Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A CUCKOO SEARCH BASED PAIRWISE STRATEGY FOR COMBINATORIAL TESTING PROBLEM

[1]ABDULLAH B. NASSER, [2]YAZAN A. SARIERA, [3]ABDUL RAHMAN A. ALSEWARI, AND [4]KAMAL Z. ZAMLI

Faculty of Computer Systems and Software Engineering,

Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia

E-mail: [1]abdullahnasser83@gmail.com, [2]alsarierah@gmail.com,[3]alsewari@gmail.com, [4]k.z.zamli@gmail.com

ABSTRACT

Combinatorial Testing (CT) is a sampling technique to generate test cases with a focus on the behavior of interaction system's components with their collaborators. Given its effectiveness to reveal faults, pairwise testing has often been chosen to perform the required sampling of test cases. The main concern for pairwise testing is to obtain the most optimal test sets (i.e. pairwise dictates that every pair of input values is covered by a test case at least once). This paper discusses the design and implementation a new pairwise strategy based on Cuckoo Search, called Pairwise Cuckoo Search strategy (PairCS). PairCS serves as our vehicle to investigate the usefulness of Cuckoo Search for pairwise testing.

Keywords: *Pairwise testing, Cuckoo Search, Test suite Generator, Software Testing, Combinatorial Testing Problem.*

1. INTRODUCTION

Combinatorial Testing (CT) is a sampling technique to generate test cases with a focus on the behavior of interaction system's components with their collaborators [1]. Given its effectiveness to reveal faults, pairwise testing has often been chosen to perform the required sampling of test cases. In the literature, many studies show that most software failures are often caused by interaction of two parameters [2]. In a study conducted by Kuhn, Wallace et al, it was found that 70% to 90% of bugs can be detected by using pairwise technique [3].

The main issue for pairwise testing is to obtain the most optimal test sets (i.e. pairwise dictates that every pair of input values is covered by a test case at least once [4, 5]). In fact, searching operation for the optimal set of test cases is an NP-hard (Non-deterministic Polynomial-time hard) problem [6-8]. To address this issue, many pairwise strategies have been designed and implemented such as strategies based on Simulated Annealing [9], Genetic algorithm[9], Ant Colony Algorithm (ACA) , Particle Swarm Optimization [4], and Harmony Search [5], to name a few.

In this paper, we introduce a new pairwise strategy, called PairCS, based on Cuckoo search algorithm. The adoption of Cuckoo Search Algorithm (CS) appears to be an attractive option as it appears more efficient than that of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [10-13]. In [13], CS was adopted to solve a milling optimization problem. A comparison between the CS and other techniques including GA, Ant Colony Optimization (ACO), hybrid PSO and Immune Algorithm (AIA) showed that CS performs better than other techniques. In a study on scheduling optimization [14], CS performs better than GA and PSO. Specifically, CS offers the following advantages [15, 16]:

- Unlike GA, and PSO, CS offers lightweight computation relying only on three parameters; max generation, nest size and probability pa.
- CS embeds elitism mechanism (via probability pa) to ensure that the best solutions are carried over the next iteration.
- CS offers balance intensification and diversification of solutions through the adoption of Lévy Flight. Essentially, Lévy Flight consists of random walks that are interspersed by long jumps which are heavy tailed according to a power law distribution. In this manner, CS often can sufficiently explore regions of interests.

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 1: Design-your-burger Example

Currently, researches on cuckoo search is very active and its applications have been proven successes in many areas such as machine learning [17], the field of truss optimization problems [18], clustering of web results [19], nurse scheduling problems [20], generating test data generation [21], generating independent paths for software testing [22].

The rest of this paper is organized as follows. Section 2 gives an overview of pairwise testing. Related works are stated in Section 3. Detailed reviews of Cuckoo Search algorithm is provided in Section 4. Section 5 presents the proposed strategy. Section 6 highlights the experimental results and discussion. Lastly, Section 7 gives the conclusion and future work.

2. BACKGROUND

In general, any system under test consists of a number of components, which interact with each other through a set of parameters with some defined values.

Definition 1 (Pairwise testing): Given a set of N parameters P_1, $P2$, $P3$,.., P_N, having v_i possible values $\{v_1,v_2,...v_m\}$, a set of test data values Tc, contains N test values, which is selected for each of the parameter values such that all test cases in Tc cover all 2-way pairs of input parameter values.

To illustrate the concept of combinatorial and pairwise testing for test suite reduction, consider the following form of design-your-burger application as given in Figure 1. In this form, the user can order a burger by selecting his favorite ingredients. Here, there are 10 inputs; each input with associated values as shown in Table 1

In order to test all the factors exhaustively, there are 3072 test cases. By using pairwise testing, each pair of input parameter values can be covered at least one time on the test case. To generate test suite for Design-your-burger example, there are 30 pairs need to be covered. By using the proposed PairCS algorithm, all the 3072 test cases can be minimized to merely 12 test cases.

Table 1 : Design-your-Burger Input Values

Test Factor	Values
Burger	Beef, Turkey, Veggie
Cooked	None, Rare, Medium , well
Cheese	No , Yes
Lettuce	No , Yes
Tomato	No , Yes
Onion	No , Yes
Ketchup	No , Yes
Mustard	No , Yes
Mayo	No , Yes
Secret sauce	No , Yes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. RELATED WORKS

This section is intended to provide an overview of the existing works for constructing a pairwise test suite. Based on [2], the existing approaches can be classified into two main categories: algebraic construction, computational construction.

3.1 Algebraic Approach:

In this approach, test data sets are constructed without enumerating any combinations. Hence, the generation process adopts lightweight computations. There are two types of algebraic approach. The first one is based on mathematical functions [23-25]. The second one employs a recursive process to construct test sets by constructing a large test sets from small test sets [26]. Strategies adopting this approach CA, MCA and TConfig, are often restricted to small configurations.

3.2 Computational construction:

This approaches use a greedy algorithm to construct the test cases. Each step tries to cover as many combinations as possible uncovered combinations. Generating test set is accomplished by either using one-test-at-a-time strategy (OTAT) or one-parameter-at-a-time strategy (OPAT). OTAT strategies start to build one complete test case per iteration and checks if this test case is the best test case to cover the most uncovered interaction. The iteration continues until all the combinations are covered. In the literature, there are many strategies that adopts OTAT techniques such as AETG [27], TConfig [28], Jenny [29], and WHITCH [30]. One-parameter-at-a-time (OPAT) strategy starts by building a completed test suite for the first two parameters, or the smallest number of components, then extends horizontal by adding one parameter per iteration, and sometimes, extends vertically until all the parameters is covered. Examples of such approach are IPO [7] and its improvement (i.e. IPOG [31], IPOG-D [6], IPOF and IPAD2 [32]).

Recently, many existing works are focusing on nature-inspired based strategies. Nature-inspired based algorithms (e.g. Simulated Annealing, Genetic Algorithm, Ant Colony Algorithm, Particle Swarm Optimization and Harmony Search) have been used successfully for pairwise testing. Simulated Annealing (SA) algorithm has been implemented to generate pairwise test cases by Cohen, 2004, and Patil and Nikumbh, 2012 [8, 33]. In order to avoid getting stuck in a local minimum solution, SA allows a poor move based on some probability. Another nature-inspired algorithm that has been used to generate pairwise test data is Genetic algorithm (GA) by Shiba, Tsuchiya et al (2004). GA is based on AETG strategy[27]. For constructing pairwise test cases, GA generates a number of objects, called chromosomes. Each chromosome is subjected to series of operation of Mutation, Crossover, Selection processes until certain stopping criteria are met. Ant Colony Algorithm (ACA) has also been used for pairwise test cases generation. Simulating the behavior of ant colony for finding food paths, the places of food represent the parameter and the food represents the value of the parameter, and each test case represents the quality of the paths to the food. The paths to the food are evaluated based on the quantity of pheromones which is reinforced by the ants. By comparison, the best path is selected to be added to final test cases [34, 35].

Particle Swarm Optimization (PSO) algorithm has been implemented for pairwise test suite generation using two different based approaches OTAT and OPAT [2, 4]. The discrete version of PSO is adopted in Particle Swarm-based Test Generator (PSTG) strategy [4]. Much recent work undertaken in this field, Harmony Search has been adopted in Harmony Search algorithm-based strategy (PHSS) to implement and generate pairwise tests suite. PHSS is pairwise test data generation. Using PHSS, the test data generation process is mimicking the improvisation process by a skilled musician [5].

4. CUCKOO SEARCH:

Cuckoo search (CS) is a nature-inspired algorithm for solving global optimization problems developed by Xin-She Yang. CS mimics the behavior of brood parasitic for some birds such as the Ani and Guira cuckoos [36].

4.1 Cuckoo Breeding Behavior:

The behavior of cuckoo bird is represented in the obligate brood parasitism of some cuckoo. Parasitic cuckoos lay eggs their eggs in the nests of other host birds. If the properties of cuckoo eggs have developed well enough, then the eggs will take a great opportunity to survive. To this end, cuckoo increase phenotypic matching between cuckoo and host eggs by mimic the external color and pattern of host eggs. Furthermore, the cuckoos often choose a nest where the host eggs recently is laid to lay their eggs. Thus, cuckoo eggs hatch early than the host

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

eggs, and then first cuckoo chicks instinctively will evict the host eggs out of the nest [37].

Figure 1 : Cuckoo Breeding Behaviour

One issue of importance, Cuckoo search has two search capabilities: global search, which allows the algorithm to jump out of local optimum, and local search by intensify search around the current best, are controlled by pa probability. If pa=0.25, the local search takes about %25 and global search takes about %75 of the total search time[16]. Local search and global search capabilities combined with search using Levy Flight makes CS exploration of the search space efficiently. In this paper, we are investigating the use of Cuckoo search (CS) algorithm for pairwise test suite generation.

4.2 Cuckoo Search algorithm:

Cuckoo search algorithm is essentially a population based algorithm for solving global optimization problems. For simplification purpose, Cuckoo Search relies upon three idealized rules [36]:

1. Each cuckoo chooses a nest randomly to lays eggs.
2. The number of available host nests is fixed, and nests with high quality of eggs will carry over to the next generations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $pa \in [0, 1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

Based on those three rules, Cuckoo Search algorithm can be summarized as shown in Figure 3.

Cuckoo Search Algorithm

Objective function f(x), x = (x1, ..., xd) ;
Initial a population of n host nests x_i *(i = 1, 2, ..., n);*
while (*t <MaxGeneration*) or (stop criterion)
  Get a cuckoo (say *i*) randomly by Lévyflights;
  Evaluate its quality/fitness *Fi;*
  Choose a nest among n (say *j*) randomly;
  IF (Fi > Fj)
    Replace j by the new solution;
  End if
  Abandon a fraction (*pa*) of worse nests and build new ones at new locations
  Keep the best solutions
  Rank the solutions and find the current best;
end while
Postprocess results and visualization;
End-Procedure

Figure 2: Cuckoo Search Algorithm

The CS algorithm is straightforward to use and implement owing to small number of parameters and needs a small population to achieve a good results. The core part of the CS algorithm is generating new solution using of Lévy Flight Equation 1, where each position of cuckoo is updated.

$$x_i^{(t+1)} = x_i(t) + \alpha \oplus \text{Lévy}(\lambda) \qquad (1)$$

where $\alpha > 0$ is the step size which should be related with problem and $Lévy \sim u = t^{-\lambda}$. Equation 1 is considered as a generic equation to update cuckoo's position either using Lévy flights or random walk. The Lévy flight essentially is a random walk interspersed by long jumps where the next step is based on the current location, and step lengths have a certain probability distribution that is heavy-tailed.

5. THE PROPOSED STRATEGY

In the following section, the application of CS outlines in PairCS. PairCS is a composition of three main steps: Generating Binary Combinations, Generating Interaction Elements and Finding the optimal set of test cases using CS as Figure 3 show. In the following, these three steps are explained.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

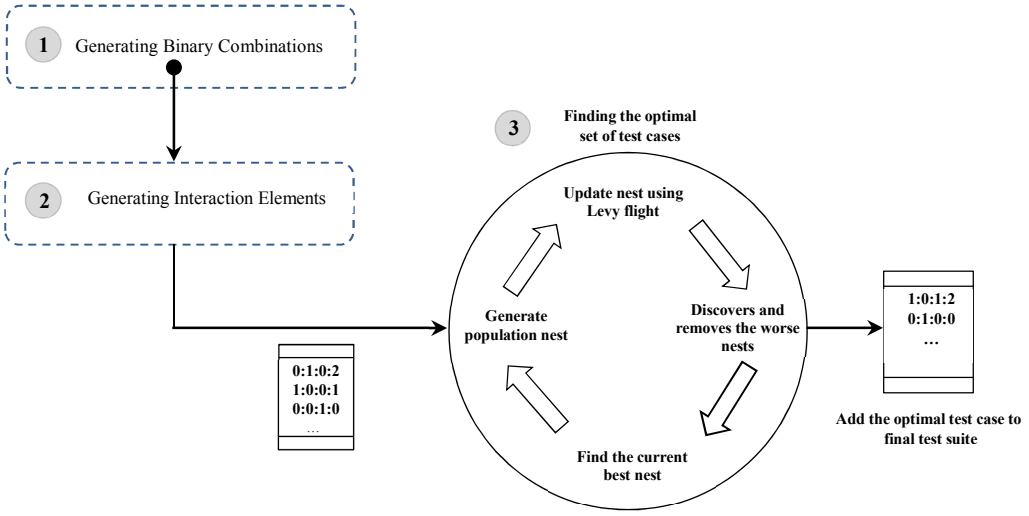

Figure 3: Graphical Representation of PairCS Strategy Steps

Based on the receiving inputs (i.e. a set of parameters P = {*p1, p2, p3, p4*} and parameter values v_i), the proposed strategy, PairCS, begins generating all possible binary combinations of *P*-digit that only contain two 1s (i.e 0011, 0101, 0110, 1001…). Based on the generated binary combinations, interaction elements list is generated accordingly. For example, if *p1, p2*, and *p3* are having two values (i.e., 0 and 1), and *p4* is having three values (0, 1, and 2), interaction elements for the first binary combination 1100 are 2×2 possible interaction elements (i.e., 0:0:0:0, 0:0:0:1, 0:0:1:0, and 0:0:1:1), while interaction elements for 1001 are 0:0:0:0, 0:0:0:1,1:0, 0:0:1:1, 0:0:2:0, and 0:0:2:1).

The complete step for the proposed strategy includes finding optimal test cases phase using CS. In PairCS, each nest or solution represents one test case. PairCS starts to generate initial nests or test cases randomly, and finds the best nest of those nests. In order to improve current nest, a new nest, $x^{new} = (x_1{}^{new}, x_2{}^{new}, \dots, x_{n-1}{}^{new}, x_n{}^{new})$, is generated by performing a Levy flight, and evaluated.

Nest weight or fitness is number of interaction elements x_i that can be cover by candidate nest, which can be expressed mathematically as follows:

$$\text{Maximize } f(x) = \sum_1^N x_i \qquad (2)$$

where *N* is covered interaction elements by candidate nest.

Based on nest weight, the new nest will be chosen as a current nest. If the new nest weight is better than current nest weight, the new nest is taken as current nest. As part of elitism process, the algorithm iterates all population and removes the worse nests based on the value of *pa* probability. The best nest will be selected and added to final test cases and the covered interaction elements are removed from the interaction list. Finding optimal test cases phase is repeated till all interaction elements are covered (i.e., the interaction elements list is empty). The proposed strategy is summarized in Figure 4.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PairCS Strategy Algorithm

Input N: Parameters number *n*, and
V: set of values for each parameter V = [v_0 ..v_j];
Output: test suite List *TS*;
Let IPairs all Interaction Pairs.
Let TS be a set of candidate tests;
Generate initial population of host nest randomly
while IPairs is not empty do
while t <MaxGeneration or stop criterion do
Get a cuckoo (say i) randomly by Lévyflights;
Evaluate its quality/fitness Fi;
Choose a nest among n (say j) randomly;
IF (Fi > Fj)
Replace j by the new solution;
End if
Abandon a fraction (pa) of worse nests and build new ones at new locations
Keep the best solutions
Rank the solutions and find the current best;
End while
Add the best test case into TS.
Remove covered interactions elements from IPairs.
End while
End-Procedure

Figure 4: PairCS Strategy

6. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed strategy (PairCS), PairCS is implemented and executed. Results of performance analysis are displayed with the use of tables. Several existing comparative experiments [2, 5] are adopted in our experiments. For our experiment, we have adopted *Pa* = 0.25, iteration = 500, and nest size 30 based on previous work [38]. To measure the performance of the proposed strategy, NetBeans 8.0.1 was used to execute the algorithms. The specification of the machine used is: Intel (R) core™ i7-3770 CPU @ 3.40 GHz, 4GB of RAM, Windows 7 professional and 32-bit Operating System. Here, our experiment alienated into two groups as following:

1. Comparison with existing pairwise strategies with 2-valued parameters and P are varied from 3 to 12 and (50, 100, and 150) to show the ability of PairCS to address high configuration system.
2. Comparison PairCS with published results of existing strategies using different systems configuration.

For a fair comparison due to PairCS is non-deterministic strategy, we run PairCS 20 times for every configuration, and the best test suite size is reported.

Table 2 and 3 show the comparisons between PairCS and existing strategies. From Table 1, our PairCS strategy produces the most optimum results in most of the configurations (as marked with *). Table 2, also, shows the ability of proposed strategy to generate test data for systems with high configurations where then number of parameter *P* can go up to 150 parameters.

In Table 3, PairCS strategy produces the most minimum test size for T1, T2 and T7. In general, table 3 shows that conventional strategies (i.e. TVG, PICT, AETG, mAETG, CTEXL and so on), generate slightly better size than Nature-Inspired Strategies. However, when we take a closer look at mAETG, AETG, ACA, SA and GA perform better that other strategies due to their randomization. By comparing only nature-inspired strategies in Table 3, we found that our PairCS outperforms most of existing strategy in some cases such as T4, T5 and T7 (as marked by *). The good obtained results is supported the fact that CS offers a good balance between global search and local search through the adoption of Lévy Flight in its core implementation and achieves good results when the systems consisting of big values due to long jumps of Lévy Flight.

7. CONCLUSION AND FURTHER WORK

In this paper, we have proposed and evaluated a new pairwise strategy based on Cuckoo search algorithm, called PairCS. Our experience with PairCS has been promising, as we have managed to obtain good test sizes for most of the considered configurations. Our results, in most cases outperform the existing nature-inspired-based as well as other computational-based strategies.

Furthermore, our case study evaluation also demonstrates the capability of proposed strategy in generating efficient test suites. As part of the future work, we plan to introduce seeding and constraints into the current implementation. We are also currently improving PairCS to support both sequence and sequence-less t-way testing.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2 : Comparison with Existing Strategies using V = 2 and P varied From 3 to 150

P	TVG	PICT	CTE_XL	TConfig	IPOG	Jenny	PPSTG	PHSS	PairCS
3	4*	4*	6	4*	4*	5	4*	4*	4*
4	6	5*	6	6	6	6	6	6	5*
5	6*	7	6*	6*	6*	7	6*	6*	6*
6	6*	6*	8	7	8	8	7	7	6*
7	8	7	8	9	8	8	7	7	7*
8	8	8	7	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		10	9	9	8	8*
12	10	9	10	9	10	10	9	9	9*
50	NA	NA	NA	NA	NA	NA	NA	NA	12
100	NA	NA	NA	NA	NA	NA	NA	NA	15
150	NA	NA	NA	NA	NA	NA	NA	NA	17

Table 3 : Comparison with Existing Strategies using Different System Configuration with Mixed Parameter Values

Config-rations	TVG	PICT	AETG	mAETG	CTEXL	TConfig	AllPairs	Jenny	IPO	IPOG	IRPS	G2Way	SA	GA	ACA	PPSTG	PHSS	PairCS
T1	11	10	NA	NA	10	10	10	9*	NA	11	9*	10	NA	NA	NA	9*	9*	9*
T2	12	13	9*	11	10	10	10	13	9	12	9*	10	9*	9*	9*	9*	9*	9*
T3	20	20	15*	17	21	20	22	20	17	20	17	19	16	17	17	17	18	18
T4	189	170	NA	NA	192	170	177	157	169	176	149*	160	NA	157	159	170	155	151
T5	473	NA	NA	NA	NA	NA	390	336	361	373	321*	343	NA	NA	NA	NA	341	333
T6	NA	NA	180	198	NA	NA	230	NA	212	NA	210	200	183*	227	225	NA	224	209
T7	50	47	NA	NA	50	48	49	45	47	50	45	46	NA	NA	NA	45	43	42*
T8	23	21	19	20	21	22	21	41	NA	19	17	23	15*	15*	16	21	20	20
T9	41	38	34	35	39	33	NA	31*	NA	36	NA	NA	NA	33	32	39	39	38
T10	52	46	45	44	53	49	NA	51	NA	44	NA	NA	NA	42*	42*	49	48	47
T11	100	101	NA	NA	102	92	NA	98	NA	91*	NA	NA	NA	NA	NA	97	95	96

The configurations are shown as follows:

T1: 3^3, T2: 3^4, T3: 3^{13}, T4: 10^{10}, T5: 15^{10}, T6: 10^{20}, T7: 5^{10}, T8: $5^1 3^8 2^1$, T9: $6^1 5^1 4^6 3^8 2^3$, T10: $7^1 6^1 5^1 4^6 3^8 2^3$, T11: $10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1$

y^x means : means that task take x parameters, each parameter with y values.

ACKNOWLEDGMENT

The work reported in this paper 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. Mr. Abdullah B. Nasser is the recipient of the Graduate Research Scheme (GRS) from Universiti Malaysia Pahang.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton, "The combinatorial design approach to automatic test generation," *IEEE software,* pp. 83-88, 1996.

[2] X. Chen, Q. Gu, J. Qi, and D. Chen, "Applying particle swarm optimization to pairwise testing," in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, 2010, pp. 107-116.

[3] D. R. Kuhn, D. R. Wallace, and J. AM Gallo, "Software fault interactions and implications for software testing," *Software Engineering, IEEE Transactions on,* vol. 30, pp. 418-421, 2004.

[4] B. S. Ahmed, K. Z. Zamli, and C. Lim, "The development of a particle swarm based optimization strategy for pairwise testing," *Journal of Artificial Intelligence,* vol. 4, pp. 156-165, 2011.

[5] A. R. A. Alsewari and K. Z. Zamli, "A harmony search based pairwise sampling strategy for combinatorial testing," *International Journal of the Physical Sciences,* vol. 7, pp. 1062-1072, 2012.

[6] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing," *Software Testing, Verification and Reliability,* vol. 18, pp. 125-148, 2008.

[7] Y. Lei and K.-C. Tai, "In-parameter-order: A test generation strategy for pairwise testing," in *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, 1998, pp. 254-261.

[8] M. B. Cohen, "Designing test suites for software interaction testing," Citeseer, 2004.

[9] J. Stardom, "Metaheuristics and the search for covering and packing arrays," Trent University, 2001.

[10] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation,* vol. 1, pp. 330-343, 2010.

[11] X.-S. Yang, *Nature-inspired metaheuristic algorithms*: Luniver press, 2010.

[12] X.-S. Yang, S. Deb, M. Karamanoglu, and X. He, "Cuckoo search for business optimization applications," 2012.

[13] A. R. Yildiz, "Cuckoo search algorithm for the selection of optimal machining parameters in milling operations," *The International Journal of Advanced Manufacturing Technology,* vol. 64, pp. 55-61, 2013.

[14] S. Burnwal and S. Deb, "Scheduling optimization of flexible manufacturing system using cuckoo search-based approach," *The International Journal of Advanced Manufacturing Technology,* vol. 64, pp. 951-959, 2013.

[15] X. S. Yang, S. Deb, and S. Fong, "Metaheuristic algorithms: optimal balance of intensification and diversification," 2013.

[16] X.-S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications,* vol. 24, pp. 169-174, 2014.

[17] R. A. Vázquez, "Training spiking neural models using cuckoo search algorithm," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 2011, pp. 679-686.

[18] A. H. Gandomi, S. Talatahari, X. S. Yang, and S. Deb, "Design optimization of truss structures using cuckoo search algorithm," *The Structural Design of Tall and Special Buildings,* vol. 22, pp. 1330-1349, 2013.

[19] C. Cobos, H. Muñoz-Collazos, R. Urbano-Muñoz, M. Mendoza, E. León, and E. Herrera-Viedma, "Clustering of Web Search Results based on the Cuckoo Search Algorithm and Balanced Bayesian Information Criterion," *Information Sciences,* 2014.

[20] L. H. Tein and R. Ramli, "Recent advancements of nurse scheduling models and a potential path," in *Proceedings of 6th IMT-GT Conference on Mathematics, Statistics and its Applications*, 2010, pp. 395-409.

[21] K. Perumal, J. M. Ungati, G. Kumar, N. Jain, R. Gaurav, and P. R. Srivastava, "Test data generation: a hybrid approach using cuckoo and tabu Search," in *Swarm, Evolutionary, and Memetic Computing*, ed: Springer, 2011, pp. 46-54.

[22] P. R. Srivastava, R. Khandelwal, S. Khandelwal, S. Kumar, and S. Santebennur Ranganatha, "Automated test data generation using cuckoo search and tabu search (csts) algorithm," 2012.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[23] A. Hartman and L. Raskin, "Problems and algorithms for covering arrays," *Discrete Mathematics,* vol. 284, pp. 149-156, 2004.

[24] R. Mandl, "Orthogonal Latin squares: an application of experiment design to compiler testing," *Communications of the ACM,* vol. 28, pp. 1054-1058, 1985.

[25] K. A. Bush, "Orthogonal arrays of index unity," *The Annals of Mathematical Statistics,* vol. 23, pp. 426-434, 1952.

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

[27] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: An approach to testing based on combinatorial design," *Software Engineering, IEEE Transactions on,* vol. 23, pp. 437-444, 1997.

[28] A. Williams, "TConfig download page [Online]," p. University of Ottawa. Available: http://www.site.uottawa.ca/~awilliam/[Accessed 23 Dec 2014]. 2008.

[29] B. Jenkins, "Jenny download page [Online]," p. Available : http://www.burtleburtle.net/bob/math. [Accessed 16 Dec 2014]. 2003.

[30] A. Hartman, T. Klinger, and L. Raskin, "IBM intelligent test case handler," *Discrete Mathematics,* vol. 284, pp. 149-156, 2010.

[31] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," in *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, 2007, pp. 549-556.

[32] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology,* vol. 113, pp. 287-297, 2008.

[33] M. Patil and P. Nikumbh, "Pair-wise testing using simulated annealing," *Procedia Technology,* vol. 4, pp. 778-782, 2012.

[34] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, 2004, pp. 72-77.

[35] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR),* vol. 43, p. 11, 2011.

[36] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210-214.

[37] J. Avilés, B. Stokke, A. Moksnes, E. Røskaft, M. Åsmul, and A. Møller, "Rapid increase in cuckoo egg matching in a recently parasitized reed warbler population," *Journal of evolutionary biology,* vol. 19, pp. 1901-1910, 2006.

[38] A. B. Nasser, A. R. A. Alsewari, and K. Z. Zamli, "Tuning of Cuckoo Search Based Strategy for T-way Testing," in *International Conference on Electrical and Electronic Engineering*, 2015.