Adopting Ant Colony Optimization Algorithm for Pairwise T-Way Test Suite Generation Strategy

Nuraminah Ramli1,2, Rozmie Razif Othman1,2, Rimuljo Hendradi3, Shukor Sanim Mohd Fauzi4, Izaidy Ismail1,2, Mohd Zamri Zahir Ahmad1, Mohd Wafi Nasrudin1,2

1Advanced Computing, Centre of Excellence (CoE), Universiti Malaysia Perlis, Pauh Putra, 02600 Arau, Perlis, Malaysia
2Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Pauh Putra, 02600 Arau, Perlis, Malaysia
3Faculty of Sciences and Technology, Universitas Airlangga, 60115, Surabaya Jawa Timur, Indonesia
4Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 02600 Arau, Perlis, Malaysia

nuraminah@unimap.edu.my

Abstract. Combinatorial testing or t-way testing (t represents strength) is useful to detect faults due to interactions. Pairwise testing is one type of t-way testing. The technique is effective in reducing the number of test cases without decreasing the level of coverage. Besides, its purpose is to overcome the problem of exhaustive testing that produces a great number of test cases and is impossible to be implemented due to time and cost constraints. Pairwise T-way Test Suite Generation Strategy based on Ant Colony Optimization (pTTSGA) is introduced to generate a near-optimum test suite size. Experiments have been conducted to evaluate the ability of this strategy for pairwise testing. The results are compared to benchmark results. Overall, pTTSGA produces a comparable test suite size.

Keywords : t-way testing; pairwise testing; ACO algorithm; metaheuristic

1. Introduction
Software systems have become crucial and complex as they pose challenges to software testers during testing activities. To test every single component, control flows, scenario or every aspect of the software may lead to exhaustive testing which involves too many test cases to be implemented. Thus, it is impossible due to time execution and cost [1], [2].

Pairwise testing is one of the well-known types of t-way testing. It strength for pairwise is equal to two. Besides 2-way testing, the term is well known as pairwise testing. Pairwise testing will pairs every two input parameters. The final test suite will have at least one test that consists of the pair [3], [4].

Consider an example of a system application in a living room that consists of three input parameters: fan, light, and air conditioner. Each input parameter has two values: on and off, as can be seen in Table 1. To test every combination (i.e. exhaustive testing), 8 test cases are required (i.e. $2^3 = 8$).
8). By employing pairwise testing, only 4 test cases are required as depicted in Table 2. This signals a 50 per cent reduction in test suite size. The example implies that pairwise testing is able to solve exhaustive testing issues.

| Table 1. Input parameters and their values of a living room electrical system application |
|-------------------------------------|-----------------|-------------------|
| Fan      | Light | Air conditioner |
| On       | On    | On               |
| Off      | Off   | Off              |

| Table 2. Pairwise test cases for living room electrical system application |
|-------------------------------------|-----------------|-------------------|
| Fan      | Light | Air conditioner |
| On       | On    | On               |
| On       | Off   | Off              |
| Off      | On    | Off              |
| Off      | Off   | On               |

The purpose of pairwise testing is to produce a near-optimum test suite size and to implement effective testing activities. Nevertheless, generating this near-optimum test suite size falls under NP-hard problem [1], [5]. In view of this, many strategies have been developed to improve the ability of producing minimal test suite size.

This paper presents a new strategy called pairwise T-way Test Suite Generation Strategy that is based on Ant Colony algorithm (pTTSGA). pTTSGA applies metaheuristic search technique in order to find the best test cases. The purpose of this strategy is to produce the minimum number of test cases with a high coverage by using pairwise t-way testing.

pTTSGA is inspired by Ant Colony Optimization algorithm (ACO) which mimics a colony of ants travelling to find food by following the shortest path from their nest to a food source [6]. The algorithm has been used in many optimization problems including in software testing field [7]–[9]. The strategy is adopting Ant Colony System random proportional rule and embedded the element into Ant Colony algorithm to offer better exploration and exploitation in the search space in order to find the best test cases. Besides, the strategy could support higher configurations (i.e. higher parameters and values).

The rest of the paper is presented as follows. The second Section illustrates the research background whereas Section 3 presents the new strategy named pTTSGA. Results and discussion are exhibited in the fourth section. Finally, Section 5 concludes about the research.

2. Research Background

Pairwise testing involves search technique to find the best test cases that can detect faults. It offers two types of search techniques, either to apply metaheuristic or computational search technique [10]. In order to construct test cases, computational search technique works by searching pair of input parameters in the search space. It continues searching until all pairs of input parameters has been covered by the test cases. Its aim is to construct test case that can cover as many pairs as it can. This technique can be used for a large configuration. An example of existing strategies that applies computational search technique is GVS [11] and Density [12] and [13]. In contrast, metaheuristic search technique begins searching randomly. In order to construct the best test cases that can cover all combinations, it has to follow some processes and calculation based on the respective algorithms. Besides that, metaheuristic is using fitness function element to assist in selecting the best test cases. CS [14], MABCTS [5] and GA [7] are example of existing strategies that apply metaheuristic search technique.
T-way testing including pairwise testing can be expressed by mathematical notations of covering arrays (CA) and mixed covering arrays (MCA) [1]. CA is used if the parameters contain the same number of values. Instead, if parameters consist of varying values, then MCA is used. The equations for CA are depicted in Equation (1).

\[ \text{CA} (N; \ t, C) \] (1)

In the equation, \( N \) is the final test suite size, \( t \) represents strength of the t-way testing and \( C \) is the value of configurations and exhibited as \( \text{value}^{\text{Parameter}} \). Here, \( \text{value} \) is the number of parameter values while \( \text{parameter} \) is the number of parameters. Equation (2) shows equation for MCA.

\[ \text{MCA} (N; \ t, C) \] (2)

In MCA equation, \( N \) is the final test suite size, \( t \) is the strength of t-way testing and \( C \) is the value of configurations as represented by \( \text{value}^{\text{Parameter}_0}, \text{value}^{\text{Parameter}_1}, \ldots, \text{value}^{\text{Parameter}_n} \). In the equation, \( \text{value} \) is the number of parameter values and \( \text{parameter} \) indicates the number of parameters.

3. pTTSGA Strategy

Figure 1 shows the pTTSGA framework. Input parameters and their values of system under test (SUT) are required for the strategy. There are three main components in pTTSGA. Firstly, tuples generator is used to generate t-way interactions (in this research, t-way interaction is pairwise or \( t=2 \)), while Search Space Generator is tasked with generating a route for the agents (i.e. ants). Thirdly, the Test Case Generator is utilised to yield the best test cases with the one best test case then being moved to test suite.

The strategy begins by interacting input parameters and their values in pairs by using Tuple Generator. The generated pair is called tuple. The Search Space Generator then proceeds to construct trail which consists of nodes and edges. The node represents parameters while their values are edges that connect the node to another node. Next, the Test Case Generator performs its task to generate test cases. Ant Colony algorithm is integrated into the strategy to assist in producing the best test cases.

Figure 2 presents the algorithm of test suite generation which embeds a modified Ant Colony algorithm to suit this proposed strategy. The generated test case is then placed in the test suite and any covered tuples produced by Tuple Generator are removed. The process continues until the tuple is empty. The algorithm is divided into three main phases, namely Initialization, Trail Construction and Pheromone Update. During Initialization phase, pheromone value is set to a constant value, 0.5, while heuristic value needs to be calculated for each edge. Heuristic value is used to collect early information on each edge. In the Trail Construction phase, random proportional rule is being used to give equal chances either to explore new route or exploit the same route with previous ants. In this
phase, fitness function is being calculated in order to choose the best edge. pTTSGA has proposed a new fitness function to suit all types of t-way interactions. Equation (3) shows the pTTSGA’s fitness function.

\[ f(t_i) = \sum_{p=0}^{\text{program output}} w_p \]  

(3)

In the equation, \( w_p \) is the number of interactions covered by current test but not covered by the previous test. In the equation, \( \text{program output} \) is the output of the SUT. Lastly, Pheromone Update is used to update the pheromone value to all edges that has been selected by the ants. The higher pheromone value on the edge, the higher chances the edge to be selected by the ants.

---

**Figure 2. Test Suite Generation Algorithm**

| Input: Interaction set, S |
| Output: Test Suite, TS |

**Begin**

Set ants, iteration_{max}, bestTest, q

for interaction set, S is not empty {  
    produce single test:
    **Initialization**
    
    Set pheromone, \( \tau \) to a constant value, 0.5
    Initialize heuristic value for every edge,
    \[ \text{heuristicValue}(i,j) = \frac{E_{ij} - E_{ij} + 1}{E_{ij} - E_{ij} + 1} \]
    \[ E_{ij} = \text{number of test cases in } S \]
    \[ E_{max} = \max_{i,j \in \text{edges}} \{ E_{ij} \} \]
    \[ E_{best} = \min_{i,j \in \text{edges}} \{ E_{ij} \} \]

    Repeat {  
        Put ants in node \( N_i \)
        Repeat {  
            **Trail Construction**
            
            Set \( q_0 \) to 0 ≤ \( q \) ≤ 1
            Set random value, \( q \) in [0,1]
            If \( q \leq q_0 \)
                Exploit path, \[ \argmax_{1 \leq r \leq Q} \{ \text{heuristicValue}(r,t) \} \]
            If \( q > q_0 \)
                Explore new path, \[ p_{ij}(t) = \frac{[\text{heuristicValue}(r,t)]^\rho \text{heuristicValue}(r,t)}{\sum_{r=1}^{Q} [\text{heuristicValue}(r,t)]^\rho \text{heuristicValue}(r,t)} \]
            Move to \( N_{ij} \) by choosing path that has the highest value of \( p_{ij} \)
        } Until \( N_i = N_{max} \)
        Calculate fitness function, fit by each ant
        Get overall best test
        if bestFitnessFunction > bestTest
            bestTest = bestFitnessFunction
            Pheromone Update:
            \[ \tau_{ji}(t + 1) = \left\{ \begin{array}{ll}
            (1 - \rho) \tau_{ij}(t) + \rho(fit) & \text{if } e_{ij} \in \text{bestTest} \\
            \tau_{ij}(t) & \text{otherwise}
            \end{array} \right. \]
        } Until iteration > iteration_{max}
        if there is interaction covered by single bestTest
            remove interactions covered by single test from S
            add single bestTest into TS
    } End }
4. Results and Discussion
Experiments have been conducted based on the pairwise experiments implemented by [15] and [7]. Those researches are the benchmarked experiments and they produce benchmarked results in t-way testing. In our research, each experiment was performed in 10 independent runs to obtain the smallest test suite size. In order to perform the experiments, there are seven design parameters has been set as illustrated in Table 3. These values are taken from [16]. The parameter values have been chosen because research conducted by [16] has produced promising results.

| Table 3. Design parameters of pTTSGA |
|-------------------------------------|
| **Design parameter** | **Value** |
| Number of ants | 20 |
| Pheromone control, $\alpha$ | 1.0 |
| Heuristic control, $\beta$ | 0.5 |
| Pheromone evaporation rate, $\rho$ | 0.1 |
| Initial pheromone, $\tau_0$ | 0.5 |
| $q_0$ | 0.5 |
| Iteration | 300 |

Results of the experiments are compared to the other 12 strategies. Those strategies are divided into two categories of search techniques, namely metaheuristic and computational. Strategies involved under metaheuristic search technique are High Level Hyper-heuristic (HHH), Harmony Search Strategy (HSS), Particle Swarm Test Generator (PSTG), Cuckoo Search (CS), Simulated Annealing (SA) and Ant Colony Optimization (ACO)), whereas strategies that apply computational search technique are Automatic Efficient Test Generator (AETG), mAETG, In-Parameter-Order-General (IPOG), Jenny and Test Vector Generator (TVG).

Table 4 and 5 demonstrate comparison results of TTSGA with other strategies for CA and MCA respectively. There are eight system configurations for CA and three system configurations for MCA performed independently. Each system configuration has no relation with other configurations. For Table 4, the input value and number of parameters are increasing for each system configurations. Whereas, for Table 5, the values for each system configurations are increasing with vary number of parameters. For both tables, the results of test suite sizes with a bold font indicate the best test suite size for each configuration. The best test suite size is the smallest number of test suite size generated by each configuration. ‘NA’ indicates that no result is available from the literature.
generates the smallest test suite size for 5 out of strategies, HHH strategy could perform better than computational based strategy generate a strategies. metaheuristic algorithm which gives benefits to the strategy as compared to other computational (i.e. generated best results for all strategies. Whereas overall, pTTSGA shows comparable results with other existing strategies as in Table 4. pTTSGA is able to generate the minimum number of test cases for the system configuration, CA (N, 2, 3^4). All metaheuristic strategies and three computational strategies, namely mAETG, AETG and IPOG also generate the same number of test cases for the configuration. For configuration CA (N; t, 2^10), CA (N;2,5^10), CA (N;2,4^4) and CA (N;2,5^9), pTTSGA generates a smaller test suite size than computational strategies. Whereas for configurations CA (N;2,3^13), MCA (N;2,6^6 5^1 4^6 3^8 2^2), MCA (N;2,2^6 5^1 4^6 3^8 2^2), CA (N;2,2^6) and CA (N;2,3^7), the test suite size generated by pTTSGA is similar to one of the computational strategies. Meanwhile, Table 5 presents that pTTSGA yields competitive results to other strategies for MCA type of system configurations. SA generates the overall best results for all three system configurations.

Both table also highlights that metaheuristic strategies outperform computational strategies for the best results for all configurations except for CA (n;2,3^13) (i.e. generated by AETG) and CA (n;2,3^4) (i.e. generated by AETG and IPOG). As mentioned previously, pTTSGA is developed using a metaheuristic algorithm which gives benefits to the strategy as compared to other computational strategies. The results obtained also are similar to other literature that states metaheuristic strategy can generate a smaller test suite size [1], [4]. It is also proven that metaheuristic search technique based strategy could perform better than computational based strategy [15]. Among the metaheuristic strategies, HHH strategy exhibits the best overall results for covering array experiments which generates the smallest test suite size for 5 out of the 8 configurations. However, it is best to ignore the SA strategy because no results are available for some of the configurations for SA in the literature.

| Table 4. Results of pairwise test suite size for CA |
|---------------------------------------------------|
| | Metaheuristic | Computational |
| | pTTSGA | mAETG | AETG | IPOG | Jenny | TVG |
| CA (N;2,2^4) | 7 | 7 | 7 | 6 | 6 | NA | NA | NA | NA | NA | 8 | 8 | 7 |
| CA (N;2,3^8) | 8 | 8 | 7 | 8 | 8 | NA | NA | NA | NA | NA | 9 | 9 | 9 |
| CA (N;2,4^4) | 15 | 14 | 14 | 15 | 15 | NA | NA | NA | NA | NA | 19 | 16 | 15 |
| CA (N;2,5^4) | 38 | 35 | 35 | 37 | 37 | NA | NA | NA | NA | NA | 42 | 37 | 42 |
| CA (N;2,5^10) | 49 | 42 | 43 | 45 | NA | NA | NA | NA | NA | NA | 50 | 45 | 51 |

| Table 5. Results of pairwise test suite size for MCA |
|---------------------------------------------------|
| | Metaheuristic | Computational |
| | pTTSGA | mAETG | AETG | IPOG | Jenny | TVG |
| MCA (N;2,5^9 3^8 2^2) | 21 | 20 | 21 | 21 | 15 | 15 | 16 | 20 | 19 | 19 | 23 | 22 |
| MCA (N;2,6^6 5^1 4^6 3^8 2^2) | 40 | 36 | 38 | 39 | 43 | 30 | 33 | 32 | 35 | 34 | 35 | 40 | 43 |
| MCA (N;2,7^7 6^1 5^1 4^6 3^8 2^2) | 51 | 48 | 50 | 48 | 51 | 42 | 42 | 42 | 44 | 45 | 43 | 50 | 51 |

Overall, pTTSGA shows comparable results with other existing strategies as in Table 4. pTTSGA is able to generate the minimum number of test cases for the system configuration, CA (N, 2, 3^4). All metaheuristic strategies and three computational strategies, namely mAETG, AETG and IPOG also generate the same number of test cases for the configuration. For configuration CA (N; t, 2^10), CA (N;2,5^10), CA (N;2,4^4) and CA (N;2,5^9), pTTSGA generates a smaller test suite size than computational strategies. Whereas for configurations CA (N;2,3^13), MCA (N;2,6^6 5^1 4^6 3^8 2^2), MCA (N;2,6^6 5^1 4^6 3^8 2^2), CA (N;2,2^6) and CA (N;2,3^7), the test suite size generated by pTTSGA is similar to one of the computational strategies. Meanwhile, Table 5 presents that pTTSGA yields competitive results to other strategies for MCA type of system configurations. SA generates the overall best results for all three system configurations.

Both table also highlights that metaheuristic strategies outperform computational strategies for the best results for all configurations except for CA (n;2,3^13) (i.e. generated by AETG) and CA (n;2,3^4) (i.e. generated by AETG and IPOG). As mentioned previously, pTTSGA is developed using a metaheuristic algorithm which gives benefits to the strategy as compared to other computational strategies. The results obtained also are similar to other literature that states metaheuristic strategy can generate a smaller test suite size [1], [4]. It is also proven that metaheuristic search technique based strategy could perform better than computational based strategy [15]. Among the metaheuristic strategies, HHH strategy exhibits the best overall results for covering array experiments which generates the smallest test suite size for 5 out of the 8 configurations. However, it is best to ignore the SA strategy because no results are available for some of the configurations for SA in the literature.
4.1. Statistical Analysis

Statistical analysis has been performed so that pTTSGA can be compared with other strategies (i.e. HHH, HSS, PSTG, CS, SA, GA, ACO, mAETG, AETG, IPOG, Jenny, TVG). However, six strategies are excluded from the analysis because the results found in the literature are not complete. Only strategies that perform all experiments are participated in the statistical analysis. The participated strategies are HHH, HSS, PSTG, IPOG, Jenny and TVG.

The null hypothesis is defined as no significant difference between pTTSGA and other strategies. Meanwhile, the alternative hypothesis is that there is a significant difference between pTTSGA and other strategies. The significant value, \( p \) is set to 0.05. The null hypothesis is rejected if the \( p \) value is less than 0.05.

Due to the results that are not normally distributed, two non-parametric statistical tests have been chosen to perform the analysis by using IBM SPSS Statistics 20. They are Friedman Test and Wilcoxon Signed Rank Test. Friedman Test is used to rank the strategies while Wilcoxon Signed Rank Test will pair pTTSGA and each participated strategy and compare their significant value.

Table 6 shows the Friedman Test result. From the table, HHH strategy produces the lowest mean rank of test suite size. pTTSGA is in the fourth place behind HSS and PSTG. The slow rate convergence could be the reason pTTSGA producing more test suite size than HHH, HSS and PSTG. HSS strategy [17] that applies Harmony Search algorithm has three parameters to be tuned (i.e. Harmony Memory Size, Harmony Memory Considering Rate and Pitch Adjustment Rate) while PSTG [18] that applies Particle Swarm algorithm requires population size, acceleration coefficients and inertia weight parameters to be tuned. Whereas, HHH strategy [15], a hyper heuristic strategy that employs one high level metaheuristic and four low level metaheuritic, needs to tune Tabu\(_{max}\), acceleration coefficient, initial weight and elitism factor parameters. On the other hand, pTTSGA strategy requires six parameters tuning as in Table 4 (i.e. exclude the iteration) to ensure the suitability of the ACO algorithm in the strategy. Thus, number of parameters could also be the reason that contributes to the slow convergence to find the best test cases.

### Table 6. Friedman Test Result

| Strategy | Mean Rank |
|----------|-----------|
| pTTSGA   | 4.36      |
| HHH      | 2.05      |
| HSS      | 2.68      |
| PSTG     | 3.09      |
| IPOG     | 4.59      |
| Jenny    | 5.55      |
| TVG      | 5.68      |

Meanwhile, TVG occupies the last place among all strategies. The results also further support that metaheuristic based strategies performed better than computational based strategies and consistent with previous research by [15].

To further analyse the results, Wilcoxon Signed Rank Test is performed to see the differences between pTTSGA and other participated strategies. Table 7 presents Wilcoxon Signed Rank Test result. From the table, significant value of HHH, HSS, PSTG and TVG are less than 0.05. Those strategies reject null hypothesis. It means that there exist significant difference between pTTSGA and those strategies respectively. Whereas, significant value for the other strategies (i.e. IPOG and Jenny) are greater than 0.05, which means that test suite size produce by those strategy have no significant difference with pTTSGA. Moreover, Figure 3 illustrates boxplot graph of four strategies including pTTSGA. Boxplot graph has been performed to see the differences between pTTSGA and HHH, HSS, PSTG and TVG. It is used to compare the median score between the strategies. Median score is the middle of the data. It is represented by a line across the box. In this research, the lowest median data
means the strategy produces the smallest median test suite size. From the table, it is obvious that TVG’s median score is the highest while HHH strategy’s has the lowest value. Even though pTTSGA is in the fourth place according to Friedman test, boxplot graph shows that the differences are small among HHH, HSS and PSTG. It can be concluded that pairwise pTTSGA produces competing results with other participating strategies.

Table 7. Wilcoxon Signed Rank Test Result

| Strategy  | Significant value |
|----------|-------------------|
| HHH - pTTSGA | 0.011  |
| HSS - pTTSGA | 0.017  |
| PSTG - pTTSGA | 0.047  |
| IPOG - pTTSGA | 0.858  |
| Jenny - pTTSGA | 0.365  |
| TVG - pTTSGA | 0.019  |

Figure 3. Boxplot graph for pTTSGA, HHH, HSS and TVG

5. Conclusion

This paper mainly discusses on t-way test suite generator, a pairwise TTSGA strategy. This strategy adopts ACO with some improvised elements by adding random proportional rule in the t-way test suite generator to assist in generating a minimum test suite size. pTTSGA consists of Tuples Generator, Search Space Generator and Test Case Generator components as its framework. Two types of experiments (i.e. covering array and mixed covering array) which consist of 11 sub-experiments have been conducted independently. The results were compared with benchmarked results and statistical analysis has been conducted. The goal is to generate a near-optimal test suite size. Results from pairwise experiments are compared to benchmarked pairwise results implemented by other 12 strategies. Overall, pTTSGA renders comparable results. Even though pTTSGA is in the fourth best strategy, the differences between pTTSGA and the other strategies are very small. It shows that pTTSGA is a competitive pairwise strategy. For future works, seeding can be introduced to the strategy to speed up the convergence rate and generates a near-optimal test suite size.
Acknowledgments
The author would like to acknowledge the support from the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2020/ICT01/UNIMAP/02/1 from the Ministry of Education Malaysia

References

[1] A. K. Alazzawi, H. M. Rais, S. Basri, and Y. A. Alsariera, 2019, “PhABC: A Hybrid Artificial Bee Colony Strategy for Pairwise test suite Generation with Constraints Support,” 2019 IEEE Student Conf. Res. Dev. SCOReD 2019, pp. 106–111.

[2] M. I. Younis, A. R. A. Alsewari, and K. Z. Zamli, 2020, “CTJ : Input-Output Based Relation Combinatorial Testing Strategy Using Jaya Algorithm,” Baghdad Sci. J., 17, no. 3, pp. 1002–1009.

[3] R. C. Bryce and C. J. Colbourn, 2007, “The density algorithm for pairwise interaction testing,” Softw. Testing, Verif. Reliab., 17, pp. 159–182.

[4] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, 2015, “A Bat-inspired strategy for pairwise testing,” ARPN J. Eng. Appl. Sci., 1, 18, pp. 8500–8506.

[5] M. Shaiful, A. Rashid, R. R. Othman, and Z. R. Yahya, 2020, “A Modified Artificial Bee Colony Based Test Suite Generation Strategy for Uniform T-Way Testing,” IOP Conf. Ser. Mater. Sci. Eng., 767, 1.

[6] M. Dorigo and T. Stützle, 2004, Ant Colony Optimization. London: The MIT Press.

[7] T. Shiba, T.atsuuya, and T. Kikuno, 2004, “Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing,” in Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004., pp. 72–77.

[8] B. Suri and S. Singal, 2012, “Literature Survey of Ant Colony Optimization in Software Testing,” in Sixth International Conference on Software Engineering (CONSEG), pp. 1–7.

[9] M. Zamri, Z. Ahmad, R. R. Othman, M. Shaiful, and A. Rashid, 2020, “A Self-Adapting Ant Colony Optimization Algorithm Using Fuzzy Logic (ACOF) for Combinatorial Test Suite Generation,” IOP Conf. Ser. Mater. Sci. Eng., 767, 1, pp. 1–10.

[10] N. Ramli, R. R. Othman, Z. I. Abdul Khalib, and M. Jusoh, 2017, “A Review on Recent T-way Combinatorial Testing Strategy,” MATEC Web Conf., 140, 1016, pp. 1–6.

[11] R. R. Othman, K. Z. Zamli, and L. E. Nugroho, 2012, “General Variable Strength T-way Strategy Supporting Flexible Interactions,” Maejo Int. J. Sci. Tech., vol. 6, no. 3, pp.415–429.

[12] C. J. Colbourn, M. B. Cohen, and R. C. Turban, 2004, “A Deterministic Density Algorithm for Pairwise Interaction Coverage,” in LATEST Conf. on Software Engineering, pp. 245–252.

[13] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, 2007, “IPOG: A General Strategy for T-way Software Testing,” in Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems, pp. 549–556.

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

[15] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, 2016, “A Tabu Search Hyper-Heuristic Strategy for T-way Test Suite Generation,” Appl. Soft Comput. J., 44, pp. 57–74.

[16] X. Chen, Q. Gu, A. Li, and D. Chen, 2009, “Variable Strength Interaction Testing with an Ant Colony System Approach,” in Asia-Pacific Software Eng. Conference, APEC, pp. 160–167.

[17] A. A. Alsewari and K. Z. Zamli, 2012, “Design and Implementation of a Harmony-Search-Based Variable-Strength T-way Testing Strategy with Constraints Support,” Inf. Softw. Technol., 54, 6, pp. 553–568.

[18] B. S. Ahmed, K. Z. Zamli, and C. Peng, 2012, “Application of Particle Swarm Optimization to uniform and variable strength covering array construction,” Appl. Soft Comput. J., 12, 4, pp. 1330–1347.