Ant Colony Optimization Algorithm Parameter Tuning for T-way IOR Testing

To cite this article: N Ramli *et al* 2018 *J. Phys.: Conf. Ser.* **1019** 012086

Ant Colony Optimization Algorithm Parameter Tuning for T-way IOR Testing

N Ramli[1], R R Othman[2,*], S S M Fauzi[3]

[1,2]School of Computer and Communication Engineering, Universiti Malaysia Perlis Pauh Putra Campus, 02600 Arau, Perlis, Malaysia
[3]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 02600 Arau, Perlis, Malaysia

*rozmie@unimap.edu.my

Abstract. In this study, Ant Colony Optimization (ACO) algorithm's parameters for t-way IOR testing were examined. ACO and its variant have been applied to t-way testing but never to t-way IOR interaction support. Tuning ACO parameters were executed to ensure that ACO could perform for IOR as good as other t-way interaction support. Parameter α, β, τ_0, q_0, ρ value and number of ant were tuned to uniform and non-uniform configuration. Each parameter was executed for 10 independent run. Average best test suite and best test suite were recorded and compared among other parameter values to find which value will produce the best result. The optimum test suite size and average test suite size were generated when the value for parameter α = 0.5, β = 3, τ_0 = 0.4 (uniform configuration) and, 0.2 and 1 (non-uniform configuration), q_0 = 0.5, ρ = 0.5 and number of ant = 20.

Keywords : t-way testing, input-output based relation, ant colony optimization

1. Introduction

Software applications become complex and crucial in our daily life. To assure the quality of software application, software testing plays an important role. Combinatorial software testing or also known as t-way (where t is the strength) testing is stated to be very effective test design technique in software testing [1]. T-way testing is different from other test design technique (e.g. Boundary Value Analysis and Cause and Effects) because it involves interaction of input parameters and its values. Its purpose is to overcome exhaustive testing problem systematically [2]. Exhaustive testing is impossible to be implemented because it generates enormous number of test cases [3] as it tests all possible values of input parameters. This situation lead to more time [4] and resources allocated for testing activities.

Input-output based Relation (IOR) is categorized as one of the t-way testing support interactions [5], [6]. It involves interactions of multi input parameters that affect an output. This type of support interaction closely related with system under test's specifications. Software tester, who performs the testing activities, must have the knowledge of the system under test to interact related inputs and its respective output [7]. T-way IOR has proven to reduce number of test cases [5], [8], [9] because it ignores any unrelated interaction and eliminates any redundant test cases. Although the number of test cases is being reduced, the strategy is maintaining its capability in finding faults.

Consider an application of Colour Scheme Generator as in Figure 1 to facilitate the understanding on t-way IOR testing. The application involves three inputs, red, blue and yellow. Each input colour consists of two values, either dark or light. It produces two (2) outputs, purple and green. However the outputs are restricted to certain combination of input. For example, combination of input red and blue will produce output purple. This application is an example of IOR t-way testing and can be represented as in (1).

$$P(\text{purple}) = f\{\text{red, blue}\}$$
$$P(\text{green}) = f\{\text{blue, yellow}\}\ldots\ldots\ldots\ldots\ldots\ldots.. \quad (1)$$

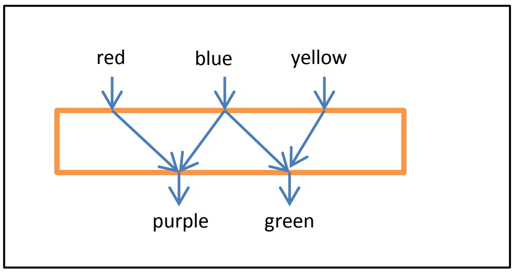
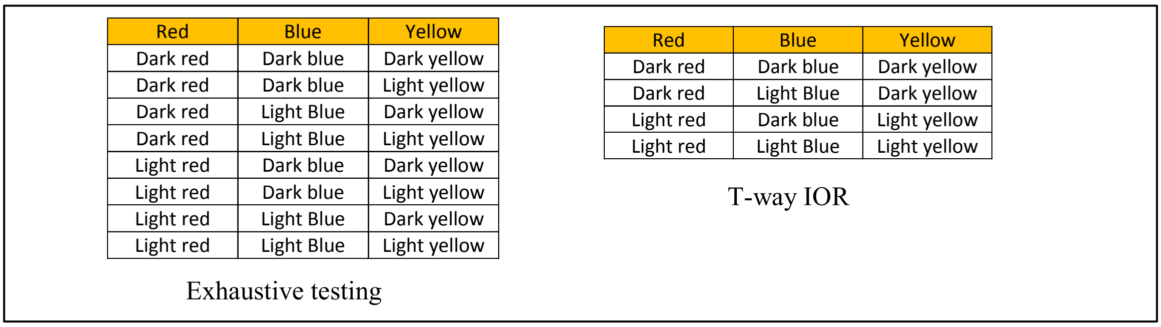


Figure 1. Color Scheme Generator

Test cases generated for Color Scheme Generator is illustrated in Figure 2. Two types of testing design techniques are presented in the figure (i.e. exhaustive testing and t-way IOR testing). Obviously, t-way IOR testing reduces half of number of test cases generated by exhaustive testing. The t-way IOR strategy only generates related input parameters that produce an output and removes any redundant test cases.



Red	Blue	Yellow
Dark red	Dark blue	Dark yellow
Dark red	Dark blue	Light yellow
Dark red	Light Blue	Dark yellow
Dark red	Light Blue	Light yellow
Light red	Dark blue	Dark yellow
Light red	Dark blue	Light yellow
Light red	Light Blue	Dark yellow
Light red	Light Blue	Light yellow

Exhaustive testing

Red	Blue	Yellow
Dark red	Dark blue	Dark yellow
Dark red	Light Blue	Dark yellow
Light red	Dark blue	Light yellow
Light red	Light Blue	Light yellow

T-way IOR

Figure 2. Exhaustive testing and t-way IOR testing

Mathematically, t-way IOR can be represented as covering array. The notation is as in Equation (2).

$$P = IOR(N, C, Rel) \quad (2)$$

From the equation (2), N is the final test suite size, C is value configuration a be represented as $V_0^{p_0}$, $V_1^{p_1}$,……., $V_n^{p_n}$ (p_0 is parameters with V_0 values, p_1 parameters with V_1 values and so on) and *Rel* is the input output relationships involve in generating the test suite.

T-way IOR capability in generating optimum test suite size encourages researchers to further study in this area. There are few strategies that support IOR exists in the market (i.e. Union [9], Greedy [10], Density[11], Test Vector Generator [12], Integrated T-Way Test Data Generator (ITTDG) [13], Aura [14], Para Order and ReqOrder [15]). Those existing strategies apply computational search technique but none of them found to utilize metaheuristic search technique that is proven to produce reasonable test suite size for other type of interaction supports [16].

A metaheuristic algorithm, Ant Colony Optimization (ACO) is known to solve combinatorial problem because the algorithm produces good results in a reasonable time [17]. T-way combinatorial testing also has no exception in using ACO to generate the optimum test suite size. Ant Colony Algorithm [1] and Ant Colony System (ACS) [18] has been introduced for other type of t-way support interactions. Whereas a proposed strategy for IOR using ACO is found in the literature, without stating any suitable parameter settings in the algorithm [19]. There are six parameters (i.e. α, β, τ_0, q_0, ρ and number of ant) to be tuned to get the best settings for the t-way IOR strategy.

This paper presents tuned parameters of Ant Colony Optimization for t-way IOR. The best parameter settings involve in ACO algorithm that can boost up the effectiveness of IOR in generating the best test suite size is examined. Even though ACO has been applied by many researchers, it has not been explored in t-way IOR [16].

The rest of this paper is structured as follows: In section 2, overview of Ant Colony Optimization algorithm is explained in detail. ACO tuning parameter for t-way IOR testing is presented in section 3. Results and discussion is further described in section 4. Finally, section 5 delivers the conclusion.

2. Overview of Ant Colony Optimization Algorithm

Ant Colony Algorithm is an optimization algorithm applied to many combinatorial optimization problem [17] including t-way testing [1], [18], [20]. This algorithm mimics a colony of ant travelling from their nest to find food source. Each ant communicates with other ants by their chemical pheromone deposited at every path travelled. The pheromone deposited by previous ant become guidance to the others in choosing which path to travel. Their aim is to choose the shortest path to the food source. Later on, the colony will build the shortest path from their nest to food source.

The capability of ACO and its family in t-way testing has been proven as it generated very promising results. It has been explored in uniform [1] and variable strength [18] (i.e. Ant Colony System). As a result of good performance by both studies, ACO is introduced to another type of t-way support interaction, IOR. The application of ACO family (i.e. Ant Colony System) towards IOR has been introduced in [19]. To further strengthen the ability of ACO in IOR, we adapted and modified the ACO strategy.

Figure 3 presents the algorithm of our ACO. The algorithm's structure is based on [19] except that some modification has been made in calculating heuristic value as in [1] and the pheromone local update has been removed to suit with IOR strategy. The algorithm begins with initializing heuristic and a constant pheromone value to each edge. Next, each ant starts to construct test case by travelling from one node to another until it reaches food node (i.e. end of the node). After all ants finish travelling to the food node, the best test case constructed is chosen based on fitness function of each test case. Fitness function is calculated based on IOR interaction and function output and it is adopted as in [19]. The process continues until the IOR interaction set is empty.

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

Begin
    Set ants, $\text{Iteration}_{max}$, bestTest, q
    for interaction set, S is not empty {
        produce single test :

            Set pheromone , τ to a constant value
            Initialize heuristic value for every edge,
                heuristicValue $(i,j) = \frac{E_{i,max} - E_{i,j} + 1}{E_{i,max} - E_{i,min} + 1}$
                    $E_{i,j}$ = number of test cases in S
                    $E_{i,max} = \max_{1 \leq j \leq \text{edge } i} \{E_{i,j}\}$
                    $E_{i,min} = \min_{1 \leq j \leq \text{edge } i} \{E_{i,j}\}$
            Repeat {
              Put ants in node $N_1$
              Repeat{
              Construct test case
                Set $q_0$ to $0 \leq q \leq 1$
                Set random value, q in [0,1]
```

```
                    If  $q \leq q_0$
                      Exploit path,  $argmax_{1\leq h\leq li}\{[\tau_{i,j}(t)^{\alpha}[\text{heuristicValue}_{i,j}(t)]^{\beta}\}$

                    If  $q>q_0$
                      Explore new path, $p_{i,j}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\text{heuristicValue}_{i,j}(t)]^{\beta}}{\sum_{h=1}^{li}[\tau_{i,h}(t)^{\alpha}[\text{heuristicValue}_{i,h}(t)]^{\beta}}$

                    Move to $N_{i+1}$ by choosing path that has the highest value of  $p_{i,j}$
                  } Until $N_i == N_{max}$
                  Calculate fitness function, ft by each ant
                  Get overall best test
                    if bestFitnessFunction > bestTest
                        bestTest = bestFitnessFunction
                  Update $\tau$ ,
                        $\tau_{i,j}(t+1) = \begin{cases} (1-\rho)\tau_{i,j}(t) + \rho(ft)\ if\ e_{i,j} \in bestTest \\ \tau_{i,j}(t) \quad otherwise \end{cases}$

                  if bestTest remains unchanged  == 5 times
                    reset $\tau$ to $\tau_0$
                  } Until Iteration > $\text{Iteration}_{max}$

              if there is interaction covered by single bestTest
                  remove interactions covered by single test from S
                  add single bestTest into TS
      }
End
```

Figure 3. Test Suite Generation Algorithm

3. ACO Parameter tuning for t-way IOR

In order to adopt ACO into t-way testing strategy, few parameters tuning has to be made. Five parameters are required to be tuned to find the best value that can produce the optimum or near optimum test suite size. Table 1 shows parameters settings as suggested by [1], [18], [20], [21] and other additional values.

Table 1. Parameters settings for ACO

Parameter	Value
Ants	5,10,20,30
α	0.2,0.4,0.5,0.6,0.8, 1.0,1.5, 1.6, 2.0
β	0.5 , 0.2, 2 – 5 , 2.0
ρ	0.1, 0.2,0.4,0.5, 0.6, 0.8
τ_0	0.2, 0.4, 0.5, 0.8,1.0, 1.5
q_0	0.2, 0.4, 0.5, 0.6, 0.8, 0.9, 1.0

To tune the parameters as in Table 1, two configurations from benchmarked experiments for IOR [16] have been executed. The configurations are as follows:

- a uniform configuration IOR (N, 3^{10} , R) - consists of 10 3-valued parameters and 10 input parameters where the relation, R = [{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}].
- a non-uniform configuration IOR (N, 2^3 $3^3 4^3 5^1$, R) – consists of 3 2-valued, 3 3-valued, 3 4-valued and 1 5-valued parameters and 20 input parameters where the relation, R = [{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}].

Each parameter has to perform 10 independent executions to get the best test suite size, the average of test suite size and average of execution time. This is due to our strategy utilizes a random q value to determine probability value that results different test suite size per execution (i.e. run).

4. Results and Discussion

The experiments' goal is to find the best value for each parameter. Value which generates the smallest average test suite size and best test suite size is the best value. Even though the aim of any strategy developed for t-way testing is to find the best test suite size, this experiments are concern with both average test suite size and best size. The smallest gap between average test suite size and best size shows that the value of parameter is consistent in generating the best size. This section will presents a graph for each parameter. Each graph will present test suite size generated based on value of parameter. The right axis presents an average test suite size and best test suite size for configuration 2 (i.e. non-uniform), while left axis presents an average test suite size and best test suite size for configuration 1 (i.e. uniform). In order to find the best value for each parameter, the differences between average test suite sizes and best sizes are calculated. To ease the graph presentation, configuration 1 represents uniform configuration, while configuration 2 represents non-uniform configuration.

Figure 4 shows the best test suite size, average test suite size and best test suite size based on α values for configuration IOR (N, 3^{10} , R) and IOR (N, 2^3 $3^3 4^3 5^1$, R). As can be seen in the graphs, both lines for average test suite size fluctuate with the smallest size at 0.5. The best test suite size for α parameter is unchanged except for value 1.5, it climbs dramatically at 1.5. Value 2 shows the smallest gap between average test suite size and best size for configuration 1 (i.e. IOR (N, 3^{10}, R)) while value 0.5 is the best value for configuration 2 (i.e. IOR (N, 2^3 $3^3 4^3 5^1$, R).). However, some researchers found that alpha value greater than 1 will cause the stagnation in the algorithm [21]. Although the study was about a general ant colony algorithm, the suggestion should be considered. In that case, 0.5 is most suitable value for α parameter.

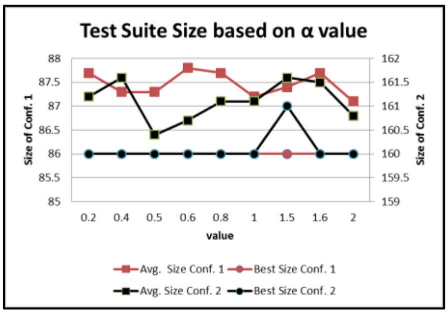


Figure 4. Test suite size based on α value

The second parameter, β parameter is given in Figure 5. A slight fluctuate graph is shown for both configurations except for a linear graph for configuration 2's best size. As discussed earlier, the smallest test suite size the better the value parameter, which implied that value 3 is the most suitable β parameter. The value also has been suggested by [21]. Parameter β, similar to α, involves in calculation of probability.

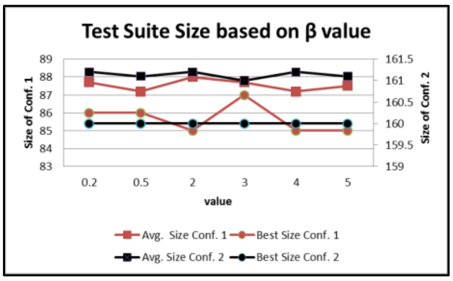


Figure 5. Parameter β values

Next parameter, τ_0 can be seen in Figure 6. It shows a similar trend as in Figure 5. There are three values recorded the smallest gap between average test suite size and best size. Value 0.4 shows the smallest gap for configuration 1, while value 0.2 and 1 hit the smallest gap for configuration 2. Value 0.4 has been suggested by [1]. This parameter is important and required during initialization. It contributes to the first move of the ant.

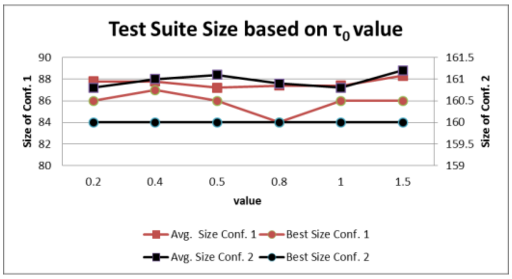


Figure 6. Parameter τ_0 values

Parameter q_0 in Figure 7 illustrates inconsistent results for the graph except the best size for configuration 2. The smallest gap between average test suite size and best size is at value 0.8. The value also reaches the smallest best size for configuration 1. However, the value illustrates the worst gap for configuration 1. Value of 0.5 shows a consistent gap between average test suite size and best size for both configurations. The value is suggested by [18]. Parameter q_0 affects ants to explore new path or exploit similar path chosen by previous ant.

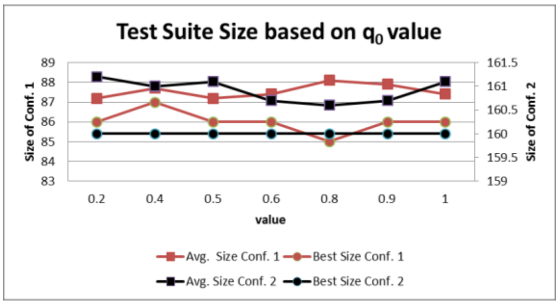


Figure 7. Parameter q_0 values

As compared to other parameters, parameter ρ projected a steady line for both average test suite size and best size from 0.1 until 0.8 for both configurations as in Figure 8. However, there has been a steep rise for ρ value is 1. Value of 0.5 shows the smallest gap between the two sizes. The value also is suggested by [1]. Parameter ρ is a decay value and it affects global pheromone value in the algorithm.

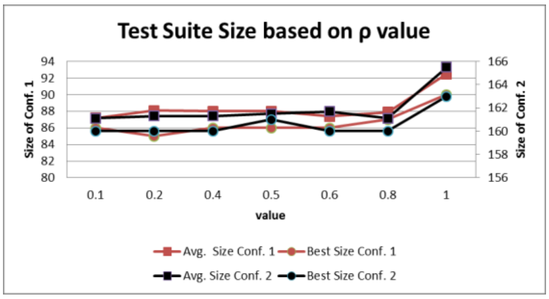


Figure 8. Parameter ρ values

The last parameter is number of ant as presented in Figure 9. As can be seen, the more ants used in the strategy, the smaller test suite size is generated by the strategy. Obviously, value of 40 shows the smallest gap between average test suite size and best size. However, time generation is increasing when the number is ant increased. Because of this, value of 20 is most suitable value as it shows a consistent gap for both configurations. Besides that, more time can be saved rather than value of 40.

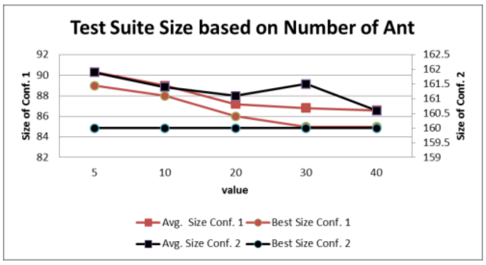


Figure 9. Parameter number of ant values

From 10 independent run of each configuration, the average test suite size and time generated has been calculated. Out of 10 results, the best size was recorded. This is the reason some of the results of the best test suite size are differ from the best average test suite size. Another factor of this situation is due to randomness in the strategy. Even though the value of parameter could generate the best test suite size, it cannot be concluded that the value also will generate the best average test suite size.

5. Conclusion

In this work, six parameters of ACO were tuned to suit t-way IOR. This was performed by experimenting 10 independent runs for each parameters to two configurations, uniform configuration IOR (N, 3^{10}, R) and non- uniform configuration IOR (N, 2^3 $3^3 4^3 5^1$, R). From the experiments, average test suite size and best test were recorded. It was found that the best value for α parameter = 0.5, β = 3, τ_0 = 0.4 (for uniform configuration) and, 0.2 and 1 (for non-uniform configuration), q_0 = 0.5, ρ = 0.5 and number of ant = 20. Those values were recorded to be the best value for both configurations except τ_0 parameter that has two different values for both configurations. Even though value for each parameter has been suggested by some researchers, but none of them has suggested for t-way IOR. Because of that, this work introduced ACO parameters for t-way IOR. Therefore, for further works, experiments based on benchmark experiments will be executed based on the values.

6. References

[1] T. Shiba, T. Tsuchiya, and T. Kikuno, “Using artificial life techniques to generate test cases for combinatorial testing,” in *Proc. of the 28th Annual Int. Computer Software and Applications Conference, 2004. COMPSAC 2004.*, 2004.

[2] 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.

[3] M. B. Cohen, “A Study in Prioritization for Higher Strength Combinatorial Testing,” in *2013 IEEE 6th Int. Conf. on Software Testing, Verification and Validation Workshops*, 2013.

[4] 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.

[5] R. Othman and K. 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.

[6] R. R. Othman, K. Z. Zamli, S. Mashita, and S. Mohamad, “T-Way Testing Strategies : A Critical Survey and Analysis,” *Int. J. Digit. Content Technol. its Appl.*, vol. **7**, May, pp. 222–235, 2013.

[7] K. Z. Zamli, R. R. Othman, M. I. Younis, and M. H. Mohamed Zabil, “Practical Adoptions of T-Way Strategies for Interaction Testing,” in *Int. Conf. on Software Engineering and Computer Systems*, 2011, pp. 1–14.

[8] A. a Al-sewari and K. Z. Zamli, “An Orchestrated Survey on T-Way Test Case Generation Strategies Based on Optimization Algorithms,” in *The 8th Int. Conf. on Robotic, Vision, Signal Processing & Power Applications*, 2014, vol. **291**, pp. 255–263.

[9] P. J. Schroeder, “Black-box Test Reduction using Input-Output Analysis,” Illinois Institute of Technology, 2001.

[10] C. Cheng, A. Dumitrescu, and P. Schroeder, “Generating small combinatorial test suites to cover input-output relationships,” in *Third Int. Conf. on Quality Software*, 2003, pp. 76–82.

[11] W. Ziyuan, X. Baowen, and N. Changhai, “Greedy Heuristic Algorithms to Generate Variable Strength Combinatorial Test Suite,” in *The 8th Int. Conf on Quality Soft.*, 2008, pp. 155–160.

[12] P. J. Schroeder, E. Kim, J. Arshem, P. Bolaki, and W. Milwaukee, “Combining Behavior and Data Modeling in Automated Test Case Generation,” in *3rd Int. Conf. On Quality Software (QSIC’03)*, 2003, pp. 1–8.

[13] R. R. Othman and K. Z. Zamli, “ITTDG : Integrated T-way test data generation strategy for interaction testing,” *Sci. Res. Essays*, vol. **6**, no. 17, pp. 3638–3648, 2011.

[14] H. Y. Ong and K. Z. Zamli, “Development of interaction test suite generation strategy with input-output mapping supports,” *Sci. Res. Essays*, vol. **6**, no. 16, pp. 3418–3430, 2011.

[15] W. Ziyuan, C. Nie, and X. Baowen, “Generating Combinatorial Test Suite for Interaction Relationship,” in *4th Int. workshop on Soft. Quality Assurance: in conjunction with the 6th ESEC/FSE joint meeting*, 2007, pp. 55–61.

[16] A. A. Alsewari, N. M. Tairan, and K. Z. Zamli, “Survey on Input Output Relation Based Combination Test Data Generation Strategies,” *ARPN J. Eng. Appl. Sci.*, vol. **10**, no. 18, pp. 8427–8430, 2015.

[17] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theor. Comput. Sci.*, vol. **344**, no. 2–3, pp. 243–278, 2005.

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

[19] N. Ramli, R. R. Othman, and M. S. A. Rashid Ali, “Optimizing Combinatorial Input-Output Based Relations Testing using Ant Colony Algorithm,” in *3rd Int. Conf. on Electronic Design (ICED), August 11-12, 2016, Phuket, Thailand*, 2016, pp. 586–590.

[20] X. Chen, Q. Gu, X. Zhang, and D. Chen, “Building prioritized pairwise interaction test suites with ant colony optimization,” in *Int. Conf. Qual. Softw.*, pp. 347–352, 2009.

[21] M. Dorigo and T. Stützle, *Ant Colony Optimization*. The MIT Press.