<td>t4.</td>
<td>a0</td>
</tr>
<tr>
<td>t5.</td>
<td>a0</td>
</tr>
<tr>
<td>t6.</td>
<td>a1</td>
</tr>
<tr>
<td>t7.</td>
<td>a1</td>
</tr>
<tr>
<td>t8.</td>
<td>a1</td>
</tr>
<tr>
<td>t9.</td>
<td>a1</td>
</tr>
<tr>
<td>t10.</td>
<td>a2</td>
</tr>
<tr>
<td>t11.</td>
<td>a2</td>
</tr>
<tr>
<td>t12.</td>
<td>a2</td>
</tr>
<tr>
<td>t13.</td>
<td>a2</td>
</tr>
</tbody>
</table>

### Step-2: Pruning the Test vectors

In the step-2 verification is made to check the extent to which the WEB pages related code has been covered using following algorithm.

Test Suite (T) contains test vectors \{t0, t1, \ldots, tn\}

**Begin**

Step-1: Consider a Test case
Step-2: Compute the number of webpages which refers to the values contained in test vector
Step-3: Compute the code coverage by determines the percentage of the pages covered over the total web pages contained in the WEB site. If the coverage is less than 80% then the vector is dropped

**End**

The test vector thus pruned based on the code coverage are shown in the Table 4
Table 4 Code Coverage by the Test Vectors

<table>
<thead>
<tr>
<th>Test vector Number</th>
<th>Test Vector elements</th>
<th>Percentage coverage of the HTML Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1.</td>
<td>a0 b0 c1 d1</td>
<td>93</td>
</tr>
<tr>
<td>t2.</td>
<td>a0 b1 c2 d0</td>
<td>89</td>
</tr>
<tr>
<td>t3.</td>
<td>a0 b1 c3 d1</td>
<td>80</td>
</tr>
<tr>
<td>t4.</td>
<td>a0 b1 c0 d1</td>
<td>91</td>
</tr>
<tr>
<td>t5.</td>
<td>a0 b0 c2 d1</td>
<td>92</td>
</tr>
<tr>
<td>t6.</td>
<td>a1 b1 c2 d1</td>
<td>93</td>
</tr>
<tr>
<td>t7.</td>
<td>a1 b0 c1 d0</td>
<td>87</td>
</tr>
<tr>
<td>t8.</td>
<td>a1 b0 c0 d0</td>
<td>79</td>
</tr>
<tr>
<td>t9.</td>
<td>a1 b0 c3 d0</td>
<td>98</td>
</tr>
<tr>
<td>t10.</td>
<td>a2 b0 c3 d0</td>
<td>97</td>
</tr>
<tr>
<td>t11.</td>
<td>a2 b0 c0 d0</td>
<td>94</td>
</tr>
<tr>
<td>t12.</td>
<td>a2 b1 c1 d1</td>
<td>88</td>
</tr>
<tr>
<td>t13.</td>
<td>a2 b1 c2 d1</td>
<td>87</td>
</tr>
</tbody>
</table>

The test vector t8 thus will not be considered for testing as it covers only 79% of the code contained in the HTML Pages.

IV. COMPARATIVE ANALYSIS

The ANN-PTCG algorithm is compared with other deterministic algorithms which are shown in Table 1. Using published results as guidelines, for a given input set an initial test set size was supplied to the ANNs algorithm. We have implemented a tool called ANN-PTCG (Artificial Neural Network–Pairwise Test case Strategy Generator) based on ANNs algorithm. Experiments have been conducted by considering different systems which can be defined using different parametric values. The parametric values considered for different systems have been shown in the table 3.
Table 3 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 variables</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>12</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>S3</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>4.</td>
<td>S4</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>5.</td>
<td>S5</td>
<td>100</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>6.</td>
<td>S6</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 4.

Table 4 Pair wise Test Set Sizes for ANN-PTCG and Other algorithms.

<table>
<thead>
<tr>
<th>System</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETG</td>
<td>11</td>
<td>17</td>
<td>35</td>
<td>25</td>
<td>12</td>
<td>193</td>
</tr>
<tr>
<td>Pair-test (IPO)</td>
<td>9</td>
<td>17</td>
<td>34</td>
<td>26</td>
<td>15</td>
<td>212</td>
</tr>
<tr>
<td>SBC</td>
<td>9</td>
<td>15</td>
<td>35</td>
<td>27</td>
<td>10</td>
<td>198</td>
</tr>
<tr>
<td>ANN-PTCG</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>12</td>
<td>4</td>
<td>100</td>
</tr>
</tbody>
</table>

It can be seen from the above table that the number of test cases generated by ANN-PTCG are minimal considering any of the system configurations.

V. CONCLUSION AND FUTURE RESEARCH WORK

Neural network is a wonderful modeling method which can be used to model the input variables and comprehensive connectivity between them. All the test paths that should be considered can be shown through neural networks. The filtering of the test paths contained in the neural networks can be filtered by using the possible value pairs. The test paths thus obtained show that minimal test cases are generated that more than accurately tests the given system.
REFERENCES


