IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

Comparative Analysis on Parameter Optimization for ARPT

K. Devika Rani Dhivya[1*], V.S. Meenakshi[2]

[1]Research Scholar, Bharathiar University, Coimbatore-8, Tamilnadu, India.
[2] Department of Computer Science, Chikkanna Government Arts College, Tirupur-2, Tamilnadu, India.

[*]*Corresponding Author: devika58@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 18/Dec/2018, Published: 31/Dec/2018

***Abstract*—** Software testing is a principal however complex piece of software development life cycle. It points towards recognizing the bugs and faults in the program of functional behavior. It always needs generation of test cases and suites for confirming their input ranges. The Optimization of Software testing is the foremost challenge. Subsequently to achieve greatest coverage the test must be created from overall dispersed regions of input areas, known Partition testing. Random testing serve better than partition testing nevertheless it also generating high computational overheads. Another technique is Adaptive Random technique having adaptive nature of recovering and finishing a portion of the test cases back to the input for correcting the next test cases lined to be passed. Adaptive Random Partition Testing (ARPT) was used to test software which utilized AT and RT in an alternative manner. The computational intricacy issue of random partitioning in ARPT strategies was resolved by utilizing clustering algorithms. It expends additional time and it prompts overhead procedure to estimate parameters of ARPT. In this paper, the parameters of ARPT 1 and ARPT 2 are optimized using Bacterial Foraging Algorithm (BFA) and Improved BAT algorithm which improves the accuracy of ARPT software testing strategies. However, the BFA has the most critical parameter step size that has strong influence in the convergence and stability of algorithm. In order to solve these problems, the improvised BAT optimization algorithm is proposed in this paper. It improves the accuracy and reduces time consumption of parameter setting of ARPT testing strategies.

***Keywords*—** Software testing, Adaptive Random Partition Testing, BFA, Improvised BA

I. INTRODUCTION

Software testing is a technique aimed at evaluating an attribute or usability or capability of a product or program and determining that it meets its quality [4]. Software testing is the key to ensuring a successful and reliable software product or service, yet testing is often considered uninteresting work compared to design or coding. [3] The process of software testing continues to challenge the software community. This is crucial to the success of a software project. Despite decades of effort to develop an alternative technology to verify the quality of software, system software testing remains the primary way. However, software testing remains a labor-intensive, imperfect and slower process. Hence it is more important to consider testing can be performed more effectively at a lower cost through the use of systematic and automated methods.

There are different systematic and automated methods are developed for software testing. One of the basic approaches to test software is randomly generating test cases from the set of all input domain which is called Random Testing (RT). Adaptive Random Testing (ART) [5] had been developed to enhance the RT in which the adjacent program inputs show a certain degree of similarity in failure revealing behaviors. Partition Testing (PT) is one of the traditional software testing strategies where the input domains are divided into subdomains categorized according to some separation conditions of test cases.

Adaptive and Random Partition Testing (ARPT) [7] testing strategy is a combination of AT and Random Partition Testing (RPT) those are used in an alternative manner to test software. By introducing RPT testing process the time complexity of AT was reduced. [7] ARPT consists of two options namely ARPT 1 and ARPT 2.

APRT 1 and ARPT 2 testing strategies consist of different parameters. Those parameters values are varied based on the software. It needs to estimate over and over for different software. Thus the parameter estimation for different software leads to computation overhead and time consumption.

The computational complexity of random partitioning in ARPT strategies was resolved by using clustering algorithms in random partitioning of ARPT strategies. The time consumption and overhead process of parameter estimation were reduced by Bacterial Foraging Algorithm and Improvised BAT (IBAT) algorithm. However, the BFA

© 2018, IJCSE All Rights Reserved

required has the most critical parameter step size C which has a strong influence in the algorithm stability and convergence. It consumes more time to optimize the parameters of ARPT. Therefore in the proposed improvised BAT algorithm with ARPT, the weighted average of solutions is calculated and then selects the best solution in between weighted average solution and best solution which reduces the time and improves the accuracy of parameter setting of ARPT strategies.

This paragraph provides organization of this paper, Section I contains the introduction of software testing strategy ARPT, Section II contains the related work of adaptive testing, random testing, and partition testing, Section III contains the methodology used for optimizing ARPT, Section IV describes results and discussion, Section V concludes research work.

II. RELATED WORK

Arcuri, A., & Briand, L.[1] presented theorems to define the probability of random testing that detects the interaction faults in software. Some faults of software are revealed only if particular features are chosen from the delivered products. Testing software with all combination of features is not feasible and it consumes more time. Combinational testing performs testing using t features in the delivered products. The presented theorems describe the probability of random testing which choosing the features for testing. However still this random testing and combinatorial testing provides minimum guarantees on the probability of fault detection at any interaction level.

Lv, J., et al., [7] proposed Adaptive Testing with Gradient Descent (AT-GD) which investigates the asymptotic behavior of adaptive testing and improves the global performance without losing local optimality of adaptive testing. Here AT-GD is a local optimal testing strategy converges to the globally optimal solution as the assessment process proceeds. Gradient is extensively utilized in deciding a search direction when a step size choice is create to solve an optimization problem. This is introduced in original Adaptive Testing framework which investigates the asymptotic behavior of AT-GD and the upper bound of AT strategies is explored. However this method still has computational overhead problem.

Shahbazi, A., et al., [11] projected the use of Centroidal Voronoi Tessellations (CVT) to address the problem in Adaptive Random Testing (ART) and Quasi Random Testing (QRT). In addition to that, a test case generation method called as Random Border CVT (RBCVT) was proposed to enhance the code coverage of the input space of Random testing strategy. The test cases generated by other testing strategies were given as input to the proposed RBCVT and it returns an improved set of test cases. Furthermore, a novel search algorithm was proposed to reduce the time complexity of RBCVT. The RBCVT outperforms than the ART and CVT methods. However the RBCVT approach is not cost effective due to their relatively high runtime.

Schwartz, A., & Do, H., [12] proposed two additional Adaptive Test Prioritization (ATP) strategies utilizing Weighted Sum Model (WSM) and fuzzy Analytical Hierarchy Process (AHP) to test software. Three strategies were developed in this paper. First strategy utilized the fuzzy AHP which address the issue of the results from AHP and the second strategy utilized a fuzzy expert system to obtain the benefits of a strategy which does not require a pair wise comparisons. The final strategy used WSM which investigate the effectiveness of strategy for ATP.

Singh, K., et al., [13] described the Anti- Random testing to found an error in software and demonstrated that this method achieve high fault coverage. Random testing, test an application based on the randomly selected test cases and it does not consider the previous information. While in the Anti Random testing, each test is applied its total distance from all tests is maximum. It is a variation of Random Testing that generates random input and send that input to a system for test. In order to measure the difference hamming distance and Cartesian distance is used.

Huang, R., et al., [5] proposed an enhanced Mirror Adaptive Random Testing called as Dynamic Mirror Adaptive Random Testing (DMART) to reduce the computation overhead of Adaptive Random Testing (ART) strategy. Mirror Adaptive Random Testing cannot decrease the order of magnitude for computational overhead of ART strategy while maintaining similar failure detection effectiveness. The proposed strategy splits the input domain incrementally along with the testing process by using new mirroring scheme. The mirroring scheme of DMART is independent of concrete ART strategy. This approach still takes more time for generating the test cases.

Machado, B. N., et al., [8] presented a framework for search based software testing (SBST) named as search based software testing framework (SBSTFrame). This framework worked as a top level layer of genetic optimization frameworks and testing software tools. The main intend of this framework is to support the software testers which are not able to utilize optimization frameworks during a testing activity. This framework supports the optimization of software testing activities through supporting the application of search techniques.

Mao, C. [9] presented a new Adaptive Random Testing algorithm based on two-point partitioning to improve the fault revealing ability of random testing. By using the new algorithm, the midpoint of two points is determined and

© 2018, IJCSE All Rights Reserved

current max region is partitioned by using the midpoints of two points instead of using single point. The first point of two points is generated randomly and second point is chosen from the candidate set based on the farthest distance criterion.

III. METHODOLOGY

In this section, the optimal parameter setting for different software using BFA and IBAT optimization algorithm is described in detail. Initially, the time consumption and complexity of ARPT testing strategies were reduced by using clustering algorithms. Then the parameters of both ARPT 1 and ARPT 2 strategy are optimized by using BFA and IBAT optimization algorithm. The new software is tested using ARPT testing strategy the BFA and IBAT algorithm optimize the parameters of ARPT 1 and ARPT 2 testing strategy. Then the optimized parameters are initialized to ARPT 1 and ARPT 2 to detect the faults in software. Thus the more effective parameters are obtained by using the IBAT algorithm.

A. Parameters of ARPT

The ARPT consists of two options such as ARPT 1 and ARPT 2. In ARPT 1 the stage of AT and RPT are initiated through parameters. The test cases are partitioned and then each partition is switch between AT and RPT testing strategies which reduced the computational overhead of test case selection. The other option of ARPT is ARPT 2 where the test case is divided into two lengths. For the first length of test cases AT testing strategy is applied and for the second half of the test case, the RPT testing strategy is applied. The parameter is set only to alter the length of the test case as there is only one switching between the two AT and RPT strategy. The parameters of ARPT 1 and ARPT 2 are optimized by using BFA and IBAT algorithm. The optimal parameters are used for different subject programs that improve fault detection efficiency and reduce time consumption. [4]

ARPT 1 consists of 5 different parameters are S, x, $Sig_R(S)$, k(0) and l(0). S represents the state of the current testing process, the state denotes an AT segment and RPT segment, $Sig_R(S)$ denotes signal retained to record whether any defect is detected. k(0) represents the test case of AT testing strategy and l(0) represents the test case of RPT testing strategy[7]. ARPT 2 consists of two parameters are x and y. x is a parameter utilized to determine when to change the testing strategy and y is the testing length for the AT process. These parameters are optimized using BFA and IBAT algorithm. The fitness function of these optimization algorithms are calculated based on the weighted sum of the time consumption t, defect detection efficiency E_d, memory consumption M, number of test cases n and code coverage C which is represented as follows:[2][4]

$$F = \sum w_1 \frac{1}{t} + w_2 E_d + w_3 M + w_4 \frac{1}{n} + w_5 C \qquad (1)$$

where $w_1 + w_2 + w_3 + w_4 + w_5 = 1$

B. Optimal parameter setting using Bacterial Foraging Algorithm [4]

Bacterial Foraging algorithm is an optimization algorithm where a Bacteria search for nutrients is a manner to maximize energy obtained per unit time. The individual bacterium also communicates with others by sending signals. A bacterium takes foraging decisions after considering two previous factors. The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemo taxis. The key idea of BFA is mimicking the chemotactic movement of virtual bacteria in the problem search space. This algorithm is processed based on the Chemotaxis, swarming, reproduction and elimination dispersal.

In the Chemotaxis process simulates the movement of an E.coli cell through swimming and tumbling via flagella. Biologically an E.coli bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$ represents the i-th bacterium at the j-the chemotactic, k-th reproductive and l-th elimination-dispersal step. *C*(*i*) is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (2)$$

In the above equation 2, Δ represents a vector in the random direction whose elements lie in [-1,1].

In swarming the cell to cell signaling in E. coli is defined and it is represented as follows:

$$Fitness_{cc}\big(\theta, P(j,k,l)\big) = \sum_{i=1}^{S} Fitness_{cc}(\theta, \theta^i(j,k,l)) = \sum_{=1}^{S}[-d_{attractant}\exp(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2 + \sum_{i=1}^{s} h_{repellant}\exp\big(-w_{repellant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\big] \qquad (3)$$

In equation 3, $Fitness_{cc}\big(\theta, P(j,k,l)\big)$ is the objective function value to be added to the actual objective function to present a time varying objective function, *S* is the total number of bacteria, p is the number of variables to be optimized, which are present in each bacterium and $\theta = [\theta_1, \theta_2, \dots, \theta_p]^T$ is a point in the *p*-dimensional search domain, $d_{attractant}$, $w_{attractant}$, $w_{repellant}$ are different coefficients.

Then the least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the

© 2018, IJCSE All Rights Reserved

objective function) asexually split into two bacteria, which are then placed in the same location. This keeps the swarm size constant. Finally in the elimination and dispersal process, Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons e.g. a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the bacteria in a region are killed or a group is dispersed into a new location. To simulate this phenomenon in BFA some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

BFA based Optimal Parameter setting algorithm [4]

Step 1: if (ARPT_strategy==ARPT 1)
Step 2: Initialize S bacteria with $S_1, \dots S_h$, $x_1, \dots x_h$, $Sig_R(S)_1, \dots Sig_R(S)_h$, $\mathrm{k}(0)_1, \dots \mathrm{k}(0)_h$ and $\mathrm{l}(0)_1, \dots \mathrm{l}(0)_h$.
Step 3: Call BFA algorithm
Step 4: else
Step 5: Initialize S bacteria with $m_1, \dots m_h$ and $n_1, \dots n_h$
Step 6: Call BFA algorithm

C. Optimal parameter setting using Improvised BAT algorithm

The BAT algorithm is used to optimize the parameters of ARPT1 and ARPT2. This algorithm is performed based on the echolocation characteristics of microbats. Initially the number of bats in the population is initialized. The Bats are flying randomly along with the velocity $vel_x(t)$, position $pos_x(t)$, fixed frequency $freq_{min}(t)$, wavelength λ and loudness A_0 . The wavelength of each bat is automatically adjusted by the emitted pulse and pulse rate between 0 and 1 that is depending upon the proximity effect. The loudness is varied in positive values A_0 to minimum constant value A_{min}. Mostly, the solution of BAT algorithm is depending on the virtual bat moment, loudness and pulse emission.[10] [14]

Consider the searching dimension n and the fixed frequency $[freq_{min}, freq_{max}]$ and the wavelength $[\lambda_{min}, \lambda_{max}]$. The minimum frequency is fixed as 0 and the n^{th} dimension solution is selected as $pos_{x,n}(t)$. The frequency $freq_x$, velocity $vel_x(t)$ and the position $pos_x(t)$ of the new x^{th} solution is updated as follows:

$$freq_x = freq_{min} + rand(0,1)(freq_{max} - freq_{min}) \quad (4)$$
$$vel_x(t) = vel_x(t-1) + freq_x(pos_x(t) - pos(t)) \quad (5)$$
$$pos_x(t) = pos_x(t-1) + vel_x(t) \quad (6)$$

In equation 4, rand (0,1) denotes the uniformly distributed random number between 0 and 1. In local search, the local solution is generated for each bat after the solution is acquired from the midst of the current best solution and the generation of local solution is achieved by using the random walk which is written as follows:

$$pos_{x,new}(t) = pos_{x,old}(t-1) + rand(-1,1)A_x(t) \quad (7)$$

In equation 7, $pos_{x,old}(t-1)$ denotes the old position, $rand(-1,1)$ denotes the uniformly distributed random number from -1 to 1 and $A_x(t)$ represents the average pulse rate of all bats at step time t. The pulse emission and loudness are expressed as follows:

$$A_x(t+1) = \beta A_x(t) \quad (8)$$
$$rate_x(t+1) = rate_x(t)[1 - \exp(-\gamma t)] \quad (9)$$

In equation 8 and 9, β and γ are the constants. In this the objective function is to decrease the time consumption t, memory utilization M and number of test cases n, increase the defect detection efficiency E_d and code coverage C. In the improvised BAT algorithm is same as the BAT algorithm, instead of generating new solution randomly, it generates new solution in between calculated weighted average of solutions and selected best solution which improvise the parameter setting of ARPT1 and ARPT2 strategies.[14] The following algorithm defines the improvised BAT algorithm for optimal parameter setting.

Improvised BAT based Optimal Parameter setting algorithm

Step 1: if (ARPT_strategy==ARPT 1)
Step 2: Initialize X BAT with $S_1, \dots S_h$, $x_1, \dots x_h$, $Sig_R(S)_1, \dots Sig_R(S)_h$, $\mathrm{k}(0)_1, \dots \mathrm{k}(0)_h$ and $\mathrm{l}(0)_1, \dots \mathrm{l}(0)_h$.
Step 3: Call BAT algorithm
Step 3: else
Step 5: Initialize S bacteria with $m_1, \dots m_h$ and $n_1, \dots n_h$
Step 6: Call BAT algorithm

BAT algorithm

Step 7: Randomly initialize the bat velocity vel_i
Step 8: Define pulse frequency $freq_x$ at x_i
Step 9: Initialize pulse rates $rate_x$ and the loudness A_x
Step 10: Calculate the objective function of each bat using equation 1
Step 11: while (t < maxiterations)
Step 12: Generate new solutions by adjusting frequency and updating velocities and locations using equation 4 to 6
Step 13: if (rand> $rate_i$)
Step 14: Select a solution among the best solution
Step 15: Calculate the weighted average of solution
Step 16: end if
Step 17: Generate a new solution by flying between weighted average of solution and best solution
Step 18: if (rand< A_x & F (x_i) < F (x_*))
Step 19: Accept the new solutions
Step 20: Increases $rate_i$ and reduce A_x
Step 21: end if
Step 22: Rank the bats and find the current best x_*
Step 23: end while

© 2018, IJCSE All Rights Reserved

The above improvised BAT algorithm optimizes the parameters of ARPT 1and ARPT 2 which is used to select the optimized test suite for software testing process.

IV. Results and Discussion

The experiment is conducted and results are analyzed in this section to prove the effectiveness of the proposed optimization algorithms in ARPT strategies. The experiments are conducted in terms of time consumption, defect detection efficiency, memory utilization, number of test cases and code coverage between existing and proposed optimization techniques for parameter setting of ARPT strategies.

There are three benchmark software used called univocityparser, marc4j, and jsoup which used in the experiment. The univocity-parser is a suite of extremely fast and reliable parsers for Java. It provides a consistent interface for handling different file formats, and a solid framework for the development of new parsers. The marc4j is software which provides an easy to use Application Programming Interface (API) for working with MARC and MARCXML in Java.[6] jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. The number of test case analyzed 1000 and Number of programs undergone for the test are more the 130. The following are parameters are following:

A. Time Consumption

Time consumption is a measure of the amount of time taken to test software based on optimized ARPT testing strategy.

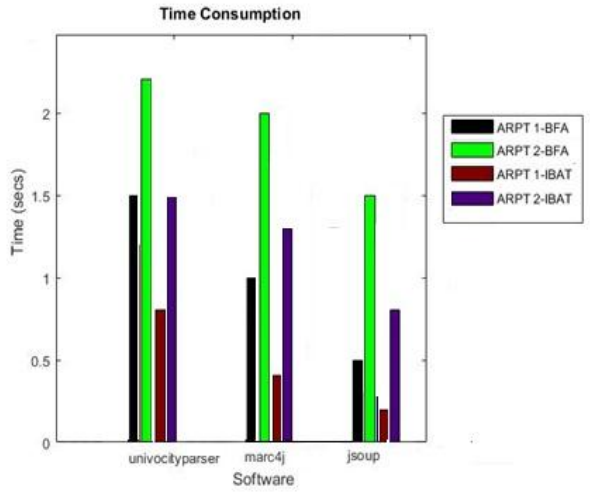

Figure 1. Comparison of Time Consumption

Figure 1, shows the comparison of time consumption between existing and proposed optimization techniques with different software. X axis represents software and Y axis represents the Time in seconds. From figure 1, it is proved that the proposed Improvised BAT optimization technique consumes less time than the other optimization technique in both ARPT testing strategies.

B. Defect Detection Efficiency

Defect detection efficiency (DDE) is the number of defects detected during a phase/stage that is injected during that same phase divided by the total number of defects injected during that phase. It can be calculated by using following formula: [4]

$$DDE = \frac{No.of\ defects\ injected\ AND\ Detected\ in\ a\ phase}{Total\ No.of\ Defects\ injected\ in\ that\ phase} \times 100\%$$

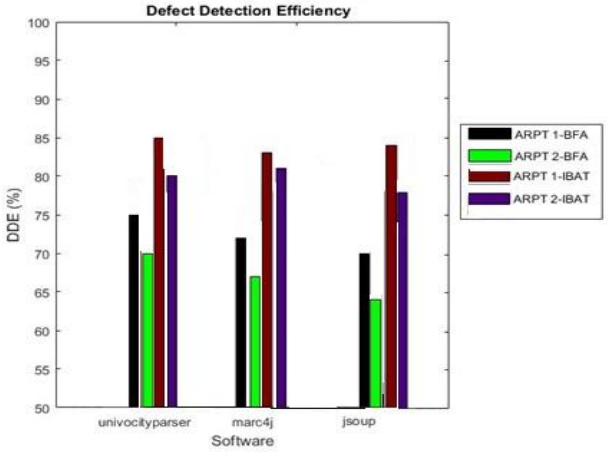

Figure 2. Comparison of Defect Detection Efficiency

Figure 2, shows the comparison of Defect Detection Efficiency between existing and proposed optimization techniques with different software. X axis represents software and Y axis represents the Defect Detection Efficiency in %. From figure 2, it is proved that the proposed Improvised BAT optimization technique has high defect detection efficiency than the other optimization techniques in both ARPT testing strategies.

V. Conclusion and Future Scope

In this paper, two optimization algorithms are introduced to improve the ARPT testing strategies. It consists of two testing strategies are ARPT 1 and ARPT 2 and which consist of different parameters. These parameters are optimized by using efficient algorithms called BFA and improvised BAT optimization algorithms. These algorithms select the most optimal parameters with less time consumption and with high accuracy that improves the ARPT testing strategies. The experimental results are conducted in three different software are university parser, marc4j and jsoup to prove the effectiveness of the proposed optimization technique in

© 2018, IJCSE All Rights Reserved

terms of time consumption and defect detection efficiency. In future advanced metaheuristic algorithms can be used for optimizing the ARPT.

REFERENCES

[1] Arcuri, A., & Briand, L." Formal analysis of the probability of interaction fault detection using random testing". *IEEE Transactions on Software Engineering*, *38*(5), 1088-1099, 2012

[2] Bashir, M. B., & Nadeem, A. (2017). Improved Genetic Algorithm to Reduce Mutation Testing Cost. *IEEE Access*.

[3] Deak, A., Stålhane, T., & Sindre, G., "Challenges and strategies for motivating software testing personnel.", *Information and Software Technology*, *73*, 1-15. 2016.

[4] Devika Rani Dhivya K., Meenakshi V.S. "An Optimized Adaptive Random Partition Software Testing by Using Bacterial Foraging Algorithm", *Lecture Notes in Computational Vision and Biomechanics*, *vol 28. Springer, Cham*, Print ISBN978-3-319-71766-1, Online ISBN978-3-319-71767-8. Pg.No. 542-555, 2018.

[5] Huang, R., Liu, H., Xie, X., & Chen, J. "Enhancing mirror adaptive random testing through dynamic partitioning". *Information and Software Technology*, *67*, 13-29, 2015.

[6] Iyad Alazzam1 , Izzat Alsmadi2 and Mohammed Akour ,"Test Cases Selection Based on Source Code Features Extraction," International Journal of Software Engineering and Its Application, Vol.8, No.1, pp.203-214. 2014.

[7] Lv, J., Hu, H., Cai, K. Y., & Chen, T. Y. , "Adaptive and random partition software testing", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(12), 1649-1664, 2014.

[8] Machado, B. N., Camilo-Junior, C. G., Rodrigues, C. L., & Quijano, E. H. "SBSTFrame: a framework to search-based software testing", In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on* (pp. 004106-004111), 2016.

[9] Mao, C. Adaptive Random Testing Based on Two-Point Partitioning. *Informatica (Slovenia)*, *36*(3), 297-303. 2012.

[10] M. Beskirli and I. Koc, "A Comparative Study of Improved Bat Algorithm and Bat Algorithm on Numerical Benchmarks," *2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, Kuala Lumpur, pp. 68-73, 2015.

[11] Shahbazi, A., Tappenden, A. F., & Miller, J. ,"Centroidal voronoi tessellations-a new approach to random testing", *IEEE Transactions on Software Engineering*, *39*(2), 163-183, 2013.

[12] Schwartz, A., & Do, H. "Cost-effective regression testing through Adaptive Test Prioritization strategies", *Journal of Systems and Software*, *115*, 61-81, 2016.

[13] Singh, K., & Rani, S. "Anti-random test generation in software testing". *Journal of Global Research in Computer Science*, *2*(5), 17-24, 2011.

[14] X.-S. Yang, A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization, (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 65-74, 2010.

© 2018, IJCSE All Rights Reserved