Volume 7, No. 3, March 2016
Journal of Global Research in Computer Science
RESEARCH PAPER
ISSN-2229-371X
http://rroij.com/global-research-in-computer-science.php

Test Data Generation Strategies Using Constraints Support

Saad Masood[1]*, Shahid Masood Butt[2]*, Azura Onnc[3]*
[1]Department of Computer and Software Engineering, Bahria University, Islamabad, Pakistan
[2]Faculty of Management, Hamdard University Islamabad Campus, Pakistan
[3]Department of Management and Human Resource, University Tenaga Nasional, Malaysia
saadmasoodbutt668@yahoo.com, shahid.masood@hamdard.edu.pk, azura@uniten.edu.my

Abstract: Our dependency on software is everywhere the software system is getting more complex due to the increase of software requirements. Therefore, testing task is getting difficult and cost more, which causes by the vast number of test cases and the high number of test feathers. For that reason, software test suite reduction is needed. Computational strategies have shown an interested result of software test suite optimization. In this paper, we intend to discuss the computational testing and investigate the support of constraints. Furthermore, analyse and compare the existing computational strategies to identify the advantages and limitations. As further work is to design and implement a computational testing tool capable to get rid of some of the limitations in the existing computational strategies

Keywords- particle swarm optimization, t-way testing, software testing

INTRODUCTION

A human action produces some errors or bugs in the software. This will lead to a lot of defects which cause failure occurs when executed. This problem will cause serious damage of system function, and especially for a critical system will involve high costs and loss of time. Therefore, software testing is important in any software development life cycle (SDLC) to make sure the quality of software and to prevent the failure of software [1]. Software execution faults are one of the greatest contributors to info safety system faintness, which make software testing a key measure of the security system. Although, the existing testing techniques like boundary value (BV), Equivalent Partition (EP) useful to generate test cases. Software testers often for defects that they anticipate, while unforeseen defects, especially those arising from the combination among components, are overlooked in this area one of the major essential inquiries is the generation of test cases [2]. To process the combinatorial testing technique (CTT) has been put forward to generate minimum effective test cases. Because of the minimum test cases will be generated, the combinatorial testing techniques can help increase the effectiveness of software testing and reduce the cost for many applications [3]. Software Developers which a large data often noticed an interesting, although it is not amazing phenomenon: When the application is used for jumps higher, Ingredients that have operated for several months lacking trouble, rapidly appeared previous were not discovered faults. For instance, the application may have been installed on a various OS-hardware database management system (DBMS) networking platform, or recently extra clienteles may have an account registered with a weird combination of values that have not happened before [4]. Many of these scarcely combinations lead to bugs or error which may have resorted to past experiences and the use of large-scale. It is well recognized that such failures such as interaction failures; due they are only risky when three or additional input values interact that reasons the program to access as an outcome of not satisfactory. Combinatorial testing can assistance find out mistakes similar this early time in the testing lifecycle. The vision original t-way (t refers to the degree of the interaction) the combinatorial testing is explain that not all parameters common to each failure and most of the failure are leads by interaction between comparatively small number of parameters or a single parameter value.

COMBINATION TESTING NOTATION

Combinatorial testing notations are not new on the testing [5]. Discussion on how to use statistically designed experiments orthogonal arrays. Interaction test suites it can be set to corresponding combinatorial objects, Codenamed covering array (CA). In order of Assistance in the debate, this section separates the useful symbols notations for expressing covering array (CA). Often, the covering array (CA) has four parameters; p, t, N, and v (i.e, CA (N, t, vp)). Here, the symbols P, v, and t are used to mention to the parameters (P), values (V), and interaction strength for the covering array (CA), respectively. For example, CA (9,2,34) Represents a test suite consisting of 4×9 arrays (i.e, the column represents the parameter (P), and the rows represent the size of test cases (N).

In this case, the test suite is cover 2-way interaction for a system with four 3-value parameters. Like to covering array (CA), mixed covering array (MCA) has three parameters; t, N, and Configuration (C) (i.e, MCA (N, t, C)). In addition to t and N that carries the same meaning as covering array (CA), mixed covering array (MCA), Dependents a new symbol, C. Consistent with earlier Submitted notations, where C appears the p and v of every configuration in the following: $v_1^{p1}, v_2^{p2}, ..., v_n^{pn}$ representing that there are P_1 parameters with v_1 values, P_2 parameters with v_2 values, and so on. For example, MCA (1265, 4, 10^2, 4^1, 3^2, 2)^7 Indicates the test size of 1265 that covers four-way interaction. Here, the configuration takes 12 parameters: one 4-value parameter, two 10-value parameters, seven 2-value parameters and two 3-value parameters [2].

PROBLEM DEFINATION MODEL

Bugs in software system can cost the loss of life and huge sum of money. Software testing plays an important role in exploring detects through possible test data to ensure its quality. Most of the software systems in this time are created using components. Often, the system errors or bugs are caused by unexpected combination between these [6]. For example, consider the

testing of Microsoft words display tab in the options dialog (Figure 1).

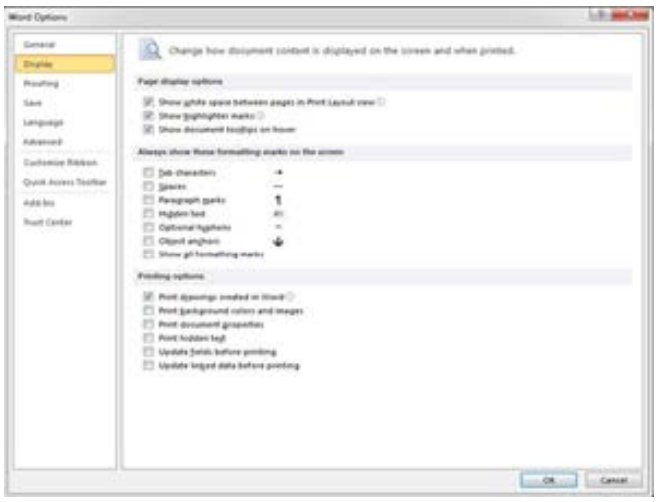

Figure 1: Microsoft word, options, display.

17 possible options that can take it two possible values 217= 131.072 to be tested. There are which is practically inefficient!!! If a test case needs 5 minutes to be tested, it will take 15 months to test only the display tab completely.

Additionally, in real world scenarios, some of the possible value combinations may not be valid. Therefore, constraints have been introduced. They allow specifying invalid combinations which will be excluded or skipped during test case generation. Constraints can be found in many specifications of a software system. They are typically given in natural language and exist for several reasons, such as limitations of the components used in the target system, available resources and even marketing decisions. While constraints reduce the number of valid test cases, their presence makes combination testing more challenging [7]. The impact of constraints varies with the (test) problem, but their presence causes problems for many existing CTT tools of the numerous existing. These tools are supportive combinatorial test design only a few offers full constraints support. These rare apparatuses with full details published are scarcer.

RELATED WORK

Computational t-way strategies are approaches which heavily depend on search processes to find the best test cases from a generated interaction test element, which is generated according to pre-identified input parameters and values. Researchers have developed several t-way strategies to optimize the test suite. Such as AETG [8], GA [9], IPO [10,11], Jenny (Jenkins, 2003), TVG [12], PSTG [13], SA [6,14].

HSS [15], As far as the review of existing computational t- way strategies is our intention. This literature review is carried out an overview of 15 related strategies. This review has classified into two main classifications according to [10] taken into account the strategies related to constraints-based testing (Figure 1).

Natural-Based T-Way Strategies

Natural based T-way strategies are based on natural Algorithm, which are inspired from the nature of the creatures behaviors (i.e, forging, meeting… etc) [16]. Those strategies have shown interesting outcomes related to the optimum test suite generation. Moreover, it has overcome many limitations related to combinatorial optimization problem [17]. Usually, Natural based strategies uses natural algorithm as a backbone search engine. Many natural based strategies has been proposed using this approach as a basis of one-test-at-a-time approach, such as, HS [15], PSTG [13], SA [6,14], GA [9].GA strategy mimics the natural selection process Which has been addressed in a lot of different research [9,18,19] the algorithm for automatic test data generation. The generated data utilizes a genetic algorithm to scan for test cases that satisfy desired testing prerequisites. A solution (chromosome) is a group of test data (i.e. a list of input values). The algorithm assesses test case by implementing the program with the test case as input, and recording the predicates in the system that implementing with that test case. This list of predicates is compared with the group of predicates found on the control-dependence predicate paths for the node representing the current test demands. GA can be stated as a non-deterministic. Regarding T, it can only support T=3 and 3 it also doesn't allow using real values as input. It can support uniform input values.

Simulated Annealing (SA) [14,19] first performs SA to support pairwise interactions. Later developed and execution SA to support up to 3-way interactions. In the execution, a large random field of search is generated. Relying on the binary search process and a probability based transformation equation, the strategy repeats to pick the best test case every iteration with a specific end goal to construct the test suite. Thus, the results indicated that SA is more effective than other approaches for discovering optimal sizes in cases of little strengths. It can support uniform input values. Similar to other natural-based counterparts, SA addresses small values of interaction strength (i.e., $t \leq 3$). Particle Swarm Test Generator (PSTG) [13,20]. It is the Natural-based t-way strategy for generating t-way test suite. PSTG is based on the particle swarm optimization (PSO) algorithm, which simulates the swarm behaviour of birds. Internally, PSTG iteratively performs local and global searches, inquiries to discover the candidate test cases solutions to be added to the final suite until all the interaction tuples are covered. PSTG Not like other Natural based strategies that address little values of t (i.e., $2 \leq t \leq 3$), where it can support up to $t = 6$. As such, the interaction support provide by PSTG is still limited. PSTG does not cater for the constraints support.

HSS algorithm has developed by (Alsewari) to generate the test suite by adopting harmony search (HS) algorithm as its core implementation. HS is an algorithm mimicking the behaviour of musicians in the process of improvisation (Geem). HSS generates random test cases in Harmony Memory HM based on harmony memory size. Then improvise the test cases in HM for several times based on local and global improvement. In each iteration. HSS will add test case to final test suite until covered all pair interactions. Unlike other competing natural based t-way strategies, HSS addresses the support for high interaction (i.e, $t > 6$), and implements seamless support for constraints. Through published results, HSS gives competitive results against most existing t-way strategies.

CONCLUSION

This paper gives us a background about Combinatorial Testing CT, where reviews the existing Combinatorial test generation strategies shows the advantages and the limitations for each. This works serve as a background study for our further work, which we intend to design and implement a new CT strategy, which may help get rid of some of the limitations in this research area.

REFERENCES

[1] Karaboga D, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[2] Alsewari ARA and Zamli KZ, "Design and implementation of a harmony-search-based variable-strength t- way testing strategy with constraints support," Information and Software Technology, vol. 54, pp. 553-568, 2012.

[3] Kuhn DR, "Introduction to Combinatorial Testing," 2011.

[4] Garvin BJ, Cohen MB, and Dwyer MB, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," Empirical Software Engineering, vol. 16, pp. 61-102, 2011.

[5] Hedayat AS, Sloane NJA, and Stufken J, Orthogonal arrays: theory and applications: Springer Science & Business Media, 1999.

[6] Cohen MB, Dwyer MB, and Shi J, "Interaction testing of highly-configurable systems in the presence of constraints," in Proceedings of the 2007 international symposium on Software testing and analysis, pp. 129-139, 2007.

[7] Myers GJ, Sandler C, and Badgett T, The art of software testing: John Wiley & Sons, 2011.

[8] Cohen DM, Dalal SR, Fredman ML and Patton GC, "The AETG system: An approach to testing based on combinatorial design," Software Engineering, IEEE Transactions on, vol. 23, pp. 437-444, 1997.

[9] Shiba T, Tsuchiya T, and Kikuno T, "Using artificial life techniques to generate test cases for combinatorial testing," in Computer Software and Applications Conference, COMPSAC, Proceedings of the 28th Annual International, pp. 72-77, 2004.

[10] Lei Y, Kacker R, Kuhn DR, Okun V, and Lawrence J, "IPOG/IPOG - D: efficient test generation for multi - way combinatorial testing," Software Testing, Verification and Reliability, vol. 18, pp. 125-148, 2008.

[11] Lei Y and Tai KC, "In-parameter-order: A test generation strategy for pairwise testing," in High-Assurance Systems Engineering Symposium, Proceedings Third IEEE International, pp. 254-261, 1998.

[12] Schroeder PJ, Kim E, Arshem J, and Bolaki P, "Combining behavior and data modeling in automated test case generation," in Quality Software, Proceedings Third International Conference on, pp. 247-254, 2003.

[13] Ahmed BS and Zamli KZ, "PSTG: a t-way strategy adopting particle swarm optimization,"in Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on, pp. 1-5, 2010.

[14] Stardom J, "Metaheuristics and the search for covering and packing arrays", Trent University, 2001.

[15] Abdul Rahman KZZ and Alsewari A, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support", vol. 54, Iss. 6, 2012.

[16] Gazi V and Passino KM, "Swarm Stability and Optimization, Part IV: Swarm Based Optimization Methods - Bacteria Foraging Optimization," in Swarm Stability and Optimization, ed: Springer, pp. 233-249, 2011.

[17] Floudas C, Pardalos P, Adjiman C, Esposito W, Gumus Z, Harding S, et al. Handbook of test problems in local and global optimization: Kluwer Academic Publishers, Dordrecht, 1999.

[18] Afzal W, Torkar R, and Feldt R, "A systematic review of search-based testing for non-functional system properties," Information and Software Technology, vol. 51, pp. 957-976, 2009.

[19] Cohen MB, Colbourn CJ, and Ling AC, "Constructing strength three covering arrays with augmented annealing," Discrete Mathematics, vol. 308, pp. 2709-2722, 2008.

[20] Ahmed BS and Zamli KZ, "A variable strength interaction test suites generation strategy using particle swarm optimization," Journal of Systems and Software, vol. 84, pp. 2171-2185, 2011.

