

An Improved Test Generation Algorithm for Pair-Wise Testing

Soumen Maity¹, Amiya Nayak¹, Marzia Zaman², Nita Bansal², and Alka Srivastava²

Abstract

Pair-wise or two-way interaction testing requires that for a given numbers of input parameters to the system, each possible combination of values for any pair of parameters be covered by at least one test. Empirical results show that pair-wise testing is practical and efficient for various types of software systems [1]. The problem of generating a minimum test set for pair-wise testing is NP-complete [3]. In this paper, we propose a test set generation strategy for pair-wise testing when all parameters can take same values. Our strategy is better than those in [3] and “AETG” [1], in terms of the number of tests.

1 INTRODUCTION

Software testing is expensive and time consuming. Given the different input parameters with multiple possible values for each parameter, performing exhaustive testing which tests all possible combinations is practically impossible. Generating an optimal set of tests which will effectively test the software system is therefore desired. Pair-wise testing is known for its effectiveness in different types of software testing [1]. To illustrate the concept of pair-wise testing, consider a system with three two-valued parameters, i.e., parameter A has values A_1 and A_2 , parameter B has values B_1 and B_2 , and parameter C has values C_1 and C_2 . For parameters A , B and C , $\{(A_1, B_1, C_2), (A_2, B_1, C_1), (A_1, B_2, C_1), (A_2, B_2, C_2)\}$ is the only pair-wise test set of size 4.

2 MAIN RESULTS

In this section, we first consider the case where each parameter takes only two values. Suppose one has strings consisting of zeros and ones which all have the same length $2k - 1$. Let us define the weight of a string to

be the number of ones in it. Let S_{2k-1} be the collection of all binary strings of length $2k - 1$ and weight k . Note that

$$|S_{2k-1}| = \binom{2k-1}{k}.$$

For example, here is the 10 strings of S_5 :

```
0 0 0 0 1 1 1 1 1 1
0 1 1 1 0 0 0 1 1 1
1 0 1 1 0 1 1 0 0 1
1 1 0 1 1 0 1 0 1 0
1 1 1 0 1 1 0 1 0 0
```

Pick any two strings (i.e. columns), say the first and the last. Each of these three combinations (0 1), (1 0) and (1 1) does appear at least once. Thus, if one appends a 0 at the bottom of each string of S_{2k-1} , then it is possible to conclude that each of the four possible combinations (0 0), (0 1), (1 0) and (1 1) appears at least once in each pair of strings.

Algorithm:

Input: Number of parameters n .

Output: A test set.

1. Compute smallest k such that $n \leq \binom{2k-1}{k}$.
2. Choose any n strings from S_{2k-1} .
3. Append one zero at the end of each chosen string to get a test set of size $2k$.

End Algorithm

Example 1 Let $n = 9$. Then $k = 3$ and a test set of size 6 is $\{(0 0 0 0 1 1 1 1 1), (0 1 1 1 0 0 0 1 1), (1 0 1 1 0 1 1 0 0), (1 1 0 1 1 0 1 0 1), (1 1 1 0 1 1 0 1 0), (0 0 0 0 0 0 0 0 0)\}$. Note that this set is obtained by choosing the first nine strings from S_5 .

Now we consider the case where each parameter takes more than two values. Our technique is based on *Mutually Orthogonal Latin Squares* (MOLS). The use of MOLS have been considered in the past by Mandl [4] for pair-wise testing in compiler designs.

¹School of Information Technology and Engineering (SITE), University of Ottawa, 800 King Edward Avenue, Ottawa, Ontario, CANADA, K1N 6N5, Email: soumenmaity@yahoo.co.in, nayak@uottawa.ca

²Cistel Technology Inc., 200-210 Colonnade Road, Ottawa, Ontario, CANADA, K2E 7L5 Email: nbansal@cistel.com, Marzia@cistel.com, Alka@cistel.com

A latin square of size n is an array of n copies of each of n different objects (typically the latin letters A, B, C, ...) so that all the objects in any row, and all the objects in any column, are different. Two size n latin squares, one with objects A, B, C, ..., and one with objects a, b, c, ..., are orthogonal if superimposing them leads to a square array containing all n^2 possible pairs (A,a), (A,b), ..., (B,a), (B,b), It is known that if, n is prime, then one can construct $n - 1$ MOLS. For example, two MOLS of size 3 are shown below.

0 1 2	0 1 2
1 2 0	2 0 1
2 0 1	1 2 0

So what happens with the other sizes? Let $L(n)$ be the number of MOLS that exist of size n . It is known that, $L(n)$ is at most $n - 1$, and this upper bound can be attained for primes and powers of primes.

If there are k ($k - 1$)-valued parameters, then $k - 2$ MOLS of size $k - 1$ are required. With a set of $k - 2$ MOLS of size $k - 1$, two parameters take their values from row and column indices, and the remaining parameters take their values from the ordered ($k - 2$)-tuple formed from the superimposed squares. For example, 4 3-valued parameters required two MOLS of size 3 as shown above. Superimposing these two MOLS we obtain:

(0, 0)	(1, 1)	(2, 2)
(1, 2)	(2, 0)	(0, 1)
(2, 1)	(0, 2)	(1, 0)

The highlighted entry represents the test configuration (2, 1, 0, 2): row label 2, column label 1, entry (0, 2). Note that we label the rows and columns with 0, 1, 2 instead of 1, 2, 3. Continuing in this way for other entries we get the following test set for 4 3-valued parameters: $\{(0000), (0111), (0222), (1012), (1120), (1201), (2021), (2102), (2210)\}$. Can one construct a test set for, say, 16 3-valued parameters, using above test set for 4 3-valued parameters? The answer is yes. We now illustrate it. For convenience of the discussion we construct a matrix A by considering the test configurations as the rows of the matrix A . Let A_i be the i -th column of the matrix A . For example, considering above test set we get $A_1 = (000111222)^T$, $A_2 = (012012012)^T$, $A_3 = (012120201)^T$ and $A_4 = (012201120)^T$. The 18 rows of the following 18×16 matrix

$$\begin{pmatrix} A_1 & A_1 & A_1 & A_1 & A_2 & \cdots & A_4 \\ A_1 & A_2 & A_3 & A_4 & A_1 & \cdots & A_4 \end{pmatrix}$$

provides the complete set of test configurations for 16 3-valued parameters. It is easy to check that any pair of these columns contains all the combinations at least once. This idea can be extended to get test set for any number of 3-valued parameters. For example, to get test set for 64 3-valued parameters, we consider the 27 rows of the following 27×64 matrix:

$$\begin{pmatrix} A_1 & A_1 & A_1 & A_1 & A_1 & A_1 & \cdots & A_4 \\ A_1 & A_1 & A_1 & A_1 & A_2 & A_2 & \cdots & A_4 \\ A_1 & A_2 & A_3 & A_4 & A_1 & A_2 & \cdots & A_4 \end{pmatrix}$$

In general, to generate test set for n m -valued parameters, our starting point would be $L(m)$ MOLS of size m . The full table of MOLS appears in [2].

3 CONCLUSIONS

In this paper, we have proposed an efficient test generating strategy for pair-wise testing. The following table shows the size of generated test set obtained by our technique as well as two other methods called "AETG" [1] and "Pair-Test" [3]. Note that our results are always better than "Pair-Test" method and better than "AETG" method in most cases. S_1 : 4 3-valued parameters, S_2 : 13

System	S_1	S_2	S_3	S_4	S_5
Pair-Test	9	17	15	40	66
Our Technique	9	18	10	32	48
AETG	11	17	12	NA	NA

3-valued parameters, S_3 : 100 2-valued parameters, S_4 and S_5 : 20 and 100 4-valued parameters.

References

- [1] Cohen, D. M., Dalal, S. R., Fredman, M. L., and Patton, G. C., The AETG system: An Approach to Testing Based on Combinatorial Design, *IEEE Transaction on Software Engineering*, 1997, 23 (7), 437-443.
- [2] Colbourn, C. and Dinitz, J. (Ed.) The CRC Handbook of Combinatorial Design, CRC Press, 1996.
- [3] Lei, Y and Tai, K. C., In-Parameter-Order: A Test Generating Strategy for Pairwise Testing, *IEEE Trans. on Software Engineering*, 2002, 28 (1), 1-3.
- [4] Mandl, R., Orthogonal Latin Squares: An Application of Experimental Design to Compiler Testing, *Comm. ACM*, 1985, 28 (10), 1054-1058.