A Self-Adapting Ant Colony Optimization Algorithm Using Fuzzy Logic (ACOF) for Combinatorial Test Suite Generation

Mohd Zamri Zahir Ahmad, Rozmie Razif Othman, Mohd Shaiful Aziz Rashid Ali and Nuraminah Ramli.
School of Computer and Communication Engineering, Kampus Pauh Putra, Universiti Malaysia Perlis, 02600 Arau, Perlis, Malaysia.
E-mail: zamri.zahir@yahoo.com, rozmie@unimap.edu.my, shaifulaziz@unimap.edu.my, nuraminah.ramli@gmail.com.

Abstract. Software testing is one of the most crucial phases in software development life cycle (SDLC). The main function of testing is to cater bugs between interactions of the inputs. It is not possible to eliminate all bugs in one system but by using a suitable testing optimization, it can provide a good enough solution for it. Reducing effort on this phase is not only could lead to numerous bugs between the input interactions, but it also leads to a greater loss such as loss of profits, reputations and even loss of life. Generally, there are three categories of combinatorial testing techniques which is computational, metaheuristic, and hyper heuristic. Ant colony optimization (ACO) is a heuristic technique where its mimic the nature of ants in finding the best route from the nest to the food node and vice versa. Many optimization problems have been solved by using ACO. This paper is to proposed a self-adapting ant colony optimization algorithm using fuzzy logic (ACOF) for combinatorial test suite generation, where it will dynamically determine number of ants and edge selection (i.e. either to explore or to exploit) based on percentage of remaining tuple list and covered test cases.

Keywords: Combinatorial testing, T-way testing, Ant Colony Optimization.

1. Introduction
In development of a software system, there are five main stages that engage throughout of the entire process. Also known as software development life cycle (SDLC), it consists of software requirement analysis, system design, implementation, testing and evolution. It is reported that, from this five stages, testing phase consume up to 50% of total cost development. This numbers could growth if the system involves with safety-critical applications [1]. Software failure is a disaster situation as it could increase the cost of software development tremendously. In order to minimize the software failure, a rigorous testing needs to be conducted. To ensure the effectiveness of the testing activity, many software testing technique have been developed in literature. Combinatorial testing (or also known as t-way testing) is one of the test design technique that has been used widely. It can be used to generate test suite that can cater bugs due to interaction between input parameters [2]. In generating the t-way test suite, t-way strategy uses either mathematical or computational methods in reducing the test suite size but at the same time still covering all interactions involving specified parameters-values combination for specific strength, t [3,4].
As the main aim of all combinatorial test suite generation strategy is to produce as small test suite size as possible, many optimization algorithm has been proposed and adopted for the purpose of generating t-way test suite (e.g IPOG, Jenny, GTWay, ITTDG and TVG). At the same time, there are researchers that use metaheuristic optimization technique in generating t-way test suite. Bestoum implements Particle Swarm Optimization technique in generating uniform and variable strength t-way test suite while Alsewari adopted Harmonic Search Algorithm for generating uniform strength t-way test suite. There is also an attempt made by Zamli of using more than one metaheuristic technique (namely hyper-heuristic technique) to generate uniform strength t-way test suite. The capability of metaheuristic techniques in producing smaller size of test suite become the motivation of researchers to implements these techniques in generating t-way test suite.

One of the most used metaheuristic technique in generating t-way test suite is Ant Colony Optimization (ACO) algorithm. In 2004, Shiba successfully implements ACO algorithm in generating uniform strength t-way test suite [5]. Enhancing Shiba works, Chen added local search in his implementation of ACO (called Ant Colony System) for generating variable strength t-way test suite [6]. Recently, Ramli proposed her implementation of ACO algorithm for input-output based t-way test suite [7]. Overall, all implementation of ACO successfully producing the smallest t-way test suite in many system configurations [5,6,8–12].

The success of ACO in producing smaller test suite size is highly depending on number of ants used in the algorithm. Increasing number of ants increase the probability of producing the optimum test suite. Nevertheless, increasing number of ants will also increase the execution time. Although the execution time are also affected by system environment used, optimizing the numbers of ant possibly could optimize the ant colony efficiency. Thus, we propose a study on a self-adapting ACO algorithm using fuzzy logic for combinatorial test suite generation. The fuzzy component will dynamically determine the suitable number of ants that will be used in ACO and edge selection either to explore or to exploit the search space.

2. Related Works

Particle Swarm Optimization (PSO) is an algorithm for t-way combinatorial testing published in 2010. The algorithm is inspired by swarm animals hunting for their food. Each individual in the swarm moves towards the best individual location and the best global location based on optimal solution calculated from their position and velocity [13]. A few algorithm or tools is developed based on PSO algorithm. The generation of PSO algorithm for t-way testing starts with the development of pairwise PSO [13]. The researchers have developed two different algorithms. One algorithm is OTAT strategy and another one is OPAT. The utilization of PSO algorithm in the previous research has encouraged the development of Particle Swarm Test Generator (PSTG) [9]. Faster convergence rate attitude, requires a few parameters to be controlled, easily apply to any optimization problem and lighter computational load are the reasons PSO has been chosen as PSTG’s basic platform [9,10]. PSTG only support uniform strength up to 6 levels. Next, VSPSTG has been developed to improve PSTG by enhancing variable strength interaction [11]. Discrete Particle Swarm Optimization (DPSO) [14] is another algorithm based on PSO and discrete particle swarm as its basis. This new algorithm is designed by adopting S-PSO that uses set-based scheme as its discrete search space. The new DPSO has improves the performance by having two auxiliary strategies; particle reinitialization 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 that motivated by PSO between 2010 and 2017 is Swarm Intelligent Test Generator (SITG) [12]. SITG could support 2 to 6 uniform strength and variable strength. SITG outperform PSTG in some cases in uniform strength while performing better results for variable strength. As PSO is a metaheuristic algorithm, its entire variant follows the similar search technique. PSTG, V-PSTG, DPSO and SITG use OTAT strategy approach while Pairwise PSO has both OTAT and OPAT strategy for different type of Pairwise PSO algorithm.

In 2011, GTWay, ITTDG and AURA were introduced as another t-way combinatorial testing algorithms. Those algorithms applied computational search technique to find the best test cases.
GTWay [15] applies backtracking concept. The algorithm needs to map the actual data with symbolic representation. Then, it generates t-way pair possible interactions. After that, the strategy backtracks the uncovered t-way pairs. The t-way pairs are merge if they are combinable and complement each other’s missing value and the new merge interaction could cover the most uncovered t-way pairs. If the pairs fail to merge, backtrack algorithm goes back to the first defined values. GTWay is an OTAT strategy that supports uniform strength for t greater than six. ITTDG [8] practices a list of candidate test data to be used in deciding a final test data. The visited tuples are put into a candidate list. Next, parameter that has a value that covers the most uncovered tuples is added to the candidate test data, one parameter at a time. In a situation where tie occurs, the corresponding test data that has the tie values is duplicated and put into the candidates list. Once the candidate list finished forming a complete test data, the strategy calculates weight for every test data. Test data with a highest weight has a value of the most covered tuples. If tie happens, the first round in the candidate list will be selected to be put in the final test suite and removed from the uncovered tuples list. The process repeated until all uncovered tuples are being covered. ITTDG could support all three types of interaction; uniform, variable and IOR.

Similar to ITTDG, AURA [16] has the same ability in supporting all type of interactions. AURA started by exploiting the interactions combination created earlier to generate test suite. Lastly actual data mapping algorithm is used to support symbolic values and actual data output. Only one type of algorithm published in 2012. Harmony Search algorithm is the basis for two other algorithms, HSS and HS-PTSGT. Another algorithm that uses metaheuristic search technique to generate t-way test suite is Harmony Search Strategy (HSS) [4,15]. It is based on Harmony Search (HS) algorithm. Similar to PSO, HS algorithm requires lightweight computation with only few parameters setting. HSS could support uniform and variable interaction strength with high interaction strength, up to 14 and constraints. Besides HSS, HS-PTSGT [6] is a pairwise generator tool that was developed based on HS algorithm.

In 2013 DA-RO and DA-FO [17] emerged. Both algorithms are a computational technique and support all type of interactions. DA-FO starts with an empty test case which contains unfixed factors. Local density for each factor is determined to get the order of coverage requirements. A coverage requirement with highest local density is selected to be fixed factor based on global density. For each combination in the test case where the values of factors are fixed, DA-RO will calculate global density. The combination which holds greatest global density is selected to fix the value of factors. Unlike DA-RO, DA-FO algorithm produces single test case by fixing value in order of factor. Priority number is defined to measure priority of different factors to determine order of factors. Coverage requirements that hold high priorities and local densities will get 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 tie problem. To break the ties, the similar factors as DA-RO can be used. GVS [18] is the only algorithm published in 2014. The algorithm is inspired by GTWay. Similar to GTway, GVS is a computational technique and improves the interaction by supporting all type of interactions. It generates one tuple at a time before the iteration starts. Each tuple which is generated in the tuple generator is assigned with a don’t care value to complete the test data. The test data with the first don’t care value, is compared with the uncovered tuples list. If the test data exists, the test data with the first don’t care is selected. However, if not, the algorithm changes the next remaining value parameter in the don’t care value space. The process of determining the don’t care values is repeated until all parameter values finish. Afterward, GVS checks which test cases could cover the most uncovered tuples. The chosen test case is then placed into 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 [19] is the algorithm that combines greedy Tabu search and random walk heuristic. Greedy Tabu search is used during initialization of test cases, and then TCA apply heuristic search technique to expand the opportunities in covering the uncovered interactions. By using both techniques, it effects the runtime test suite generation. Even though TCA utilizes combination of search techniques, it is still under metaheuristic category because Tabu search is a metaheuristic algorithm. TCA supports 3-way constraint uniform strength. Cuckoo Search [20] has been explored in t-way combinatorial testing
as a capacity that Cuckoo algorithm offers search capabilities using Levi flights [21] to update search space and it only consists of few parameters to be tuned [20]. Cuckoo Search supports uniform, up to 3 interaction strength and also variable strength. Flower Strategy (FS) [22] evolved from the beauty of Flower Pollination Algorithm (FPA). FPA is a simple, flexible and requires lightweight computation. It also offers a balance between exploitation and exploration by employing levi flight. FS is a metaheuristic that supports uniform strength.

In 2016, hyper-heuristic strategy emerged. High Level Hyperheuristic (HHH) [23] is the pioneer in utilizing hyper-heuristic strategy in t-way combinatorial testing. It consists of high level metaheuristic Tabu Search and other four (4) low level metaheuristic algorithms (i.e. Teaching Learning based Optimization, Global Neighbourhood Algorithm, Particle Swarm Optimization, and Cuckoo Search Algorithm). The algorithm uses metaheuristic search technique and could support uniform strength from 2 until 6 level of strength. There are two proposed t-way testing strategies found in the literature. Artificial Bee Colony (ABC) [24] has been proposed to generate an optimum test suite. The strategy is inspired by a group of bees searching for nectar for their hive. ABC was chosen by the researchers as it was proven to be a great strategy for combinatorial field. Another proposed strategy found in 2016 by Ramli is Ant Colony System (ACS) [7]. ACS is a variant of Ant Colony Optimization (ACO) algorithm. It has been successfully solving many combinatorial optimization problems. Ant Colony Algorithm (ACA) [5] and ACS [6] have been built to generate an optimum test suite to cater uniform and variable strength respectively. The proposed ACS strategy is to cater all type of interactions especially IOR. Adaptive TLBO (ATLBO) is a metaheuristic t-way testing that emerged from Teaching Learning-based optimization (TLBO) algorithm in the first quarter of 2017. It was based on Mamdani fuzzy inference system. ATLBO could support uniform and variable strength [25].

3. Problem Definition Model
To demonstrate uniform strength of t-way testing, let’s consider a system for scholarship sponsor application. The system is to help filtering application from student to determine which student is eligible to obtain the scholarship. There system will filter the application based on four factor which is cumulative grade point average (CGPA), course, study level and parent salary as shown in Table 1. To ease the discussion, a representation of parameters and its value is shown in Table 2. The CGPA is represented as ‘C’, course represented as ‘K’, study level represented as ‘S’ and parent salary represented as ‘P’. For the values, each of the values represented by small letter of the represented filtering factor. For example, c1 referring to input parameter CGPA and the value is less and equal to 3.

| Filtering Factor | CGPA | Course     | Study Level | Parent Salary |
|------------------|------|------------|-------------|---------------|
| Values           | ≤ 3  | Engineering| Diploma     | ≤ RM2000      |
|                  | > 3  | Business   | Bachelor Degree | > RM2000    |

| Input Parameter | C   | K   | S   | P   |
|-----------------|-----|-----|-----|-----|
| Values          | c1  | k1  | s1  | p1  |
|                 | c2  | k2  | s2  | p2  |

To minimize the potential bugs in this system, appropriate testing must be made before release the system to end user. Interaction testing is the crucial test that need to be conduct as fault may occurred between this input parameter. One of the technique to test the system is by using exhaustive testing. In exhaustive testing, all possible input combinations need to be tested. This means, all input is interacting with each other where faults are expected to arise between the interactions. As a result, 16 test cases (i.e. $2^4 = 16$) need to be executed. The list of exhaustive test case is shown in Table 3.
Table 3. Exhaustive Test Suite for Scholarships Application.

| C  | K  | S  | P  |
|----|----|----|----|
| c1 | k1 | s1 | p1 |
| c1 | k1 | s1 | p2 |
| c1 | k1 | s2 | p1 |
| c1 | k1 | s2 | p2 |
| c1 | k2 | s1 | p1 |
| c1 | k2 | s1 | p2 |
| c1 | k2 | s2 | p1 |
| c1 | k2 | s2 | p2 |
| c2 | k1 | s1 | p1 |
| c2 | k1 | s1 | p2 |
| c2 | k1 | s2 | p1 |
| c2 | k1 | s2 | p2 |
| c2 | k2 | s1 | p1 |
| c2 | k2 | s1 | p2 |
| c2 | k2 | s2 | p1 |
| c2 | k2 | s2 | p2 |

Table 3 shown exhaustive testing of the system where the strength, $t$ is 4. The interaction strength is possible to be reduce to $2$ ($t=2$) or also known as pairwise testing. This means, only two input parameter are interacting and faults is expected to arise within these interactions as compared to four input parameter in the exhaustive testing. The rest of input parameter that not related to interaction are considered as ‘don’t care’ (DC) value, as it will not affect the input interactions testing. The purpose to reduce the strength is to minimize the generated test case without compromise the interacting input. The possible input interaction for 2-way testing is CK, CS, CP, KS, KP and SP. As the result, the generated test case now become smaller where the repeated test case is removed from the list as shown in Figure 1. Noted that, it almost achieves 44 percent reduction of test cases as compared to exhaustive testing. The reduction may not significant for a small and simple application, but it is significantly reduced for a larger and complex applications environment. For example, system with 40 input with 2 possible value each will resulted $1099511627776$ (i.e. $2^{40}$) test cases need to be executed. It is impractical to do the exhaustive testing as it will consume a lot of time as well as many software testers to be part in, therefore the overall cost of testing will increase enormously. Hence, by implementing t-way testing technique, it is proven could reduce the test cases and save a lot of resources with an acceptable fault detection.

![Figure 1. Test Suite for 2-Way Testing.](image-url)
4. Overview of ACO

Ant algorithm is a metaheuristic approach where it is inspired by nature life behavior of ant colony in finding shortest path to foods from their nest. Some ants’ species are completely blind and they can’t see anything, yet they have a very powerful searching system where they can find shortest path to the foods source. Ants sense the path by a special chemical deposited by them while travelling the path called as pheromone. The more pheromone deposited to the trail increases the probability for the ants to choose that path.

There were several algorithms have been proposed inspired from the nature behavior of ant colony. In early nineties, an Ant System (AS) was the first proposed by Dorigo et al. [26] for travelling salesman problem (TSP). Ant system consist of three main techniques which is ant-cycle, ant-density, and ant-quantity. Among of this three techniques, ant-cycle is the most performed compared the other two hence it was later established as the Ant System [27]. This algorithm was then being enhance producing several more algorithm based on this Ant System. Among of the rise algorithm are Ant System Elitist, rank-based Ant System (ASrank), MAX–MIN Ant System (MMAS), and Ant Colony System (ACS). ACS is the tuned version of AS where several modifications been made to enhance the searching effectiveness. It is reported that ACS give the most competitive performance compared with others AS strategies because it goods in the exploitation and exploration in searching of the best solutions [6,28]. To be comprehensive, ACS is improved because of;

a) Edge selection

The edge selection rule provides, which called as random-proportional rule; a through probability method to balance searching solutions, between exploration of the new path or exploitation of the path with previous information. A random variable of $q$ and user-define $q_0$ is used to define the probability of the ants in selecting the edges. Variable $q$ is uniformly distributed in $[0,1]$ and variable $q_0$ ($0 \leq q_0 \leq 1$) is used to determine either the ant to do the exploration or exploitation.

- Exploration; if ($q_0 > q$)

$$ p_{i,j}(t) = \frac{[\tau_{i,j}(t)]^{\alpha} [\eta_{i,j}(t)]^{\beta}}{\sum_{h=1}^{\nu_i} [\tau_{i,h}(t)]^{\alpha} [\eta_{i,h}(t)]^{\beta}} $$

(1)

- Exploitation; if ($q_0 \leq q$)

$$ \text{Argmax}_{1 \leq h \leq \nu_i} \left( [\tau_{i,h}(t)]^{\alpha} [\eta_{i,h}(t)]^{\beta} \right) $$

(2)

where;

- $[\tau_{i,j}(t)]$ is the amount of pheromone values of edges $e_{i,j}$.
- $\eta_{i,j}(t)$ is the heuristic values value of edge $e_{i,j}$, and
- $\alpha(\alpha \leq 0)$ and $\beta(\beta > 0)$ are the factor that determine relative influence of pheromone and heuristic value on the edge decision of the ant.

b) The pheromone update

- Global pheromone update

Ants that travel along the path will construct the path based on probability of the edge chosen. Eventually there is only one best ant identified as the single best ant (or globally best ant) where it constructs the shortest path compare to the others. This single best ant is the only ant that allowed to deposit pheromone to the trail. Meanwhile, pheromone evaporation occurs in order to avoid the algorithm stuck in a local minima solution. This is process done after all the ants have successfully complete construct the path in the current iteration.

$$ \tau_{i,j}(t + 1) = \begin{cases} (1 - \rho)\tau_{i,j} + \rho \Delta \tau_{i,j}^{bs} & \text{if } e_{i,j} \in \text{test}^{bs} \\ \tau_{i,j}(t) & \text{otherwise} \end{cases} $$

(3)
where;
- $\rho$ is the pheromone evaporation rate and,
- $\Delta \tau_{(i,j)}^{bs}$ is the fitness function.

- The local pheromone update
  The main intention of local pheromone update is to make the search performed by following ants during the iteration diversified. While travelling in trail path, each ant is allowed to deposit their pheromone at the last edges. Meanwhile, pheromone evaporation occurs as to make sure subsequent ant explore the different edges. Ants will update the pheromone level by applying local update rule;

$$
\tau_{i,j}(t) = (1 - \varphi)\tau_{i,j}(t) + \varphi \tau_0
$$

where;
- $\tau_0$ is the initial value of pheromone and
- $\varphi$ ($0 < \varphi < 1$) is the evaporation of the pheromone.

5. Proposed Strategy
The aim of t-way testing test is to find faults between interaction input parameter. In previous section, the algorithm of ant and variant of ant colony optimization is explained. However, there is room for improvement that still can be made to the algorithm, where number of ants is still not optimized. As ACOF will adapt OTAT searching technique, number of ant’s agent can be optimized based on remaining uncovered tuple using fuzzy logic algorithm. Besides, the probability in selecting the edge to travers also need to be tuned with exploration in early iteration and exploitation in middle or near end. Figure 2 shows the main component function of ACOF. It consists of four main components that drive towards the effectiveness of ACOF.

![Figure 2. Main Framework of ACOF](image)

5.1. Ant Colony Path Generator (ACPG)
This component function is to generate path of the ant based on configuration setting set by software tester. It consists of virtual nodes where it represents the inputs and branch for each node where it represents value of each input. This path will be used by ant agents to travel from each input node, in finding the optimum path. This will help in generating the optimum test case.
5.2. Tuple List Generator (TLG)
This function is to generate a full list of interaction tuples. This function retrieves inputs parameters (i.e. number of input and its values), interaction parameters as well as interaction strength prior generating the tuple list. The tuple list then will be used by test case generator to determine which test case covered the most tuple. Tuple list will eventually be empty as every covered tuple by test cases will be removed from the list.

5.3. Ant Agent Fuzzy Controller (AAFC)
Different input configuration tends to have different amount of ant agent to be released. This function is to dynamically assign number of ant agent without regarding the input setting. Different from others ACO algorithm, number of ant agent will be optimized automatically using fuzzy logic algorithm. It is based on the percentage of uncovered tuples in the tuple list and percentage of covered test case of every iteration. For the early iteration, less number of ant will be assigned as there are still many uncovered tuples in the list. While the iterations reach to half or near end, more ant will be assigned as to help test case generator to find the most uncovered tuple.

5.4. Test Case Generator (TCG)
TCG is the main component to generate test case. As ACOF will adapt OTAT, one best test case will be generated in one process iteration. Test case generator uses ACPG as the path for ants to traverse in constructing the test case. In traveling from node to node, ant will have to make decision in which branch they will choose. The decision is based on heuristic value, pheromone, probability and fitness function. However, as mentioned in AAFC earlier, the probability need to be tune as to control the exploration and exploitation of the path travel by the ants. In finding the best test case for early iteration, ant will explore the search space while in middle or near end of the process, ants will exploit path from the last best test case. This is since, at beginning of the iteration, there are still a lot of uncovered tuples remain in the tuple list. Thus it is to avoid the algorithm stuck in local minima solution, where there might be other best solution in other space. Eventually, ant will do the exploitation as remaining of uncovered tuple now become fewer to empty.

5.5. Flowchart of ACOF
An overview of how the strategy works is presented in Figure 3. The flowchart presents the process in general. The strategy accepts input parameters and its values as input to the strategy and produces test suite. Firstly, TLG generates a tuple list. In the meantime, ACPG starts to generate path to be travelled by ant’s agent based on input parameters and its values. After both TLG and ACPG finished their tasks, TCG starts to generate a single best test case.

Next, the generated single best test case will construct its own tuples. Each tuple then is compared to the tuples list. Any similar tuples will be removed from the tuple list. The similar tuples mean that the single best test case covers at least one combination of input parameters. The single best test case is then stored in the Test Suite. However, if none of the constructed tuple is similar to the tuples in the tuple list, the TCG continues to generate another single best test. Throughout the process, the tuple list is becoming smaller as it has been removed whenever it is covered by a single best test. The process iterates until the tuple list is empty.
Figure 3. Flowchart of ACOF

6. Conclusion and Future Works
In this paper, we proposed a t-way strategy that based on ant colony optimization with fuzzy logic. The framework of ACOF is consist of four main components which is Tuple Generator, Ant Colony Path Generator, Ant Agent Controller and Test Case Generator. Next, the algorithm will be implemented and the performance of the ACOF (i.e. test suite size) will be evaluated and compare against other strategies with several benchmarking experiment that have been published.

Acknowledgement
The author would like to acknowledge the support from Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2018/ICT01/UNIMAP/02/1 from the Ministry of Education Malaysia.

References
[1] Ammann P and Offutt J 2008 Introduction to Software Engineering (Cambridge University Press)
[2] Ahmad M Z Z, Othman R R and Rashid Ali M S A 2016 Sequence Covering Array Generator (SCAT) for Sequence Based Combinatorial Testing Int. J. Appl. Eng. Res. 11 5984–91
[3] Kuhn D R, Kacker R N and Lei Y 2009 Combinatorial Software Testing Computers 42 94–6
[4] Alsewari A R A and Zamli K Z 2011 Interaction Test Data Generation Using Harmony Search Algorithm 2011 IEEE Symp. Ind. Electron. Appl. ISIEA 2011 559–64
[5] Shiba T, Tsuchiya T and Kikuno T 2004 Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing Proc. 28th Annu. Int. Comput. Softw. Appl. Conf. 2004. COMPSAC 2004. 72–7
[6] Chen X, Gu Q, Li A and Chen D 2009 Variable Strength Interaction Testing with an Ant Colony System Approach Proc. - Asia-Pacific Softw. Eng. Conf. APSEC 160–7
[7] Ramli N, Othman R R and Ali M S A R 2017 Optimizing Combinatorial Input-Output Based Relations Testing Using Ant Colony Algorithm 2016 3rd Int. Conf. Electron. Des. ICED 2016
Othman R R and Zamli K Z 2011 ITTDG: Integrated T-way Test Data Generation Strategy for Interaction Testing Sci. Res. Essays 6 3638–48

Ahmed B S and Zamli K Z 2010 PSTG: A t-way strategy adopting particle Swarm Optimization AMS2010 Asia Model. Symp. 2010 - 4th Int. Conf. Math. Model. Comput. Simul. 1–5

Ahmed B S, Zamli K Z and Lim C P 2012 Application of Particle Swarm Optimization to uniform and variable strength covering array construction Appl. Soft Comput. J. 12 1330–47

Ahmed B S and Zamli K Z 2011 A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization J. Syst. Softw. 84 2171–85

Rabbi K, Mamun Q and Islam R 2015 An Efficient Particle Swarm Intelligence Based Strategy to Generate Optimum Test Data in T-way Testing 2015 IEEE 10th Conf. Ind. Electron. Appl. 123–8

Chen X, Gu Q, Qi J and Chen D 2010 Applying Particle Swarm Optimization to Pairwise Testing Proc. - Int. Comput. Softw. Appl. Conf. 107–16

Wu H, Nie C, Kuo F C, Leung H and Colbourn C J 2015 A Discrete Particle Swarm Optimization for Covering Array Generation IEEE Trans. Evol. Comput. 19 575–91

Zamli K Z, Klaib M F J, Younis M I, Isa N A M and Abdullah R 2011 Design And Implementation Of A T-Way Test Data Generation Strategy With Automated Execution Tool Support Inf. Sci. (Ny). 181 1741–58

H Y O and Kamal Z Z 2016 Development of interaction test suite generation strategy with input-output mapping supports Sci. Res. Essays 6 3418–30

Wang Z and He H 2013 Generating variable strength covering array for combinatorial software testing with greedy strategy J. Softw. 8 3173–81

Othman R R, Khamis N and Zamli K Z 2014 Variable Strength T-Way Test Suite Generator With Constraints Support Malaysian J. Comput. Sci. 27 204–17

Lin J, Luo C, Cai S, Su K, Hao D and Zhang L 2016 TCA: An Efficient Two-Mode Meta-Heuristic Algorithm for Combinatorial Test Generation Jinkun Proc. - 2015 30th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2015 494–505

Ahmed B S, Abdulsamad T S and Potrus M Y 2015 Achievement of Minimized Combinatorial Test Suite for Configuration-aware Software Functional Testing Using The Cuckoo Search algorithm Inf. Softw. Technol. 66 13–29

Yang X S and Deb S 2009 Cuckoo search via Lévy flights 2009 World Congr. Nat. Biol. Inspired Comput. NABIC 2009 - Proc. 210–4

Nasser A B, Sariera Y A, Alsewari A R A and Zamli K Z 2016 Assessing Optimization Based Strategies for t-way Test Suite Generation: The Case for Flower-based Strategy Proc. - 5th IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2015 150–5

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