A Review on Recent T-way Combinatorial Testing Strategy

Nuraminah Ramli , *Rozmie Razif* Othman*, *Zahereel Ishwar* Abdul Khalib and *Muzammil* Jusoh

School of Computer and Communication, University of Malaysia Perlis

Abstract. T-way combinatorial testing aims to generate a smaller test suite size. The purpose of t-way combinatorial testing is to overcome exhaustive testing. Although many existing strategies have been developed for t-way combinatorial testing, study in this area is encouraging as it falls under NP-hard optimization problem. This paper focuses on the analysis of existing algorithms or tools for the past seven years. Taxonomy of combinatorial testing is proposed to ease the analysis. 20 algorithms or tools were analysed based on strategy approach, search technique, supported interaction and year published. 2015 was the most active year in which researchers developed t-way algorithms or tools. OTAT strategy and metaheuristic search technique are the most encouraging research areas for t-way combinatorial testing. There is a slight difference in the type of interaction support. However, uniform strength is the most utilized form of interaction from 2010 to the first quarter of 2017.

1 Introduction

Software application has become an integral part of our daily lives due to its numerous benefits. One's life can turn complicated if software applications fail to perform their tasks. Software failures can be prevented by implementing software testing activities. Exhaustive testing can be carried out to the software to ensure the software is bug-free. However software testers will be pushed to the limit if exhaustive testing is implemented. Enormous number of test cases needs to be performed. This situation is impossible to be exercised by software tester [1-2]. An increase in the number of test cases contributes to increased time of software development and cost. Combinatorial software testing is a type of testing technique to overcome the problem of exhaustive testing which involves interaction of parameters. This is because software failures are detected due to interaction of few input parameters, or known as t-way testing, where t is the interaction strength [1], [3-4].

To construct t-way combinatorial testing, Covering Array (CA) is used. It is the most active research in combinatorial testing [4]. Covering array can be defined as CA (N; t, k, v) where N represents the rows of the array, t is the strength, k is the degree and v is the order [3].

Many strategies and techniques have been developed to cater to t-way combinatorial testing. Optimization algorithm is supposed to assist in generating smaller test suite size or near optimum number of test cases. Generating the optimum number of test cases is NP-hard problem [6]. This situation has motivated many researchers in this area. Thus, it is the aim of this paper to investigate the existing algorithms or tools, current and potential knowledge related to t-way combinatorial testing. In addition, this paper also explores potential improvements in t-way testing. This paper focuses on studies published on combinatorial testing from 2010 to the first quarter of 2017. Published papers from various types of available databases were collected and analysed according to strategy approach, search technique, supported interaction and the year published.

This paper is organized as follows: In section 2, Combinatorial Software Testing Taxonomy is presented. Existing Algorithms and Tools are described in section 3 while analysis and discussion are highlighted in section 4. Lastly, section 5 delivers the conclusion.

2 Combinatorial Testing Taxonomy

Taxonomy of combinatorial software testing has been developed based on studies done between 2010 and 2017. Figure 1 presents taxonomy of combinatorial software testing. It can be divided into three main areas namely strategy approach, search technique and supported interaction.

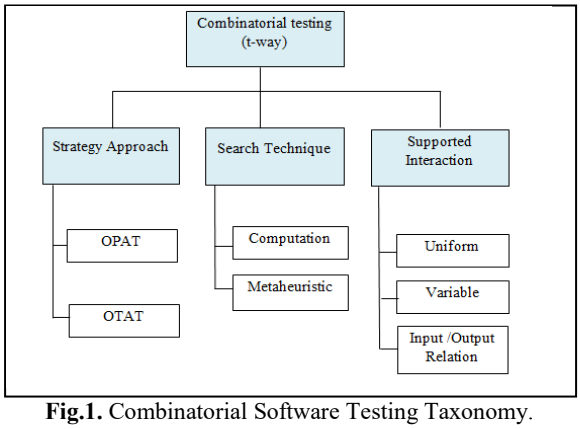

Fig.1. Combinatorial Software Testing Taxonomy.

* Corresponding author: rozmie@unimap.edu.my

2.1. Strategy Approach

Strategy approach can be categorized into two types of strategies. The first strategy is one-test-at-a-time (OTAT). OTAT strategies generate only one test case at one time. Each test case generated must cover at least one t-way combination or uncovered tuples. The best test case generated is the one that covers the most t-way combinations for all parameters. Once the test case is selected, it will be added into a final test suite. This is known as vertical extension. AETG [7] is the pioneer of OTAT strategy.

Another strategy is called one-parameter-at-a-time (OPAT). This strategy starts with t parameter. It produces test suites for t-wise combination. Next, another parameter and a new test are added (horizontal extension). During the horizontal extension, a new test data might be incorporated into test suite to cover all the uncovered t-wise combination. OPAT was popularized by IPOG [8] and its families.

2.2. Search Technique

The two types of search techniques are computational and metaheuristic. General computational approach is based on interaction possibilities. It is able to support very large configurations. The approach repeatedly searches combinations in the searching space until all combinations are covered. This is not practical and can be costly if there are too many combinations [9]. General computational approach is primarily uses greedy technique. Greedy technique constructs test cases that can cover as many uncovered combinations as possible. This technique has been quite effective in generating test cases. However, it is easily trapped in local optima [10].

Metaheuristic search technique generates a small number of rows in any type of covering arrays, support constraints and prioritization [7-8]. This approach starts with a random set of solutions. These solutions will undergo a series of processes to find the best test case using a fitness function. The process repeats until all combinations of parameter input are covered. Metaheuristic search technique has proven to produce an optimum test suite size compared to computational search technique. However, the limitation of metaheuristic is that it requires more time for test suite construction [9–11].

2.3. Supported Interactions

The type of interactions is also an important component in constructing combinatorial testing. There are three types of interactions to construct test cases namely uniform strength, variable strength and Input Output Based Relation (IOR).

Uniform strength or t-wise combines values of parameters with other parameters. The number of related parameters is based on the t (i.e. strength). For example, pair wise or 2-way testing involves combination of values for every two parameters. Pairwise is widely used by software testers. Research has found that faults are detected by smaller interactions (i.e. $t < 2$). Nonetheless, more research has been conducted and it is discovered that more faults can be detected by interactions greater than six [13]. The latest strategy has been developed to support the coverage strength of 14 [14].

Variable strength interaction testing is an extension of uniform strength. To ensure that all parameter combinations are covered at least once, software testers need to run testing at a higher strength or at the highest interaction level. Increasing the strength is unnecessary because that requires using more resources to do unimportant parameter combinations. In view of this problem, variable strength is introduced. Different level of strength can be set to different subset of parameters. This case is applicable for any application or system that needs to be run by multiple configurations [15].

Another form of interaction is IOR. It involves combinations of parameters that impact a particular output. It caters for any combination of related parameter inputs that affect certain output. IOR was initiated as not every application or system has the same characteristics and also to prevent redundancy of test cases or running unnecessary test cases [16]. Through implementation of IOR, both uniform and variable strength strategies are also applied [17]. Uniform strength, variable strength and IOR strategy are further explained in [18].

3 Existing Works

Many algorithms and tools have been explored to be used in t-way combinatorial testing. In this section, existing algorithms or tools developed from 2010 to 2017 are further examined. The existing algorithms and tools are identified according to the year published or their variant. Table 1 presents algorithms/tools published by year.

Table 1 Algorithms/tools published by year

Year	Algorithms / Tools
2010	Particle Swarm Optimization (PSO) • Pairwise PSO (2010) • Particle Swarm Test Generator (PSTG) (2010) • VS-PSTG (2011) • Discrete Particle Swarm Optimization (DPSO) (2015) • Swarm Intelligent Test Generator (SITG) (2015)
2011	• GTWay • Integrated T-Way Test Data Generation (ITTDG) • AURA
2012	Harmony Search • Harmony Search Strategy (HSS) • HS-PTSGT
2013	• DA-RO • DA-FO
2014	• General Variable Strength (GVS)
2015	• TCA • Cuckoo Search (CS) • Flower Strategy (FS)
2016	• High Level Hyperheuristic (HHH) • Artificial Bee Colony (ABC) • Ant Colony System (ACS)

2017	• Adaptive Teaching Learning Based Optimization (ATLBO)

Particle Swarm Optimization (PSO) is an algorithm for t-way combinatorial testing published in 2010. The algorithm is inspired by animal swarms hunting for food. Each individual in the swarm moves towards the best individual location and the best global location based on optimal solution that is calculated according to their position and velocity [19]. A few algorithms or tools have been developed based on PSO algorithm.

The generation of PSO algorithm for t-way testing starts with the development of pairwise PSO [19]. Researchers have developed two different algorithms based on OTAT and OPAT strategies respectively.

The utilization of PSO algorithm in previous research has encouraged the development of Particle Swarm Test Generator (PSTG) [20]. There are several reasons as to why PSO is chosen as the basic platform of PSTG including faster convergence rate attitude, only a few parameters are required to be controlled, can be easily applied to any optimization problem and lighter computational load. PSTG only supports uniform strength up to six levels. Next, VS-PSTG was developed to improve PSTG by enhancing variable strength interaction [21].

Discrete Particle Swarm Optimization (DPSO) [10] is another algorithm based on PSO and uses discrete particle swarm as its basis. The algorithm was designed by adopting S-PSO that uses set-based scheme as its discrete search space. The new DPSO improves performance by having two auxiliary strategies; particle re-initialization and additional evaluation of *gbest*. It also offers guidelines for parameter settings. DPSO is reported to be a promising improvement of PSO.

The last algorithm motivated by PSO between 2010 and 2017 is Swarm Intelligent Test Generator (SITG) [22]. SITG supports 2 to 6 levels of uniform strength and variable strength. SITG outperforms PSTG in some cases of uniform strength while exhibiting better results for variable strength.

As PSO is a metaheuristic algorithm, its entire variant follows a similar search technique. PSTG, V-PSTG, DPSO, SITG and Pairwise PSO (OTAT) use OTAT strategy approach while only Pairwise PSO (OPAT) applies OPAT strategy.

In 2011, three algorithms or tools that apply computational search technique were introduced. GTWay [23] uses backtracking concept. The algorithm needs to map actual data with symbolic representation. Then, it generates possible interactions of t-way pair. Once this is done, the strategy backtracks the uncovered t-way pairs. The t-way pairs are merged if they are able to be combined, complement the missing value of each other and if the new merge interaction can cover the most uncovered t-way pairs. If the pairs fail to merge, the strategy backtracks to the first defined values. GTWay is an OTAT strategy that supports uniform strength for t greater than six.

Another algorithm or tool is ITTDG [17]. It forms a list of candidate test data to be used in deciding a final test data. A visited tuples is put into the candidate list. Next, parameters that have a value that cover the most uncovered tuples is added to the candidate test data, one parameter at a time. In a situation where a tie occurs, the corresponding test data that contains the tied values are duplicated and put into the candidate list. Next, the weight for every test data is calculated. Test data with the highest weight holds the value of the most covered tuples. If a tie happens, the first round in the candidate list will be selected to be put in the final test suite and subsequently removed from the uncovered tuples list. This process is repeated until all uncovered tuples are eventually covered. ITTDG is able to support all three types of interaction; uniform, variable and IOR.

Similar to ITTDG, AURA [24] has the same ability in supporting all types of interactions. AURA starts by exploiting the combination of interactions created earlier to generate test suite. Then, actual data mapping algorithm is used to support symbolic values and actual data output.

Only one type of algorithm was published in 2012, which was Harmony Search (HS). It is the basis for two other algorithms, Harmony Search Strategy (HSS) and HS-PTSGT. HS was inspired by a musician tuning their music.

Another algorithm that uses metaheuristic search technique to generate t-way test suite is Harmony Search Strategy (HSS). It is based on Harmony Search (HS) algorithm. Similar to PSO, HS algorithm requires minimal computation with only a few parameter settings. HSS [14] can support uniform and variable interaction strength with high interaction strength, up to 14 and constraints. Besides HSS, HS-PTSGT [9] is a pairwise generator tool that was developed based on HS algorithm.

In 2013, DA-RO and DA-FO [25] emerged. Both algorithms are computational techniques and support all types of interactions. DA-FO starts with an empty test case which contains unfixed factors. Local density for each factor is determined to obtain the order of coverage requirements. A coverage requirement with the highest local density is selected to be fixed based on global density. For each combination in the test case where the values of factors are fixed, DA-RO will calculate the global density. The combination with the greatest global density is selected to fix the value of factors.

Unlike DA-RO, DA-FO algorithm produces a single test case by fixing values in order of factors. Priority numbers are defined to measure priority of different factors to determine the order of factors. Coverage requirements that hold high priorities and local densities will obtain a great factor density and will be chosen to fix the values. Similar to DA-RO, after fixing values, the local, global and factor densities may change due to re-modification of densities. DA-FO also suffers from the problem of tied values.

GVS [26] was the only algorithm published in 2014. This algorithm is inspired by GTWay. Similar to GTway, GVS is a computational technique and improves

the interaction by supporting all type of interactions. The algorithm chooses a don't care value sequentially. It generates one tuple at a time before the repetition starts. Each tuple is assigned with a don't care value to complete the test data. Next, GVS checks which test cases can cover the most uncovered tuples. The chosen test case is then placed in the final test suite and covered tuples are added in the covered tuples list.

In 2015, TCA, Cuckoo Search (CS) and Flower Strategy (FS) were published. They are metaheuristic algorithms.

TCA [27] is an algorithm that combines greedy Tabu search and random walk heuristic. Greedy Tabu search is used during initialization of test cases. TCA applies heuristic search technique to expand opportunities in covering the uncovered interactions. Through both techniques, the runtime for test suite generation is affected. Although TCA utilizes combination of search techniques, it still falls under the metaheuristic category because Tabu search is a metaheuristic algorithm. TCA supports 3-way constraint uniform strength.

Cuckoo Search [28] has been explored in t-way combinatorial testing as Cuckoo algorithm offers search capabilities using Levi flights [29] to update search space. It consists of a few parameters to be tuned [28]. Cuckoo Search supports uniform strength (up to three) and also variable strength.

Flower Strategy (FS) [30] evolved from the effectiveness of Flower Pollination Algorithm (FPA). FPA is a simple, flexible and requires light computation. It also strikes a balance between exploitation and exploration by employing levy flight. FS is a metaheuristic algorithm that supports uniform strength.

In 2016, hyper-heuristic strategy emerged. High Level Hyperheuristic (HHH) [31] is a pioneer in incorporating hyper-heuristic strategy in t-way combinatorial testing. It consists of high level metaheuristic Tabu Search and other four low level metaheuristic algorithms (i.e. Teaching Learning based Optimization, Global Neighborhood Algorithm, Particle Swarm Optimization, and Cuckoo Search Algorithm). The algorithm uses metaheuristic search technique and support uniform strength from level 2 to 6.

There are two proposed t-way testing strategies found in the literature. Artificial Bee Colony (ABC) [32] is proposed to generate an optimum test suite. This strategy was inspired by a group of bees searching for nectar for their hive. ABC was chosen by researchers as it has proven to be a great strategy in the combinatorial field.

Another proposed strategy found in 2016 is Ant Colony System (ACS) [33]. ACS is a variant of the Ant Colony Optimization (ACO) algorithm. It has successfully solved many combinatorial optimization problems. Ant Colony Algorithm (ACA)[34] and ACS [5] were developed to generate an optimum test suite size to cater uniform and variable strength respectively. The proposed ACS strategy is envisioned to serve all types of interactions especially IOR.

Adaptive TLBO (ATLBO) [35] is a metaheuristic t-way testing that emerged from Teaching Learning-based optimization (TLBO) algorithm in the first quarter of 2017. It is based on Mamdani fuzzy inference system. ATLBO is able to support uniform and variable strength.

4 Analysis and Discussion

This section will discuss the analysis from section 3, existing works and other related matters.

4.1. Analysis of Algorithms and Tools

20 algorithms or tools have been identified since 2010 until 2017. Figure 2 presents the number of uniform, variable and IOR supported interactions by year. The number of algorithms or tools exhibited excludes ABC algorithm as no type of suitable interaction for the algorithm was reported. From the graph, it can be inferred that uniform strength is the most favourable type of interaction within the seven years. IOR comes in second, while variable strength is last by only a slight difference. The number of algorithms and tools that support uniform and variable strength in 2015 is comparable. Six algorithms or tools were developed in 2015, the highest among all the years. The algorithm or tool development was reported to be the lowest in 2012, 2014 and 2017. However, as this report contains data as of the first quarter of 2017, it cannot be concluded that only one algorithm was produced in 2017.

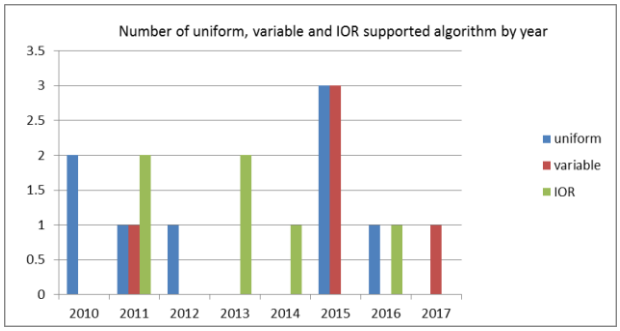

Fig. 2. Number of Uniform, Variable and IOR Supported Interaction by Year

Among these algorithms, PSO is the most widely used within seven years. Five algorithms were developed based on PSO including Pairwise PSO, PSTG, VS-PSTG, DPSO and SITG. PSO requires a smaller number of parameters to be tuned with a simple computational formula. PSO is still popular among researchers for this very reason. PSO has also proven to support uniform strength up to 6 levels and variable strength.

4.2. Analysis of Strategy Approach

There are two categories of strategy approach. From the literature, all 20 algorithms or tools developed employ OTAT strategy. Only PSO makes use of OPAT strategy. OTAT is more popular and favourable as a result of its ability to support higher configuration as opposed to OPAT. The latest trend in uniform strength is likely to offer higher strength such as HSS that supports 14 levels

of strength. Most of the latest uniform strength based algorithm can support up to 6 levels of strength.

4.3. Analysis of Search Technique and Supported Interaction

Table 2 presents algorithms or tools based on their search technique and supported interactions. Six of them apply computational search technique, while the remaining 14 algorithms or tools use metaheuristic search technique. The capability of metaheuristic in solving optimization problem is well-known as it is used in Search-Based Software Engineering (SBSE) [36]. This could possibly be one of the reasons many researchers are interested in studying the metaheuristic search technique.

Among the three supported interactions, uniform strength is the most popular and the most widely used among researchers. In addition, algorithms and tools that support variable strength and IOR are also supporting uniform strength. This could be due to variable strength being an enhancement of uniform strength. While for the IOR, literature proves that IOR can support all types of interaction as mentioned by [18] and Table 2 shows that all six algorithms or tools supports both uniform and variable strength. The second most popular type of interaction is variable strength. 60% of the algorithms or tools support variable strength. Clearly, algorithms or tools that support IOR are the least developed type of interaction at only 30%. Algorithms or tools that support IOR fall under computation category. They were published between 2011 till 2014. It is apparent from the table that none of the metaheuristic algorithms support IOR. However, ACS is proposed to support all types of interactions especially IOR. ABC algorithm is also suggested. However, types of supported interaction are not reported.

Table 2. Summary of algorithms/tools

		Computational						Metaheuristic													
		GVS	GTWay	ITTDG	AURA	DA-RO	DA-EO	HHH	TCA	CS	PSO	PSTG	VS-PSTG	DPSO	HSS	HS-PTSGT	FS	SITG	ABC	ACS	ATLBO
Supported Interaction (strength)	Uniform	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	NR	√	√
	Variable	√	x	√	√	√	√	x	x	√	x	x	√	√	√	x	x	√	NR	√	√
	IOR	√	x	√	√	√	√	x	x	x	x	x	x	x	x	x	x	x	NR	√	x
		√ : Supported , x : Not Supported, NR : Not reported																			

5 Conclusion

T-way combinatorial testing can be explored through various ways or categories such as strategy approach, search technique or supported interactions. In strategy approach, OTAT is the dominant strategy chosen by the algorithms or tools. OPAT strategy is selected only by PSO algorithms (i.e. OPAT PSO). Metaheuristic search technique is the most widely used search technique as compared to computational technique. 70% of algorithms or tools use metaheuristic search technique. Out of three types of interactions, all 20 algorithms or tools is able to support uniform strength. HSS holds up highest strength at level 14. The number of algorithms or tools that support variable strength reached a peak in 2015 and this contributed to the highest number of algorithms or tools in that particular year. 58% of the algorithms or tools utilize metaheuristic search technique. In contrast to variable strength, only one proposed strategy that incorporates metaheuristic search technique is reported to support IOR (i.e. ACS).

References

[1] R. C. Bryce, Y. Lei, D. R. Kuhn, and R. N. Kacker, “Combinatorial Testing,” *Handb. Res. Softw. Eng. Product. Technol. Implic. Glob.*,196–208 (2010).

[2] M. I. Younis, K. Z. Zamli, and R. R. Othman, “Effectiveness of the Cumulative vs . Normal Mode of Operation for Combinatorial Testing,” in *IEEE Symposium on Industrial Electronics and Applications (ISIEA 2010)*, 350–354 (2010).

[3] R. Kuhn, R. Kacker, Y. Lei, and J. Hunter, “Combinatorial Software Testing,” Computers, **42**, 8, 94–96, (2009).

[4] C. Nie and H. Leung, “A survey of combinatorial testing,” ACM Comput. Surv., **43**, 2, 1–29 (2011).

[5] X. Chen, Q. Gu, A. Li, and D. Chen, “Variable strength interaction testing with an ant colony system approach,” in *Asia-Pacific Softw. Eng. Conf. APSEC*, 160–167 (2009).

[6] M. Rahman, R. R. Othman, R. B. Ahmad, and M. Rahman, “Event Driven Input Sequence T-way Test Strategy Using Simulated Annealing,” in *Fifth Int. Conf. on Intelligent Systems, Modelling and Simulation*, 663–667 (2014).

[7] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, “The AETG system: an approach to testing based on combinatorial design,” IEEE Trans. Softw. Eng., **23**,7, 437–444 (1997).

[8] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, “IPOG: A general strategy for T-way software testing,” in *Proceedings of the Int. Symp and Workshop on Eng. of Comp. Based Systems*, 549–556 (2007).

[9] L. Y. Xiang, A. A. Alsewari, and K. Z. Zamli, “Pairwise Test Suite Generator Tool Based On Harmony Search Algorithm (HS-PTSGT),” NNGT Int. J. Artif. Intell.,**2**, 62–65 (2015).

[10] H. Wu, C. Nie, F. Kuo, H. Leung, and C. J. Colbourn, “A Discrete Particle Swarm Optimization for Covering Array Generation,” IEEE Trans. Evol. Comput.,**19**,4,575–591, (2015).

[11] J. Torres-jimenez, C. V. Tamps, and C. V.

Tamps, "Survey of Covering Arrays," in *15th Int. Symp. on Symbolic and Numeric Algorithms for Scientific Computing*, 20–27 (2013).

[12] M. Rahman, R. R. Othman, R. B. Ahmad, and M. Rahman, "A Meta Heuristic Search based T-way Event Driven Input Sequence Test Case Generator," Int. J. Simul. Syst. Sci. Technol., **15**, 3, 65–71(2014).

[13] R. Kuhn, Y. Lei, and R. Kacker, "Practical Combinatorial Testing : Beyond Pairwise," IEEE IT Professional,**10**, 3,19–23 (2008).

[14] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t -way testing strategy with constraints support," Inf. Softw. Technol., **54**, 6, 553–568 (2012).

[15] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, and J. S. Collofello, "A variable strength interaction testing of components," in *27th Annual Int. Comp. Software and Applications Conf.* (2003).

[16] P. J. Schroeder and B. Korel, "Black-box test reduction using input-output analysis," ACM SIGSOFT Softw. Eng. Notes, **25**, 5,173–177, (2000).

[17] R. R. Othman and K. Z. Zamli, "ITTDG : Integrated T-way test data generation strategy for interaction testing," Sci. Res. Essays, **6**,17, 3638–3648 (2011).

[18] R. Othman and K. Zamli, "T-Way Strategies and Its Applications for Combinatorial Testing," Int. J. New Comput. Archit. Their Appl., **1**,2,459–473 (2011).

[19] X. Chen, Q. Gu, J. Qi, and D. Chen, "Applying particle swarm optimization to pairwise testing," in *IEEE 34th Annual Computer Software and Applications Conference*, 107–116 (2010).

[20] B. S. Ahmed and K. Z. Zamli, "PSTG : A T-Way Strategy Adopting Particle Swarm Optimization," in *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, 1–5 (2010).

[21] B. S. Ahmed and K. Z. Zamli, "A variable strength interaction test suites generation strategy using Particle Swarm Optimization," *J. Syst. Softw.*, vol. 84, pp. 2171–2185, 2011.

[22] K. Rabbi, Q. Mamun, and R. Islam, "An Efficient Particle Swarm Intelligence Based Strategy to Generate Optimum Test Data in T-way Testing," in *IEEE 10th Conf. on Industrial Electronics and App. (ICIEA)*, 123–128 (2015).

[23] K. Z. Zamli, M. F. J. Klaib, M. I. Younis, N. Ashidi, M. Isa, and R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support," Inf. Sci. (Ny)., **181**,9, 1741–1758 (2011).

[24] H. Y. Ong and K. Z. Zamli, "Development of interaction test suite generation strategy with input-output mapping supports," Sci. Res. Essays, **6**,16, 3418–3430 (2011).

[25] Z. Wang and H. He, "Generating Variable Strength Covering Array for Combinatorial Software Testing with Greedy Strategy," J. Softw., **8**,12, 3173–3181 (2013).

[26] R. R. Othman, N. Khamis, and K. Z. Zamli, "Variable Strength t-way Test Suite Generator with Constraints Support," Malaysian J. Comput. Sci.,**27**, 3, 204–217 (2014).

[27] J. Lin, C. Luo, S. Cai, K. Su, D. Hao, and L. Zhang, "TCA : An Efficient Two-Mode Meta-Heuristic Algorithm for Combinatorial Test Generation," in *30th IEEE/ACM International Conference on Automated Software Engineering* 494–505 (2015).

[28] B. S. Ahmed, T. S. Abdulsamad, and M. Y. Potrus, "Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm," Inf. Softw. Technol., **66**,13–29 (2015).

[29] X. S. Yang and S. Deb, "Cuckoo search via Levy flights," in *World Congress on Nature and Biologically Inspired Computing,*210–214 (2009).

[30] A. B. Nasser, Y. A. Sariera, A. A. Alsewari, and K. Z. Zamli, "Assessing Optimization Based Strategies for t-way Test Suite Generation : The Case for Flower-based Strategy," in *IEEE Int. Conf. on Control System, Computing and Eng*, 150–155 (2015).

[31] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, "A Tabu Search hyper-heuristic strategy for t-way test suite generation," Appl. Soft Comput. J.,**44**, 57–74 (2016).

[32] M. Shaiful, A. Rashid, R. R. Othman, Z. R. Yahya, M. Zamri, and Z. Ahmad, "Implementation of Artificial Bee Colony Algorithm for T-way Testing," in *3rd Int. Conf. on Electronic Design (ICED)*, 591–594 (2016).

[33] N. Ramli, R. R. Othman, M. Shaiful, and A. Rashid, "Optimizing Combinatorial Input-Output Based Relations Testing using Ant Colony Algorithm," in *3rd Int. Conf. on Electronic Design (ICED)*, 586–590 (2016)

[34] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Proceedings of the 28th Annual Int. Comp. Soft. and App. Conf*, (2004).

[35] K. Z. Zamli, F. Din, S. Baharom, and B. S. Ahmed, "Engineering Applications of Arti fi cial Intelligence Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t -way test suites," Eng. Appl. Artif. Intell., **59**, 35–50 (2017).

[36] M. Harman, Y. Jia, J. Krinke, W. B. Langdon, J. Petke, and Y. Zhang, "Search based software engineering for software product line engineering : a survey and directions for future work," in *15th Soft. Product Line Conference*, 5–18 (2014)