Int. J. Advance Soft Compu. Appl, Vol. 5, No. 3, December 2013
ISSN 2074-8523; Copyright © SCRG Publication, 2013

Variable-Strength Interaction for T-Way Test Generation Strategy

Syahrul A. C. Abdullah, Zainal H. C. Soh, and Kamal Z. Zamli

Faculty of Electrical Engineering,
Universiti Teknologi MARA (UiTM), Shah Alam, Malaysia.
e-mail: bekabox181343@salam.uitm.edu.my

Faculty of Electrical Engineering,
Universiti Teknologi MARA (UiTM), Pulau Pinang, Malaysia.
e-mail: zainal872@ppinang.uitm.edu.my

Faculty of Computer Systems & Software Engineering,
Universiti Malaysia Pahang (UMP), Pahang, Malaysia.
e-mail: kamalz@ump.edu.my

Abstract

Often, t-way testing is usually adopted to trigger faults due to interactions. As a result, a myriad of useful t-way test generation strategies have been developed in order to generate t-way interaction test suite that is small in size while maintaining adequate t-way interaction coverage. Even though finding an efficient strategy to construct an optimal test suite is very valuable, another aspect to consider is the cost benefits of running the tests, i.e. at high level of interaction strength; the test suite size can become enormous. To balance the need for stronger interaction with the cost of running the tests, variable-strength interaction has been recommended. Hence, to enable support for construction of test suite with variable-strength interaction, a step by step procedure that extends our t-way test generation strategy, called Test Suite Generator (TSG) is highlighted. Benchmarking results against most existing strategies that support variable-strength interaction demonstrate that TSG is able to give competitive results.

Keywords: *combinatorial testing, software testing, t-way testing, t-way minimization strategy, variable-strength interaction.*

1 Introduction

One of the key challenges in software testing is to identify selection of input values to test as most software systems have a very high range of valid and invalid inputs. Moreover, it is not always feasible to test for all the input values because of constraint in cost, resources and time. To balance between these desirable but conflicting features, software tester wishes that the testing activities will eventually demonstrate that the software systems are fit for purpose and to

detect defects. However, the relevance is to complete some testing before time to deliver the software product to the market as it is not always possible to test everything.

Furthermore, the associate risk with testing must also be well comprehended. A successful testing can be cost effective and quick but in the case of discovering a failure, it will take a substantial time to rectify the code which had errors and to do retesting. As a result, often the test activities are guided by the risk analysis where it covers the testing of essential part of software systems with a list of test that must be executed first and in which sequence.

In the context of t-way testing, often, the size of t-way interaction test suite at high level of interaction strength can become enormous. Even though triggering all the faults is important, much research demonstrates that the percentage of faulty interaction from four-way to seven-way interactions among variables is relatively rare [1-3]. Thus, as with most aspects of engineering, trade-offs must be considered and must occur. Hence, to balance the need for stronger interaction strength in t-way interaction test suite with the cost of running the tests, variable-strength interaction has been recommended. This approach is often favored because of the compromise in term of test suite size where the strategy focuses testing where it has the most potential value, which usually is associated with the risk analysis.

In the previous work, a t-way test generation strategy called Test Suite generator (TSG) has been implemented using a multilevel Greedy Algorithm [4-6]. To enable support for construction of test suite with variable-strength interaction, this paper discusses a step by step procedure taken to extend TSG with this feature. Benchmarking results demonstrate that TSG produces competitive results while addressing support for variable-strength interaction.

This paper is organized as follows. Section 2 outlines the related works on variable strength t-way test generation strategy. Section 3 introduces the step by step procedure taken to convert the existing strategy. Section 4 elaborates the benchmarking of TSG against most existing strategies that support variable-strength interaction. Finally, Section 5 provides the conclusion.

2 Related Work on Variable-Strength Interaction

There are myriad of useful variable strength t-way test generation strategies that have been implemented in order to sample test cases and construct a variable-strength t-way interaction test suite for t-way testing. Initially, it was recommended by Cohen in a strategy called Simulated Annealing (SA) as the strategy adopted Simulated Annealing algorithm to construct the test suite with variable-strength interaction [7]. Eventually, many strategies begin to support construction of test suite with variable-strength interaction like Ant Colony System (ACS), Density, Input Parameter Order Generalized (IPOG), ParaOrder, PICT, Test Vector Generator (TVG), VS-PSTG (Variable-Strength Particle Swarm Test Generator) and WHITCH (IBM Intelligent Test Case Handler).

Like SA, ACS [8] and VS-PSTG [9] used AI Search technique in sampling and selection of the test cases to construct the variable-strength t-way interaction test suite. As the name suggests, ACS used Ant Colony Optimization while VS-PSTG used Particle Swarm Optimization (PSO) in their implementation. These strategies often able to produce very good results in term of test suite size as their implementation are able to escape local optimal.

On the other hand, the rest of the above-mentioned strategies used pure computational approaches. The difference among them is on the design technique used, IPOG [10] and ParaOrder [11] are algorithms that implement one-parameter-at-a-time design technique while Density [11], PICT [12], TVG [13] and WHITCH [14] are based on one-test-at-a-time design technique. Both ParaOrder and Density consider actual interaction relationship in their implementation and they are able to produce acceptable results in certain special inputs.

PICT is a public domain testing tool developed by Microsoft. Often, the PICT generated test suite is not optimal since the core generation algorithm is based on some random selection. Unlike PICT, as far as TVG and WHITCH are concerned, there is limited literature on the implementation. However, based on our experience in running the tool, TVG generation algorithm is based on input-output relationship, while WHITCH generation algorithm is based on exhaustive search, which may not conducive to run for large input parameters.

3 Approach and Method

The t-way test generation strategy is usually divided into two phase, generation of uncovered interaction elements and selection of test case to cover as many number of uncovered interaction elements. Generating candidate test cases to select a decent test case that conforms to certain rules by using the characteristics of the running algorithm and later on becomes part of a suitable subset of t-way interaction test suite is a multi-objective optimization problem.

The objective of this kind of minimization strategy is to generate a t-way interaction test suite that has full interaction coverage. This means covering all possible interaction at least once by the selected test cases while trying to minimize the total number of test cases as small as possible by maximizing the fitness (the total number of interaction elements covered) of each selected test case.

Hence, the size of the test suite is optimal based on the minimum total number of test cases when each interaction elements is covered only once by any of the test case within the test suite. To further address this matter, the following sub-sections present the notation and the implementation flavors.

3.1 Notation

To improve the t-way test suite abstractions, a common structure to describe and formulate these interactions is adopted [9], which is as follows: (1) Input parameters (p) with a fixed constant number of values (v) is represented by Covering Array, i.e. CA (N: t, v^p), where t implies the interaction strength of parameters with the corresponding generated test suite size (N). (2) Input parameters with a different or mixed number of values are represented by Mixed Covering Array, i.e. MCA (N: t, $v_1^{p1}v_2^{p2} \ldots v_n^{pn}$). (3) The variable-strength notation VSCA (N: t, $v_1^{p1}v_2^{p2} \ldots v_n^{pn}$, {C}) represents covering array (either as CA or MCA) of strength t containing C, which is a subset of the p columns each of strength >t. C is a covering array and can be represents either as CA or MCA.

3.2 The TSG strategy

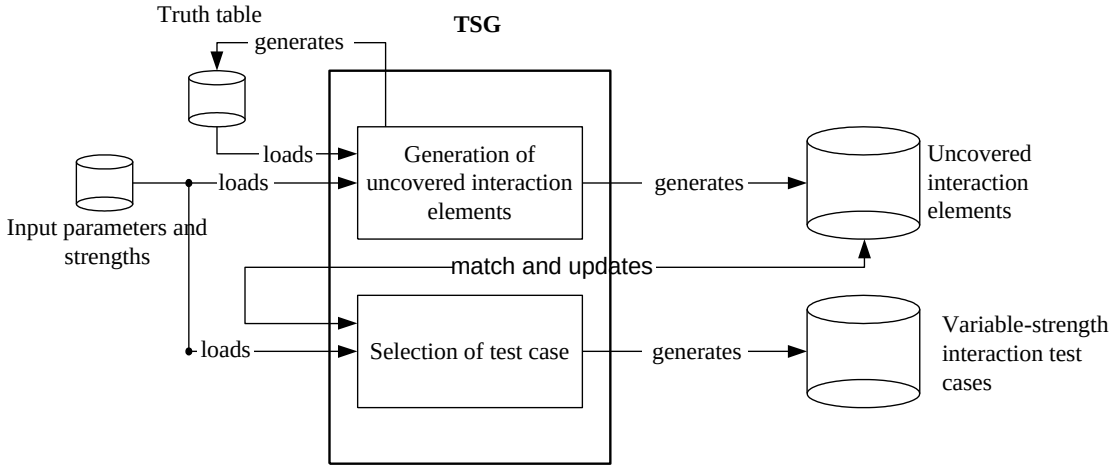

Fig. 1: Overview of the TSG strategy and its implementation

Referring to Fig. 1, addressing the selection of test case in TSG is by using a multilevel Greedy Algorithm [4]. Here, when there is more than one value (candidate test case) that satisfies best that is maximum fitness, a decision on how to choose one of these values is delayed until it satisfies certain rules. In the implementation, all best values are stored inside a container and delayed breaking the rules until the size of container, *s* is bigger than 100. This setting is based on the implementation of another greedy algorithm called Automatic Efficient Test Generator (AETG), which ran a few experiments to select the minimum value of *s* [15].

Therefore, to enable support for variable-strength interaction, the generation of uncovered interaction elements' procedure in TSG (as shown in Fig. 1) is updated so that it can able to support multiple strengths in its constructor. Hence, it can generate variable-strength interaction elements.

To describe the procedure in detail, it is necessary to state that the inner workings of TSG are based on set operations. In the beginning, based on sets of input parameters and strength, *t*, (or strengths for variable-strength interaction), TSG computes using Cartesian product to generate a truth table.

The table contains lists of Boolean true and false, which are later used to generate combination of values to be covered, called uncovered interaction elements. The generation of interaction elements is described below:

(i) TSG selects lists that contain t-number of Boolean true.

(ii) For every list, TSG enumerates its elements.

(iii) For every element, if Boolean true, TSG loads set of input parameters indicated by element's position.

(iv) Based on sets of input parameters, TSG computes using Cartesian product to generate uncovered interaction elements.

(v) Uncovered interaction elements are grouped according to their respective list for easier search later on in sampling and selection of test case.

4 Evaluation

Evaluation of the TSG focuses on the efficiency of the strategy to generate better test suites size against existing strategies that support variable-strength interaction. Based on well-known standard benchmark configurations, TSG's generated test suite size is compared with other strategies, including ACS, Density, IPOG, ParaOrder, PICT, SA, TVG, VS-PSTG and WHITCH. They are selected because their results for variable-strength interaction strengths are available publically.

Hence, three sets of experiments that are used for the comparison by Ahmed [9], Xiang [8], Ziyuan [11] and Cohen [7], are adopted as shown in Table 1, Table 2 and Table 3. All the results for above-mentioned strategies are also taken from the adopted experiments [7-9, 11]. By adopting the same experiments that have been used in other research, an objective comparison can be made with existing strategies (i.e. in terms of the generated test suite size). Cells marked NA (Not Available) indicates that the result is unavailable due to the lengthy test generation time while cells marked NS (Not Supported) indicates that the tool cannot generate the test suite for the specific configuration.

The first experiment caters for uniform input parameters that is a fixed constant number of values, and the remaining two sets of experiments were for input parameters with a mixed number of values. The experiments were designed to compare the size of the generated test suite with varying of the number of parameters (P), the number of values (V) and the interaction strength (t) but focusing on Mixed Covering Array (MCA). Referring to Table 1, for the uniform input parameters using configuration, $VSCA\ (N: 2, 3^{15}, \{C\})$, the test suite size results for TSG were only able to get two optimum test suite size as shown in the shaded area with italics font.

While for the mixed input parameters using configuration, $VSCA\ (N: 2, 4^3 5^3 6^2, \{C\})$ and $VSCA\ (N: 2, 3^{20} 10^2, \{C\})$ (referring to Table 2 and Table 3 respectively), the test suite size results for TSG was quite satisfactory and competitive as compared to other existing strategies. The TSGCR also has a majority number of second best solutions in terms of the most optimum test suite size as shown in the shaded area with italics font.

As shown in all these tables, in decreasing order by performance of generated test suite size, in general, the strategy that used AI Search technique, SA, ACS and VS-PSTG respectively are able to get the most optimal test suite size in the specified configurations. However, in mixed input parameters, TSG are able to produce better results than VS-PSTG. Hence, TSG appears to produce competitive results like the rest of the strategies.

5 Conclusion

Evaluation of the TSG focuses on the efficiency of the strategy to generate better test suites size against existing strategies that support variable-strength interaction. Three different sets of experiments were performed. The main intention here was to show that the strategy was sufficiently competitive as compared to other strategies in terms of the generated test suite size. The results show that TSG is able to compete with other existing strategies in supporting variable-strength interaction.

ACKNOWLEDGEMENTS
This research is partially funded by the generous ERGS Grant: CSTWay: A Computational Strategy for Sequence Based T-Way Testing" from Ministry of Education (MOE), Malaysia and Research Intensive Faculty (RIF) grant – "Integrating Seamless Crash Recovery Support For T-way Test Generation Strategy" (File No : 600-RMI/DANA 5/3/RIF (304/2012)) from Research Management Institute (RMI), Universiti Teknologi MARA (UiTM), Malaysia.

Table 1: Sizes of VS interactional test suites for the configuration VSCA (N: $2, 3^{15}, \{C\}$)

{C}	Number of interaction elements	TSG	ACS	Density	IPOG	ParaOrder	PICT	SA	TVG	VS-PSTG	WHITCH
( )	945	20	19	21	21	33	35	*16*	22	19	31
$CA(3, 3^3)$	972	*27*	*27*	28	*27*	*27*	81	*27*	*27*	*27*	48
$CA(3, 3^3)^2$	999	*27*	*27*	28	30	33	729	*27*	30	*27*	59
$CA(3, 3^3)^3$	1,026	28	*27*	28	33	33	785	*27*	30	*27*	69
$CA(3, 3^4)$	1,053	33	*27*	32	39	*27*	105	*27*	35	30	59
$CA(3, 3^5)$	1,215	40	38	40	39	45	131	*33*	41	38	62
$CA(3, 3^4)$, $CA(3, 3^5)$, $CA(3, 3^6)$	1,485	48	40	46	51	44	1,376	*34*	53	45	114
$CA(3, 3^6)$	1,485	48	45	46	53	49	146	*34*	48	45	61
$CA(3, 3^7)$	1,890	51	48	53	58	54	154	*41*	54	49	68
$CA(3, 3^9)$	3,213	59	57	60	65	62	177	*50*	62	57	94
$CA(3, 3^{15})$	13,230	82	76	70	NS	82	83	*67*	81	74	132

Table 2: Sizes of VS interactional test suites for the configuration VSCA (N: $2, 4^3 5^3 6^2, \{C\}$)

{C}	Number of interaction elements	TSG	ACS	Density	IPOG	ParaOrder	PICT	SA	TVG	VS-PSTG	WHITCH
( )	663	39	41	41	43	49	43	*36*	44	42	48
$CA(3, 4^3)$	727	*64*	*64*	*64*	83	*64*	384	*64*	67	*64*	97
$MCA(3, 4^3 5^2)$	1,507	125	104	131	147	141	781	*100*	132	124	164
$CA(3, 5^3)$	788	*125*	*125*	*125*	136	126	750	*125*	*125*	*125*	145
$CA(3, 4^3)$, $CA(3, 5^3)$	852	*125*	*125*	*125*	136	129	8,000	*125*	*125*	*125*	194
$MCA(3, 4^3 5^3 6^1)$	4,290	197	201	207	215	247	1,266	*171*	237	206	254
$MCA(3, 5^1 6^2)$	843	*180*	*180*	*180*	*180*	*180*	900	*180*	*180*	*180*	188
$MCA(3, 4^3 5^3 6^2)$	7,080	239	255	256	NS	307	261	*214*	302	260	312

Table 3: Sizes of VS interactional test suites for the configuration VSCA (N: $2, 3^{20} 10^2, \{C\}$)

{C}	Number of interaction elements	TSGCR	ACS	Density	IPOG	ParaOrder	PICT	SA	TVG	VS-PSTG	WHITCH
( )	3,010	*100*	*100*	*100*	101	*100*	*100*	*100*	101	102	NA
$CA(3, 3^{20})$	33,790	*100*	*100*	*100*	*100*	103	940	*100*	103	105	NA
$MCA(3, 3^{20} 10^2)$	73,990	411	396	401	NS	442	423	*304*	423	481	NA

References

[1] D. R. Kuhn and M. J. Reilly. 2002. An Investigation of the Applicability of Design of Experiments to Software Testing, *Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop*.

[2] D. R. Kuhn, D. R. Wallace and A. M. Gallo, Jr. 2004. Software Fault Interactions and Implications for Software Testing, *IEEE Transactions on Software Engineering*, vol. 30, pp. 418-421.

[3] D. R. Kuhn and V. Okum. 2006. Pseudo-Exhaustive Testing for Software, *Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop*.

[4] S. A. C. Abdullah, Z. H. C. Soh and K. Z. Zamli. 2012. Design and Implementation of a Multilevel Greedy Algorithm on t-way Minimization Strategy, *Proceedings of the 6th Malaysian Software Engineering Conference (MySEC 2012)*.

[5] N. Bouhmala and X. Cai. 2008. A Multilevel Greedy Algorithm for the Satisfiability Problem, Greedy Algorithms, InTech.

[6] M. Chen. 2008. A Greedy Algorithm with Forward-Looking Strategy, Greedy Algorithms, InTech.

[7] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn and J. S. Collofello. 2003. A Variable Strength Interaction Testing of Components, *Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC 2003)*, pp. 413-418.

[8] C. Xiang, G. Qing, L. Ang and C. Daoxu. 2009. "Variable Strength Interaction Testing with an Ant Colony System Approach", *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC 2009)*, pp. 160-167.

[9] B. S. Ahmed and K. Z. Zamli. 2011. A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization, *Journal of Systems and Software*, vol. 84, pp. 2171-2185.

[10] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun and J. Lawrence. 2007. IPOG: A General Strategy for t-Way Software Testing, *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*.

[11] W. Ziyuan, X. Baowen and N. Changhai. 2008. Greedy Heuristic Algorithms to Generate Variable Strength Combinatorial Test Suite, *Proceedings of the 8th International Conference on Quality Software (QSIC 2008)*, pp. 155-160.

[12] J. Czerwonka. 2006. Pairwise Testing in Real World. Practical Extensions to Test Case Generators, *Proceedings of 24th Pacific Northwest Software Quality Conference*, pp. 419-430.

[13] P. J. Schroeder, K. Eok, J. Arshem and P. Bolaki. 2003. Combining Behavior and Data Modeling in Automated Test Case Generation,

Proceedings of 3rd International Conference on Quality Software, pp. 247-254.

[14] A. Hartman, T. Klinger and L. Raskin. 2013. IBM Intelligent Test Case Handler. Available: http://www.alphaworks.ibm.com/tech/whitch (last accessed 01.01.2013)

[15] D. M. Cohen, S. R. Dalal, M. L. Fredman and G. C. Patton. 1997. The AETG System: An Approach to Testing Based on Combinatorial Design, *IEEE Transactions on Software Engineering*, vol. 23, pp. 437-444.