ISSN:0975-9646

Orthogonal Testing Using Genetic Algorithms

Harsh Bhasin[#], Harish Kumar[*], Vikas Singh[*]

[#]*Department of Computer Science,*
Delhi Technological University
Delhi, India

[*]*YMCAUST, CE Department*
Sector – 6, Faridabad, India

***Abstract*— Orthogonal Array Testing is one of the most important techniques that produce test cases which are much lesser in number than black box testing, but are more relevant. In spite of its importance, the technique has not been explored as much as other techniques. The present work, therefore, explores the literature to find the gaps in the literature and hence propose a new technique based on Genetic Algorithms. The work, also, verifies the technique on a set of web pages and compares it with the existing techniques. The work proposes a novel algorithm of orthogonal testing. The process has been applied on 20 web pages and is being applied on another set of 25 pages. The initial results are encouraging and the technique paves way for the application of orthogonal arrays with a new perspective.**

***Keywords*— Orthogonal Testing, Genetic algorithms, Orthogonal Array Testing Strategy, Black Box Testing, Web Testing.**

I. INTRODUCTION

Software testing is an integral part of software development life cycle. As and when software evolves, testing is needed. It is essential at every level, every phase and at every step. Testing also helps to cut cost during the maintenance phase. There are very many methods of testing. Orthogonal testing is one of the unexplored methodologies of testing. The orthogonal testing paradigm calls for the crafting of feasible permutations of the values of attributes. Different testing techniques have been extensively studied by various researchers. But, orthogonal testing has seldom been explored. The work proposed intends to fill the gap. The proposed work amalgamates artificial intelligence with orthogonal testing to produce a novel technique which will be pivotal in the area of web testing. The manual labor and the challenges faced in the web testing can be greatly overcome by this path breaking work. An extensive literature review has been carried out and the proposed technique is being applied to selected set of web pages. The work of verification and validation of technique is being done day and out and the results are encouraging.

The remaining sections are arranged as follows: Section 2 provides overview of the Artificial Intelligence Technique namely Genetic Algorithms (GAs) used. Next Section gives an overview of testing followed by introduction to Orthogonal Testing. The proposed work is described in Section 5 and Section 6 will describe our experimental analysis. Finally, in Section 7, conclusions and future work are discussed.

II. GENETIC ALGORITHMS

Genetic Algorithms are adaptive heuristic search algorithms which are based on Charles Darwin theory of the survival of the fittest [1]. They are known to perform efficiently if sample space is huge [2]. The depiction of a natural population is done using, what is called chromosomes, which are nothing but a set of numbers, generally binary. Each number represents a cell and can be perceived as an affirmative or negative answer. For example, a chromosome 10110 if applied to knapsack problem can be assumed as selecting the first, third and fourth item from amongst a set of five items, as there is 1 at the first, third and fourth position. The initial population can be generated using any Pseudo Random Number Generator. New population is generated by applying genetic operators such as crossover and mutation on initial population [3]. Each chromosome is then assigned a fitness value [4]. Population is selected to form new population as per the fitness value. More is the fitness value; more is the chance to be selected. Hence, the new population obtained will be better than the old one [5]. The above process is repeated until some condition is satisfied.

A. Crossover Operator

It is genetic operator that combines two chromosomes to produce a new chromosome. The child chromosome takes one section of the chromosome from each parent. The point at which chromosome is broken depends on the randomly selected crossover point. The number of crossover is determined by the crossover rate which is generally 2-5% [6]. GAs have following type of crossover:

1) One Point Crossover: Single Point Crossover is performed by selecting a random gene along the length of the chromosomes and swapping all the genes after that point [7].

2) Two Point Crossover: Here two crossover points are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent [8].

3) Uniform Crossover: In this method bits are randomly copied from the first or from the second parent [8].

B. Mutation Operator

It randomly changes its genetic makeup. This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at

each bit position in a string with some probability, usually very small (e.g., 0.001) [9].

C. Selection Operator

It is quantitative criterion based on fitness value to choose which chromosomes from population will go to reproduce. Intuitively the chromosome with more fitness value will be considered better and in order to implement proportionate random choice, Roulette wheel selection is one of the methods used for the selection procedure [10].

III. Testing

Software Testing is the process of executing a program or system with the intent of finding errors. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Software testing is used to increase the quality and confidence in a software product [11]. The two issues in testing that need to be considered are test effectiveness and test efficiency. Increasing test effectiveness could be accomplished by using test cases that have a high likelihood of discovering a fault and increasing test efficiency could be achieved by reducing the number of test cases that need to be executed without significantly compromising effectiveness [11]. To ensure complete testing of software, all the possible combinations of input are to be tested, known as exhaustive testing. But, in practical it is impossible to run all the test cases due to limited Time and cost. Thus, various techniques have been developed to reduce the number of test cases, of which use of Orthogonal Arrays is the one.

IV. Orthogonal Testing

Orthogonal testing is a black box testing technique, used for testing pair wise interactions. It calls for generating test cases apt for combination of different variables. Orthogonal array can be applied in user interface testing, system testing, regression testing, integration testing of software components, testing combinations of configurable options. If the number of input variables is such that testing all the possible combinations is not feasible owing to large number of test cases then orthogonal array testing can be used thereby to reduce the number of possible test cases by filtering out some impossible combination based on certain constraints. The basics of Orthogonal Testing are as follows:

- Runs: It is the number of rows in the array. This directly translates to the number of test cases that will be generated by the OATS technique.
- Factors: It refers to the number of columns in an array. This directly translates to the maximum number of variables that can be handled by this array.
- Levels: It is the maximum number of values that can be taken on by any single factor. An orthogonal array will contain values from 0 to levels-1.
- Strength: It is the number of columns it takes to see each of the levels^{Stength} possibilities equally often.
- Orthogonal arrays are most often named as follows # $L_{Runs}(levels^{Factors})$.

V. Proposed Work

The section proposes a novel way of generating test cases via orthogonal testing. The proposed technique is sound and efficient. Yet, at the same time, it is simple enough to be implemented. The proposed technique is as follows:

A. Web Page Selection

Select a web page, which is to be tested.

B. Attribute Selection

Attributes which are to be tested, are selected from the webpage.

C. Range calculation

Possible range of values is inferred from the web page for each attribute.

D. Attribute Selection

Attributes with minimum and maximum number of values are selected.

E. Column

The two selected attributes act as the first and last column of 2D orthogonal array. If X and Y be the minimum and maximum number of possible values, then 2D array obtained will have X * Y rows.

F. Intermediate Columns

The remaining attributes acts as the intermediate columns, for the array, in the ascending order.

G. Intermediate Values

Values for intermediate columns are selected using a GAs technique. Following steps are used:

1) Initial Population generation: Generate initial population using PRNG (Pseudo Random Number Generator).

2) Crossover: The crossover rate of 2.2 is taken in the implementation.

3) Mutation: For every set of child generated mutation is performed as against taking an overall mutation rate.

Fig. 1 presents the algorithm in a concise way. The technique has been implemented in C#. .NET framework 4.0.

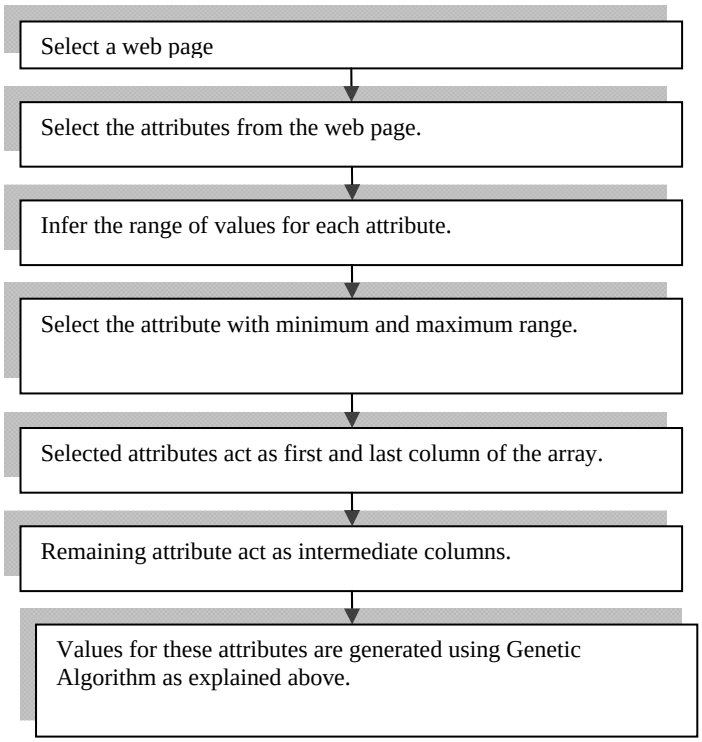

Fig. 1 Flowgraph of the proposed algorithm.

VI. Experimental Analysis

The proposed technique has been applied on a generated web page. The fields of the page are given in Table 1.

TABLE I
ATTRIBUTES

Sr. No.	Name of Attribute	Range of Values
1	First Name	26^{20}
2	Last Name	26^{20}
3	Gender	2
4	Date	31
5	Month	12
6	Year	54
7	Mother Tongue	62
8	Caste	420
9	Religion	12
10	Phone Number	10^{10}
11	Country	71
12	Country Code	$10^2 = 100$

In the above table, the fields like 'First Name, last Name and Phone Number' will have all valid combination of legal characters. The field 'First Name' has a maximum of 20 characters, therefore, 26^{20} cases would be generated in total if exhaustive test cases are to be considered. But there is no point of generating such large number of cases because this will not only add to the bulkiness of the test case suite but will also fail to add any more confidence in the robustness of the test case suite. Such cases may be neglected in the second step and a few valid values of the above fields may be added in the test case suite being formed. Table 2 depicts the remaining values.

TABLE III
SELECTION OF FILTERED ATTRIBUTES

Sr. No.	Name of Attribute	Range of Values
1	Gender	2
2	Date	31
3	Month	12
4	Year	54
5	Mother Tongue	62
6	Caste	420
7	Religion	12
8	Country	71
9	Country Code	$10^2 = 100$

The attribute with maximum range is Caste, whereas that with minimum range is Gender. So, Total number of test cases generated will be 420 * 2 = 840.

In black box testing, total number of test cases which will be generated result is:

$26^{20} \times 26^{20} \times 2 \times 31 \times 12 \times 54 \times 62 \times 420 \times 12 \times 10^{10} \times 71 \times 100$. On the other hand orthogonal testing reduces the number of test cases to a large extend.

Table 3 shows the values of first and last attributes. The shaded region depicts the intermediate values which are generated using the GAs. Attribute 1 can have 420 values. Therefore, v1 represents the value 1; v2 represents the value 2, and so on.

VII. Conclusions

The work proposes a new technique of orthogonal testing. The technique amalgamates the orthogonal testing with genetic algorithms. The applicability of genetic algorithms is apt as the values in the test cases are to be searched from a vast set of values. The genetic algorithms are a time tested way of selection which not only gives optimal results but also ensure robustness. The technique has been applied to 20 web pages and is being applied to another set of 25 web pages with varying attributes and frameworks. The results obtained so far are encouraging. The technique is bound to change the way we look at orthogonal testing.

TABLE IIIII
SELECTION OF FILTERED ATTRIBUTES

Attributes / Values	1	2	3	4	5	6	7	8	9
1	v1								v1
2	v2								v1
3	v3								v1
4	v4								v1
--	--								--
--	--								--
--	--								--
419	v419								v1
420	v420								v1
421	v1								v2
422	v2								v2
423	v3								v2
424	v4								v2
--	--								--
--	--								--
--	--								--
839	v419								v2
840	v420								v2

References

[1] Harsh Bhasin and Manoj, "Regression Testing Using Coupling and Genetic Algorithms", International Journal of Computer Science and Information Technology (IJCSIT), 3(1), pp. 3255 – 3259.

[2] Harsh Bhasin and Rohan Mahajan, "Genetic Algorithms Based Solution to Maximum Clique Problem", International Journal on Computer Science and Engineering (IJCSE), 2012, 4(8), pp. 1443-1448.

[3] Harsh Bhasin and Neha Singla, "Modified Genetic Algorithms Based Solution to Subset Sum Problem", International Journal of Advanced Research in Artificial Intelligence (IJARAI), Volume 1, No 1, April 2012.

[4] Harsh Bhasin and Nakul Arora, "Cryptography using Genetic Algorithms", International Conference on Reliability Infocom Technology and Optimization (ICRITO), 2010, Conference Proceedings pages 226- 230.

[5] Harsh Bhasin and Neha Singla, "Harnessing Cellular Automana and Genetic Algorithms to solve Travelling Salesman Problem", International Conference on Information, Computing and Telecommunications (ICICT), conference proceedins, pp 72 – 77.

[6] Harsh Bhasin and Geetanjali Ahuja, "Harnessing Genetic Algorithm for Vertex Cover Problem", International Journal on Computer Science and Engineering (IJCSE), Vol. 1, Issue 2, pp. 218 - 223.

[7] Harsh Bhasin and Neha Singla, "Genetic based Algorithm for N – Puzzle problem", International Journal of Computer Applications (IJCA), Volume 51, issue 22, pp. 44 – 50.

[8] Harsh Bhasin and Surbhi Bhatia, "Application of Genetic Algorithms in Machine learning", 2011, International Journal of Computer Science and Information Technology (IJCSIT), Volume 2, Issue 5, pp. 2412-2415.

[9] Harsh Bhasin and Surbhi Bhatia, "Use of Genetic Algorithms for Finding Roots of Algebraic Equations", International Journal of Computer Science and Information Technology (IJCSIT), 2011, Volume 2, Issue 4, pp. 693-696.

[10] Harsh Bhasin and Supreet Singh, "GA-Correlation Based Rule Generation for Expert Systems", International Journal of Computer Science and Information Technology (IJCSIT), Volume 3, Issue 2, pp. 3733-3736.

[11] Karnig Agop Derderian, "Automated test sequence generation for Finite State Machine using Genetic Algorithms", PhD Thesis, Brunel University (2006).