T-way Test Suite Generation Strategy based on Ant Colony Algorithm to Support T-way Variable Strength

N Ramli¹, R R Othman¹, R Hendradi², I Izsaidy¹

¹School of Computer and Communication Engineering, Universiti Malaysia Perlis, Malaysia
²Faculty of Sciences and Technology, Universitas Airlangga, 60115, Surabaya Jawa Timur, Indonesia

E-mail: nuraminah@unimap.edu.my

Abstract. T-way test suite generation strategy based on Ant Colony algorithm (TTSGA) has been developed to support t-way variable strength testing which tackles exhaustive testing issues. It employs the ant colony optimization algorithm to generate near-optimal number of test suite size. Even though the test suite size is smaller than exhaustive testing, the strategy covers every possible combination of interacting parameters. The strategy has been evaluated by using benchmarked experiments. Results obtained were compared with other existing strategies that support variable strength. It was found that TTSGA produces comparable results with other existing strategies especially for higher strength configurations. Two non-parametric tests, which are Wilcoxon Rank and Friedman test, have been conducted to analyze the results statistically between TTSGA and HSS as only both strategies have complete experiments results. Although the results shows that there is no significant difference of test suite size among them, TTSGA is in the first rank in the Friedman test.

Keywords : ant colony optimization algorithm, t-way testing, test suite generation, variable strength

1. Introduction

Software testing plays important activity in any software development project. It could assure the quality of the software being produced [1]. Testing techniques are used to assist in producing the most suitable set of test cases. Number of test cases reflects time to run the test cases [2]. Indirectly, it will also affect cost of the software development. Because of that, choosing the suitable test cases is very important to ensure a minimal number of test cases to be run.

By implementing exhaustive testing (test all combinations of input parameters) towards a system that consists of one 2-value parameter and three 3-value parameters, 54 test cases are required to be tested by software tester (i.e. 2x3x3x3). The number of test cases seems to be small. However, if we have 100 parameters with 2 values, number of test cases is 2100. It is impossible to be practiced because the size is too large [3]–[6]. Therefore, many testing techniques are used to overcome the problems.

T-way (t represents strength) testing is one type of testing techniques existed to generate test cases.
It concerns on interactions of input parameters that will cover the required interaction strength at least once [7], [8]. The rational of t-way is due to most of the faults detected are from interactions of some parameters rather that by each parameters [9], [10]. T-way testing started with pairwise testing (i.e. the value of $t = 2$) or known as uniform strength because the input parameters are interacted uniformly. The need for $t$ greater than 2 is desired to detect faults [11], [12]. However, in some situation, increasing value of $t$ might be expensive [13]–[15]. Therefore, multi strength of input parameter interactions becomes reasonable. These situations lead to the emergence of variable strength which provides more flexibility to software tester to set the strength and it is in accordance to the real application systems [16].

There are several numbers of existing t-way variable strength strategies. Studies reported that 6 variable strength strategies have been developed from 2010 until the first quarter of 2017 [17]. Basically, those strategies are categorized as computational or metaheuristic search technique. As far as optimality of covering all interactions with minimal test suite size is concerned, metaheuristic search technique is proven to generate smaller test suite size than computational strategy [18]. In addition, metaheuristic search technique is flexible as it can be used with different input factors and level. This is why it is more suitable for the real system scenarios [19]. Example of the metaheuristic strategies are Harmony Search Strategy [18], Cuckoo Search [19], Discrete Particle Swarm Optimization [20], VS- Particle Swarm Test Generator [21] and Event Driven Input Sequence Test Case – Simulated Annealing (EDISTC-SA) [22]. Nevertheless, there are many strategies published before 2010 such as Simulated Annealing [16], Ant Colony System [23] and Genetic Algorithm [24]. Whereas computational strategies that support variable strength are such as Density [25], ParaOrder [25], IPOG [9] and General Variable Strength with Constraints (GVSCONST) [26].

Even though many strategies have been developed, due to the t-way test suite generation fall under NP-hard problems [19][6], [27], new strategies that concern on near optimum test case generation will emerge. Driven by the issue, this paper introduces a design and implementation of a new strategy that supports t-way variable strength, T-way Test Suite Generation Strategy based on Ant Colony Algorithm (TTSGA). The strategy embeds an optimization algorithm, Ant Colony (ACO) algorithm to assist in producing a minimal test suite size. The algorithm is inspired by a colony of ants travel to search for the shortest path to the food source. The algorithm can be used for many optimization problem [28]. The algorithm also has been used by other researchers in t-way testing and gives promising results. However, the research focuses on small interaction strength and small configuration system [24]. The new strategy, TTSGA is developed to support higher interaction strength and configuration system.

The rest of this paper is structured as follows. Section 2 describes the background study on t-way variable strength, while Section 3 describes the proposed strategy, TTSGA. Meanwhile, Section 4 explains how the experiment is being implemented. Section 5 presents results and discussion. Finally, Section 6 concludes and summarizes the paper.

2. Related Works

2.1. T-way Variable Strength

Covering arrays (CA) is used by t-way testing to represent mathematical notations. It consists of all t-way combinations where each row is a test while each column is a parameter [10]. For variable strength covering array, it consists of covering arrays and its subsets of covering arrays that have different strength [14]. CA for t-way variable strength can be found in equation (1) is adapted from [16].

$$P = VCA(N, t, C, S)$$

In the equation, $N$ is the final test suite size and $t$ is the dominant interaction strength. $C$ is the value
configuration and can be represented as $V_0^{p_0}, V_1^{p_1}, \ldots, V_n^{p_n}$ whereas $p_n$ is a parameter with $V_n$ values. $S$ is the multi-set of disjoint covering array with strength larger than $t$ as given in equation (2).

$$S = CA (N, ts, Cs)$$

In the equation is final test suite size, $ts =$ interaction strength and $Cs$ is value configuration can be represented as $V_0^{p_0}, V_1^{p_1}, \ldots, V_n^{p_n}$. $p_n$ is parameter with $V_n$ values. This notation is used widely in any t-way study.

2.2. Case Study

Consider Course Registration System as an example of software testing situation. There are four types of courses, namely university, major, minor and elective course. Students are required to register one course offered by each type of courses. Table 1 shows the types of courses and courses offered. It consists of one 2-value parameter for University Course and three 3-value parameters for Major Course, Minor Course and Elective Course. University Course offers English Language and Thinking Skills. Students may choose Computer Programming, Computer Architecture or Organization and Data Structures for their Major Course, while Principles of Software Engineering, Computer Networking and Data Structures for their Minor Course. Besides that, students also may register Internet of Things, Big Data or Technopreneurship and E-Commerce Application as their Elective Course.

| University Course | Major Course | Minor Course | Elective Course |
|-------------------|--------------|--------------|-----------------|
| English Language (EL) | Computer Programming (CP) | Principles of Software Engineering (PSE) | Internet of Things (IoT) |
| Thinking Skills (TS) | Computer Architecture & Organization (CAO) | Computer Networking (CN) | Big Data (BD) |
| - | Data Structures (DS) | Human-Computer Interaction (HCI) | Technopreneurship & E-Commerce Application (TEA) |

The input parameters (i.e. University, Major, Minor and Elective Course) have been set to have strength of two and input parameters University, Major and Minor Course has been set to the strength of three. CA for this configuration is $VCA (N, 2, 3^1 3^2, CA(3,2^3 3^3))$. Figure 1 demonstrates tuples generated by the strength of two and three. The first six tables show interaction of every two parameters. The last table illustrates interaction of three input parameters: University, Major and Minor course. Then, any repeating tuples is joined and discarded from the table. Finally, the final test suite is illustrated in Table 2.

By implementing exhaustive testing (test all combinations of input parameters) towards Course Registration System, 54 test cases are required to be tested by software tester (i.e. $2x3x3x3$). However, t-way variable strength testing only generates 18 test cases. It proves that t-way testing can reduce more than 50 per cent of the test suite size.
**Figure 1.** Tuples generated for VCA (N, 2, $2^{1,3^3}$, CA($3,2^{1,3^3}$))

### t = 2

| University Course | Major Course | Minor Course |
|-------------------|--------------|--------------|
| EL                | CP           | PSE          |
| EL                | CAO          | CN           |
| EL                | DS           | HCl          |
| TS                | CP           | PSE          |
| TS                | CAO          | CN           |
| TS                | DS           | HCl          |

| University Course | Elective Course |
|-------------------|-----------------|
| EL                | IoT             |
| EL                | BD              |
| EL                | TEA             |
| TS                | IoT             |
| TS                | BD              |
| TS                | TEA             |

### t = 3

| University Course | Major Course | Minor Course |
|-------------------|--------------|--------------|
| EL                | CP           | PSE          |
| EL                | CP           | CN           |
| EL                | CP           | HCl          |
| EL                | CAO          | PSE          |
| EL                | CAO          | CN           |
| EL                | CAO          | HCl          |
| EL                | DS           | PSE          |
| EL                | DS           | CN           |
| EL                | DS           | HCl          |
| TS                | CP           | PSE          |
| TS                | CP           | CN           |
| TS                | CP           | HCl          |
| TS                | CAO          | PSE          |
| TS                | CAO          | CN           |
| TS                | CAO          | HCl          |
| TS                | DS           | PSE          |
| TS                | DS           | CN           |
| TS                | DS           | HCl          |

| University Course | Elective Course |
|-------------------|-----------------|
| EL                | IoT             |
| EL                | PSE             |
| EL                | BD              |
| EL                | TEA             |
| CN                | IoT             |
| CN                | BD              |
| CN                | TEA             |
| HCl               | IoT             |
| HCl               | BD              |
| HCl               | TEA             |
Table 2. T-way variable strength test suite

| University | Major Course | Minor Course | Elective Course |
|------------|--------------|--------------|-----------------|
| EL         | CP           | PSE          | IoT             |
| EL         | CP           | CN           | BD              |
| EL         | CP           | HCI          | TEA             |
| EL         | CAO          | PSE          | IoT             |
| EL         | CAO          | CN           | BD              |
| EL         | CAO          | HCI          | TEA             |
| EL         | DS           | PSE          | IoT             |
| EL         | DS           | CN           | BD              |
| EL         | DS           | HCI          | TEA             |
| TS         | CP           | PSE          | IoT             |
| TS         | CP           | CN           | TEA             |
| TS         | CP           | HCI          | BD              |
| TS         | CAO          | PSE          | TEA             |
| TS         | CAO          | CN           | BD              |
| TS         | CAO          | HCI          | IoT             |
| TS         | DS           | PSE          | BD              |
| TS         | DS           | CN           | IoT             |
| TS         | DS           | HCI          | TEA             |

3. Methodology

The TTSGA has been developed to support variable strength in generating t-way test suite. In order to find the best test cases, metaheuristic search technique is adopted to the strategy. Metaheuristic is chosen because it has been proven to solve many optimization problem [29]. Although many metaheuristic algorithms have been used in t-way testing, Ant Colony family has proven to give competitive results. (i.e. Ant Colony System algorithm for variable strength [23] and Ant Colony algorithm for uniform strength [24]). It becomes a challenge to adopt Ant Colony Optimization in the strategy to see its performance for higher strength and configurations.

Figure 2 shows the TTSGA framework and its components. There are three main components in the strategy. The components are VS-Tuple Generator, Search Space Generator and Test Case Generator.

![Figure 2. TTSGA Framework](image-url)
VS-Tuples Generator is used to generate tuples. Parameters, values and its respective strength are the input to generate tuples. The strength for every parameter interactions may vary. VS-Tuples Generator generates tuples based on strength and input parameters assigned for that strength. The generator will produce a list of tuples that will be used in producing the best test case.

Next, Search Space Generator works to generate route. The route consists of nodes and paths from each node. Node represents inputs parameter, while path denotes the input parameter values. The last node is Food node. It is a dummy node to end the last path. Agents will travel along the route. Information (i.e. heuristic and pheromone) will help agents in choosing the best route.

The last component is Test Case Generator. Its purpose is to generate test cases. This component uses route generated by Search Space Generator and VS Tuple List generated by VS-Tuple Generator. ACO algorithm is embedded in this component to generate the best test cases. Figure 3 shows the algorithm for test case generator.

\[
\eta_{i,j} = \frac{E_{i,\text{max}} - E_{i,j} + 1}{E_{i,\text{max}} - E_{i,\text{min}} + 1}
\]

In the equation, \(E_{i,j} = \) number of test cases in the interaction set, \(E_{i,\text{max}} = \max_{1 \leq j \leq \text{path} \cdot i} \{E_{i,j}\}\) and \(E_{i,\text{min}} = \min_{1 \leq j \leq \text{path} \cdot i} \{E_{i,j}\}\).

The ants will be set at the first node. Based on value of the \(q^0\) and other value initialized earlier, each ant will travel from one node to another until food node. In order to travel, the ant needs to choose the best path in every node. The ant will then explore a new path or exploit a similar path as the previous ant by using Route Selection Rule. A probability \(q_0 (0 \leq q \leq 1)\) is used to compare with a random variable, \(q\) that is uniformly distributed in \([0,1]\). If \(q \leq q_0\), ant is exploiting its path and calculated by equation (4).
\[ p_{i,j} = \arg\max_{1 \leq h \leq l_i} \{[\tau_{i,h}]^\alpha[\eta_{i,h}]^\beta \} \] (4)

In the equation, \(l_i\) is the number of path for node \(i\), \(\tau_{i,h}\) is the total amount of pheromone for all path of the node and \(\eta_{i,h}\) is the total heuristic value for all path of the node. Nevertheless, if \(q > q_0\), ant is exploring new path and calculated by equation (5).

\[ p_{i,j} = \frac{[\tau_{i,j}]^\alpha[\eta_{i,j}]^\beta}{\sum_{h=1}^l[\tau_{i,h}]^\alpha[\eta_{i,h}]^\beta} \] (5)

In the equation, \(\tau_{i,j}\) is the amount of pheromone deposited at the path, \(\eta_{i,j}\) is the heuristic value at the path, while \(\tau_{i,h}, \eta_{i,h}\) and \(l_i\) have the same functions as Equation (4).

Once the ant reaches the food node, it updates the pheromone value. The purpose of updating the pheromone value is to inform other ants that it chose that path. The chosen path will be evaluated by fitness function to find the best test case among other ants. Equation for fitness function is as in equation (6).

\[ \text{fit}(s_i) = \sum_{s=0}^{bs} w_s \] (6)

In the equation, \(w_s\) is the number of interaction covered by current test but not covered by the previous test and \(s\) is the interactions. Path with the highest fitness function value is selected as the best test case and will update the pheromone as in equation (7).

\[ \tau_{i,j} = \begin{cases} (1 - \rho)\tau_{i,j} + \rho \Delta \tau_{i,j}^b, & \text{if } e_{i,j} \in \text{test}^b, \\ \tau_{i,j} & \text{otherwise} \end{cases} \] (7)

In the equation, \(\rho\) is the pheromone decay, \(e_{i,j}\) is the path \(i,j\), test\(^b\) is the best test and \(\Delta \tau_{i,j}^b\) is the fitness function value for the path \(i,j\). The best test case is compared to the VS-Tuple List and any related tuples is removed from the list. The best test case is then placed in the test suite. This process continues until VS-Tuple List is empty.

4. Experiments

TTSGA has been evaluated by conducting experiments. Six benchmark system configurations involved in the experiments as in [18]. Each configuration consists of several other sub-configurations. The configurations are VCA (N, 2, 3\(^{15}\), \{C\}), VCA (N, 2, 4\(^5\)\(^6\), \{C\}), VCA (N, 2, 3\(^{20}\) 10\(^7\), \{C\}), VCA (N, 3, 3\(^{15}\), \{C\}), VCA (N, 3, 4\(^{1}\)\(^7\)\(^2\), \{C\}) and VCA (N, 2, 10\(^1\) 9\(^1\) 8\(^1\) 7\(^1\) 6\(^1\) 5\(^1\) 4\(^1\) 3\(^1\) 2\(^1\), \{C\}). Each sub-configuration is conducted 10 independent run. The running environment for the experiment consists of a desktop PC with Windows 7, 3.2 GHz Core 2 Duo CPU, 4 GB of RAM. The TTSGA strategy is coded in Java. Parameters used by TTSGA are as in Table 3.

| 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   |
| Stale period              | 5     |
These experiments were conducted to find the minimum test suite size for each configuration. Next, the results obtained are compared with other existing variable strength strategies. The ability of TTSGA to produce better test suites sizes is evaluated.

5. Results and Discussion

Table 4 – 9 present results for all system configurations conducted by TTSGA as well as other existing variable strength strategies. The bold font for each experiment illustrates the smallest test suite size (i.e. the best test suite size). Cells with NA (not available) symbol means that no result is published in any literature. While cells with NS (not supported) present that the strategy does not support the specified configurations.

Table 4 demonstrates results for system configuration VCA (N, 2, 315, {C}). SA produces the minimum results for all sub-configurations except CA(4,3), CA(4,3), CA(4,3), CA(5,3), CA(5,3), CA(6,3) and CA(6,3). TTSGA gives a comparable results with other strategies and produces minimum results for all sub-configurations except CA(4,34), CA(4,35), CA(4,37), CA(5,35), CA(5,37), CA(6,36) and CA(6,37).

The minimum number of test suite size for system configuration VCA (N, 2, 435362, {C}) produced by 12 strategies are depicted in Table 5. TTSGA manages to produce minimum test suite size for 10 out of 15 sub-configurations. It shows that this strategy performs well for this type of system configuration. From the table, HSS and VS-PSTG also performed well in this configuration, while PICT is the worst of all.
Another configuration that has been experimented is VCA (N, 2, 3^{20}, \{C\}) and the results are shown in Table 6. TTSGA, HSS, VS-PSTG and TVG produced the minimum results for last three sub-configurations. Meanwhile, SA produced the minimum results for the first three sub-configurations.

### Table 6. Test suite for VCA (N, 2, 3^{20}, \{C\})

| C | TTSGA | HSS | VS-PSTG | IPOG | WHITCH |
|---|-------|-----|---------|------|--------|
| Ø | 102   | 106 | 102     | 102  | 101    |
| CA(3,3^{20}) | 107  | 109 | 105     | 102  | 100    |
| CA(3,5^{20}) | 493  | 450 | 481     | 416  | 396    |
| CA(4,3^{20}) | 270  | 270 | 270     | 270  | 270    |
| CA(5,3^{20}) | 2700 | 2700| 2700    | 2700 | 2700   |
| CA(6,3^{20}) | 8100 | 8100| 8100    | 8100 | 8100   |

HSS produces the minimum test suite size for most of the sub-configurations for VCA (N, 3, 3^{15}, \{C\}) as displayed in Table 7. However, TTSGA gives comparable results with a slight difference with HSS. Meanwhile, TTSGA produces the minimum test suite size for sub-configurations CA(6,3), CA(7,3), CA(4,3), CA(4,5), CA(5,3), CA(6,3) and CA(6,3). PICT produces the worst test suite size for all sub-configurations but no result is available for CA(4,34)3 CA(3,3), CA(5,3), CA(6,3), CA(7,3), CA(6,3)² CA(3,3), CA(4,3) and CA(3,3) CA(4,3) CA(5,3). For all sub-configurations, no result is available for Density and ParaOrder in the literature while ACS and SA strategy do not support the sub-configurations.

### Table 7. Test suite for VCA (N, 3, 3^{15}, \{C\})

| C | TTSGA | HSS | VS-PSTG | ACS | SA | PICT | TVG | Density | ParaOrder | IPOG | WHITCH |
|---|-------|-----|---------|-----|----|------|-----|---------|-----------|------|--------|
| 0 | 84    | 75  | 75      | NS  | NS | 83   | 84  | NA      | NA        | 82   | 75     |
| CA(4,3) | 93   | 87  | 91      | NS  | NS | 1507 | 93  | NA      | NA        | 87   | 129    |
| CA(4,3)² | 98   | 90  | 91      | NS  | NS | 19749| 97  | NA      | NA        | 91   | 183    |
| CA(4,3)³ | 99   | 91  | 91      | NS  | NS | 531441| 97  | NA      | NA        | 106  | 237    |
| CA(4,3)⁴ | 100  | 89  | 90      | NS  | NS | 98   | NA  | NA      | NA        | 106  | 237    |
| CA(3,3) | 244  | 243 | 243     | NS  | NS | 5366 | 244 | NA      | NA        | 243  | 273    |
| CA(5,3)² | 248  | 243 | 245     | NS  | NS | 177300| 245 | NA      | NA        | 250  | 459    |
the minimum test suite size all sub-configurations except for CA(4,4 34) CA(5,3 22). PICT produces configuration CA(5,4 1 34) CA(5,3 3 22) while results for sub-configuration CA(6,4 1 35) is not the worst test suite size except for an empty sub-configuration and no result is stated for sub-configurations turn are at least the same as TTSGA, HSS and VS-PSTG. Interestingly, IPOG produces Apparently, TVG and IPOG provide comparable results to other strategies with over half of sub-configurations support the configuration, while results for Density and ParaOrder are not available.

| C    | TTSGA | HSS | VS-PSTG | ACS | SA | PICT | TVG | Density | ParaOrder | IPOG | WHITCH |
|------|-------|-----|---------|-----|----|------|-----|---------|-----------|------|--------|
| Ø    | 65    | 66  | 65      | NS  | NS | 72   | 70  | NA      | NA        | 73   | 112    |
| CA(4,4 34) | 84  | 108 | 108     | NS  | NS | 1377 | 111 | NA      | NA        | 108  | 193    |
| CA(4,4 35) | 87  | 108 | 108     | NS  | NS | 17496| 112 | NA      | NA        | 108  | 253    |
| CA(4,4 36) | 324 | 324 | 324     | NS  | NS | NA   | 324 | NA      | NA        | 326  | 497    |
| CA(4,4 37) | 105 | 133 | 136     | NS  | NS | 1500 | 141 | NA      | NA        | 149  | 217    |
| CA(4,4 38) | 164 | 170 | 171     | NS  | NS | 1547 | 183 | NA      | NA        | 207  | 226    |
| CA(4,5 1 37) | 134 | 122 | 136     | NS  | NS | 1500 | 141 | NA      | NA        | 149  | 307    |
| CA(4,5 2 3) | 243 | 324 | 324     | NS  | NS | 3586 | 325 | NA      | NA        | 324  | 369    |
| CA(5,3 3 2) | 244 | 324 | 324     | NS  | NS | 325  | NA  | NA      | NA        | 324  | 482    |
| CA(6,4 3 3) | 972 | 972 | NS      | NS  | NS | NA   | NA  | NA      | NA        | NS   | NS     |

Table 8 presents results for the system configuration VCA (N, 3, 4\(^{1} 3\^{7} 2^{2}, \{C\})). TTSGA produces the minimum test suite size all sub-configurations except for CA(4,4 34) CA(5,3 22). PICT produces the worst test suite size except for an empty sub-configuration and no result is stated for sub-configuration CA(5,4 33) CA(5,3 22) while results for sub-configuration CA(6,4 33) is not available. Similar to the previous configuration, no result is stated for ACS and SA as they do not support the configuration, while results for Density and ParaOrder are not available.

Table 8. Test suite for VCA (N, 3, 4\(^{1} 3\^{7} 2^{2}, \{C\})

Results for configuration VCA (N, 2, 10\(^{1} 9\^{1} 8\^{1} 7\^{1} 6\^{1} 5\^{1} 4\^{1} 3\^{1} 2^{1}, \{C\}) is depicted in Table 9. TTSGA, HSS and VS-PSTG produce the same number of test suite size for all sub-configurations except for CA(3,10\(^{1} 9\^{1} 8\^{1} 7\^{1})). However, VS-PSTG produces slightly more number of test cases. Apparently, TVG and IPOG provide comparable results to other strategies with over half of sub-configurations turn at least the same as TTSGA, HSS and VS-PSTG. Interestingly, IPOG produces the minimum test suite size for sub-configuration CA(3,4\(^{1} 3\^{1} 2^{1})>. In general, PICT produces the worst test suite size except for empty sub-configuration and not available results for sub-configuration CA(3,10\(^{1} 9\^{1} 8\^{1}) CA(3,7\^{1} 6\^{1} 5^{1}), CA(3,10\(^{1} 9\^{1} 8\^{1}) CA(6,7\^{1} 6\^{1} 5^{1} 4\^{1} 3\^{1} 2^{1}), CA(3,10\(^{1} 9\^{1} 8\^{1}) CA(3,7\^{1} 6\^{1} 5^{1})
CA(3,4 1 3 1 2 1) and CA(6,7 1 6 1 5 1 4 1 3 1 2 1). No result is available for ACS, SA, Density and ParaOrder for this configuration.

**Table 9.** Test suite for VCA (N, 2, 10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1, {C})

| C          | TTSGA | HSS | VS-PSTG | ACS | SA | PICT | TVG | Density | ParaOrder | IPOG | WHITCH |
|------------|-------|-----|---------|-----|----|------|-----|---------|-----------|------|---------|
| Ø          | 94    | 94  | 97      | NA  | NA | 102  | 99  | NA      | NA        | 91   | 119     |
| CA(3,10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1) | 720   | 720 | 720     | NA  | NA | 31256| 720 | NA      | NA        | 720  | 465     |
| CA(3,7 1 6 1 5 1 4 1 3 1 2 1) | 96    | 94  | 97      | NA  | NA | 2397 | 99  | NA      | NA        | 91   | 140     |
| CA(3,10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1) | 720   | 720 | 720     | NA  | NA | 22878| 784 | NA      | NA        | 772  | 806     |
| CA(3,7 1 6 1 5 1 4 1 3 1 2 1) | 720   | 720 | 720     | NA  | NA | 720  | NA  | NA      | NA        | 720  | 947     |
| CA(3,10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1) | 5040  | 5040| 5040    | NA  | NA | 19515| 210 | NA      | NA        | 5041 | 5803    |
| CA(3,7 1 6 1 5 1 4 1 3 1 2 1) | 720   | 720 | 720     | NA  | NA | 2397 | 99  | NA      | NA        | 720  | 968     |
| CA(3,10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1) | 120   | 120 | 120     | NA  | NA | 1200 | 123 | NA      | NA        | 142  | 237     |
| CA(5,10 1 9 1 8 1 7 1 6 1 5 1 4 1 3 1 2 1) | 2160  | 2160| 2160    | NA  | NA | 124157| 2160| NA      | NA        | 2160 | 2276    |
| CA(3,4 1 3 1 2 1) | 5040  | 5040| 5040    | NA  | NA | 720  | NA  | NA      | NA        | 5041 | 5157    |

6. Analysis

Two type of statistical analysis tests, Wilcoxon Rank Test and Friedman Test have been conducted to see TTSGA’s performance in producing test suite size. Both tests need to be performed as the results are not normally distributed [30]. The significant difference of produced test suite size between TTSGA and other strategies is analyzed with 95 percent confident level. The null hypothesis is rejected if the significant value is less than 0.5. It is defines as there is a significant difference of test suite size between TTSGA and other strategies.

84 results of experiment for variable strength as in Table 4 to 9 were combined to execute the tests. However, the statistical analysis can only be executed between TTSGA and HSS. It was because the other strategies did not perform complete experiments. The analysis in Table 10 shows that the null hypothesis cannot be rejected because the significant value is greater than 0.05. Therefore, there is no significant difference between TTSGA and HSS in terms of producing test cases. Nevertheless, Friedman Test mean rank shows that TTSGA is in the first rank and beats the HSS strategy.

**Table 10.** Statistical analysis for variable strength

| Strategy  | Wilcoxon Rank Test | Friedman Test |
|-----------|--------------------|---------------|
|           | Pair   | Ties | Significant value | Mean Rank |
| TTSGA     | 1.49   |      |                  |           |
| HSS       | HSS - TTSGA     | 33   | 0.336            | 1.51      |

7. Conclusion

In this paper, a strategy, TTSGA that supports t-way variable strength test suite generation was developed based on ACO algorithm. TTSGA strategy is concerned in optimization of test suite size generation. Evaluation on this strategy has been made based on benchmarked experiments to
determine the ability of this strategy in test suite generation and compared with other existing variable strength strategies. It was found that TTSGA managed to produce similar or better results for configuration VCA (N, 2, 45^2, \{C\}), VCA (N, 3, 4^3 \cdot 2^2, \{C\}) and VCA (N, 2, 10^1 \cdot 9^1 \cdot 8^1 \cdot 7^1 \cdot 6^1 \cdot 5^1 \cdot 4^1 \cdot 3^1 \cdot 2^1, \{C\}). The results are comparable with other strategies especially for higher configuration (i.e. strength > 3 and higher parameter values). Statistical analysis shows that, even though there is no significant difference between TTSGA and HSS in terms of producing test cases, Friedman Test mean rank shows that TTSGA is in the first rank and beats the HSS strategy. For future works, seeding and constraints will be added into the strategy.

8. References

[1] S. F. Ahmad and P. Abd Samat, “Extraction Cost of Quality and Testing in Software Project,” in 2018 IEEE Conference on e-Learning, e-Management and e-Services, IC3e 2018, 2019, pp. 109–115.
[2] S. Vegas and V. R. Basili, “A Characterization Schema for Software Testing Techniques,” Empir. Softw. Eng., vol. 10, no. 4, pp. 437–466, 2005.
[3] Xiao Qu and M. B. Cohen, “A Study in Prioritization for Higher Strength Combinatorial Testing,” in 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, 2013.
[4] K. Z. Zamli, F. Din, B. S. Ahmed, and M. Bures, “A Hybrid Q-learning Sine-Cosine-Based Strategy for Addressing the Combinatorial Test Suite Minimization Problem,” PLoS One, vol. 13, no. 5, pp. 1–29, 2018.
[5] M. H. M. Zabil, K. Z. Zamli, and K. C. Lim, “Evaluating Bees Algorithm for Sequence-based T-way Testing Test Data Generation,” Indian J. Sci. Technol., vol. 11, no. 4, pp. 1–20, 2018.
[6] A. R. A. Alsewari, R. Poston, K. Z. Zamli, M. Balfaqih, and K. S. Aloufi, “Combinatorial test list generation based on Harmony Search Algorithm,” J. Ambient Intell. Humaniz. Comput., 2020.
[7] R. R. Othman and K. Z. Zamli, “T-Way Strategies and Its Applications for Combinatorial Testing,” Int. J. New Comput. Archit. Their Appl., vol. 1, no. 2, pp. 459–473, 2011.
[8] M. I. Younis, K. Z. Zamli, and R. R. Othman, “Effectiveness of the Cumulative vs. Normal Mode of Operation for Combinatorial Testing,” in 2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2010), 2010, pp. 350–354.
[9] 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 International Symposium and Workshop on Engineering of Computer Based Systems, 2007, pp. 549–556.
[10] D. R. Kuhn, R. N. Kacker, and Y. Lei, Introduction to Combinatorial Testing. Boca Raton: CRC Press, 2013.
[11] R. Kuhn, Y. Lei, and R. Kacker, “Practical Combinatorial Testing : Beyond Pairwise,” IEEE IT Professional, vol. 10, no. 3, pp. 19–23, 2008.
[12] R. Kuhn, R. Kacker, Y. Lei, and J. Hunter, “Combinatorial Software Testing,” Computers, vol. 42, no. 8, pp. 94–96, 2009.
[13] Y. Cemal, S. Fouché, M. B. Cohen, P. Adam, D. Gulsen, and K. Ugur, “Moving Forward with Combinatorial Interaction Testing,” Computers, no. February, pp. 37–45, 2014.
[14] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, “Constructing Test Suites for Interaction Testing,” in 25th International Conference on Software Engineering, 2003, pp. 38–48.
[15] S. Esfandyari and V. Rafe, “A Tuned Version of Genetic Algorithm for Efficient Test Suite Generation in Interactive T-way Testing Strategy,” Inf. Softw. Technol., vol. 94, no. January 2017, pp. 165–185, 2018.
[16] 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 International Computer Software and Applications Conference. COMPAC 2003, 2003, pp. 413–418.
[17] N. Ramli, R. R. Othman, Z. I. Abdul Khalib, and M. Jusoh, “A Review on Recent T-way Combinatorial Testing Strategy,” MATEC Web Conf., vol. 140, no. 1016, pp. 1–6, 2017.
[18] A. 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.*, vol. 54, no. 6, pp. 553–568, 2012.

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

[20] 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.*, vol. 19, no. 4, pp. 575–591, 2015.

[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] 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.*, vol. 15, no. 3, pp. 65–71, 2014.

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

[24] T. Shiba, T. Tsuchiya, and T. Kikuno, “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*, 2004, pp. 72–77.

[25] W. Ziyuan, X. Baowen, and N. Changhai, “Greedy Heuristic Algorithms to Generate Variable Strength Combinatorial Test Suite,” in *The Eighth International Conference on Quality Software*, 2008, pp. 155–160.

[26] R. R. Othman, N. Khamis, and K. Z. Zamli, “Variable Strength t-way Test Suite Generator with Constraints Support,” *Malaysian J. Comput. Sci.*, vol. 27, no. 3, pp. 204–217, 2014.

[27] N. Ramli, R. R. Othman, Z. Ishwar, and A. Khalib, “Ant colony algorithm to generate t-way test suite with constraints,” *J. Phys. Conf. Ser. Pap.*, vol. 1529, no. 4, pp. 1–8, 2020.

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

[29] 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.*, vol. 2, pp. 62–65, 2015.

[30] C. Y. Piaw, *Asas Statistik Penyelidikan*, Second. McGraw-Hill Education (Malaysia) Sdn. Bhd., 2013.