Journal of Engineering and Applied Sciences 14 (10): 3271-3276, 2019
ISSN: 1816-949X

HSS Competitive Rate of Interaction Coverage over Other AI-Based Strategies with Less Number of Interactions

[1]Jalal Mohammed Hachim, [2]Waleed Noori Hussein and [1]Baha Alden Jawad Swadi
[1]Department of Computer Engineering, Iraq University College, Basra, Iraq
[2]Department of Mathematics, Faculty of Education for Pure Science, University of Basra, Basra, Iraq

Abstract: Artificial intelligence methods are adopted recently by several researchers for T-way testing. Whenever attainable, associate degree interaction is roofed at the most once underneath AI-based methods. Priority is given once sampling the foremost optimum action in several AI-based methods. However, the live of interaction coverage drained metric per action is commonly neglected. This study seeks to check the systematic analysis through victimization the present computer science primarily based methods that consist of Tabu Search (TS), simulated hardening sturmarbeiteilung, Harmonic Search Strategy (HSS), Hill climb (HC), Particle Swarm improvement (PSTG) and nice Flood (GF) and as way as their rate of taking a look at coverage worries.

Key words: Interaction coverage, harmonic search strategy, T-way, combinatorial, AI-based strategy, action

INTRODUCTION

Most of the modern (Ahmed, 2012) software that is designed to support various environments and platforms have unique hardware setup, running on different platforms and installed in different types of devices (Ahmed, 2012). In view of the above analysis, the software can be customized in order to suit various user's needs and requirements which are indifferent. The software engineers ought to develop highly configurable and more flexible software to accommodate these indifferences. Chateauneuf *et al*. (1999) suggests that it is important to note that the more such requirements are accommodated, so that, it becomes highly configurable, the more they tend to be complex and complicated in terms of design, planning, size**,** number of inputs and implementation. Under such there are many potential entangled needs among various inputs parameters, hence, exposing faults in the software (Chen *et al*., 2009). In the software development lifecycle, one of the most perhaps important activities is software testing. The tests are conducted in order to identify defects as well (Chateauneuf *et al*., 1999) as ensure conformance against the laid down specification. The techniques that have been used to conduct software testing have not changed as much despite its importance in establishing quality (NIST ACTS 2010). Some of these techniques are dated back in early 80 sec. In these early years, the most significant program consisted of lines of codes that were <1000. The trained testers were still able to develop test suite (Cohen, 2004) through reading the **(**Czerwonka, 2006) whole program line-by-line **(**Cohen, 2004), variable identity and interaction of variable in order to trace (Grindal *et al*., 2005) the main paths of the program (Grindal *et al*., 2005) even in the absence of proper documentation. There are countless programs with a few million lines of code in our society today this is as a result of reuse of commercial-of-the-shelf components (Schroeder *et al*., 2002). This complexity have made it difficult for any single individual to understand the internals by Taguchi and Phadke (1989) the program application totally, thus, rendering effective testing and analysis difficult. Exhaustive testing has become practically and prohibitively impossible due to the significant growth of test suites for all combined inputs with increased lines of codes. This has led to more research techniques with a focus on sampling interaction test suites also known as T-way strategy. Interaction testing has focused on fault assess caused by T-way variable interactions with t indicating the interaction strength this is done in order to complement the existing test strategies available such as boundary value analysis and equivalence partitioning. Findings from the empirical observations indicate that the numbers of variable inputs that are used in the software are relatively small, they are usually in the order of 3-6 in the class of other software and the above findings stemmed the rational used for T-way testing. Considerations are always made in order to

Corresponding Author: Jalal Mohammed Hachim, Department of Computer Engineering, Iraq University College, Basra, Iraq

generate test cases on some T-way combinations, so as to end up with smaller test cases in a case where t variables are known to cause a fault (Taguchi and Phadke, 1989). Yan and Zhang (2008) adopted artificial intelligence AI-based (Taguchi and Phadke, 1989) strategies for T-way (Taguchi and Phadke, 1989) testing bearing in mind the aforementioned test case as an optimization problem generated. According to Cohen *et al.* (1997) "AI-based strategies often offer results that are optimal with a minimum number of (Cohen *et al.*, 1997) test cases compared (Cohen *et al.*, 1997) to conventional counterparts."

Though the AI-based strategy is useful, some issues are still an impediment: even as the search of the (Cohen *et al.*, 1997) most optimal AI-based strategies (Cohen *et al.*, 1997) continues, sufficient focus has not been put when it comes to measuring the interactions coverage metric per (Yan and Zhang, 2008) test case. Coverage metric per (Yan and Zhang, 2008) test case is useful (Yan and Zhang, 2008) when it comes to fault detection in the early stage as it schedules and prioritizes test case execution (Cohen *et al.*, 1997). Test cases with higher coverage have more interactions and hence they ought to be executed first. Test engineers in cases where they are faced with tight deadlines and they ought to execute many tests they should consider test cases with higher coverage. This will help them in detecting faults and defects in the early stages. It is not easy to verify (Cohen *et al.*, 1997) the correctness by Cohen *et al.* (1997) generated test cases (Cohen *et al.*, 1997) given the test suites when the number of parameters and its values is large this is in relation to coverage of all required T-way interactions concerns. Undesirable consequences are expected due to failure in coverage of a particular interaction.

MATERIALS AND METHODS

Theoretical framework: As deduced by NP-hard a T-way test generation problem, it is next to impossible for a single strategy to always generate optimal results at all times. For this reason, researchers have resorted to combining many different approaches in the quest of obtaining the most optimal strategies. The T-way strategies that exist are categorized into two; AI-based and pure computational strategies. According to Ahmed and Zamli (2011a, b), "in the second case, pure computational algorithms are used in searching and sampling of optimal test suite. The first case adopts a well-known artificial intelligence based optimization algorithm. When it comes to implementation both strategies take either one-test-at-a-time or one-parameter-at-a-time approach to generate the final test suite results (Ahmed and Zamli, 2011a, b).

The one-parameter-at-a-time approach came as a result of In-Parameter-Order-Generator IPOG (Othman *et al.*, 2014). IPOG has its groups of families that include IPOD, IPOF, IPOF2 and MIPOG. The first two parameters are subjected to testing under IPOG, this is then extended by generating another pair for the first three parameters, the processes are repeated until all parameters in the system are covered. Whenever necessary, the tests are followed by a vertical extension in order to take care of the uncovered interactions. The one-test-at-a-time consists of the following approach, interactive searches are performed via. all interaction elements and a complete test case is constructed per interaction until completion such that all interaction elements are covered. Strategies that have adopted this approach include Othman *et al.* (2014), TConfig, GTWay, Automatic Efficient Test Generator AETG, Jenny and Test Vector Generator TVG. The one-test-at-a-time approach has also been adopted by AI-based strategy as elaborated earlier. The search algorithm method that is employed between AI-based and conventional strategies makes a big difference. AI-based algorithms are adopted by AI-based strategy to conduct the searching process. Various AI-based algorithms have been adopted for constructing interacting test suites, they include Particle Swarm Optimization (PSO), Tabu Search (TS), Hill Climbing (HC), harmony search algorithm, Great Flood (GF) and Simulated Annealing (SA). If the AI-based algorithms and computational algorithm were to be compared, the former in most cases often achieves better results when it comes to getting the most optimal test suite size while the later achieves better in generation time.

Interaction elements coverage analysis: This research has adopted a pizza ordering system shown in Fig. 1 in order to demonstrate both the construction and covering of interaction elements. Five input parameter system is represented in Table 1. The parameters consist of pizza type, crust, delivery, topping and size.

In order to ease discussions, symbolic values and parameters are used instead of real data (Younis, 2010). The equality symbol ≈ is used to denote values and parameters symbols assigned. This has been depicted in Table 1 (Younis, 2010).

PizzaType (P1) ≈ {VC, ML} Crust (P2) ≈ {TC, ET} Toppings (P3) ≈ {RC, GB, Mu} Size (P4) ≈ {La, Me, Sm} Delivery (P5) ≈ {EI, TA}

Fig. 1: Pizza ordering system

Table 1: Pizza options parameters (Cohen *et al.*, 1997)

Pizza type P1	Crust P2	Topping P3	Size P4	Delivery P5
Vegetarian Cheese ≈ VC	Thin Crust ≈ TC	Roasted Chicken ≈ RC	Large ≈ La	Eat in ≈ EI
Meat Lover ≈ ML	Extra Thick ≈ ET	Ground Beef ≈ GB	Medium ≈ Me	Take AIway ≈ TA
		Mushroom ≈ Mu	Small ≈ Sm	

Table 2: All 2-way possible interaction elements for the pizza system

P1P2	P1P3	P1P4	P1P5	P2P3	P2P4	P2P5	P3P4	P3P5	P4P5
VC, TC	VC, RC	VC, La	VC, EI	TC, RC	TC, La	TC, EI	RC, La	RC, EI	La, EI
VC, ET	VC, GB	VC, Me	VC, TA	TC, GB	TC, Me	TC, TA	RC, Me	RC, TA	La, TA
ML, TC	VC, Mu	VC, Sm	ML, EI	TC, Mu	TC, Sm	ET, EI	RC, Sm	GB, EI	Me, EI
ML, ET	ML, RC	ML, La	ML, TA	ET, RC	ET, La	ET, TA	GB, La	GB, TA	Me, TA
	ML, GB	ML, Me		ET, GB	ET, Me		GB, Me	Mu, EI	Sm, EI
	ML, Mu	ML, Sm		ET, Mu	ET, Sm		GB, Sm	Mu, TA	Sm, TA
							Mu, La		
							Mu, Me		
							Mu, Sm		

The full interactive strength or exhaustive test suite with t = 5 of the pizza ordering system can be achieved by use of 2×2× 3×3× 2 = 72 testcases (Cohen *et al.*, 1997). It is expected that the above exhaustive test suite will grow exponentially with an increase in the number of values and parameters. If t was equal to 2 with an assumption that the testers were looking for a pairwise interaction, the test cases would have decreased significantly with full coverage of the pairwise interactions elements. All the interactions of the system parameters are taken then assignment of related values for each interaction follows in this way, the interaction elements are constructed. All pairwise interaction elements are depicted in Table 2 they demonstrate the pizza ordering configuration system. Cohen *et al.* (1997) states that. "The contribution of all the interaction elements are summed up to obtain all pairwise interaction elements, this is from each possible interaction between P1P2, P1P3, P1P4, P1P5, P2P3, P2P4, P2P5, P3P4, P3P5 and P4P5 correspondingly". Total number of interaction elements can be obtained mathematically that is;(2×2)+(2×3)+(2×3)+(2×2)+(2×3)+(2×3)+(2×2)+(3×3)+(3×2)+(3×2) = 57 (Cohen *et al.*, 1997).

My implementation of HSS in generating test suite is demonstrated in Table 3. It shows the interaction element coverage analysis for t = 2 (Younis *et al.*, 2008). Nine test cases for the aforementioned pizza system is produced by HSS. For the interaction coverage metric per test case to be satisfied, there has to be a definition of a number of variables. There has to be a definition of weight which is referred to the number of interactions covered per test case. In this research, number 10 can be regarded as the maximum weight for any particular case. This means that the number 10 is the two way interactions between parameters. Therefore, the total number of interaction elements in the pizza system in this research is equal to the expected cumulative weight for the last case which is equal to 57. The initials UI denote the uncovered interactions (Taguchi and Phadke, 1989). The uncovered ratio which is defined as the ratio of Uncovered Interactions (UI) over the total number of interaction is denoted by UCR. UCR= Uncovered Interactions /Total number of interaction.

Table 3 shows this research denotes the interaction elements that are covered in redundant using underline and bold fonts. Initially, this study notes that the primary 2 take a look at cases acquire the most attainable accumulative weight of 20 within the third test suit, the interaction part consisting of is redundant because it has already been lined in initial test suit. For this reason, the accumulative weight for the third test suit is 29.

Table 3: A possible 2-way test suite covering all interactions for pizza system (Mahmoud and Ahmed, 2015)

P_1, P_2, P_3, P_4, P_5	Interaction covered	Cumulative weight	UI	UCR
VC, ET, RC, La, TA	{VC, ET}, {VC, RC}, {VC, La}, {VC, TA}, {ET, RC}, {ET, La}, {ET, TA}, {RC, La}, {La, TA}	10	47	0.824
ML, TC, RC, Me, EL	{ML, TC}, {ML, RC}, {ML, Me}, {ML, EL}, {TC, RC}, {TC, Me}, {TC, EL}, {RC, Me}, {RC, EI}, {Me, EI}	20	37	0.650
VC, ET, Mu, Sm, EI	{VC, ET}, {VC, Mu}, {VC, Sm}, {VC, EI}, {ET, Mu}, {ET, Sm}, {ET, EI}, {Mu, Sm}, {Mu, EI}, {Sm, EI}	29	28	0.490
ML, TC, GB, Sm, TA	{ML, TC}, {ML, GB}, {ML, Sm}, {ML, TA}, {TC, GB}, {TC, Sm}, {TC, TA}, {GB, Sm}, {GB, TA}, {Sm, TA}	38	19	0.333
VC, ET, GB, Me, EI	{VC, EI}, {VC, GB}, {VC, Me}, {VC, EI}, {ET, GB}, {ET, Me}, {ET, EI}, {GB, Me}, {GB, EI}, {Me, EI}	44	13	0.280
ML, TC, Mu, La, EI	{ML, TC}, {ML, Mu}, {ML, La}, {ML, EI}, {TC, Mu}, {TC, La}, {TC, EL} {Mu, La}, {Mu, EI}, {La, EI}	50	7	0.123
VC, TC, Mu, Me, TA	{VC, TC}, {VC, Mu}, {VC, Me}, {VC, TA}, {TC, Mu}, {TC, Sm}, {TC, TA} {Mu, Me}, {Mu, TA}, {Me, TA}	54	3	0.053
ML, ET, RC, Sm, EI	{ML, ET}, {ML, RC}, {ML, Sm}, {ML, EL}, {ET, RC}, {EI, Sm}, {ET, EI}, {RC, Sm}, {RC, EI}, {Sm, EI}	56	1	0.018
ML, TC, GB, La, EI	{ML, TC}, {ML, GB}, {ML, La}, {ML, EI}, {TC, GB}, {TC, La}, {TC, EI}, {GB, La}, {GB, EI}, {La, EI}	57	0	0.000

From the fifth check cases onward, there exist multiple redundant interaction components that have already been lined by earlier check cases (Mahmoud and Ahmed, 2015). Also, within the final action at law, this analysis notes that just one new interaction component is roofed to complete all 57 interaction components (Mahmoud and Ahmed, 2015). Regarding UI and UCR, their individual values step by step amendment with every action at law (Mahmoud and Ahmed, 2015). It is expected that UI and UCR worth are happening a downward trend (Mahmoud and Ahmed, 2015). Except for its direct relationship with UCR as cited equation one, UI worth is additionally relating to accumulative weight (Mahmoud and Ahmed, 2015). For the same dish example at any action at law, the add of UI and accumulative weight should always adequate the entire variety of interaction components (Mahmoud and Ahmed, 2015). For this reason, the last worth for UI and UCR is zero.

RESULTS AND DISCUSSION

Methodology and data analysis: This study will use the same analysis that was discussed earlier in benchmarking and analyzing the existing artificial intelligence based strategies where coverage interactions are in terms of metric per test case. AI-based strategies include harmony search algorithm, Tabu Search (TS), Particle Swarm Test Generator (PSTG), Hill Climbing (HC), Great Flood (GF) and Simulated Annealing (SA). Using the interaction strength of four that is t = 4, two configurations based on thirteen 3-valued parameters for instance 3-valued parameters, ten 2-valued parameters, one 5-valued parameter and two 4-valued parameters have been designated for the analysis. The results for GF, SA, TS, PSTG and HC are obtained from the above analysis. Since, the AI-based strategies are used with the greedy algorithm, there is a need for comparing every given result against the greedy algorithm. PSO, TS, HC, GF and SA are put into an experiment for at least five times each. After that, we get the average of the runs which are later summarized as the final result. This experiment has considered the same conditions applied under HSS.

In order to obtain the best results, the experiment undertakes a thousand repetitions for each algorithm. However, the HSS strategy is able to achieve the same results merely using five repetitions only. A representation of the analysis has been depicted in Table 4 and 5. Figure 2 and shows the XY plot of uncovered ratios against test case number. When compared to other strategies in the initial 50 test cases, it can be noted that Table 4 demonstrates HSS as the greediest strategy with an initial low uncovered ratio. It is important to note that towards the end PSTG steadily outshines HSS. It takes a total of 252 test cases to cover the whole interactions using HSS strategy. Without considering HSS and PSTG, SA has the lowest UCR metric compared to the other strategies. The TS and GF results appear to be competitively sufficient. The HC and greedy algorithm are the worst performing overall. Strategies with large uncovered interaction UI still consists of those of HC and greedy algorithm as indicated in the last test case 6 and 12, respectively. The HC and the greedy algorithm require more than 350 test cases in order for them to cover all the interactions this indicates poor generation performance over test considering test size optimality. Table 5 shows the best overall performances by both HSS and PSTG with low UCR metrics. The table indicates the HSS managing to outperform PSTG towards

Table 4: The 4-way interaction elements coverage metric for the input 3^{13}

Test No.	Greedy Repetition No. 1000 UI	Greedy Repetition No. 1000 UCR	HC Repetition No. 1000 UI	HC Repetition No. 1000 UCR	SA Repetition No. 1000 UI	SA Repetition No. 1000 UCR	TS Repetition No. 1000 UI	TS Repetition No. 1000 UCR	GF Repetition No. 1000 UI	GF Repetition No. 1000 UCR	PSTG Repetition No. 20 UI	PSTG Repetition No. 20 USR	HSS Repetition No. 5 UI	HSS Repetition No. 5 USR
25	41,692	07.190	41,629	0.719	40,719	0.703	40,947	0.707	40,845	0.705	40,811	0.705	40,689	0.703
50	29,365	0.507	29,460	0.509	26,895	0.464	27,359	0.472	27,007	0.466	26,878	0.464	26,837	0.463
75	20,497	0.354	20,056	0.346	16,989	0.293	17,634	0.305	16,952	0.293	16,799	0.290	16,811	0.290
100	14,039	0.242	13,868	0.239	10,165	0.176	10,986	0.189	10,187	0.176	9,933	0.172	9,942	0.172
125	9,383	0.162	9,363	0.162	5,641	0.097	6,595	0.114	5,724	0.099	5,538	0.096	5,584	0.096
150	6,271	0.108	6,267	0.108	2,976	0.051	3,795	0.006	3,002	0.052	2,835	0.049	23951	0.051
175	4,154	0.071	4,101	0.070	1,453	0.025	2,075	0.036	1,488	0.026	1,327	0.023	1,432	0.024
225	1,545	0.071	1,592	0.027	180	0.003	492	0.008	214	0.004	155	0.003	174	0.003
250	916	0.026	932	0.016	10	0.0001	188	0.003	13	0.0002	4	6E-05	8	0.0001
275	484	0.008	523	0.009	0	0	32	0.0005	0	0	0	0	0	0
300	201	0.008	252	0.004	0	0	0	0	0	0	0	0	0	0
325	59	0.001	81	0.001	0	0	0	0	0	0	0	0	0	0
0350	6	0.0001	12	0.0002	0	0	0	0	0	0	0	0	0	0

Table 5: The 4-way interaction elements coverage metric for the input 3^{10} 3^3 4^2 5^1

Test No.	Greedy Repetition No. 0 UI	Greedy Repetition No. 0 UCR	HC Repetition No. 1000 UI	HC Repetition No. 1000 UCR	SA Repetition No. 1000 UI	SA Repetition No. 1000 UCR	TS Repetition No. 1000 UI	TS Repetition No. 1000 UCR	GF Repetition No. 1000 UI	GF Repetition No. 1000 UCR	PSTG Repetition No. 20 UI	PSTG Repetition No. 20 USR	HSS Repetition No. 5 UI	HSS Repetition No. 5 USR
25	47,357	0.575	45,010	0.547	44,420	0.539	45,049	0.547	44,422	0.539	44,267	0.538	44,311	0.538
50	29,766	0.362	25,764	0.313	24,625	0.299	25,580	0.310	24,491	0.298	24,445	0.297	24,481	0.297
75	19,724	0.239	15,485	0.188	14,438	0.175	15,318	0.186	14,430	0.174	14,255	0.173	14,279	0.174
100	13,143	0.159	9,496	0.115	8,522	0.104	9,425	0.115	8,493	0.103	8,845	0.103	8,506	0.103
150	6,051	0.074	3,822	0.045	3,029	0.037	3,734	0.045	3,079	0.037	2,969	0.036	3,014	0.037
200	2,786	0.034	1,510	0.018	1,065	0.013	1,480	0.018	1,059	0.013	1,043	0.013	1,071	0.013
250	1,146	0.014	566	0.007	345	0.004	543	0.007	333	0.004	330	0.004	344	0.004
300	376	0.005	177	0.002	101	0.001	151	0.002	93	0.001	91	0.001	90	0.001
350	96	0.001	28	0.0003	16	0.0002	11	0.0001	8	9E-05	4	4E-05	2	2E-05

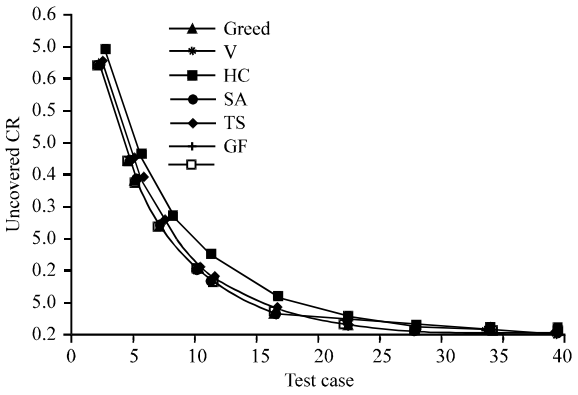

Fig. 2: UCR versus test case number for 3^{13}

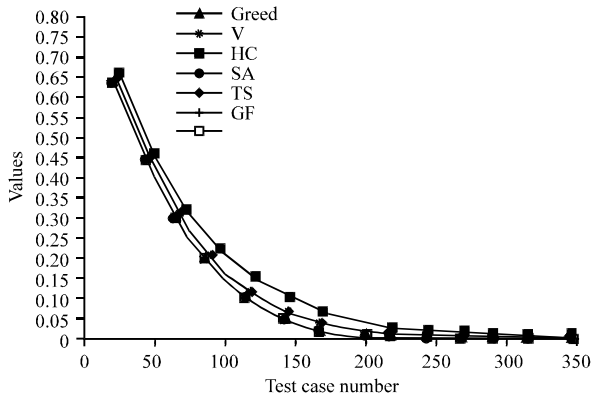

Fig. 3: UCR versus test case number for 2^{10} 3^3 4^2 5^1

the end. About 352 tests are required by HSS to cover all the interactions. Besides HSS and PSTG, the early part of test indicates GF and SA also yield low UCR metric. Comparing SA to GF, SA performs poorly towards the endwith the former performing better. The poorest results are depicted by Greedy algorithm and HS with 0.001 and 0.0003 UCR metric values, respectively.

The fact that lines overlap with each other based on the test case number plot versus UCR as shown in Fig. 2 and 3 shows that there is no significant difference in terms of the greedy performance between all artificial intelligence based strategies. AI-based strategies are usually good optimizers, hence, the above occurrences.

As far as the resulting test cases are concerned, greedy algorithm appears to be out-of-the league. In a nut shell, the above research denotes that for good results to be achieved, 1000 interactions are needed for HC, SA, TS and GF strategies. PSO. Therefore, it requires 20 interactions. Harmony search algorithm for T-way testing

can be said to be superior over others as it requires only 5 interactions to achieve the aforementioned result.

CONCLUSION

In measuring how greedy is a particular strategy, this research highlighted a systematic analysis of existing AI-based strategy using interaction coverage metric per test case. The potentiality of analyzing the test suite is also highlighted which provide the testers with various options to choose from as priority is likely to be given to a strategy with the best test case and high coverage ratio. Best results were obtained through, the experiment which undertakes a thousand repetitions for each algorithm. HSS strategy was able to achieve the same results merely using five repetitions only.

REFERENCES

Ahmed, B.S. and K.Z. Zamli, 2011b. A variable strength interaction test suites generation strategy using particle swarm optimization. J. Syst. Software, 84: 2171-2185.

Ahmed, B.S. and K.Z. Zamli, 2011a. Comparison of metahuristic test generation strategies based on interaction elements coverage criterion. Proceedings of the International Symposium on Industrial Electronics and Applications (ISIEA), September 25-28, 2011, IEEE, Langkawi, Malaysia, ISBN:978-1-4577-1418-4, pp: 550-554.

Ahmed, B.S., 2012. Adopting a particle swarm-based test generator strategy for variable-strength and T-way testing. Ph.D Thesis, Universiti Sains Malaysia, ?Penang, Malaysia.

Chateauneuf, M.A., C.J. Colbourn and D.L. Kreher, 1999. Covering arrays of strength three. Des. Codes Cryptography, 16: 235-242.

Chen, X., Q. Gu, A. Li and D. Chen, 2009. Variable strength interaction testing with an ant colony system approach. Proceedings of the International 16th Asia-Pacific Conference on Software Engineering Conference (APSEC'09), December 1-3, 2009, IEEE, Penang, Malaysia, ISBN:978-0-7695-3909-6, pp: 160-167.

Cohen, D.M., S.R. Dalal, M.L. Fredman and G.C. Patton, 1997. The AETG system: An approach to testing based on combinatorial design. IEEE Trans. Software Eng., 23: 437-444.

Cohen, M.B., 2004. Designing test suites for software interaction testing. Ph.D. Thesis, University of Auckland, New Zealand.

Czerwonka, J., 2006. Pairwise testing in real world. Proceedings of the 24th International Conference on Pacific Northwest Software Quality Vol. 200, October 9-11, 2006, Citeseer, New Jersey, USA., pp: 1-12.

Grindal, M., J. Offutt and S.F. Andler, 2005. Combination testing strategies: A survey. Softw. Test. Verification Reliab., 15: 167-199.

Mahmoud, T. and B.S. Ahmed, 2015. An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. Expert Sys. Appl., 42: 8753-8765.

Othman, R.R., N. Khamis and K.Z. Zamli, 2014. Variable strength T-way test suite generator with constraints support. Malaysian J. Comput. Sci., 27: 204-217.

Schroeder, P.J., P. Faherty and B. Korel, 2002. Generating expected results for automated black-box testing. Proceedings of the 17th International Conference on Automated Software Engineering, September 23-27, 2002, Washington, DC., USA., pp: 139-148.

Taguchi, G. and M.S. Phadke, 1989. Quality Engineering through Design Optimization. In: Quality Control, Robust Design and the Taguchi Method, Dehnad, K. (Ed.). Springer, Boston, Massachusetts, USA., ISBN:978-1-4684-1474-5, pp: 77-96.

Yan, J. and J. Zhang, 2008. A backtracking search tool for constructing combinatorial test suites. J. Syst. Software, 81: 1681-1693.

Younis, M.I., 2010. MIPOG: A parallel t-way minimization strategy for combinatorial testing. Ph.D Thesis, School of Electrical and Electronics, Universiti Sains Malaysia, Penang, Malaysia.

Younis, M.I., K.Z. Zamli and N.M. Isa, 2008. MIPOG-modification of the IPOG strategy for T-Way software testing. Proc. Distrib. Frameworks Appl., 1: 1-6.