VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

A BAT-INSPIRED STRATEGY FOR PAIRWISE TESTING

Yazan A. Alsariera, Mazlina A. Majid and Kamal Z. Zamli
Software Engineering Research Group, Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Gambang, Kuantan Pahang, Malaysia
E-Mail: kamalz@ump.edu.my

ABSTRACT

Owing to exponential growth of software lines of codes (LOC)s, testing becomes painstakingly difficult activities. Test engineers are often under pressure to test more and more LOCs yet within the same targeted deadline. For this reason, efficient testing strategy is required. Pairwise testing is amongst the most common strategies for minimizing and sampling of tests for testing consideration. Recently, there are growing interests for adapting optimization algorithms as the basis of the newly developed strategies. Complementing the existing work, we propose a novel design and implementation of Bat-inspired algorithm (BA) for pairwise strategy, called Bat-inspired pairwise testing strategy (BPTS). Based on the benchmarking results, BPTS outperforms most existing strategies in terms of the generated test suite size. BPTS serves as our research vehicle to investigate the effectiveness of Bat-inspired algorithm for pairwise test generation, which is going to be helpful to reduce the time and cost of software testing by reducing the number of test cases.

Keywords: bat algorithm, meta-heuristics optimization algorithms, pairwise testing, combinatorial explosion problem.

INTRODUCTION

Software testing can be viewed as the gatekeeper of quality, that is, in term of minimizing the risk software failure (Alsewari and Zamli 2012). Considering the need to deal with ever increasing user requirements (and exponential growth of software lines of codes (LOC)), exhaustive testing is practically impossible. Recent studies have suggested that the use of 2-way (or pairwise) testing can be effective to systematically sample data for testing consideration (Soh, Abdullah *et al.* , Kuhn, Wallace *et al.* 2004, Kuhn, Lei *et al.* 2008). Complementing existing sampling strategies (such as equivalence partitioning, boundary value analysis), pairwise testing focuses on the bug due to 2-way interaction between two parameters within a typical software of interest.

Several strategies have been adopted for pairwise testing in the past, including TConfig (Williams 2000), Test Vector Generator (TVG) (Arshem 2009) Pairwise Independent Combinatorial Testing (PICT) (Czerwonka 2006), Classification-Tree Editor eXtended Logics (CTE-XL) (Lehmann and Wegener 2000), In Parameter Order Generator (IPOG) (Lei, Kacker *et al.* 2007), Jenny (Pallas 2003), Particle Swarm Test Generator (PPSTG) (Ahmed and Zamli 2011) and pairwise Harmony Search algorithm-based strategy (PHSS) (Alsewari and Zamli 2012), Ant Colony algorithm (ACA) (Shiba, Tsuchiya *et al.* 2004), genetic algorithm (GA) (McCaffrey 2009, Huang, Cohen *et al.* 2010, McCaffrey 2010), Simulated Annealing (SA) (Cohen, Gibbons *et al.* 2003) and Harmony Search (HS)(Alsewari and Zamli 2012)). While most strategies give competitive results, recent benchmarks (as published in (Alsewari and Zamli 2012) and (Ahmed and Zamli 2011)) demonstrate the use of optimization algorithms as the backbone engine of the pairwise strategy prove to be more superior in terms of the test suite size than that of the computational based counterparts. For this reason, we have opted to explore the use of optimization algorithms further.

The use of Bat Algorithm appears alluring given that its use has not been sufficiently explored as the basis of a pairwise strategy. Furthermore, the superiority of the Bat Algorithm over existing counterparts is also commendable. For instance, Khan and Sahai (Khan and Sahai 2012) reported that the Bat Algorithm outperforms Particle Swarm Optimization, and Genetic Algorithm for training Artificial Neural Networks (ANNs) (Khan and Sahai 2012) within e-Learning Context. In other work, Yang (Yang 2010) also demonstrates that the Bat Algorithm gives the best performance as compared to PSO, GA, and Harmony Search against standard benchmark functions. In fact, Yang proves that PSO and Harmony Search can be considered as the generalization of the Bat Algorithm.

Given its potential, we propose a novel design and implementation of Bat-inspired algorithm (BA) for pairwise strategy, called Bat-inspired pairwise testing strategy (BPTS). Based on the benchmarking results, BPTS outperforms most existing strategies in terms of the generated test suite size. BPTS serves as our research vehicle to investigate the effectiveness of Bat-inspired algorithm for pairwise test generation.

The remainder of the paper is organized as follows. Section two reviews the existing related work for pairwise testing strategies. Section three outlines the Bat algorithm and elaborates on our implementation. Section four highlights the correctness of the implementation along with the benchmarking experiments. Finally, Section five provides our closing remarks along with our plan for further work.

RELATED WORK

In general, existing interaction strategies for pairwise testing can be categorized into two categories, that is, algebraic approaches or computational approaches respectively (Lei, Kacker *et al.* 2007). Algebraic approaches construct test sets using mathematical

properties of covering arrays (Lei, Kacker *et al.* 2007). For this reason, the computations involved in algebraic approaches are lightweight. On the negative note, the applicability of algebraic approaches is often restricted to small configurations (Yan and Zhang 2006, Lei, Kacker *et al.* 2007). Orthogonal arrays (OA) (Hedayat, Sloane *et al.* 1999, Hartman and Raskin 2004), MOA (Mandl 1985) and TConfig (Williams 2002) are a typical example of the strategies that are based on algebraic approach.

Test Vector Generator (TVG) (Arshem 2009) implements a public domain strategy supporting pairwise testing. Our experience exploring TVG indicates that the pairwise generation can be summoned based on three algorithms, namely, T-reduced algorithm, Plus-one algorithm, and Random sets algorithm respectively. Although useful, not much information can be implied as the details implementation has not been made available in the literature. Other pairwise strategy implementation that is lacking as far as documentation is concerned includes CTE_XL (Lehmann and Wegener 2000). Unlike, Jenny is another public domain strategy (Pallas 2003). In the nutshell, Jenny generates the pairwise test in stages. Firstly, Jenny generates test data to cover all the 1-way interaction. Then, Jenny will extend the first stage test data to greedily cover the 2-way interactions. Optionally, this process can continue until the nth-way interactions as specified by the user. Similarly, Pairwise Independent Combinatorial Testing (PICT) (Czerwonka 2006, Ahmed and Zamli 2011) is the public domain strategy implementation developed by Microsoft. Adopting random selection for completing the uncovered pair interaction, PICT often offers non-optimal results.

In Parameter Order (IPO) (Lei and Tai 1998) builds a pairwise test suite using horizontal and vertical extensions. Ideally, IPO works as follows. Firstly, IPO builds the pairwise tests for the first parameter. Then, IPO strategy extends the test set to cover the first three parameters, and continues to extend the test set no further horizontal extension is possible. If there are still uncovered pairs, the IPO will then proceed to its vertical extension essentially filling in the missing coverage. Recently, a number of IPO variants have developed by researchers (i.e., IPOG (Lei, Kacker *et al.* 2007), MIPOG, MC-MIPOG (Younis, Zamli *et al.* 2008, Younis and Zamli 2010, Younis and Zamli 2011) and IPOG-D (Lei, Kacker *et al.* 2008)).

Lastly, PPSTG (Ahmed and Zamli 2011) and PHSS (Alsewari and Zamli 2012) are two of the most recent pairwise strategies that are based on optimization algorithms. The former (i.e. PPSTG), a Particle Swarm based Optimization algorithm, iteratively performs local and global searches to find the candidate solution to be added to the final suite until all the pairwise interactions are covered. The latter (i.e. PHSS) is a strategy based on Harmony Search Algorithm. Like PPSTG, PHSS adopts both global and local search. Unlike PPSTG, PHSS adopts two probability values (i.e. the considering rate and pitch adjustment rate). Here, global search is iteratively performed by randomizing values in the Harmony memory whereby the local best value can be selected given a considering rate probability. Here, local best value can be considered for improvements for further improvements in the local search (i.e. with pitch adjustment probability). Upon completing each iteration, the best value will be added to the final test suite until all pairwise interactions are covered.

BAT-INSPIRED PAIRWISE TESTING STRATEGY (BPTS)

Bat-inspired algorithm (BA)

In a nutshell, BA is a population optimization algorithm founded on the hunting behavior of Microbats by using echolocation. Typically, every pulse lasts for a few milliseconds (i.e., 8-10ms) with a frequency of 25–150 kHz on consistent wavelengths of 2–14 mm. The algorithm has been built on the assumption that the bat is able to find its prey in complete darkness. The bat position represents a possible solution of the problem. The best position of a bat to its prey indicates the quality of the solution. Here, obstacles are avoided using echolocation. In such a case, different frequencies are returned. Generally, the BA has three main assumptions:

Assumption 1: All bats are using echolocation to intelligently calculate distance. They know the difference between food/prey and the surrounding environment background in a magical way.

Assumption 2: Bats are flying randomly using velocity vi at position xi. They automatically adjust emitted pulses and adjust the rate of pulse emission r ⌜ [0, 1], of their echolocation frequency.

Assumption 3: Although the loudness could be different in several ways, Here, it is assumed that the loudness change from a large (positive) A0 to a minimum value Amin.

[1].
[2]. Objective function $f(x_i)$, $_{xi} = (x_{i1}....x_{ii})^T$
[3]. Initialize the bat population x_i and velocities v_i for i = (1,2..no bats)
[4]. Define pulse frequency Q_i within $[Q_{min}, Q_{max}]$
[5]. Initialize pulse rates r_i and the loudness A_i
[6]. While ($t<T_{max}$)
Generate new solutions by adjusting frequency, and update velocities and locations [Eq. 1 to 3]
$f_i = f_{min} + (f_{max} + f_{min})\beta$ (1)
$v_i^{t+1} = v_i^t + (x_i^t - x_o)f_i$ (2)
$x_i^{t+1} = x_i^t + v_i^{t+1}$ (3)
if $(rand(0,1) > r_i)$
Select the best solution in the current population
Generate a local solution around the best solution
End if

if $(rand(0,1) < A_i and f(x_i) < f(x))$
Accept the new solutions
Increase r_i and reduce A_i
End if
Rank the bats and find the current best
End while
[7]. Process results and visualization

Figure-1. Pseudo code of the BA according to (Yang 2010).

The complete step of BA is shown in the pseudo code in Figure 1. Briefly, for every bat *i* a position *xi* and velocity *vi* in a D- dimensional search space that is randomly initialized. After the initialization for the first set of position *xi* and velocity *vi* the fitness is calculated for each solution. Those solutions are being updated and improved by each iterative using the equations 1-3 shown in Figure-1 until the best fitness for the problem is found.

The implementation of bat-inspired pairwise testing strategy (BPTS)

We have developed our strategy in the Java programming language. BA attributes have been initialized in sequence. We also introduce population lists to represent the interaction elements in our implementation. The class structure of our implementation can be seen in Figure-2.

Building from the aforementioned class implementation, our BPTS strategy consists of two main processes; generating process and bat-inspired based searching process respectively. As for generating process, BPTS initializes pairing criteria by matching all the possible combination binary value with the pair interaction elements.

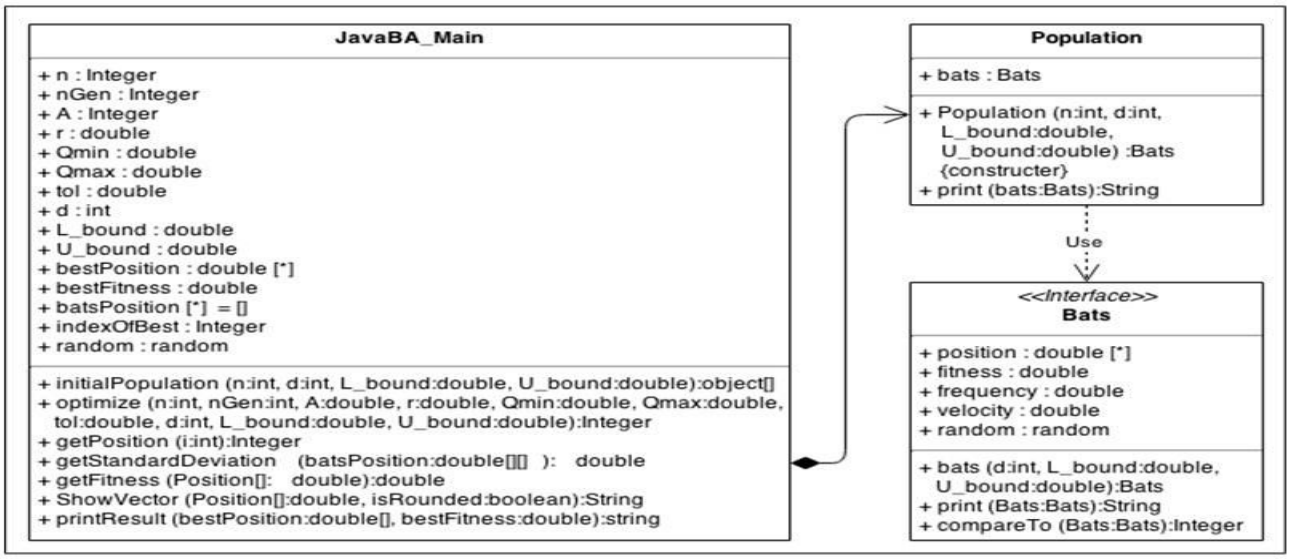

Figure-2. Class diagram for java BA implementation.

The searching process starts with the initialization of the bat population. BPTS then iterates each individual bat in search for the best fitness values. At each iteration, the test case stored in each individual bat is updated based on the updated individual frequency, velocity, and displacement (see the equations 1-3 on Figure-1). Upon completing the maximum iteration, the best bat (i.e. covering maximum uncovered tests) will be selected and be added to the final test list. Then, all the covered interaction elements are removed from the interaction elements list. Finally, the results and visualization for the final test suite is processed for auxiliary intentions (i.e. to view the result and make them readable). The pseudo code of the bat-inspired pairwise testing strategy (BPTS) can be seen in Figure-3.

```
[1]. Begin
[2]. Define empty interaction element list (IEL), final test
     list (FTL) BA attributes.
[3]. Generate all interaction elements (IE) for selected pair
     and store in IEL.
[4]. Initialize Bat population
[5]. Loop until IEL is empty
     Loop until the specified number of Bat generation
          Evaluate Bat fitness for all the IE's
          Select maximum fitness Bat as best.
          Loop until a specific number of Bat
               Calculate frequency f_i
               Get the v_{i+1} according to the old v_i, x_i and f_i
               Move x_i to x_{i+1} according to the new v_{i+1}
               if the best x_{i+1} cover more pair interaction
               Accept the new solutions Best x_i = Best x_{i+1}
               Increase r_i and reduce A_i
               End if
          End Loop
          Accept Best x_{i+1} test case as best case
     End Loop
     Add best test case to FTL
     Remove the covered IE from IEL
[6]. End loop
[7]. process results and visualization
```

Figure-3. The pseudo code of the BPTS.

EXPERIMENTAL RESULTS

Our experimental results consist of two parts. For the first part, the goal is to verify the correctness of our Bat algorithm implementation. For the second part, the goal is to benchmark against competing pairwise strategies as well as to investigate the effect of incremental changes of V and P values to BPTS.

As far as verifying the correctness of our implementation is concerned, we run the optimization tests for 6 selected benchmarking functions (Jamil and Yang 2013) and recorded the results. Here, we use the two diminutions for all functions and obtain the best and worst solution for each function. We also calculate the mean and standard deviation for the 20 runs result (for statistical significance).

Function-1. The Rastrigin's function

$$f_1(x) = 10D + \sum_{i}^{D} \left(x_i^2 - 10cos(2\pi x_i)\right) \quad (4)$$

Subject to [-5.12 ≤ *xi* ≤ 5.12].
It has global minimum $f(x^*) = 0$ at $x^* = (0, ..., 0)$.

Function-2. The Egg Crate benchmarking function.

$$f_2(x) = x_1^2 + x_2^2 + 25(sin^2(x_1) + sin^2(x_2)) \quad (5)$$

Subject to [-5 ≤ xi ≤ 5].
It has global minimum $f(x^*) = 0$ at $x^* = (0, 0)$.

Function-3. The frist Bohachevsky benchmarking function.

$$f_3(x) = x_1^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7 \quad (6)$$

Subject to [-100 ≤ *xi* ≤ 100].
It has global minimum $f(x^*) = 0$ at $x^* = (0, 0)$.

Function-4. The Booth benchmarking function

$$f_4(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \quad (7)$$

Subject to [-10 ≤ *xi* ≤ 10].
It has global minimum $f(x^*) = 0$ at $x^* = (1, 3)$.

Function-5. The Himmelblau benchmarking function.

$$f_5(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \quad (8)$$

Subject to [-5 ≤ xi ≤ 5].
It has a global minimum $f(x^*) = 0$ at $x^* = (3, 2)$.

Function-6. The Parsopoulos benchmarking function

$$f_6(x) = sin(x_1)^2 cos(x_2)^2 \quad (9)$$

Subject to [-5 ≤ *xi* ≤ 5].
It has 12 global minimum $f(x^*) = 0$ at $x^* \in R^2$.

Tables-1 until 3 demonstrates our results for all the experiments.

Concerning benchmarking of BPTS, we run two comparative experiments taken from (Ahmed and Zamli 2011) and (Alsewari and Zamli 2012). In the first experiment, we have used a system configuration with 10 V-valued parameters, where V is varied from 3 to 10. In the second experiment, we have used a system configuration with P 2-valued parameters, where P is varied from 3 to 15 (see Table-3).

For both parts of our experimental results, we use the following configuration; Intel® Core™ i7-3770 (3.40GHz, 3MB L3, 256KB L2, 32KB L1 cache) with 4GB of RAM on Windows 7 professional Operating System with Java SE version 8. The result of time performance is shown (in milliseconds).

The Bat algorithm takes the following attributes (or parameter) values. The population size nBats = 50, number of generations = 100, loudness = 0.9, rate of pulse emission Q = 0.7 with a frequency range of [0, 1] and tolerance = 0.001.

Regarding the first part, Table-1 demonstrates that our implementation of the Bat algorithm according to the pseudocode provided by Yang (Yang 2010). BA gives accurate results. In fact, we have done this experiment using the 6 benchmarking functions (Jamil and Yang 2013) to verify the correctness of our implementation. The result shows a very good solution quality compared to the actual minimum value.

As for the second part, Table-2 and 3 highlights the overall performance of BPTS against existing strategies. For both tables, the cell with an asterisk (*) depicts the optimal test suite size achieved by each strategy.

Referring to Table-2, BPTS's performance appears to be significantly better by gradual changes in V. With the exception of 10 9-valued parameters and 10 9-valued parameters where PHSS offers the best result. In the case of 10 [4 and 7]-valued parameters BPTS gives similar results as the overall best results offered by PHSS. BPTS outperforms all existing strategies with 10 3-valued parameters, 10 5-valued parameters, 10 6-valued parameters and 10 8-valued parameters. BPTS offers the best overall results, Moreover, BPTS shows a high reduction in the case of 10 8-valued parameters where it outperform Jenny with redaction of two test cases. BPTS outperforms all existing strategies in most cases considered.

Considering Table-3, we observe that BPTS gives the best overall result. At a glance, BPTS matches with the best results in most cases except in three instances (4 2-valued parameters, 6 2-valued parameters, and 8 2-valued parameters respectively). A closer look reveals that BPTS outperforms all existing strategies in the cases of high P from 12 onwards (i.e. where P is the number of parameters) suggesting its superiority for the high number of parameters.

Table-1. Result of BA for six functions.

Functions	Best	Worst	Means	St.Dev	Time(ms)
f_1	0.000387	2.286493	1.213432	0.064132	178
f_2	0.000087	1.200364	0.317627	0.348780	364
f_3	0.000001	0.000841	0.003744	0.002654	337
f_4	0.000008	0.298273	0.094310	0.082812	181
f_5	0.000013	0.361683	0.074512	0.090207	315
f_6	0.000010	0.025467	0.006542	0.006409	128

Table-2. Test suite size for a configuration with 10 paramters V-value, values are varied from 3 to 10.

V	TVG	PICT	CTE_XL	TConfig	IPOG	Jenny	PPSTG	PHSS	BPTS
3	18	18	18	17	20	19	17	17	**16***
4	33	31	33	31	31	30	29	**28***	**28***
5	50	47	50	48	50	45	45	43	**42***
6	72	66	71	64	68	62	62	60	**59***
7	98	88	97	85	90	83	81	**79***	**79***
8	124	112	125	114	117	104	109	105	**102***
9	152	139	161	139	142	129	139	**127***	131
10	189	170	192	170	176	157	170	**155***	162

Table-3. Test suite size for a configuration with P-parameters 2-values, parameters are varied from 3 to 15.

P	TVG	PICT	CTE_XL	TConfig	IPOG	Jenny	PPSTG	PHSS	BPTS
3	**4***	**4***	6	**4***	**4***	5	**4***	**4***	**4***
4	6	**5***	6	6	6	6	6	6	6
5	**6***	7	**6***	**6***	**6***	7	**6***	**6***	**6***
6	**6***	**6***	8	7	8	8	7	7	7
7	8	**7***	8	9	8	8	**7***	**7***	**7***
8	8	**7***	8	9	8	8	8	8	8
9	**8***	9	9	9	**8***	**8***	**8***	**8***	**8***
10	9	9	9	9	10	10	**8***	**8***	**8***
11	9	9	10	9	10	9	9	**8***	**8***
12	10	9	10	9	10	10	9	9	**8***
13	10	9	10	9	10	10	9	9	**8***
14	10	10	10	**9***	10	10	**9***	10	**9***
15	10	10	10	**9***	10	10	10	10	**9***

CONCLUSIONS

Bat algorithm is a new meta-heuristics intelligence optimization algorithm. Despite that, BA has shown lately very good result in various types of research, which is the main reason to use it in our research. In this research work, we have proposed a novel strategy called BPTS, based on the Bat algorithm. Our experimental results are encouraging, especially at the prospect of supporting a high number of parameters. As the scope for future work, we are also working to support the constraints interaction in order to support the testing of software product lines.

ACKNOWLEDGEMENTS

This research is funded by MOSTI eScience fund for the project titled: Constraints T-Way Testing Strategy with Modified Condition/Decision Coverage from the Ministry of Science, Technology, and Innovation, Malaysia. We thank MOSTI for the contribution and support.

REFERENCES

[1] Ahmed B. S. and Zamli K. Z. 2011. The development of a particle swarm based optimization strategy for pairwise testing. Journal of Artificial Intelligence, Vol. 4, No. 2, pp. 156-165.

[2] Ahmed B. S. and Zamli K. Z. 2011. A variable strength interaction test suites generation strategy using Particle Swarm Optimization. Journal of Systems and Software, Vol. 84, No. 12, pp. 2171-2185.

[3] Alsewari A. R. A. and Zamli K. Z. 2012. Design and implementation of a harmony-search-based variable-strength t way testing strategy with constraints support. Information and Software Technology, Vol. 54, No. 6, pp. 553-568.

[4] Alsewari A. R. A. and Zamli K. Z. 2012. A harmony search based pairwise sampling strategy for combinatorial testing. International Journal of the Physical Sciences Vol. 7, No. 7, pp. 1062--1072.

[5] Arshem J. 2009. TVG.

[6] Cohen M. B., Gibbons P. B., Mugridge W. B. and Colbourn C. J. 2003. Constructing test suites for interaction testing. 2003. Proceedings. 25th International Conference on Software Engineering.

[7] Czerwonka J. 2006. Pairwise testing in the real world, pp. Practical extensions to test-case scenarios. Proceedings of 24th Pacific Northwest Software Quality Conference, Citeseer.

[8] Hartman A. and Raskin L. 2004. Problems and Algorithms for Covering Arrays. Discrete Mathematics, Vol. 284, No. 1-3, pp. 149--156.

[9] Hedayat A. S., Sloane N. J. A. and Stufken J. 1999. Orthogonal Arrays: Theory and Applications. New York, Springer Verlag.

[10] Huang S., M. Cohen B. and Memon A. M. 2010. Repairing GUI test suites using a genetic algorithm. 2010 Proceedings of the Third International Conference on Software Testing, Verification and Validation (ICST.

[11] Jamil M. and Yang X. S. 2013. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150--194.

[12] Khan K. and Sahai A. 2012. A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context. International Journal of Intelligent Systems and Applications (IJISA) Vol. 4, No. 7, pp. 23.

[13] Kuhn D. R., Wallace D. R. and Gallo A. M. Jr. 2004. Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering, IEEE Vol. 30, No. 6, pp. 418--421.

[14] Kuhn, R., Lei, Y. and Kacker, R. (2008). Practical combinatorial testing: Beyond pairwise. IT Professional 10(3), pp. 19--23.

[15] Lehmann E. and Wegener J. 2000. Test case design by means of the CTE XL. Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark.

[16] Lei Y., Kacker R., Kuhn D. R., Okun V. and Lawrence J. 2007. IPOG: A general strategy for t-way software testing. 2007 Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based System. (ECBS'07).

[17] Lei Y., Kacker R., Kuhn D. R., Okun V. and Lawrence J. 2007. IPOG: A general strategy for t-way software testing. Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Tucson, AZ U.S.A, IEEE Computer Society.

[18] Lei Y., Kacker R., Kuhn D. R., Okun V. and Lawrence J. 2008. IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing. Software Testing, Verification and Reliability, Vol. 18, No. 3, pp. 125--148.

[19] Lei Y. and Tai K.-C. 1998. In-parameter-order: A test generation strategy for pairwise testing. 1998 Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium, IEEE.

[20] Mandl R. 1985. Orthogonal latin squares: An application of experiment design to compiler testing. Communications of the ACM, Vol. 28, No. 10, pp. 1054--1058.

[21] McCaffrey J. D. 2009. Generation of pairwise test sets using a genetic algorithm. 2009 Proceedings of the 33rd Annual IEEE International of Computer Software and Applications Conference (COMPSAC'09).

[22] McCaffrey J. D. 2010. An empirical study of pairwise test set generation using a genetic algorithm. 2010 the Seventh International Conference on Information Technology New Generations (ITNG),

[23] Pallas. 2003. Jenny.

[24] Shiba T., Tsuchiya T. and Kikuno T. 2004. Using artificial life techniques to generate test cases for combinatorial testing. Proceedings of the 28th Annual International, Computer Software and Applications Conference (COMPSAC 2004).

[25] Soh Z. H. C., Abdullah S. A. C., Younis M. I. and Zamli K. Z. A. parallelization strategy of test suite generation for pairwise testing using tspaces.

[26] Williams A. W. 2000. Determination of test configurations for pair-wise interaction coverage. Testing of Communicating Systems, Springer, pp. 59--74.

[27] Williams A. W. 2002. TConfig. http://www.site.uottawa.ca/~awilliam.

[28] Yan J. and Zhang J. 2006. Backtracking algorithms and search heuristics to generate test suites for combinatorial testing. 2006 Proceeding of the 30th Annual International Computer Software and Applications Conference, IEEE Computer Society.

[29] Yang X. 2010. A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, pp. 65--74.

[30] Younis M. I. and Zamli K. Z. 2010. MC-MIPOG: A parallel t-way test generation strategy for multicore systems. ETRI journal, Vol. 32, No. 1.

[31] Younis M. I. and Zamli K. Z. 2011. MIPOG-An efficient t-way minimization strategy for combinatorial testing. International Journal of Computer Theory and Engineering, Vol. 3, No. 3, pp. 388--397.

[32] Younis M. I., Zamli K. Z. and Isa N. M. 2008. MIPOG-Modification of the ipog strategy for t-way software testing. 2011 Proceeding of The Distributed Frameworks and Applications (DFmA), Penang, Malaysia.