International Journal of Engineering Trends and Technology Volume 69 Issue 3, 39-45, March 2021
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I3P208 © 2021 Seventh Sense Research Group®

# Optimized Test Case Selection using Scout-less Hybrid Artificial Bee Colony Approach and Crossover Operator

Palak [1*], Preeti Gulia[2], Nasib Singh Gill[3]

[1]*Department of Computer Science and Applications, Maharshi Dayanand University, India*
[2]*Department of Computer Science and Applications, Maharshi Dayanand University, India*
[3]*Department of Computer Science and Applications, Maharshi Dayanand University, India*

[1]palak.aug6@gmail.com, [2]preeti.gulia81@gmail.com, [3]nasibsgill@gmail.com

***Abstract*** - *Efficient software testing depends on the quality of test cases that are capable of catching defects from every corner of the software and achieving higher coverage. In this article, a hybrid artificial bee colony optimization-based technique is proposed. The proposed approach defines the scout bee phase for abandoned solutions and incorporates features of a genetic algorithm for diversification. The proposed approach selects a minimal test suite with equivalent or better efficiency of its superset. It offers time and money-saving and contributes towards early product delivery. The proposed technique is assessed using five widely used programming problems and their mutants. When compared with similar existing techniques (i.e., Particle Swarm Optimization, Ant Colony Optimization, and Original Artificial Bee Colony) over various fitness ranges, the performance of the proposed approach shows better results and outperforms in terms of overall execution time and coverage.*

**Keywords** —*Artificial bee colony, Genetic Algorithm, Software testing, Swarm intelligence, Test case selection*.

## I. INTRODUCTION

Software engineering applies to myriad fields of industry like decision making, maintenance scheme, security services, education, health care, system dynamics, and many more. With the emergence of smart devices, the need for robust software is growing exponentially. There are remarkable welfares offered through technology for both business and society, but there are key questions around safety, privacy, sustainability, and trust. Software has become an inevitable part of life since its inception. The software comprises the programs, libraries, and concerned non-executable information, like digitized media that are used to accomplish various system activities. The ever-growing expectations of users place big challenges in front of the practitioners. One such challenge is to deliver highly available and robust software that meets the user's expectation and rightly match to the requirements. The software industry is emerging at a very fast pace with revolutionary coding styles and development paradigms. Machine learning and artificial intelligence have changed the traditional software engineering practices [1].

The majority of the software programs are written in high-level programming languages nowadays, which is simpler and more consistent for the programmers because it is nearer to human language than machine language [2]. The software can be written in more than one programming language, and each comprises a set of program development tools. Software quality is significant for system software and overall performance, including user satisfaction. The quality of software largely depends on the coding style of the programmer, the algorithm complexity, system compatibility, and the list go on [3]. To enhance the overall quality, the concept of software testing comes into the picture.

Testing itself is a challenging, costly, but important activity. However, the key issues in software testing are the generation, selection, and execution of the test cases. Effective testing largely depends on the quality of test cases. Various factors that determine test case quality include tester knowledge, clarity in requirements, coverage, manageability, etc. [4]. The other issue is that the dimension of the selected test cases may be very large. Thus, the large size of the test cases may affect the performance and delivery time of the software development life cycle. The dimensionality of test cases makes testing an expensive task. Several efforts have been made in lowering the cost of testing by using automated tools. Automated software testing employs optimization techniques for the generation, selection, and prioritization of test cases. The automated testing process is easy to repeat and is flexible. Various researches are being carried out using machine learning and soft computing approaches in the field of automated software testing [5][6].

This article presents a hybrid approach for test automation by selecting an efficient set of test cases by employing concepts of nature-inspired optimization techniques. Our proposed approach utilizes the efficiency of "Artificial Bee Colony (ABC) Optimizer" [D. Karaboga, 2007] by hybridizing it with Genetic Algorithm (GA) operators.

An efficient algorithm inspired by the social behavior of bees in search of food was given by Dervis Karaboga et al. [7] named "Artificial Bee Colony (ABC)" Optimization. They considered three types of bees and converted their behavior to a mathematical model. Initially, half of the bees in the beehive are termed "Employed Bees." They search for the food randomly near the hive and come back to the hive. They dance in front of the second set of the bees called "Onlooker Bees." This dance serves as the probability function for comparison and selection of the better food source. If the food source of any Employed Bee is exhausted, then it becomes the scout and serves as the stopping criteria for the algorithm. Due to its lightweight deployment with very small amounts of controller factors, numerous hard works have been done to discover ABC research. Originally the ABC technique employs three types of bees: Employed, Onlooker, and Scout bees [7]. The employed bees are linked to a definite food source. Initially, one employed bee is assigned to a food source. They transmit vital information such as navigation information, location, and the profitability of the food source and carry the data with the rest of the bees at the beehive. The onlooker bees are accountable for food source detection exploiting the information delivered by employed bees. The scout bees dispensed randomly to hunt the new food source whenever there is no further improved solution is found by either employed or onlooker bees [D. Karaboga, 2007]. The assumption is that the employed bees whose food source is exhausted are transformed into "scout bees" and commence a new exploration for the food source. The parallel conduct of these three bees speeds up the generation of feasible independent paths and software test suite optimization. ABC performs competitively to other conventional soft computing techniques and has gained popularity over the last decade due to its easy implementation.

The rest of the article is organized as follows: section 2 gives a brief summary of the related work and literature survey of various ongoing researches on hybrid ABC. The hybrid proposed approach is given in section 3. The proposed technique is evaluated and compared to various well-known existing techniques, and the results are given in section IV. Finally, we will conclude the research findings in section V.

## II. LITERATURE REVIEW

This section presents available knowledge in the field of related research work present in the form of vast literature over the past few decades. Several efforts have been made in this area, covering many application areas, including the software industry, numerical optimization, data mining, networking, and many more [8]. Swarm Intelligence (SI) is a popular field of research that is motivated by the natural phenomenon of the population (group) of various living organisms in their natural habitat for the search of food, shelter, and security. The community behavior of real living organisms dwelling in nature to protect and feed their community is the real inspiration behind SI [9].

Being a powerful yet simple technique, ABC is employed by researchers for optimization of the testing process and also for automation. Artificial bee colony (ABC) is well explored in the field of software testing over the past decade by augmenting it with other techniques to enhance its performance. To make the best use of it and to achieve the highest efficiency, several hybrid methods have been proposed over the years, which resulted in vast literature available online and offline.

Cuckoo search has been applied to a wide range of applications, including software testing. The same is applied along with bee colony on model-based testing for automated test case generation and selection [10]. L. P. et al. utilized the benefits of both optimizers for software testing using statecharts and sequence diagrams of the system under test. They considered ATM machine authentication and withdrawal functionality for evaluating the performance of their proposed hybrid technique. The "Levy flight" from the cuckoo search method is exploited to search for the best nest and abandon the others. Various modifications in the original ABC is carried out over the years. One such example is "Scout less ABC with modified onlooker bees," proposed by K. Hussain et al. It was argued by the authors that scout bees are counterproductive. To reach more diversity among the population, onlooker bees are modified, and the performance is evaluated using well-known classification problems by training a fuzzy neural network [11]. This methodology provides a lightweight optimizer for test case optimization. State table-based testing is also gaining importance day by day. A "comprehensive improved ACO" is proposed on the state table that is generated from the state graph. ACO is applied to achieve high coverage and efficiency [12]. The native exploration of the ABC algorithm lacks an information-based procedure [13]. So, to improve the global information-based solution, a memory element is added to employed bees that remember the global best solution so far. This provides a hybrid ABC approach that is based on PSO's gbest and pbest parameters [14]. Hu Peng et al. also proposed a similar approach for optimization that is based on "best neighbor guided ABC." They argued that it is risky to merely abandon the exhausted food source at the scout bee phase because the discarded one may have more beneficial information. The authors compared the results of their proposed technique with several other variants of ABC and achieved higher efficiency. S. Sheoran et al. focused more on data flow testing, which is a white box testing technique and utilized the concept of memory based ABC for path saving and achieved overall test suite saving [15]. Thus, it can be clinched that hybrid ABC with memory element for saving the global best solution has gained more popularity and shown effective results. A. P. Agrawal et al. compared various swarm intelligence techniques for regression test case selection over benchmark problems. Through experimental results, they concluded that hybrid PSO outperforms ACO with 0.7 % test case selection [16]. F.

Hamad employed ABC for path coverage by finding optimal fitness value among a range of values [17]. Researchers are still interested in well established Genetic Algorithm (GA) approach for the automated generation of test cases. The authors presented various encoding techniques for the same using GA [18]. The two main operators, i.e., crossover and mutation, are very powerful for producing new generations of solutions and can be amalgamated with other techniques to achieve hybrid properties. GA is also exploited in various areas of engineering such as machine learning [19], numerical optimization, image processing, etc. Several other techniques for prioritizing test cases have evolved over the years. Omdev et al. have proposed a hybrid cuckoo and neural network-based technique for the same [20].

The study of vast literature gave us insights into the importance of swarm intelligence-based techniques for optimizing the search problems. It is also clinched that researchers are interested in utilizing the best out of the various techniques by augmenting one with the other keeping the tradeoff balanced. ABC also has the potential to discover more faults with lesser execution time and a shorter test suite.

## III. THE PROPOSED APPROACH

Artificial bee colony optimization is a lightweight technique for optimizing search problems. It has a very smaller number of control parameters that provide both local and global search. In this article enhanced version of scout-less ABC is proposed to select a minimal test suite with the efficiency of higher fault coverage. The flowchart of the proposed approach is given below in Fig 1. The proposed approach starts with the initialization of parameters such as initial population size (i.e., No. of test cases in our case) TN. Each test case is represented by Ti, where i = 1, 2, . . ., TN.

Before applying the proposed technique, the selected programs are injected with mutants. Each mutant creates a point of error in the program which the test cases need to detect. A mutant can also change the flow of control of the program for a given set of input values. Based on these mutants, the initial fault matrix is created. Each solution (test case) is related to D dimensional parameter vector that defines a particular solution based on fault matrix i.e.

Xi = $\{x_i^1, x_i^2, \ldots, x_i^D\}$, i = 1, 2, . . ., TN.

For a fault j in fault matrix for the ith test case, the initial value $x_i^j$ is generated by

$$x_i^j = x_{min}^j + \text{rand}(0, 1) \times (x_{max}^j - x_{min}^j), \qquad (1)$$

where, i = 1, 2, . . ., TN and j = 1, 2, . . ., D. rand(0, 1) is a random number whose value belongs to [0, 1], max and min are the maximum and minimum value in case of each parameter respectively. Each employed bee maintains individual solutions, so their number is equal to the total number of test cases, that is, TN. For each test case i, employed bees to generate a new vector, Yi.

The neighbor search is performed by modifying the jth parameter of Yi where j ∈ {1, 2, . . ., D}. The following equation is used for updates done by employed bees (Karaboga and Basturk 2007):

$$y_i^j = x_i^j + \Phi_i^j \times (x_i^j - x_k^j). \qquad (2)$$

Here k is randomly selected, and i ≠ k. Xi will be replaced by Yi in the population greedily if Yi is better. $\Phi_i^j$ is for randomness ranging in [−1, 1]. The probability of selecting a test case "i" by an onlooker bee is denoted by pi, which is calculated by

$$p_i = \frac{fit_i}{\sum_{j=1}^{TN} fit_j} \qquad (3)$$

where fit$_i$ denotes the fitness value of ith test case, which is calculated on the basis of the probability of test cases to find a given set of errors. The onlooker bee also generates a new solution Yi using equation (2) similar to the employed bee.

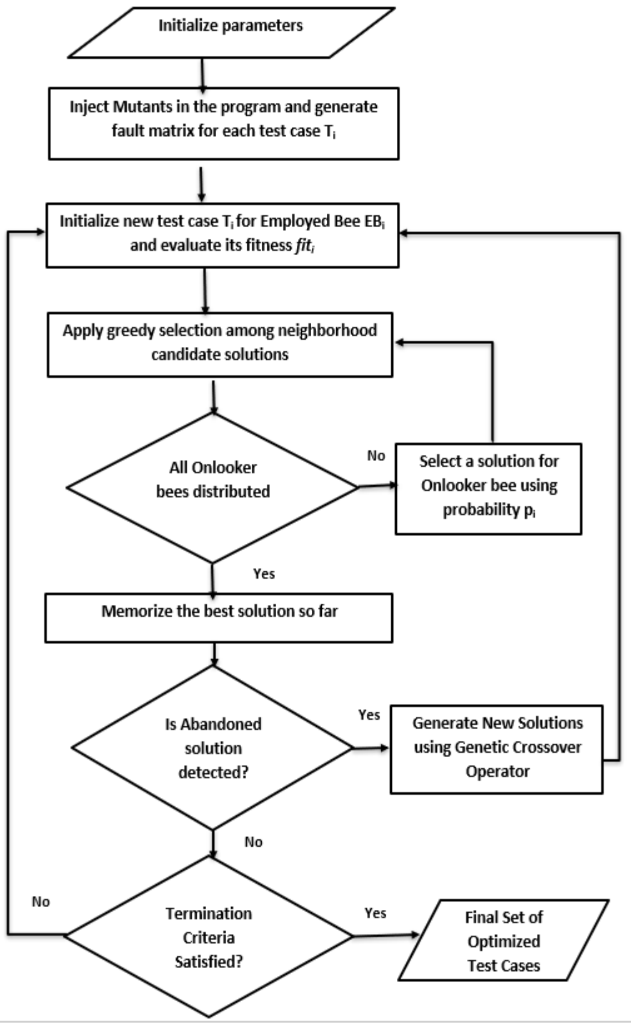
Initialize parameters
Inject Mutants in the program and generate fault matrix for each test case T$_i$
Initialize new test case T$_i$ for Employed Bee EB$_i$ and evaluate its fitness $fit_i$
Apply greedy selection among neighborhood candidate solutions
All Onlooker bees distributed
No
Select a solution for Onlooker bee using probability p$_i$
Yes
Memorize the best solution so far
Is Abandoned solution detected?
Yes
Generate New Solutions using Genetic Crossover Operator
No
Termination Criteria Satisfied?
No
Yes
Final Set of Optimized Test Cases

**Figure 1: Flowchart of Hybrid Scout Less ABC Technique**

The employed bee and onlooker bee in the proposed technique works similar to the basic ABC optimizer, but the scout bees are compromised here. Instead of the scout bees, whenever an abandoned solution is detected (i.e., a solution with no further improvement), we took advantage of hybridization with GA. Instead of random search by scout bees for a new solution, it is preferred to use the genetic crossover operator of GA to produce new solutions. The crossover operator is similar to the biological crossover of DNA in which a new child population is created by crossing parts of the parent population. Thus, the abandoned solution is crossed with the best solution so far to produce new offspring. The new offspring is then evaluated using fitness $fit_i$, and the process goes on until the stopping criteria are met.

## IV. RESULTS AND DISCUSSION

In this section, implementation setup and performance evaluation of the proposed approach are given along with the graphical results. The proposed methodology is implemented by taking some well-known programming problems into consideration.

### A. Implementation Setup

The proposed hybrid ABC approach is implemented in Visual Studio C# 2010 on Microsoft Windows 10 (64 bits), Intel Core i5 @2.40 GHz, and 8 GB memory. The performance of the proposed technique is assessed by taking five well-known programming problems into consideration. The details of the programs take into consideration are given below, consisting of the number of inputs, input domain, line of code (LOC), total number of possible paths, and description. These programs are written in C#.Net 2010.

**Table 1. Program Details for Evaluation of Proposed Approach**

| Sr. No. | Program | #input | Input domain | LOC | #Possible independent paths | Description |
|---|---|---|---|---|---|---|
| 1 | BinarySearch | 2 + array size | Integer (32 Bits) | 47 | 6 | Searches an element in a sorted list |
| 2 | Quadratic Equation | 3 | Integer (32 Bits) | 43 | 3 | Finds roots of a quadratic equation |
| 3 | TriangleType | 3 | Integer (32 Bits) | 38 | 4 | Tells about the type of triangle based on its sides |
| 4 | MergeSort | 1+ array size | Integer (32 Bits) | 74 | 5 | Recursive sorting using Merge sort |
| 5 | MatrixMultiplication | 4+ matrix 1 size + matrix 2 size | Integer (16 Bits) | 80 | 8 | Multiplies two matrices if they fulfill the condition |

### B. Performance Evaluation

Branch coverage and execution time have remained the most critical and worthy performance evaluation criteria for any test case selection technique. The proposed technique is assessed on the basis of the following three performance evaluation criteria, namely "Average Coverage," "Execution Time," "Percentage of test case selected." The obtained values of the following metrices are also used to compare the proposed technique with other competitive approaches. Description of these criteria is given below:

#### a) Average coverage

The percentage of branches covered is calculated against the total number of branches as given in eq. 4. Then, the average is calculated for all the repeated runs [19].

$$\text{Average coverage} = \frac{\sum_{i=1}^{n} BrC_i}{n} * 100 \qquad (4)$$

Where BrCi is the Branch coverage in ith run and n is the total number of runs.

The experimental results are given in Table 2 for 100 runs. The proposed technique is compared with ACO, PSO, original ABC. It is hereby depicted from the following experimental data that the proposed hybrid approach shows significant improvement in average coverage and provides near-optimum results (Fig. 2).

**Table 2: Average Coverage**

| Sr No. | Program | ACO | PSO | ABC | Proposed Hybrid ABC |
|---|---|---|---|---|---|
| 1 | BinarySearch | 98.3 | 99.2 | 99.00 | 99.2 |
| 2 | QuadraticEquation | 99.4 | 99.3 | 98.6 | 99.4 |
| 3 | TriangleType | 98.5 | 98.3 | 99.4 | 99.5 |
| 4 | MergeSort | 98.3 | 98.5 | 98.2 | 100 |
| 5 | MatrixMultiplication | 97.4 | 99.2 | 97.6 | 99.0 |

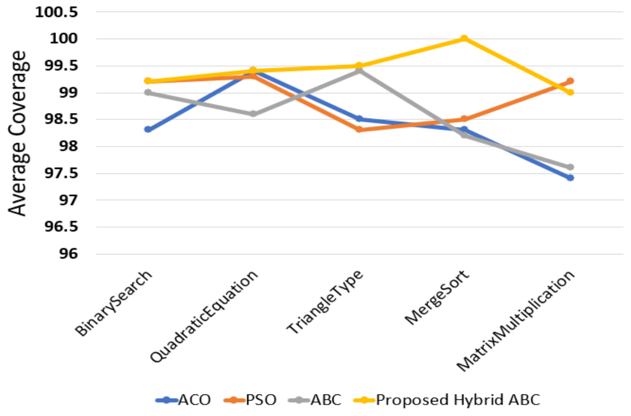

**Fig 2: Comparison of Average Coverage**

***b) Execution Time***

Execution time to run the proposed technique is measured in milliseconds for all the runs individually, and then their average is calculated. The results are given in Table 3. The experimental results of 100 runs show that the proposed approach significantly reduces the execution time and outperforms all its competitors with great margins Fig 3.

**Table 3. Execution Time in millisecond(ms)**

| Sr No. | Program | ACO | PSO | ABC | Proposed Hybrid ABC |
|---|---|---|---|---|---|
| 1 | BinarySearch | 5.34 | 1.18 | 0.88 | 0.298 |
| 2 | QuadraticEquation | 6.44 | 3.52 | 1.12 | 0.302 |
| 3 | TriangleType | 5.43 | 2.34 | 0.87 | 0.256 |
| 4 | MergeSort | 8.55 | 3.78 | 0.67 | 0.483 |
| 5 | MatrixMultiplication | 9.87 | 4.36 | 0.97 | 0.504 |

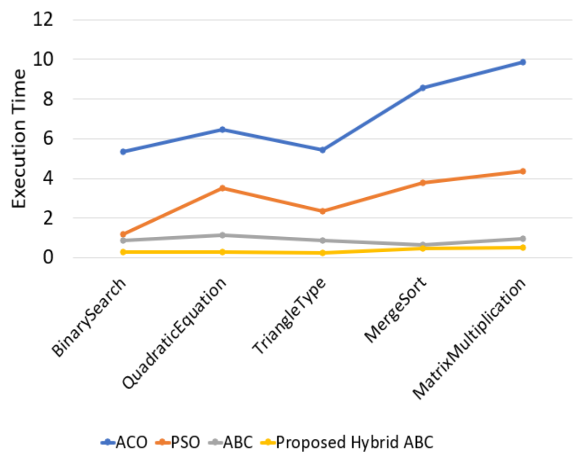

**Fig 3: Comparison of Execution Time (ms)**

***c) Percentage of selected test cases***

Table 4 shows the experimental data regarding the percentage of selected test cases over 100 runs over various fitness ranges. Fitness range is divided into two groups i.e., $0 \leq f(x) < 0.5$ and $0.5 \leq f(x) < 1.0$. It can be argued that the proposed technique is capable of selecting a shorter subset of efficient test cases with lower execution time over a given fitness range (Fig 4).

**Table 4. Percentage of test case selected over various fitness range**

| Program | Fitness Value Range | ACO | PSO | ABC | Proposed Hybrid ABC |
|---|---|---|---|---|---|
| BinarySearch | $0 \leq f(x) < 0.5$ | 40.75 | 42.5 | 35.5 | 35.5 |
| | $0.5 \leq f(x) < 1.0$ | 36.25 | 38.75 | 32 | 31 |
| QuadraticEquation | $0 \leq f(x) < 0.5$ | 43.75 | 47.5 | 40.75 | 35.75 |
| | $0.5 \leq f(x) < 1.0$ | 33.7 | 40.5 | 31.45 | 30 |
| TriangleType | $0 \leq f(x) < 0.5$ | 49.6 | 40 | 45.5 | 42.75 |
| | $0.5 \leq f(x) < 1.0$ | 39.75 | 38.75 | 37.75 | 35.5 |

| | | | | | |
|---|---|---|---|---|---|
| MergeSort | 0 ≤ f(x) < 0.5 | 42.75 | 39.5 | 41.5 | 34.75 |
| | 0.5≤ f(x) < 1.0 | 31.5 | 32.75 | 30 | 30.25 |
| MatrixMultiplication | 0 ≤ f(x) < 0.5 | 50.5 | 45.5 | 47 | 40.75 |
| | 0.5≤ f(x) < 1.0 | 40.25 | 38.3 | 35.5 | 32 |

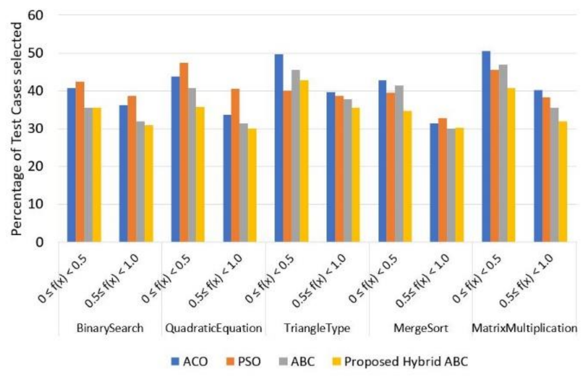

**Fig 4: Comparison of Percentage of test cases selected**

It is hereby depicted from the graphical comparison between various competitive techniques and the proposed methodology that the later one is much more economic and efficient. The comparison is made over various fitness ranges, and from the experimental results, it is argued that the proposed technique provides a shorter test suite with equivalent efficiency.

## V. CONCLUSION

Swarm-based search techniques are very helpful in optimizing the test case selection process. The same is exploited in this research work. A hybrid ABC technique for optimizing overall branch coverage in minimum time is proposed. The proposed approach uses the concept of scout-less bees with a Genetic crossover operator. The proposed technique is compared with its competitors, i.e., ACO, PSO, and original ABC, by considering five well-known programming problems. Initially, the programs under consideration are injected with mutants for better control and assessment. Three performance evaluation criteria are considered, namely "Average Coverage," "Execution Time," "Percentage of test case selected." The results verify the better performance of the proposed technique in comparison to other swarm intelligence techniques at a various fitness value range. The proposed model is capable of providing higher branch coverage and lesser execution time with a comparatively lower percentage of selected test cases. In the future, the proposed technique will be evaluated by taking more complex problems into consideration.

## REFERENCES

[1] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, Software Engineering for Machine-Learning Applications: The Road Ahead, IEEE Softw., 35(5) 81–84.

[2] doi: 10.1109/MS.2018.3571224.

[3] W. E. Lewis, Software Testing and Continuous Quality Improvement. CRC Press, (2017).

[4] M. S. Hemayati and H. Rashidi, Software Quality Models : A Comprehensive Review and Analysis, J. Electr. Comput. Eng. Innov.,6(1)(2019) 59–76, doi: 10.22061/JECEI.2019.1076.

[5] S. O. Barraood, H. M. Haslina, F. Baharom, and M. Intelligences, Test Case Quality Factors : Content Analysis of Software Testing Websites, Webology Spec. Issue Artif. Intell. Cloud Comput., 18, 75–87doi: 10.14704/WEB/V18SI01/WEB18007.

[6] J. Kim and J. W. Ryu, Machine Learning Frameworks for Automated Software Testing Tools : A Study, Int. J. Contents,13(1)(2017) 38–44.

[7] D. B. Mishra, R. Mishra, and K. N. Das, A Systematic Review of Software Testing Using Evolutionary Techniques, in Sixth International Conference on Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing, 2(2017) 546 174–184, doi: 10.1007/978-981-10-3322-3.

[8] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization : artificial bee colony ( ABC ) algorithm, J. Glob. Optim., 39(3) (2007) 459–471 doi: 10.1007/s10898-007-9149-x.

[9] K. Singh Kaswan, S. Choudhary, and K. Sharma, Applications of Artificial Bee Colony Optimization Technique Survey, in 2nd International Conference on Computing for Sustainable Global Development (INDIACom), (2015) 1660–1664.

[10] A. Chakraborty and A. K. Kar, Swarm Intelligence : A Review of Algorithms Swarm Intelligence : A Review of Algorithms, in Nature-Inspired Computing and Optimization, Modeling and Optimization in Science and Technologies 10(2017) 475–494.

[11] L. P and T. V Suresh Kumar, Automatic Generation and Optimization of Test case using Hybrid Cuckoo Search and Bee Colony Algorithm, J. Intell. Syst., 30(1)(2021) 59–72.

[12] K. Hussain, M. N. Mohd Salleh, S. Cheng, Y. Shi, and R. Naseem, Artificial bee colony algorithm: A component-wise analysis using diversity measurement, J. King Saud Univ. - Comput. Inf. Sci., 32(7)(2020) 794–808 doi: 10.1016/j.jksuci.2018.09.017.

[13] S. S. S. B and V. C. S. S. B, An Ant Colony Optimization Algorithm Based Automated Generation of Software Test Cases, in International Conference on Swarm Intelligence. (ICSI 2020), 1(2020) 231–239, doi: 10.1007/978-3-030-53956-6.

[14] A. K. Alazzawi, H. M. Rais, and S. Basri, HABC: Hybrid artificial bee colony for generating variable T-way test sets, J. Eng. Sci. Technol., 15(2)(2020) 746–767.

[15] A. K. Alazzawi, H. Rais, S. Basri, and Y. A. Alsariera, PhABC : A Hybrid Artificial Bee Colony Strategy for Pairwise test suite P h ABC : A Hybrid Artificial Bee Colony Strategy for Pairwise test suite Generation with Constraints Support, in IEEE Student Conference on Research and Development (SCOReD), (2019) 106–111, doi: 10.1109/SCORED.2019.8896324.

[16] S. Sheoran, N. Mittal, and A. Gelbukh, Artificial bee colony algorithm in data flow testing for optimal test suite generation, Int. J. Syst. Assur. Eng. Manag., 11(2)(2020) 340–349 doi: 10.1007/s13198-019-00862-1.

[17] A. Agrawal and A. Kaur, A Comprehensive Comparison of Ant Colony and Hybrid Particle Swarm Optimization Algorithms Through Test Case Selection A Comprehensive Comparison of Ant Colony and Hybrid Particle Swarm Optimization Algorithms Through Test Case Selection, Adv. Intell. Syst. Comput., (2018) 397–405, doi: 10.1007/978-981-10-3223-3.

[18] F. Hamad, Using Artificial Bee Colony Algorithm for Test Data Generation and Path Testing Coverage, Mod. Appl. Sci., 12(7)(2018) doi: 10.5539/mas.v12n7p99.

[19] Baswaraju Swathi, Dr. Harshvardhan Tiwari. Genetic Algorithm Approach to Optimize Test Cases, International Journal of Engineering Trends and Technology (IJETT), 68(10) 112-116.

[20] Jayakumar Sadhasivam, Senthil Jayavel, Arpit Rathore. Survey Of Genetic Algorithm Approach In Machine Learning International Journal of Engineering Trends and Technology (IJETT), 68(2) 115-133.

[21] Omdev Dahiya, Kamna Solanki. An Efficient Requirement-based Test Case Prioritization Technique using Optimized TFC-SVM Approach International Journal of Engineering Trends and Technology (IJETT), 69(1) 5-16.

[22] Z. K. Aghdam and B. Arasteh, An Efficient Method to Generate Test Data for Software Structural Testing Using Artificial Bee Colony Optimization Algorithm, Int. J. Softw. Eng. Knowl. Eng., 27(6)(2017) 951–966 doi: 10.1142/S0218194017500358.