Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Input-Output Based Relations T-Way Test Suite Generation Strategy Based on Ant Colony Optimization Algorithm (iTTSGA)

Nuraminah Ramli[1,2*], Rozmie Razif Othman[1,2], Rimuljo Hendradi[3], Hasneeza Liza Zakaria[1,2], Iszaidy Ismail[1,2], Nurul Ain Mohd Zaki[4], Nik Afiqah N. Ahmad Yani[5]

[1] Advanced Computing, Centre of Excellence (CoE), Universiti Malaysia Perlis, Pauh Putra, 02600 Arau, Perlis, Malaysia
[2] Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis, Pauh Putra, 02600 Arau, Perlis, Malaysia
[3] Faculty of Sciences and Technology, Universitas Airlangga, 60115, Surabaya Jawa Timur, Indonesia
[4] School of Geomatics Science and Natural Resources, College of Built Environment (CBE), Universiti Teknologi MARA, Cawangan Perlis, 02600 Arau, Perlis, Malaysia
[5] College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Cawangan Perlis, 02600 Arau, Perlis, Malaysia

ABSTRACT

Input-output based relations (IOR) testing is one of the support interactions in t-way testing (t is referring to the interaction strength). T-way testing is useful to solve the issue of exhaustive testing, which generates a large number of test cases and is impractical to execute because of financial and schedule limitations. Several IOR test suite generation strategies have been put forth in the literature. Although these strategies can produce a sizable IOR test suite, very few of them use the metaheuristic search technique which can produce smaller test suite sizes for uniform and variable strength interactions. Because the T-way test suite generation problem is NP-hard, no strategy can guarantee that it can produce the ideal test suite size for every possible system configuration. Motivated by these challenges, this paper presents an IOR test suite generation strategy based on a metaheuristic algorithm, Ant Colony Optimization (ACO) called iTTSGA. The results of two benchmark experiments for IOR uniform and non-uniform configurations were compared with those of other IOR strategies that have been published. The performance of iTTSGA has been statistically analyzed using the Friedman test and Wilcoxon Sum test. Results show that, except for |R|=10 and the top ranking in the Friedman Test, iTTSGA outperforms other strategies in the majority of uniform configurations. In addition, the iTTSGA performs exceptionally well in non-uniform configuration for |R|=30 and 60 and ranks second in the Friedman test. It demonstrates that iTTSGA can produce test suites with a smaller file size for IOR t-way testing.

Keywords:
T-way testing; input-output based relations; ant colony algorithm; metaheuristic optimization algorithm

1. Introduction

Software testing is a crucial activity in any software development project. Testing activities receive more allocation of resources for software development [1]. It is important because this activity can increase the client's confidence in the software project and ensure compliance with the

[*] *Corresponding author.*
E-mail address: nuraminah@unimap.edu.my

https://doi.org/10.37934/araset.63.1.116

requirements. Testing activities involve generating test cases [2]. Test cases are created based on scenarios, and combinations of inputs within the system under test (SUT). The number of test cases will increase exponentially as the number of functionalities (or inputs) is added to the system and eventually will lead to combinatorial explosion problem where there are too many test cases that need to be executed [1, 2].

Testing methods have therefore been developed to deal with this problem. One testing method that operates by interacting input parameters and their values at specific strengths is known as combinatorial testing (also referred to as t-way testing) [3-5]. As a general rule of thumb, the t-way test suite size is reduced when the interaction strength (i.e., the number of input parameters that are interacted) is low. If the interaction strength value is similar to total input parameters, the t-way test suite produced is identical to exhaustive testing.

Three different types of interaction support, including input-output-based relations (IOR), uniform and variable strength, are offered by t-way testing. In the case of uniform strength interaction, all input parameters are given the same strength, whereas in the case of variable strength interaction, some input parameters are given different interaction strengths. Regarding IOR, the relationships between the SUT's input and output are used to derive the interactions. For detail explanation regarding different type of interactions, readers are suggested to refer to the following papers [4, 6, 7].

Each type of interaction supports has its advantages. As far as test suite size is concern, IOR produces the smallest size of all [8]. Nevertheless, IOR can be beneficial and more effective in generating smaller test suite size if the software tester has information of interactions between parameter inputs and outputs of the system to be tested. Unlike uniform and variable strength, IOR will ignore other unrelated interactions.

Although many existing works have been carried out on uniform [4, 9-15] and variable strength [14, 16-26] very few studies have explored IOR [27]. To date, only a few strategies have been reported in the literature that support IOR [3] (e.g. Density [28] ParaOrder [29], ReqOrder [29], Union [30], Greedy [31], TVG [32], ITTDG [33], AURA [34], DA-RO, DA-FO [35] and CTJ [36]). Interestingly, only a few of the existing IOR based strategies utilize metaheuristic search technique in generating the test suite such as CTJ whereas the others apply computational search technique [37]. Although computational search technique able to generate good results, metaheuristic is able to produce smaller size of t-way test suite for both uniform and variable strength [4, 13, 38, 39]. Besides that, metaheuristic requires only a few or no assumptions which make it adaptable and flexible in many situations. It also has the capability of solving bigger search space [40].

Based on this motivation, this paper presents an IOR test suite generation strategy developed by embedding Ant Colony Optimization (ACO) algorithm, iTTSGA. In addition, generating IOR test suite is challenging problem, known as NP-Hard, meaning no single strategy can claimed to produce the optimum test suite size for all system configurations) [15, 41, 42]. Therefore, research in this area is always open for new contributions. The iTTSGA strategy benefits the IOR t-way testing field with the adoption of metaheuristic ACO algorithm. Besides, this paper presents results of experiments based on benchmarked experiments that can be used to see the performance of each strategy.

To facilitate the discussion, the paper is organized into five sections. Elaboration on the related works and research context are described in the second section. Subsequently, the proposed strategy is discussed thoroughly in the third section. Meanwhile, the fourth section elaborates the evaluation for our strategy. The final section concludes the discussion.

2. Related Works

2.1 Background Research

A College Advisory System will be discussed as an example of IOR testing scenario. The system advises student on which college programs they can apply based on their final examination marks for the selected courses such as Economic, Mathematics, English Language and Information Technology. There are 3 programs offered by the college which are Business Administration, Technoprenuership and Computer Science Program. The system will suggest the students for:

i. Business Administration Program if students pass Economic and English Language courses.
ii. Technopreneurship Program if students pass Economic and Information Technology courses.
iii. Computer Science Program if students pass Mathematics, English Language and Information Technology courses.

Here, marks for all courses are between 0 and 100 inclusive and the passing marks are 50. By using Equivalent Partitioning Technique [43] and ignoring the invalid partition, the input marks for all courses can be grouped into two partitions either passed or failed partition. One value for each partition is selected as the testing value. The value of 50 is for passed partition and 49 for failed partition. The testing input value for each course (or input parameter) is presented in Table 1.

Table 1
College advisory system's input value

Economic / Mathematics / English Language / Information Technology	
Pass	50
Fail	49

To ease the discussion, a symbolic representation as shown in Table 2 will be used to represent input values throughout this section where E, M, G and N represent Economic, Mathematics, English Language and Information Technology respectively. Meanwhile, the lowercase letter and number 1 represent pass, while fail is represented by 2. To exhaustively test all combinations, 16 test cases are required to be conducted as depicted in Table 3.

However, a closer look on the system requirements reveals three distinct outputs with following input-output relationships:

i. Acceptance for Business Administration Programme, F(V), directly related to input for Economics and English Language (i.e. E and G respectively)
ii. Acceptance for Technopreneurship Programme, F(W), directly related to input for Economics and Information Technology (i.e. E and N respectively)
iii. Acceptance for Computer Science Programme, F(X), directly related to input for Mathematics, English Language and Information Technology (i.e. M, G and N respectively)

Table 2
Input parameter values representation

E	M	G	N
e1	m1	g1	n1
e2	m2	g2	n2

Table 3
Exhaustive combination of the SUT

E	M	G	N
e1	m1	g1	n1
e1	m1	g1	n2
e1	m1	g2	n1
e1	m1	g2	n2
e1	m2	g1	n1
e1	m2	g1	n2
e1	m2	g2	n1
e1	m2	g2	n2
e2	m1	g1	n1
e2	m1	g1	n2
e2	m1	g2	n1
e2	m1	g2	n2
e2	m2	g1	n1
e2	m2	g1	n2
e2	m2	g2	n1
e2	m2	g2	n2

The input-output relationships for College Advisory System can be graphically described by Figure 1. From the system requirements with the three outputs mentioned above, the IOR testing for College Advisory System is implemented. Final test cases are produced as in Figure 2. It also shows the three outputs which are f(V), f(W) and f(X)) and the related input values. Symbol (*) indicates a don't care value where it is not needed to cover the output. The final test cases illustrate that the number of test cases produced is smaller than that of exhaustive testing. The reduction is due to IOR testing has removed any repeating test cases (i.e. e1m1g1n1, e2m1g1n2, e1m1g2n1, e2m2g1n2 and e2m2g2n2). Finally, only 10 test cases need to be performed as compared to 16 test cases produced by exhaustive testing.

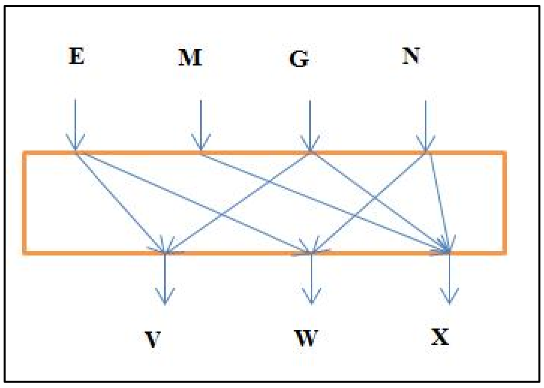


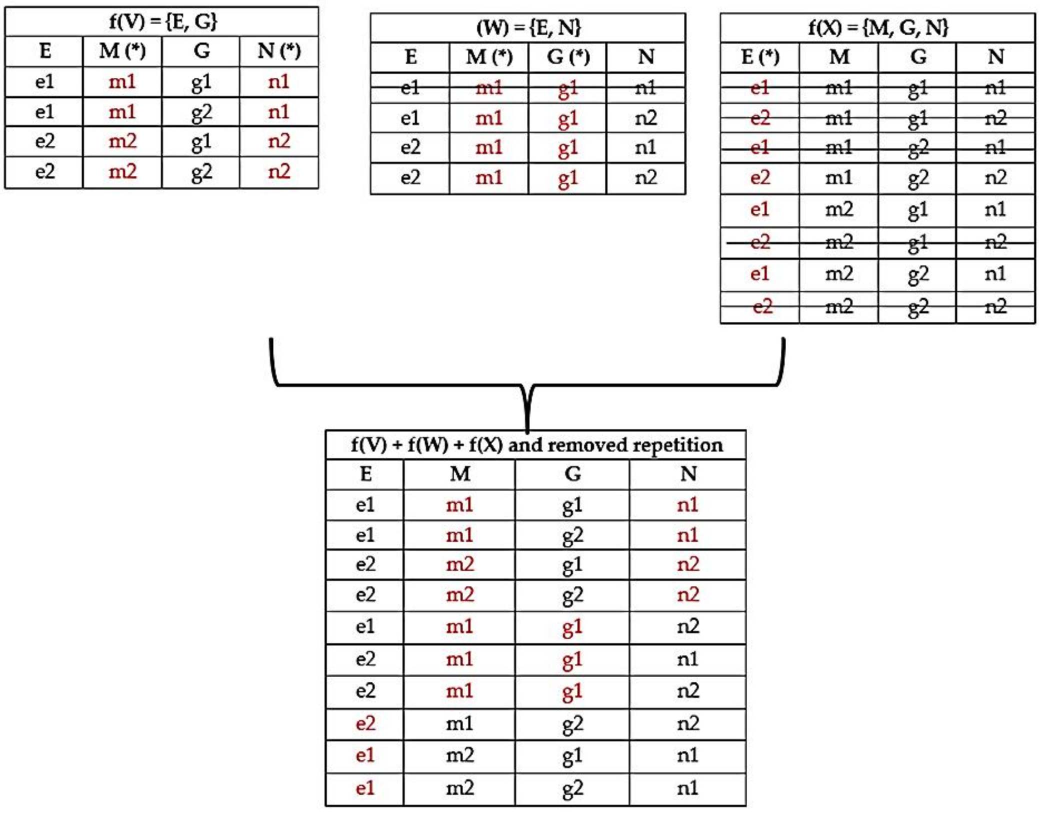
Fig. 1. IOR t-way testing for college advisory system

According to [33], IOR t-way test suite can be mathematically represented using covering array (CA) notation, P, as in Eq. (1).

P = IOR (N, C, R,) (1)

where N is size of final test suite, C is configuration value represented as $V_0^{p_0}$, $V_1^{p_1}$,......., $V_n^{p_n}$ indicating that there are p_0 parameters with V_0 values, p_1 parameters with V_1 values and so on and R is the IOR involve in generating the test suite and |R| refers to the number of defined IOR definition set. Therefore, the representation of test suite in Figure 3 using CA notation, P is as in Eq. (2).

$$P = IOR\left(10, 2^4, \left\{\{E, G\}, \{M, G, N\}, \{E, N\}\right\}\right) \tag{2}$$



f(V) = {E, G}			
E	M (*)	G	N (*)
e1	m1	g1	n1
e1	m1	g2	n1
e2	m2	g1	n2
e2	m2	g2	n2

(W) = {E, N}			
E	M (*)	G (*)	N
~~e1~~	~~m1~~	~~g1~~	~~n1~~
e1	m1	g1	n2
e2	m1	g1	n1
e2	m1	g1	n2

f(X) = {M, G, N}			
E (*)	M	G	N
~~e1~~	~~m1~~	~~g1~~	~~n1~~
~~e2~~	~~m1~~	~~g1~~	~~n2~~
~~e1~~	~~m1~~	~~g2~~	~~n1~~
e2	m1	g2	n2
e1	m2	g1	n1
~~e2~~	~~m2~~	~~g1~~	~~n2~~
e1	m2	g2	n1
~~e2~~	~~m2~~	~~g2~~	~~n2~~

f(V) + f(W) + f(X) and removed repetition			
E	M	G	N
e1	m1	g1	n1
e1	m1	g2	n1
e2	m2	g1	n2
e2	m2	g2	n2
e1	m1	g1	n2
e2	m1	g1	n1
e2	m1	g1	n2
e2	m1	g2	n2
e1	m2	g1	n1
e1	m2	g2	n1

Fig. 2. IOR testing for college advisory system

2.2 Existing Strategies

There are two types of search technique that has been used widely by researchers, namely computational and metaheuristic [44]. As discussed earlier, most of the existing strategies that support IOR apply computational which involves greedy technique.

Schroeder introduced the Union strategy [30], which assigns unrelated IOR parameters a random value at the "don't care" value. This strategy then combines redundant test cases. While it can generate test suites quickly, the resulting size tends to be large. Two years later, the Greedy strategy was proposed. It assigns a do not care value to any parameter values that are unaffected. It selects test cases from the initial test suite that cover the largest number of remaining uncovered interactions. Greedy yields a more optimized test suite size compared to the Union strategy. However, it is worth mentioning that Greedy requires a longer time to generate the test suite [31].

ITTDG works by having a candidate test data list to determine final test data. The strategy compares a parameter at a time with a value that can covers the most uncovered tuples and uses test data weight to be in the final test suite [33].

The AURA strategy [33] consists of Interaction Pair Generation, Test Suite Construction, and Actual Data Mapping algorithms that work together to create the final test suite. Both symbolic values and actual data output are supported by the strategy. It is possible to shorten the time needed to load fault files and analyze parameters by mapping the actual data values. The strategy also incorporates a post-processing method that enables software testers to change the settings in accordance with their preferences, like the number of iterations.

DARO and DAFO are another example of strategies that support IOR. The strategies were developed based on the Density strategy. Both strategies focus on determining the values of factors classified on different criteria. The DA-RO strategy considers the highest coverage requirements to fix the values of factors, while the DA-FO strategy determines the order of factors to fix their values [35].

CTJ is using a different search technique than the other strategies discussed earlier. It embeds Java algorithms and applies a metaheuristic search technique. The Jaya algorithm consists of only two parameters to be controlled (i.e. population size and maximum iterations) and it only depends on global search to search for the best results. Its main strength is to produce test suite size within a short time [36].

Every strategy has its unique strengths in generating an optimal size of test suite. CTJ is the only strategy that utilizes metaheuristic search technique. In contrast, Union, Greedy, ITTDG, AURA, DARO dan DAFO strategies apply greedy search techniques. The utilization of search techniques and algorithms can greatly assist in achieving the best possible results.

2.2 Ant Colony Optimization (ACO) Algorithm

ACO algorithm is inspired by a colony of ant searches for the shortest route from its nest in an attempt to mimic the behavior of an ant colony looking for food. Each ant will lay down a pheromone at each path it chooses, informing other ants in its colony that this path has been chosen. In the end, the ants use the greatest quantity of pheromones deposited by their colony to create the shortest routes from the nest to the food source. Details on ACO applied in the strategy are described in Section 3.

3. Proposed Strategy

iTTSGA adopts a metaheuristic algorithm, ACO. It has solved a variety of optimization issues, including those pertaining to software testing. Additionally, ACO has been applied in the t-way testing field. The first strategy named as ACO strategy supports uniform strength [14] while Ant Colony System strategy supports variable strength [20]. Nevertheless, iTTSGA is different than both strategies whereby iTTSGA supports other types of t-way, namely IOR. iTTSGA implements Route Selection Rule in order to exploit the same route or explore new route [20]. In our strategy, iTTSGA will explore more new routes than exploit the same route to ensure all routes are covered at least once. Besides that, iTTSGA also introduces a new IOR fitness function that will be explained later in this section.

Figure 3 demonstrates the proposed strategy, iTTSGA strategy which consists of three generators namely Tuples, Search Space and Test Case Generator. Relationship of input parameters and outputs of the SUT are required to trigger the strategy to produce a test suite.

In the Tuples Generator component, the relationship of input parameters and its output are interacted by using IOR t-way testing technique. Each output is called function output. It produces interactions between values of related input parameters or known as tuples. All tuples produced are stored in the IOR Tuples List. Figure 4 presents a structure of IOR Tuples List. The structure contains the interaction of related input parameter values based on function output (i.e. f(n)). The symbol (*) means the input parameter does not have any interaction in the function output. The component's algorithm is illustrated in Figure 5. The algorithm requires function outputs and the related input parameters. For each function output, the algorithm interacts with the related input parameters and its values. This algorithm will produce IOR interactions for each function output, namely IOR interaction set, S.

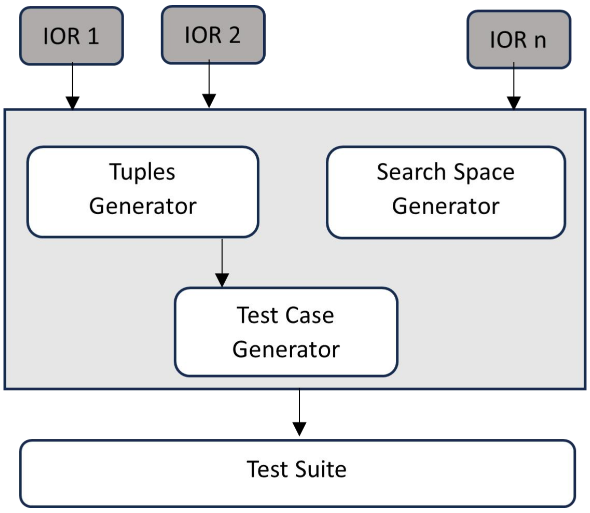
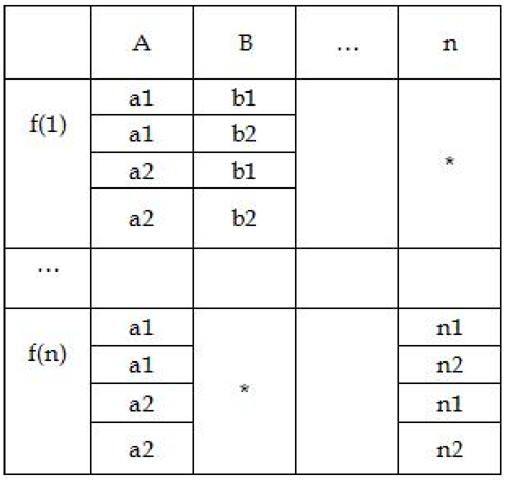
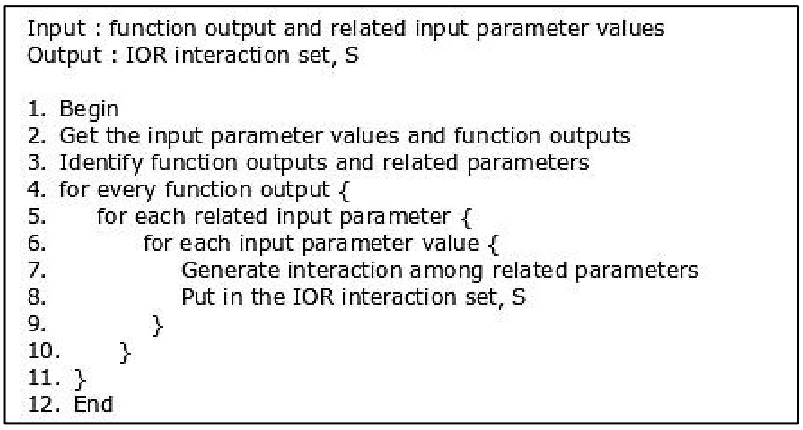


Fig. 3. Illustration of iTTSGA strategy framework



	A	B	…	n
f(1)	a1	b1		*
	a1	b2		
	a2	b1		
	a2	b2		
…				
f(n)	a1	*		n1
	a1			n2
	a2			n1
	a2			n2

Fig. 4. A structure of IOR tuple list



```
Input : function output and related input parameter values
Output : IOR interaction set, S

1. Begin
2. Get the input parameter values and function outputs
3. Identify function outputs and related parameters
4. for every function output {
5.      for each related input parameter {
6.           for each input parameter value {
7.                Generate interaction among related parameters
8.                Put in the IOR interaction set, S
9.           }
10.     }
11. }
12. End
```

Fig. 5. Tuple generator algorithm

While the Tuple Generator is focused on determining the interaction among input parameters, the Search Space Generator component concentrates on generating a path for the ants to travel. This path is created based on the input parameters and the potential values associated with them. Since this strategy applies ACO algorithm, this route corresponds to the ants' route, from a nest to a food source. Search Space Generator converts input parameters of SUT to nodes. Each value of input parameter will be the edge of a node. Figure 6 displays the structure of Search Space Generator. For example, input parameter A has two values. In the Search Space Generator, node A, has two edges. This process continues for every input parameter of SUT. The last node is a food node which only contains incoming edges.

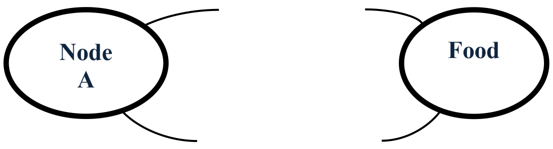


Fig. 6. Search space generator

The final component in our strategy is the Test Case Generator. This component constructs a single best test. ACO algorithm is adopted in this component. Heuristic and pheromone values are two important parameters to assist ants to make a decision. Heuristic is the early information before ants start travelling to search for food. It refers to the uncovered tuples in the IOR Tuples List. Whilst pheromone is a value deposited by the previous ants to tell the next ants that this edge has been travelled by him. The next ants have a high tendency to choose edge with the highest pheromone value. At the last node or known as Food node, the chosen edges for each node will form a single test case. Figure 7 depicts the Test Case Generator's structure.

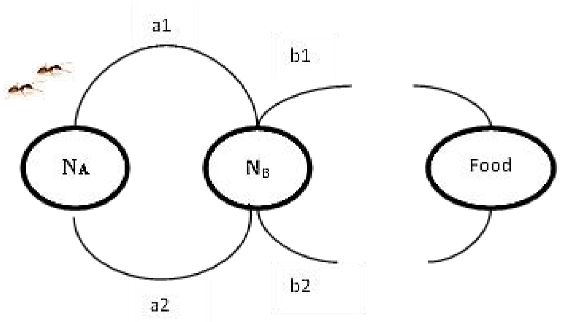
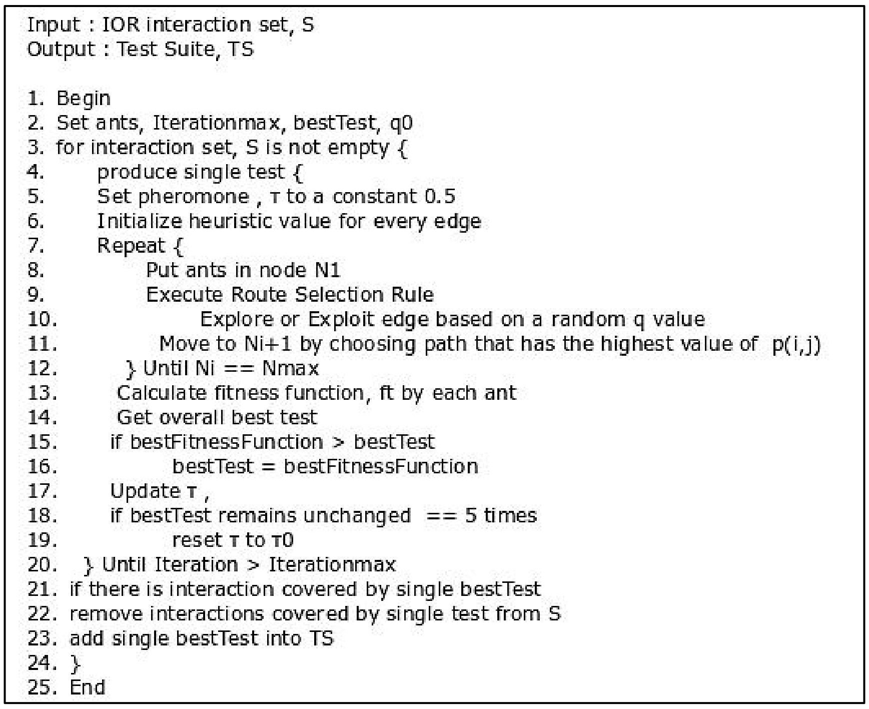


Fig. 7. Test case generator

The algorithm for this component is depicted in Figure 8. The IOR interaction set, S is used as a stopping criterion in generating Test Suite Algorithm. Once IOR interaction set, S is empty, the algorithm finishes its task and final test suite is completed. This algorithm aims to generate the most covered single test of IOR interaction test, S by using formulas or rules to calculate certain criteria.



```
Input : IOR interaction set, S
Output : Test Suite, TS

1.  Begin
2.  Set ants, Iterationmax, bestTest, q0
3.  for interaction set, S is not empty {
4.        produce single test {
5.        Set pheromone , τ to a constant 0.5
6.        Initialize heuristic value for every edge
7.        Repeat {
8.              Put ants in node N1
9.              Execute Route Selection Rule
10.                    Explore or Exploit edge based on a random q value
11.               Move to Ni+1 by choosing path that has the highest value of  p(i,j)
12.          } Until Ni == Nmax
13.        Calculate fitness function, ft by each ant
14.        Get overall best test
15.       if bestFitnessFunction > bestTest
16.                bestTest = bestFitnessFunction
17.       Update τ ,
18.       if bestTest remains unchanged  == 5 times
19.                reset τ to τ0
20.   } Until Iteration > Iterationmax
21. if there is interaction covered by single bestTest
22. remove interactions covered by single test from S
23. add single bestTest into TS
24. }
25. End
```

Fig. 8. Test suite generation algorithm

At the initialization stage (i.e. line 5-6), pheromone, $\tau_{i,j}(t)$ is initialized to 0.5 (i.e. constant value). In contrast, heuristic value, the information of uncovered tuples in the IOR Tuple List is calculated using Eq. (3).

$$\eta_{(i,j)} = \frac{E_{i,max} - E_{i,j} + 1}{E_{i,max} - E_{i,min} + 1} \quad (3)$$

where $E_{i,j}$ is the number of test cases that contain edgei,j in the given test set, $E_{i,min}$ is $\min_{1 \le j \le value_i}$ {Ei,j} and $E_{i,max}$ is $\max_{1 \le j \le value_i}$ {Ei,j}.

While travelling from one node to another, Route Selection Rule is being used as in line 9. Ant either explores new edge or exploits similar edge as previous ant. A probability indicator, q0 (where $0 \le q \le 1$), is utilized. A random variable, q, is generated uniformly within the range of [0,1] and compared to q0. If q is less than or equal to q0, the exploitation of edge is happened by applying the formula mentioned in Eq. (4).

$$\text{argmax}_{1 \le h \le li} \left\{ [\tau_{i,h}(t)^{\alpha}([\eta_{i,h}(t)]^{\beta} \right\} \quad (4)$$

However, if q>q0, ant is exploring new edge using a Eq. (5).

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

Ant will travel until it reaches the food node. Then, the ant deposits a pheromone at the chosen edge as in line 17. Indirectly, the ant is updating any pheromone value deposited by the previous ants. The formula for updating a pheromone is as in Eq. (6).

$$\tau_{i,j}(t+1) = \begin{cases} (1-\rho)\tau_{i,j}(t) + \rho\Delta\tau_{i,j}^{bs} & \text{if } e_{i,j} \in test^{bs} \\ \tau_{i,j}(t) & \text{otherwise} \end{cases} \quad (6)$$

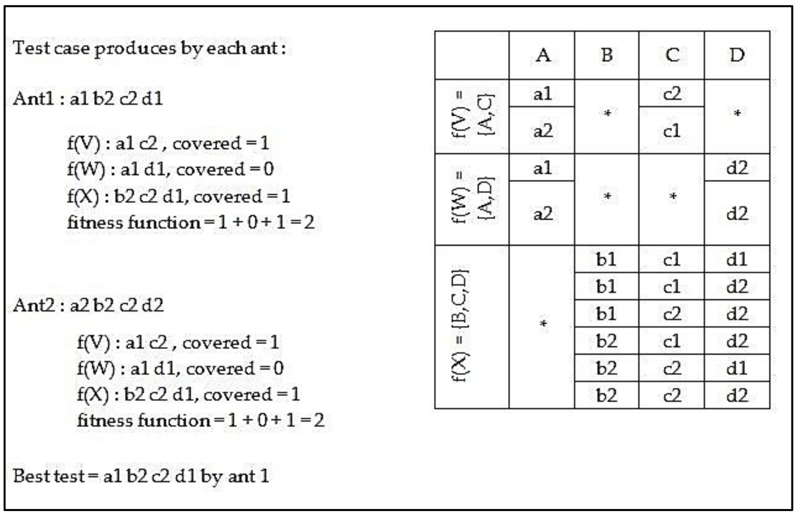
where $\Delta\tau_{i,j}^{bs}$ is fitness function.

In addition of incorporating ACO algorithm, this research also focuses on metaheuristic search technique for IOR t-way testing. Metaheuristic search technique can produce smaller test suite size in many cases [17, 45]. It also has been applied to search-based software testing and produces good result in a reasonable of time [46]. Metaheuristic works by producing a set of test cases. Rather than making assumptions like greedy-based algorithm, metaheuristic uses fitness function to search for the best coverage test case. Later on, a set of high-quality test cases can be found. For that reason, this paper is introducing fitness function for IOR t-way testing. Fitness function of IOR t-way testing is number of interactions that covered the current test but were not covered by the earlier test. An Eq. (7) presents the IOR t-way testing for fitness function which is adopted from previous researchers [47].

$$f(ti) = \sum_{p=0}^{programOutput} w_p \quad (7)$$

where w_p is the number of interactions covered by current test but not covered by the previous test and programOutput is the function output of a SUT.

An illustration of how to calculate the fitness function is shown in Figure 9. There are three function outputs in this scenario: f(V), f(W), and f(X). The IOR interaction that needs to be covered is the interaction that IOR t-way testing on the basis of these function outputs produces. The fitness function is used to identify which agent (ant) generates the best test, presuming there are two ants acting as agents to generate their respective tests. To assess the caliber of the generated test, the fitness function computes the IOR interaction derived from the function output. Then, by comparing each IOR interaction to the IOR interaction table, the highest number of coverage interactions is determined. The best test is chosen as the one with the highest fitness function. Finally, the best test I placed into the final test suite.


Test case produces by each ant :

Ant1 : a1 b2 c2 d1

f(V) : a1 c2 , covered = 1
f(W) : a1 d1, covered = 0
f(X) : b2 c2 d1, covered = 1
fitness function = 1 + 0 + 1 = 2

Ant2 : a2 b2 c2 d2

f(V) : a1 c2 , covered = 1
f(W) : a1 d1, covered = 0
f(X) : b2 c2 d1, covered = 1
fitness function = 1 + 0 + 1 = 2

Best test = a1 b2 c2 d1 by ant 1

	A	B	C	D
f(V) = {A,C}	a1	*	c2	*
	a2		c1	
f(W) = {A,D}	a1	*	*	d2
	a2			d2
f(X) = {B,C,D}	*	b1	c1	d1
		b1	c1	d2
		b1	c2	d2
		b2	c1	d2
		b2	c2	d1
		b2	c2	d2

Fig. 9. Fitness functions calculation example

4. Results
4.1 Evaluations with Benchmarking IOR Strategies

Evaluation of iTTSGA with current benchmarking IOR strategies has been performed. Two different experiments were conducted based on parameters presented in Table 4. Some of the parameters are based on the parameter tuning done by researchers [48].

Table 4. iTTSGA design parameters

Parameter	Value
Number of ants	20
Pheromone control, α	1.0
Heuristic control, β	0.5
Pheromone evaporation rate, ρ	0.1
Initial pheromone, τ_0	0.5
q0	0.5
Iteration	300
Stale period	5

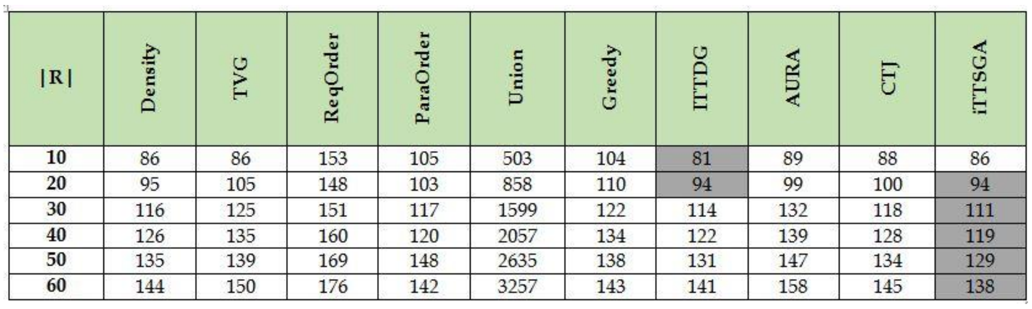
The first experiment is to represent uniform configurations, IOR (N, 3^3, R) with 10 parameters. Each parameter has three possible values. The second experiment consists of 10 non-uniform configurations, IOR (N, 2^3 $3^3 4^3 5^1$, R). The parameters contain 3 parameters of 2 values, 3 parameters of 3 values, 3 parameters of 4 values and a parameter of 5 values. For both experiments, the relationship, R are as follows:

i. R1 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, {4,6}]
ii. R2 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, {4,6}, {2,3,4,8}, {2,3,5}, {5,6}, {0,6,8}, {8,9}, {0,5}, {1,3,5,9}, {1,6,7,9}, {0,4}, {0,2,3}]
iii. R3 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, {4,6}, {2,3,4,8}, {2,3,5}, {5,6}, {0,6,8}, {8,9}, {0,5}, {1,3,5,9}, {1,6,7,9}, {0,4}, {0,2,3}, {1,3,6,9}, {2,4,7,8}, {0,2,6,9}, {0,1,7,8}, {0,3,7,9}, {3,4,7,8}, {1,5,7,9}, {1,3,6,8}, {1,2,5}, {3,4,5,7}]
iv. R4 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, {4,6}, {2,3,4,8}, {2,3,5}, {5,6}, {0,6,8}, {8,9}, {0,5}, {1,3,5,9}, {1,6,7,9}, {0,4}, {0,2,3}, {1,3,6,9}, {2,4,7,8}, {0,2,6,9}, {0,1,7,8}, {0,3,7,9}, {3,4,7,8}, {1,5,7,9}, {1,3,6,8}, {1,2,5}, {3,4,5,7}, {0,2,7,9}, {1,2,3}, {1,2,6}, {2,5,9}, {3,6,7},{1,2,4,7}, {2,5,8}, {0,1,6,7}, {3,5,8}, {0,1,2,8}]
v. R5 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, { 4,6}, {2,3,4,8}, {2,3,5}, {5,6}, {0,6,8}, {8,9}, {0,5}, {1,3,5,9}, {1,6,7,9}, {0,4}, {0,2,3}, {1,3,6,9}, {2,4,7,8}, {0,2,6,9}, {0,1,7,8}, {0,3,7,9}, {3,4,7,8}, {1,5,7,9}, {1,3,6,8}, {1,2,5}, {3,4,5,7}, {0,2,7,9}, {1,2,3}, {1,2,6}, {2,5,9}, {3,6,7}, {1,2,4,7}, {2,5,8}, {0,1,6,7}, {3,5,8}, {0,1,2,8}, {2,3,9}, {1,5,8}, {1,3,5,7}, {0,1,2,7}, {2,4,5,7}, {1,4,5}, {0,1,7,9}, {0,1,3,6}, {1,4,8}, {3,5,7,9}
vi. R6 = [{1,2,7,8}, {0,1,2,9}, {4,5,7,8}, {0,1,3,9}, {0,3,8}, {6,7,8}, {4,9}, {1,3,4}, {0,2,6,7}, {4,6}, {2,3,4,8}, {2,3,5},{ 5,6}, {0,6,8}, {8,9}, {0,5}, {1,3,5,9}, {1,6,7,9}, {0,4}, {0,2,3}, {1,3,6,9}, {2,4,7,8}, {0,2,6,9}, {0,1,7,8}, {0,3,7,9}, {3,4,7,8}, {1,5,7,9}, {1,3,6,8}, {1,2,5}, {3,4,5,7}, {0,2,7,9},{1,2,3}, {1,2,6}, {2,5,9}, {3,6,7}, {1,2,4,7}, {2,5,8}, {0,1,6,7}, {3,5,8}, {0,1,2,8}, {2,3,9}, {1,5,8}, {1,3,5,7}, {0,1,2,7}, {2,4,5,7}, {1,4,5},{0,1,7,9}, {0,1,3,6}, {1,4,8}, {3,5,7,9}, {0,6,7,9}, {2,6,7,9}, {2,6,8}, {2,3,6}, {1,3,7,9}, {2,3,7}, {0,2,7,8}, {0,1,6,9}, {1,3,7,8}, {0,1,3,7}].

For both experiments, R starts with the first 10 relationships. Then, it adds another 10 relationships and continues until all 60 relationships. As a result, 12 independent sub-experiments have been conducted for both experiments.

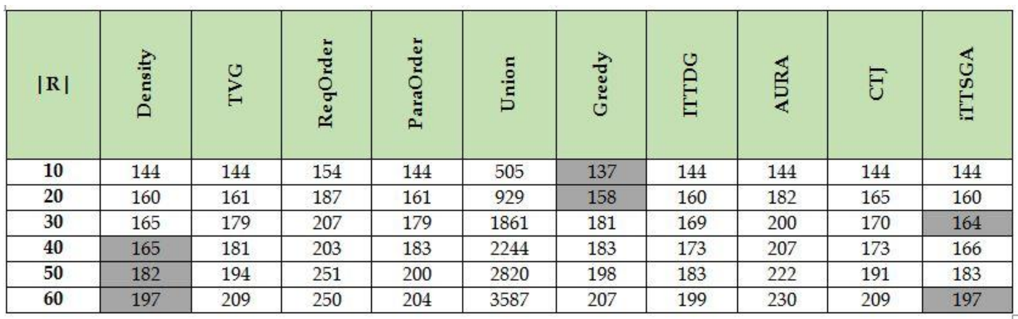
Results obtained from the experiments is compared to benchmark experiments of IOR strategies as in [27] and [36]. The comparison of our strategy to benchmarked strategies has been widely used in the t-way research field to evaluate iTTSGA's performance in producing test suite size.

Figures 10 and 11 demonstrate results from experiments for uniform configurations and non-uniform configurations respectively. Coloured cell is the best test suite size produced by each |R|. For example, ITTDG produced the best test suite size as compared to other strategies for |R|=10. The best test suite size is referring to the smallest number of test suite, which means the strategy produces a near optimum test suit size.



|R|	Density	TVG	ReqOrder	ParaOrder	Union	Greedy	ITTDG	AURA	CTJ	iTTSGA
10	86	86	153	105	503	104	81	89	88	86
20	95	105	148	103	858	110	94	99	100	94
30	116	125	151	117	1599	122	114	132	118	111
40	126	135	160	120	2057	134	122	139	128	119
50	135	139	169	148	2635	138	131	147	134	129
60	144	150	176	142	3257	143	141	158	145	138

Fig. 10. Test suite size for IOR (N, 3^{10}, R)



|R|	Density	TVG	ReqOrder	ParaOrder	Union	Greedy	ITTDG	AURA	CTJ	iTTSGA
10	144	144	154	144	505	137	144	144	144	144
20	160	161	187	161	929	158	160	182	165	160
30	165	179	207	179	1861	181	169	200	170	164
40	165	181	203	183	2244	183	173	207	173	166
50	182	194	251	200	2820	198	183	222	191	183
60	197	209	250	204	3587	207	199	230	209	197

Fig. 11. Test suite size for IOR (N, 2^3 $3^3 4^3 5^1$, R)

Figure 10 compares iTTSGA with other existing IOR strategies in the first experiment, focused on uniform configurations. It is worth noting that most of these strategies employ computational search techniques, except for the CTJ strategy. From the table, iTTSGA consistently achieves the optimal test suite size compared to other algorithms for |R|=30 to |R|=60. It shares the same test suite size with ITTDG for |R|=20. However, for |R|=10, ITTDG outperforms iTTSGA and produces the best test suite size with only 81 test cases. The results indicate that strategies utilizing metaheuristic search techniques outperform those based on computational methods, as reported in the literature [3]. This finding further supports the advantage of employing metaheuristic approaches in optimizing various tasks or problems.

In the second experiment, the performance of iTTSGA is not as strong as in the first experiment. It achieves the best test suite size for |R|=30 and 60. While iTTSGA may not outperform other strategies in all cases, it still delivers highly promising results. For |R|=10 to 20, Greedy produces the smallest test suite size, whereas Density performs the best for |R|=40 to 60. Interestingly, Density and iTTSGA produce the same number of test cases for |R|=60. Overall, these findings demonstrate the capability of these strategies to generate near-optimal test suites. Producing an optimum test suite size is NP-hard problem [41, 49]. Due to this reason, no strategies could perform the best results for all configurations. In contrast to the results that are the closest best, iTTSGA displays results that are very competitive.

4.2. Statistical Analysis

The performance of iTTSGA against other IOR based strategies (i.e. Density, TVG, ReqOrder, ParaOrder, Union, Greedy, ITTDG and AURA) has been conducted using statistical analysis. The significant value has been set to 0.05. The hypothesis null for this analysis is there is no significant difference between iTTSGA and other strategies. This hypothesis is rejected if the significant value is less than 0.05 where there are significant differences between iTTSGA and other strategies. The non-parametric tests (i.e. Friedman and Wilcoxon Rank Test) are required to conduct the test due to the result is not normally distributed. Friedman test is used to rank the strategies involved based on mean rank. Next, Wilcoxon Rank Test is executed to analyse the significant differences of test suite size produced by all strategies. The differences can be either iTTSGA produces larger or smaller test suite size than other strategies. Based on Friedman Test Rank, it can be concluded which is the differences (smaller or larger size) based on the rank.

The results obtained in Figures 10 and 11 have been used to conduct the statistical tests. Tables 5 and 6 present statistical analyses for Wilcoxon Rank and Friedman test respectively. In Table 5, the Wilcoxon Rank test presents that there is no significant difference of test suite size produced between iTTSGA and ITTDG strategy as well as GVS strategy. From this result, it is proven that iTTSGA strategy is at the same level with both strategies. Whereas, there are significant differences of test suite size with other strategies. Nevertheless, the Friedman test mean rank shows that iTTSGA is at the first rank, which means that, iTTSGA produces smaller test suite size as compared to the other strategies. Meanwhile, Table 6 exhibits statistical analysis of the Wilcoxon Rank and the Friedman test for IOR (N, 23 334351, R). Based on the Wilcoxon Rank test significant value, it shows that there is no significant difference between iTTSGA strategy and ITTDG, Greedy and Density strategy in producing number of test cases. On the other hand, the Friedman test mean rank demonstrates that the iTTSGA strategy is the second best after Density.

Table 7
Statistical analysis for IOR (N, 3^{10}, R)

Strategy	Wilcoxon rank test		Friedman test
	Pair	Significant value	Mean rank
iTTSGA			1.75
ReqOrder	ReqOrder - iTTSGA	0.027	10.00
Union	Union - iTTSGA	0.028	11.00
Greedy	Greedy - iTTSGA	0.028	6.92
ITTDG	ITTDG - iTTSGA	0.492	2.42
AURA	AURA - iTTSGA	0.027	7.67
TVG	TVG - iTTSGA	0.043	7.00
Density	Density - iTTSGA	0.042	4.25
ParaOrder	ParaOrder - iTTSGA	0.027	6.17
GVS	GVS - iTTSGA	0.168	3.17
CTJ	CTJ - iTTSGA	0.027	5.67

Table 8
Statistical analysis for IOR (N, 2^3 $3^3 4^3 5^1$, R)

Strategy	Wilcoxon rank test		Friedman test
	Pair	Significant value	Mean rank
iTTSGA			2.58
ReqOrder	ReqOrder - iTTSGA	0.028	9.83
Union	Union - iTTSGA	0.028	11.00
Greedy	Greedy - iTTSGA	0.115	4.92
ITTDG	ITTDG - iTTSGA	0.109	3.67
AURA	AURA - iTTSGA	0.043	8.58
TVG	TVG - iTTSGA	0.042	6.00
Density	Density - iTTSGA	0.564	2.33
ParaOrder	ParaOrder - iTTSGA	0.042	6.25
GVS	GVS - iTTSGA	0.043	5.08
CTJ	CTJ - iTTSGA	0.043	5.75

5. Conclusions

The development of iTTSGA, a new metaheuristic t-way strategy that supports IOR t-way testing is discussed in this paper. iTTSGA consists of three components, namely tuples generator, search space generator and test case generator. the strategy has introduced IOR t-way fitness functions to choose the best test cases. Benchmark experiments for uniform and non-uniform configurations (i.e. IOR (N, 3^3, R) and IOR (N, 2^3 $3^3 4^3 5^1$, R) respectively) has been executed and results have been compared to other existing IOR strategies. The obtained results by iTTSGA were compared with other existing IOR strategies. As far as test suite size is concerned, iTTSGA produced the optimum test suite size for uniform configuration except for $|R|=10$. Based on statistical tests, iTTSGA achieved the first mean rank in Friedman Test although the test suite size produced by iTTSGA has no significant difference with ITTDG and GVS. Meanwhile, for non-uniform configurations, the results are very competitive and iTTSGA is in the second mean rank in Friedman Tests and has no significant difference with ITTDG, Greedy and Density strategy. The statistical tests have proven that iTTSGA manage to compete other existing IOR t-way testing strategies. For the future works, iTTSGA is planned to support other constraints in t-way testing. Besides that, the strategy is planned to hybrid with other algorithm to produce better results

Acknowledgement
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] Muazu, Aminu Aminu, Ahmad Sobri Hashim, and Aliza Sarlan. "Application and adjustment of “don’t care” Values in t-way testing techniques for generating an optimal test suite." *Journal of Advances in Information Technology Vol* 13, no. 4 (2022). https://doi.org/10.12720/jait.13.4.347-357

[2] Muazu, Aminu Aminu, Ahmad Sobri Hashim, Aliza Sarlan, and Mujaheed Abdullahi. "SCIPOG: Seeding and constraint support in IPOG strategy for combinatorial t-way testing to generate optimum test cases." *Journal of King Saud University-Computer and Information Sciences* 35, no. 1 (2023): 185-201. https://doi.org/10.1016/j.jksuci.2022.11.010

[3] Zamli, Kamal Z., Fakhrud Din, Graham Kendall, and Bestoun S. Ahmed. "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation." *Information Sciences* 399 (2017): 121-153. https://doi.org/10.1016/j.ins.2017.03.007

[4] Zamli, Kamal Z., Basem Y. Alkazemi, and Graham Kendall. "A tabu search hyper-heuristic strategy for t-way test suite generation." *Applied Soft Computing* 44 (2016): 57-74. https://doi.org/10.1016/j.asoc.2016.03.021

[5] Hu, Linghuan, W. Eric Wong, D. Richard Kuhn, and Raghu N. Kacker. "How does combinatorial testing perform in the real world: An empirical study." *Empirical Software Engineering* 25 (2020): 2661-2693. https://doi.org/10.1007/s10664-019-09799-2

[6] Othman, Rozmie R., Kamal Z. Zamli, and Sharifah Mashita Syed Mohamad. "T-way testing strategies: A critical survey and analysis." *International Journal of Digital Content Technology and its Applications* 7, no. 9 (2013): 222.

[7] Zamli, Kamal Z., Fakhrud Din, Salmi Baharom, and Bestoun S. Ahmed. "Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites." *Engineering Applications of Artificial Intelligence* 59 (2017): 35-50. https://doi.org/10.1016/j.engappai.2016.12.014

[8] Othman, Rozmie R., and Kamal Z. Zamli. "T-way strategies and its applications for combinatorial testing." *International Journal of New Computer Architectures and their Applications* 1, no. 2 (2011): 459-73.

[9] Lei, Yu, Raghu Kacker, D. Richard Kuhn, Vadim Okun, and James Lawrence. "IPOG: A general strategy for t-way software testing." In *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, p. 549-556. IEEE, 2007. https://doi.org/10.1109/ECBS.2007.47

[10] Zamli, Kamal Z., Mohammad FJ Klaib, Mohammed I. Younis, Nor Ashidi Mat Isa, and Rusli Abdullah. "Design and implementation of a t-way test data generation strategy with automated execution tool support." *Information Sciences* 181, no. 9 (2011): 1741-1758. https://doi.org/10.1016/j.ins.2011.01.002

[11] Lin, Jinkun, Chuan Luo, Shaowei Cai, Kaile Su, Dan Hao, and Lu Zhang. "TCA: An efficient two-mode meta-heuristic algorithm for combinatorial test generation (T)." In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, p. 494-505. IEEE, 2015. https://doi.org/10.1109/ASE.2015.61

[12] Xiang, Lai Y., A. A. Alsewari, and Kamal Z. Zamli. "Pairwise test suite generator tool based on harmony search algorithm (HS-PTSGT)." *International Journal on Artificial Intelligence* 2, p. 62-65. 2015.

[13] Chen, Xiang, Qing Gu, Jingxian Qi, and Daoxu Chen. "Applying particle swarm optimization to pairwise testing." In *2010 IEEE 34th Annual Computer Software and Applications Conference*, p. 107-116. IEEE, 2010. https://doi.org/10.1109/COMPSAC.2010.17

[14] Shiba, Toshiaki, Tatsuhiro Tsuchiya, and Tohru 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.*, p. 72-77. IEEE, 2004. https://doi.org/10.1109/CMPSAC.2004.1342808

[15] Ali, Mohd Shaiful Aziz Rashid, Rozmie Razif Othman, Zainor Ridzuan Yahya, and Mohd Zamri Zahir. "A modified artificial bee colony based test suite generation strategy for uniform T-way testing." In *IOP Conference Series: Materials Science and Engineering*, vol. 767, no. 1, p. 012020. IOP Publishing, 2020. https://doi.org/10.1088/1757-899X/767/1/012020

[16] Ahmed, Bestoun S., Taib Sh Abdulsamad, and Moayad Y. Potrus. "Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm." *Information and Software Technology* 66 (2015): 13-29. https://doi.org/10.1016/j.infsof.2015.05.005

[17] Wu, Huayao, Changhai Nie, Fei-Ching Kuo, Hareton Leung, and Charles J. Colbourn. "A discrete particle swarm optimization for covering array generation." *IEEE Transactions on Evolutionary Computation* 19, no. 4 (2014): 575-591. https://doi.org/10.1109/TEVC.2014.2362532

[18] Ahmed, Bestoun S., and Kamal Z. Zamli. "A variable strength interaction test suites generation strategy using particle swarm optimization." *Journal of Systems and Software* 84, no. 12 (2011): 2171-2185. https://doi.org/10.1016/j.jss.2011.06.004

[19] Alsewari, Abdul Rahman A., and Kamal Z. Zamli. "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support." *Information and Software Technology* 54, no. 6 (2012): 553-568. https://doi.org/10.1016/j.infsof.2012.01.002

[20] Chen, Xiang, Qing Gu, Ang Li, and Daoxu Chen. "Variable strength interaction testing with an ant colony system approach." In *2009 16th Asia-Pacific Software Engineering Conference*, p. 160-167. IEEE, 2009. https://doi.org/10.1109/APSEC.2009.18

[21] Cohen, Myra B., Charles J. Colbourn, and Alan CH Ling. "Augmenting simulated annealing to build interaction test suites." In *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, p. 394-405. IEEE, 2003. https://doi.org/10.1109/ISSRE.2003.1251061

[22] Othman, Rozmie Razif, Kamal Zuhairi Zamli, and Lukito Edi Nugroho. "General variable strength t-way strategy supporting flexible interactions." *Maejo International Journal of Science and Technology* 6, no. 3 (2012): 415.

[23] Othman, Rozmie R., Norazlina Khamis, and Kamal Z. Zamli. "Variable strength t-way test suite generator with constraints support." *Malaysian Journal of Computer Science* 27, no. 3 (2014): 204-217.

[24] Ahmed, Bestoun S., and Kamal Z. Zamli. "A variable strength interaction test suites generation strategy using particle swarm optimization." *Journal of Systems and Software* 84, no. 12 (2011): 2171-2185. https://doi.org/10.1016/j.jss.2011.06.004

[25] Rahman, Mostafijur, Rozmie Razif Othman, R. Badlishah Ahmad, and Md Mijanur Rahman. "Event driven input sequence t-way test strategy using simulated annealing." In *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, p. 663-667. IEEE, 2014. https://doi.org/10.1109/ISMS.2014.119

[26] Ramli, N., R. R. Othman, R. Hendradi, and I. Iszaidy. "T-way test suite generation strategy based on ant colony algorithm to support t-way variable strength." In *Journal of Physics: Conference Series* 1755, no. 1, p. 012034. IOP Publishing, 2021. https://doi.org/10.1088/1742-6596/1755/1/012034

[27] Alsewari, A. A., Nasser M. Tairan, and Kamal Z. Zamli. "Survey on input output relation based combination test data generation strategies." *ARPN J Eng Appl Sci* 10, no. 18 (2015): 8427-8430.

[28] Wang, Ziyuan, Baowen Xu, and Changhai Nie. "Greedy heuristic algorithms to generate variable strength combinatorial test suite." In *2008 The Eighth International Conference on Quality Software*, p. 155-160. IEEE, 2008. https://doi.org/10.1109/QSIC.2008.52

[29] Ziyuan, Wang, Nie Changhai, and Xu Baowen. "Generating combinatorial test suite for interaction relationship." In *Fourth International Workshop on Software Quality assurance: in conjunction with the 6th ESEC/FSE joint meeting*, p. 55-61. 2007. https://doi.org/10.1145/1295074.1295085

[30] Schroeder, Patrick J., and Bogdan Korel. "Black-box test reduction using input-output analysis." *ACM SIGSOFT Software Engineering Notes* 25, no. 5 (2000): 173-177. https://doi.org/10.1145/347636.349042

[31] Cheng, Christine, Adrian Dumitrescu, and Patrick Schroeder. "Generating small combinatorial test suites to cover input-output relationships." In *Third International Conference on Quality Software, 2003. Proceedings.*, p. 76-82. IEEE, 2003. https://doi.org/10.1109/QSIC.2003.1319088

[32] Schroeder, Patrick J., Eok Kim, Jerry Arshem, and Pankaj Bolaki. "Combining behavior and data modeling in automated test case generation." In *Third International Conference on Quality Software, 2003. Proceedings.*, p. 247-254. IEEE, 2003. https://doi.org/10.1109/QSIC.2003.1319108

[33] Othman, Rozmie R., and Kamal Z. Zamli. "ITTDG: Integrated T-way test data generation strategy for interaction testing." *Scientific Research and Essays* 6, no. 17 (2011): 3638-3648. https://doi.org/10.5897/SRE10.1196

[34] Ong, H. Y., and Kamal Z. Zamli. "Development of interaction test suite generation strategy with input-output mapping supports." *Scientific Research and Essays* 6, no. 16 (2011): 3418-3430.

[35] Wang, Ziyuan, and Haixiao He. "Generating variable strength covering array for combinatorial software testing with greedy strategy." *Journal of Software* 8, no. 12 (2013): 3173-3181. https://doi.org/10.4304/jsw.8.12.3173-3181

[36] Younis, Mohammed Issam, Abdul Rahman A. Alsewari, Ng Yeong Khang, and Kamal Z. Zamli. "CTJ: Input-output based relation combinatorial testing strategy using jaya algorithm." *Baghdad Science Journal* 17, no. 3 (Suppl.) (2020): 1002-1002. http://dx.doi.org/10.21123/bsj.2020.17.3(Suppl.).1002

[37] Muazu, Aminu Aminu, Ahmad Sobri Hashim, and Aliza Sarlan. "Review of nature inspired metaheuristic algorithm selection for combinatorial t-way testing." *IEEE Access* 10 (2022): 27404-27431. https://doi.org/10.1109/ACCESS.2022.3157400

[38] Mahmoud, Thair, and Bestoun S. Ahmed. "An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use." *Expert Systems with Applications* 42, no. 22 (2015): 8753-8765. https://doi.org/10.1016/j.eswa.2015.07.029

[39] Rahman, Mostafijur, Rozmie Razif Othman, R. Badlishah Ahmad, and Md Mijanur Rahman. "A meta heuristic search based t-way event driven input sequence test case generator." *Int. J. Simul. Syst. Sci. Technol* 15, no. 3 (2014): 65-71. https://doi.org/10.5013/IJSSST.a.15.03.10

[40] Beheshti, Zahra, and Siti Mariyam Hj Shamsuddin. "A review of population-based meta-heuristic algorithms." *Int. j. adv. soft comput. appl* 5, no. 1 (2013): 1-35.

[41] Al-Sewari, AbdulRahman A., and Kamal Z. Zamli. "An orchestrated survey on t-way test case generation strategies based on optimization algorithms." In *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications: Innovation Excellence Towards Humanistic Technology*, p. 255-263. Springer Singapore, 2014. https://doi.org/10.1007/978-981-4585-42-2_30

[42] Alazzawi, Ammar K., Helmi Md Rais, Shuib Basri, and Yazan A. Alsariera. "PhABC: A hybrid artificial bee colony strategy for pairwise test suite generation with constraints support." In *2019 IEEE Student Conference on Research and Development (SCOReD)*, p. 106-111. IEEE, 2019. https://doi.org/10.1109/SCORED.2019.8896324

[43] Reid, Stuart C. "An empirical analysis of equivalence partitioning, boundary value analysis and random testing." In *Proceedings fourth international software metrics symposium*, p. 64-73. IEEE, 1997. https://doi.org/10.1109/METRIC.1997.637166

[44] Ramli, Nuraminah, Rozmie Razif Othman, Zahereel Ishwar Abdul Khalib, and Muzammil Jusoh. "A review on recent t-way combinatorial testing strategy." In *MATEC Web of Conferences*, 140, p. 01016. EDP Sciences, 2017. https://doi.org/10.1051/matecconf/201714001016

[45] Torres-Jimenez, Jose, and Idelfonso Izquierdo-Marquez. "Survey of covering arrays." In *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, p. 20-27. IEEE, 2013. https://doi.org/10.1109/SYNASC.2013.10

[46] Sahin, Omur, and Bahriye Akay. "Comparisons of metaheuristic algorithms and fitness functions on software test data generation." *Applied Soft Computing* 49 (2016): 1202-1214. https://doi.org/10.1016/j.asoc.2016.09.045

[47] Ramli, Nuraminah, Rozmie R. Othman, and Mohd Shaiful Aziz Rashid Ali. "Optimizing combinatorial input-output based relations testing using Ant Colony algorithm." In *2016 3rd International Conference on Electronic Design (ICED)*, p. 586-590. IEEE, 2016. https://doi.org/10.1109/ICED.2016.7804713

[48] Ramli, N., R. R. Othman, and S. S. M. Fauzi. "Ant colony optimization algorithm parameter tuning for t-way IOR testing." In *Journal of Physics: Conference Series*, vol. 1019, no. 1, p. 012086. IOP Publishing, 2018. https://doi.org/10.1088/1742-6596/1019/1/012086

[49] Yeh, Ong Hui, and Kamal Zuhairi Zamli. "Development of a non-deterministic input-output based relationship test data set minimization strategy." In *2011 IEEE Symposium on Computers & Informatics*, pp. 800-805. IEEE, 2011. https://doi.org/10.1109/ISCI.2011.5959020