A Modified Artificial Bee Colony Based Test Suite Generation Strategy for Uniform T-Way Testing

Mohd Shaiful Aziz Rashid Ali, Rozmie Razif Othman, Zainor Ridzuan Yahya and Mohd Zamri Zahir

School of Computer and Communication Engineering, Kampus Pauh Putra, Universiti Malaysia Perlis, 02600 Arau, Perlis, Malaysia.

E-mail: shaifulaziz@unimap.edu.my, rozmie@unimap.edu.my, zainoryahya@unimap.edu.my, zamri.zahir@yahoo.com.

Abstract. Today, t-way testing has been widely known with the ability to reduce test suite size compared to exhaustive testing. At the same time, it has been proven by many researchers to provide maximum bug detection capability. Thus, various t-way strategies were developed since the past three decades. The paper proposed a new test generation strategy, named Modified Artificial Bee Colony T-Way Test Suite Generation (MABCTS). It supports uniform strength t-way testing. Experimentation results are compared with present strategies and produced comparable results. Since t-way testing is considered an NP-hard problem, there are no strategies that can be demanded to produce the best results.

Keywords: Artificial bee colony, uniform strength, test suite, t-way testing.

1. Introduction

Every day, we are dealing with software, whether it is directly or indirectly. For example, when we use the washing machine, we are indirectly using the software imbedded inside the machine by pressing the buttons on the panel. This appliance is controlled by the software in the controller board. To ensure the software interact with the inputs and outputs of the machine, the test engineer must test the software correctly. Exhaustive testing is the best solution where all of the possible combinations of configurations are tested to detect any faulty interactions. Unfortunately, this testing is impossible to be completed due to the time and resources constraint especially the manufacturing cost. Consider a device with 20 inputs with two possible values. If the whole testing process is applying exhaustive testing, there are $2^{20}$ (1,048,576) test cases to execute.

As a consequence, there are several t-way generation test suite strategies that have been recommended by researchers. It will aid in generating a minimal test suite to cover all intended interaction strengths [1]. Strategies such as Automatic Efficient Test Generator (AETG) [2], Myra Implementation of AETG (mAETG) [3], In-Parameter Order (IPO) [4], In-Parameter-Order General (IPOG) [5] and Jenny [6] generate test cases for uniform strength t-way and pure computational based. [1, 7]

At the same time, several researchers have attempted to use the metaheuristic technique in generating t-way test suite. Shiva began the work by applying the Genetic Algorithm (GA) [8, 9] and Ant Colony Optimization (ACO) [10] to produce uniform strength t-way test suite. It was then continued by Simulated Annealing (SA) [11, 12], Particle Swarm Optimization (PSTG) [13], Harmony Search Strategy (HSS) [14-17], Cuckoo Search (CS) [18] and others until recently it has been occupied by
Hybrid Hyper Heuristic (HHH) [19]. From the literature review, test suite size can be improved when the metaheuristic in t-way strategies [7, 19] is applied. These results suggest that there is a room further improvement, this paper manage to produce a solution to generate an optimum test suite by adopting an artificial bee colony as a searching algorithm. The original artificial bee colony was modified to suit with the T-way problem as well as to act as a balance between exploration and exploitation.

This paper is structured as follows. Section 2 describes the general idea of T-way testing, while Section 3 explains the proposed strategy. It is followed by the presentation of results and discussion in Section 4. Lastly, in section 5, conclusions were stated.

2. The general idea of t-way testing

In order to further explain the idea of t-way, try to imagine an embedded system consisting of 4 toggle switches as inputs. We name all switches with the letters A, B, C and D, as typically known, by toggling the switch it gives two possible values ("High" and "Low"). Table 1, represents all possible values with symbolic representation of the switches.

To ensure that the system is working accurately, exhaustive testing can be applied to test all possible combination of inputs. There are 16 tests cases (2^4=16) that can be generated from the exhaustive testing where it can be referred to in table 2.

Table 1. Symbolic representation for input switches of embedded system

| Input Parameter | A | B | C | D |
|-----------------|---|---|---|---|
| Symbolic Representation | a1 | b1 | c1 | d1 |

| Test Case | A | B | C | D |
|-----------|---|---|---|---|
| 1         | a1 | b1 | c1 | d1 |
| 2         | a1 | b1 | c1 | d2 |
| 3         | a1 | b1 | c2 | d1 |
| 4         | a1 | b1 | c2 | d2 |
| 5         | a1 | b2 | c1 | d1 |
| 6         | a1 | b2 | c1 | d2 |
| 7         | a1 | b2 | c2 | d1 |
| 8         | a1 | b2 | c2 | d2 |
| 9         | a2 | b1 | c1 | d1 |
| 10        | a2 | b1 | c1 | d2 |
| 11        | a2 | b1 | c2 | d1 |
| 12        | a2 | b1 | c2 | d2 |
| 13        | a2 | b2 | c1 | d1 |
| 14        | a2 | b2 | c1 | d2 |
| 15        | a2 | b2 | c2 | d1 |
| 16        | a2 | b2 | c2 | d2 |
As an alternative method, T-way testing can be applied to test all possible combination inputs. In t-way testing, the interaction strength (t) must be decided first before the test suite is generated. The number of test cases can be decreased when lower interaction strength is picked. The ‘t’ can range from 2 until maximum number of input parameters is reached. In this example, we will choose two as the interaction strength. The 2-way test suite will be generated, it is commonly known as pairwise. AB, BC, CD, AC, AD and BD are the combination of input parameters that have produced tuples that was charted in Figure 2. The result produced in figure 2 is considered as only tuples which was covered by each combination. By applied t-way testing method, 25 per cent of test cases can be reduced compared to exhaustive testing. Only 9 test cases are required to test the complete system. The most glaring advantage of using the t-way testing is it will reduce the number of test cases will severely save time and countless resources.

The covering array notation is used to represent the final test suite (F) as in Equation 1. Therefore for the system in Figure 1, the final test suite is written as CA (9, 2, 2^4).

\[ F = CA(N; t, C) \]

Where,

- \( N \) = the number of test data inside the final test suite.
- \( t \) = the interaction strength
- \( C \) = value configuration can be represented as following:
- \( v_{f^1}, v_{f^2}, ..., v_{f^m} \) which indicates that there are \( p_1 \) parameters with \( v_1 \) values, \( p_2 \) parameters with \( v_2 \) values and continue

![Combining each 2-way combinations](image)

**Figure 1.** Uniform Strength with two interaction strength

3. Proposed strategy

Karaboga has introduced the artificial Bee Colony (ABC) algorithm in 2005[21]. Since then, it has been recommended in solving problems which involve non-linear optimization[22]. Fundamentally, the idea of adopting ABC is by mimicking the behaviour of bees on how they find a food source for their hive. The grouping of bees according to ABC is characterized into three different characters as below:

- Employer bees are the workers who find the food source and bring the information to the onlooker bees.

![Base Values](image)
Onlooker bees are in charge to evaluate the quality of the food source. The evaluation process is determined by researching the Waggle dance move via the employer bees. Scout bees are bees that randomly search for a new food source. The employer bees on the hand can transform into scout bees if the current food source is exhausted.

As explained earlier on the matter of exploitation and exploration of the bees’ behaviour, this algorithm focuses more on exploitation compared to exploration. The restriction on the exploration behaviour is that it only occurs once in the scout bee phase compared to the exploitation behaviour which happened twice in the employer and onlooker bee phase. This fact is the main reason why we are choosing the ABC as the core of our searching algorithm in generating the optimum test suite.

By relating the ABC algorithm with our problem, tuples produced from exhaustive testing can become one of the potential solutions or in ABC it is known as a food source. Any tuples are randomly chosen from the search space. The number of food sources is equal to the number of employer bees. Afterwards each food sources will be assigned to one employer bee.

$$x_{ij} = x_{j\,\text{min}} + \text{rand}(0,1).\left(x_{j\,\text{max}} - x_{j\,\text{min}}\right)$$  \hspace{1cm} (2)

Where:
- $x_{ij}$: Initialization possible solutions i with parameter j
- $x_{j\,\text{min}}$: The value of the smallest possible solutions based on parameters j.
- $x_{j\,\text{max}}$: Greatest value possible solutions based on parameters j
- $\text{rand}(0,1)$: A random value between 0 and 1
- $i = 1 \ldots SN$, SN is the number of possible solutions (food source).
- $j = 1 \ldots D$, D is the number of parameters

These employer bees will exploit their position in finding food source hence they will find the best local optimum source in their area.

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right)$$  \hspace{1cm} (3)

Where:
- $v_{ij}$: New food source
- $x_{ij}$: the value of the possible solution i with parameter j
- $i = 1 \ldots SN$, SN is the number of possible solutions (food source)
- $j = 1 \ldots D$, D is the number of parameters
- $k = 1 \ldots SN$, SN is the number of parameters used
- $\phi_{ij}$: Random real number between [-1, 1]

Then the amount of nectar for each food source will be evaluated by using the fitness function. The function has been modified to suit our problem, $f_n$ was given by:

$$f_n = \sum_{k=0}^{\text{Total numbers of tuples pair}} w_k$$  \hspace{1cm} (4)

Where:
- $w_k$ is the number of uncovered tuples for kth uniform strength.

The best food source has the highest amount of nectar which is determined by the onlooker bees. This indicates the best solution or the best tuples pair which can cover the highest number of test cases.
If a solution is not improved further through a predetermined number of cycles, then that food source is assumed to be exhausted. This exhausted food source will be replaced by a scout bee. A standard searching method by a scout bee is by randomly searching. The method has a high potential to be trapped in a local minima or a cycle of repeated old solutions. The second reason is that there is too much exploitation in the employee and onlooker bee phases hence it might not cover the whole solution space.

For these reasons, a mutation is applied to replace the random search. As a result, a new food source will be explored in a new region from the search space. This modification will overcome the problem of being trapped in the local minima and avoid the possibility of returning back the same location.

As an example, we select test case 2 [1, 1, 1, 2] in Table 2. We take the first parameter to be flipped as below.

Step 1: Finding the number of bit binary to represent. The formula is:

\[ N = \log_2 (UB + 1) \]  

(5)

Where:

- \( N \) = number of bits
- \( UB \) = upper bound

In this example \( UB = 2 \), so

\[ N = \log_2 (2 + 1) \]

=1.58 (round it up to next integer)

=2

Step 2: Convert the first parameter into a binary and then flip all bits. Finish, return to decimal. Now the new value is two and a new food source [2, 1, 1, 2]. By referring to Table 2, the position has jumped to number 10. This proves that the embedded mutation in the scout bee phase has better exploration process.

Two scenarios are in need of a normalized process. In the first scenario, the mutation result is above the upper bound. In this case, the value from mutation is subtracted with the upper bound. In another scenario, mutation result is less than the lower bound. Therefore it needs to be added with the lower bound. Either one is repeated until the result obtained is between the ranges.

Finally, the ending criteria of MABCTS will be checked. The generation process will be stopped when there is no further uncovered tuples. Otherwise, the searching algorithm will be repeated. The flow chart in figure 2 summarizes the proposed method.
Figure 2. Proposed MABCTS flow chart
4. Results

Table 3. Benchmarking CA configurations. [19]

| System configuration | Meta-heuristic-based strategies | General Computational-based Strategies |
|----------------------|---------------------------------|----------------------------------------|
|                      | MABCTS  | HHH    | HSS    | PSTG   | CS    | SA    | GA    | ACO    | mAETG | AETG | IPOG | Jenny | TGV |
| CA(N;2,3⁴)           | 9       | 9      | 9      | 9      | 9     | 9     | 9     | 9      | 9     | 9    | 9    | 10   | 11   |
| CA(N;2,3¹²)          | 18      | 17     | 18     | 17     | 20    | 16    | 17    | 17     | 17    | 15   | 20   | 20   | 19   |
| CA(N;2,5¹⁰)          | 42      | 42     | 43     | 45     | NA    | NA    | NA    | NA     | NA    | NA   | 50   | 45   | 51   |
| CA(N;3,3⁶)           | 39      | 33     | 39     | 42     | 43    | 33    | 33    | 33     | 38    | 47   | 53   | 51   | 49   |
| CA(N;3,4⁶)           | 96      | 64     | 70     | 102    | 105   | 64    | 64    | 64     | 77    | 105  | 64   | 112  | 123  |

*Bold value specifies the most minimum results (i.e. optimal)*

Table 4. CA (N; t, 2¹⁰) with t varied from 2 to 6.[19]

| t   | Meta-heuristic-based strategies | General Computational-based Strategies |
|-----|---------------------------------|----------------------------------------|
|     | MABCTS  | HHH    | HSS    | PSTG   | CS    | IPOG | ITCH | Jenny | TConfig | TVG |
| 2   | 8       | 8      | 7      | 8      | 8     | 10   | 6    | 10    | 9       | 10  |
| 3   | 36      | 36     | 37     | 37     | 36    | 19   | 18   | 18    | 20      | 17  |
| 4   | 78      | 79     | 81     | 82     | 79    | 49   | 58   | 39    | 45      | 41  |
| 5   | 153     | 153    | 158    | 158    | 157   | 128  | NA   | 87    | 95      | 84  |
| 6   | 153     | 153    | 158    | 158    | 157   | 352  | NA   | 169   | 183     | 168 |

*Bold value specifies the most minimum results (i.e. optimal)*
### Table 5. CA (N; t, 510) with t varied from 2 to 6. [19]

| t  | Meta-heuristic-based strategies | General Computational-based Strategies |
|----|---------------------------------|----------------------------------------|
|    | MABCTS  | HHH   | HSS   | PSTG  | CS   | IPOG | ITCH | Jenny | PICT | TConfig | TVG | GTWay | MIPOG |
| 2  | 42      | 42    | 43    | 45    | 45   | 50   | 45   | 45    | 47   | 48      | 50  | 46    | 45    |
| 3  | 271     | 280   | 276   | 287   | 297  | 313  | 225  | 290   | 310  | 312     | 342 | 293   | 281   |
| 4  | 1623    | 1638  | 1624  | 1716  | 1731 | 1965 | 1750 | 1719  | 1812 | 1878    | 1971| 1714  | 1643  |
| 5  | 8524    | 8704  | 8866  | 9425  | 9616 | 11009| NA   | 9437  | 9706 | NA      | NA  | 9487  | 8169  |
| 6  | 42886   | 47800 | 47550 | 50350 | 50489| 57290| NA   | NA    | 47978| NA      | NA  | 44884 | 45168 |

*Bold value specifies the most minimum results (i.e. optimal)*

### Table 6. CA (N; 4, v10) with v varied from 2 to 7[19]

| v  | Meta-heuristic-based strategies | General Computational-based Strategies |
|----|---------------------------------|----------------------------------------|
|    | MABCTS  | HHH   | HSS   | PSTG  | CS   | IPOG | ITCH | Jenny | PICT | TConfig | TVG | GTWay | MIPOG |
| 2  | 36      | 36    | 37    | 34    | 28   | 49   | 58   | 39    | 43   | 45      | 40  | 46    | 43    |
| 3  | 208     | 207   | 211   | 213   | 211  | 241  | 336  | 221   | 231  | 235     | 228 | 224   | 217   |
| 4  | 660     | 668   | 691   | 685   | 698  | 707  | 704  | 703   | 742  | 718     | 782 | 621   | 637   |
| 5  | 1623    | 1635  | 1624  | 1716  | 1731 | 1965 | 1750 | 1719  | 1812 | 1878    | 1917| 1714  | 1643  |
| 6  | 3375    | 3405  | 3475  | 3880  | 3894 | 3935 | NA   | 3519  | 3735 | NA      | 4159| 3514  | 3657  |
| 7  | 6215    | 6412  | 6398  | NA    | NA   | 7061 | NA   | 6462  | NA   | NA      | 7854| 6459  | 5927  |

*Bold value specifies the most minimum results (i.e. optimal)*
To validate the performance of the strategy, MABCTS is compared with existing strategies which is by grouping it in two, meta-heuristic-based strategies and general Computational-based Strategies. The configuration is set up to meet the standard benchmarking [19]. Only the final number of the test suite is evaluated. Table 3, Table 4, Table 5 and Table 6 represent those results.

Table 3 shows results for the interaction strength $t \leq 3$, which is the low interaction strength. At this interaction strength, MABCTS’s performance is average. At covering arrays CA(N; 2, 3$^4$) and CA(N; 2, 5$^{10}$) are best configurations for MABCTS compare with others. SA strategy can be assumed as the best for most configurations.

For Table 4 and 5, the covering arrays CA (N; t, 2$^{10}$) and CA (N; t, 5$^{10}$) with t varied from 2 to 6. From the result, MABCTS is showing the best result with other strategies. For the last result in Table 6, MABCTS has improved the results when more input values are configured.

5. Conclusion and Future Works
In this study, a new strategy for t-way test suite generation has been proposed based on the Modified ABC algorithm. As mentioned earlier, problems generated by the test suite is considered NP-hard problem. It can be seen in the results where none of the strategy’s best result can be used for all configuration and interaction strength. Therefore, this can be improved upon as a niche for future studies and the MABCTS strategy can further be optimized for other types of t-way strength.

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

References
[1] Othman RR, Zamli KZ. T-way strategies and its applications for combinatorial testing. International Journal of New Computer Architectures and their Applications. 2011;1(2):459-73.
[2] Cohen DM, Dalal SR, Kajla A, Patton GC. The automatic efficient test generator (AETG) system. InProceedings of 1994 IEEE International Symposium on Software Reliability Engineering 1994 Nov 6 (pp. 303-309). IEEE.
[3] Cohen DM, Dalal SR, Fredman ML, Patton GC. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering. 1997 Jul;23(7):437-44.
[4] Lei Y, Tai KC. In-parameter-order: A test generation strategy for pairwise testing. InProceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No. 98EX231) 1998 Nov 13 (pp. 254-261). IEEE.
[5] Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J. IPOG: A general strategy for t-way software testing. In14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07) 2007 Mar 26 (pp. 549-556). IEEE.
[6] Jenkins RJ. Jenny: a pairwise testing tool [Internet]. 2005 Available from: http://burtleburtle.net/bob/math/jenny.html [Accessed 16 Nov. 2019].
[7] Al-Sewari AA, Zamli KZ. An orchestrated survey on t-way test case generation strategies based on optimization algorithms. InThe 8th International Conference on Robotic, Vision, Signal Processing & Power Applications 2014 (pp. 255-263). Springer, Singapore.
[8] Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing. InProceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004. 2004 Sep 28 (pp. 72-77). IEEE.
[9] McCaffrey JD. An empirical study of pairwise test set generation using a genetic algorithm. In2010 Seventh International Conference on Information Technology: New Generations 2010 Apr 12 (pp. 992-997). IEEE.
[10] Chen X, Gu Q, Li A, Chen D. Variable strength interaction testing with an ant colony system
approach. In 2009 16th Asia-Pacific Software Engineering Conference 2009 Dec 1 (pp. 160-167). IEEE.

[11] Stardom J. *Metaheuristics and the search for covering and packing arrays*. Burnaby: Simon Fraser University; 2001 May 16.

[12] Colbourn CJ, Ling AC. *Constructing strength three covering arrays with augmented annealing*. Discrete Mathematics. 2008 Jul 6;308(13):2709-22.

[13] Ahmed BS, Zamli KZ, Lim CP. *Constructing a t-way interaction test suite using the particle swarm optimization approach*. International Journal of Innovative Computing, Information and Control. 2012 Jan 1;8(1):431-52.

[14] Alsewari AA, Younis MI, Zamli KZ. *Generation of pairwise test sets using a harmony search algorithm*. Comput Sci Lett. 2011 Apr 2;3(1).

[15] Alsewari AA, Zamli KZ. *Interaction test data generation using harmony search algorithm*. In 2011 IEEE Symposium on Industrial Electronics and Applications 2011 Sep 25 (pp. 559-564). IEEE.

[16] Alsewari AR, Zamli KZ. *Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support*. Information and Software Technology. 2012 Jun 1;54(6):553-68.

[17] Alsewari AR, Zamli KZ. *A harmony search based pairwise sampling strategy for combinatorial testing*. International Journal of Physical Sciences. 2012 Feb 9;7(7):1062-72.

[18] Yang XS, Deb S. *Cuckoo search via Lévy flights*. In 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC) 2009 Dec 9 (pp. 210-214). IEEE.

[19] Zamli KZ, Alkazemi BY, Kendall G. *A tabu search hyper-heuristic strategy for t-way test suite generation*. Applied Soft Computing. 2016 Jul 1;44:57-74.

[20] Ali MS, Othman RR, Yahya ZR, Ahmad MZ, Ramli N. *Implementation of artificial bee colony algorithm for T-way testing*. In 2016 3rd International Conference on Electronic Design (ICED) 2016 Aug 11 (pp. 591-594). IEEE.

[21] Karaboga D, Akay B. *A comparative study of artificial bee colony algorithm*. Applied mathematics and computation. 2009 Aug 1;214(1):108-32.

[22] Lindfield G, Penny J. *Introduction to Nature-Inspired Optimization*. Academic Press; 2017 Aug 10.