Indonesian Journal of Electrical Engineering and Computer Science
Vol. 15, No. 1, July 2019, pp. 504~510
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v15.i1.pp504-510

Evolutionary algorithms for path coverage test data generation and optimization: a review

Deepti Bala Mishra[1], Arup Abhinna Acharya[2], Rajashree Mishra[3]
[1,2]School of Computer Engineering, KIIT University, India
[3]School of Applied Sciences, KIIT University, India

Article Info

Article history:

Received Aug 25, 2018
Revised Nov 17, 2018
Accepted Mar 3, 2019

Keywords:

Ant colony optimization (ACO)
Artificial bee colony Optimization (ABCO)
Genetic algorithm (GA)
Particle swarm optimization (PSO)
Path Testing

ABSTRACT

Software testing is very time consuming, labor-intensive and complex process. It is found that 50% of the resources of the software development are consumed for testing. Testing can be done in two different ways such as manual testing and automatic testing. Automatic testing can overcomes the limitations of manual testing by decreasing the cost and time of testing process. Path testing is the strongest coverage criteria among all white box testing techniques as it can detect about 65% of defects present in a SUT. With the help of path testing, the test cases are created and executed for all possible paths which results in 100% statement coverage and 100% branch coverage. This paper presents a systematic review of test data generation and optimization for path testing using Evolutionary Algorithms (EAs). Different EAs like GA, PSO, ACO, and ABCO based methods has been already proposed for automatic test case generation and optimization to achieve maximum path coverage.

Corresponding Author:

Deepti Bala Mishra,
School of Computer Engineering,
KIIT Deemed to be University, Bhubaneswar, India.
Email: dbm2980@gmail.com

1. INTRODUCTION

The runtime standard of the software is tested to maximum limits for qualitative software [1]. Software industry suffers with a heavy loss of $500 billion per year due to decrease in software quality or software failure [2]. Software failure caused by different faults and those faults can be detected by software testing. For high quality software that, satisfies the user specifications and requirements, testing are required [3]. Software testing is the process of finding and resolving the error (s) through which, software quality can be improved. The error(s) can be identified by executing the code with a set of test inputs called as test data or test case [4-5], where test case is a triplet defined as [I, S, O], I is the input data to the system, S is the state of the system at which data is input and O is the expected output [6-7]. Test case generation and test case execution require lots of effort. It is not possible always as there is no limit on test data generation but we have a limit to the cost and time of the testing process [8]. It is very time consuming, less reliable, incomplete coverage and risky process as it suffers from the drawbacks such as operation speed, high investment of cost and time, limited availability of resources, redundancy of test cases, inefficient and inaccurate test checking [9]. These drawbacks can be overcome by automated testing, which leads to decrease in cost and time of testing process. It is the most important aspect of automatic testing. So in recent day's automated software testing, and developing of high quality test cases, are two main objectives for each and every software industry [9-10]. Software testing can be broadly divided in two different ways as random based testing and search based testing [11].

a. Random Based Software Testing (RBST)
b. Search Based Software Testing (SBST)

Random process is the simplest way for generating test data, but the probability of satisfying the constraints of the tested programs is very low. It simply executes the program with random inputs and check whether the expected output is satisfied or not. One of the major problem in RBST is sometimes none of the test data reaches the target test data often called as critical data [12]. But search based approach is widely used in recent years to solve many optimization problems in the field of Search Based Software Testing (SBST). In search based technique the target criterion is converted to an optimization problem, so that different types of Evolutionary Algorithms (EAs) such as GA, PSO, ACO, ABCO etc. are used to solve the specified problem by providing a global optimum or nearer optimal solution [13-14].

This paper presents a systematic review on test data generation for path testing using different EAs. The rest of the paper is organized as: Section 2 describes some basic concepts, which are used in our research paper. Section 3 discusses related work on path coverage based testing using different EAs. In Section 4, the conclusion of the paper along with some future works are outlined.

2. BASIC CONCEPTS

In this section a few background concepts and definitions arediscussed, which are used throughout the research work.

2.1. Path Testing

Basically, testing is done in four different levelsviz. unit testing, integration testing, system testing and acceptance testing. Among all kinds of software testing techniques,unit testing is the base of all other types of testing [15]. It is done in the coding stage and if it is not done properly, thecost and time for other testing will increase. So unit testing plays an important role in maintaining the software quality [14]. There are two different ways to generate test cases for unit level testing as: [16-18].

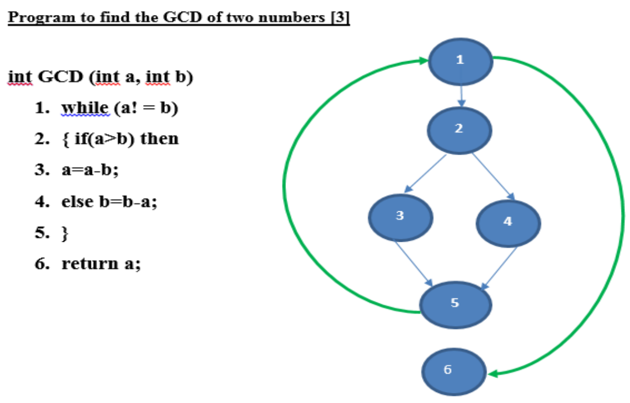
a. White box approach (Glass box or structural approach).
b. Black box approach (Functional approach).
c. Structural testing mainly involves testing process of a unit or modules and is very important for software developer. To test an unit or a module, different coverage based testing techniques are used such as statement coverage, condition coverage, multiple conditions coverage and path coverage [17], [19]. Among all coverage based testing, path coverage based testing is the strongest criterion based testing as it can detects about 65% of defects present in a Software Under Test (SUT). Literature says that many studies have been already done for unit testing but it is seen that a less focus has been paid towards path testing [19-20].
d. Path testing was first introduced by Howden in 1976.It allows finding a logical error(s) as errors/faults associated with different number of iterations that are exposed in different paths. The detection of logical errors may not possible in case of branch or statement coverage based testing [18]. In path testing, test cases are designed in such a way that all linearly independent paths of a particular SUT, should be executed at least once. A linearly independent path can be obtained from the Control Flow Graph (CFG) of a program which shows the flow of sequence in a program [20-21]. McCabe's Cyclomatic Complexity gives the upper bound value of the linearly independent paths present in a program. The CC of a program can be found by using (1).

$$V(G) = E - N + 2 \qquad (1)$$

One example is shown in Figure 1, which shows the different steps carried out to find the linearly independent paths for a specific program. With the help of path testing, the test cases are created and executed for all possible paths which results in 100% statement coverage and 100% branch coverage [21-22].

2.2. Critical Path

During path testing, the path for which there is no test data generated and searching for the test data to cover that particular path can never be succeed, is called as Critical path.In such cases some criterion is needed to stop the searching process for the test data that covers the critical path and it is almost sure that further searching is worthless [23-26].



Program to find the GCD of two numbers [3]

```
int GCD (int a, int b)
   1. while (a! = b)
   2. { if(a>b) then
   3. a=a-b;
   4. else b=b-a;
   5. }
   6. return a;
```

Figure 1. CFG for GCD program

The linearly independent paths are as follows

1→6
1→2→3→5→6
1→2→4→5→6

2.3. Evolutionary Algorithm (EA)

Evolutionary algorithms (EAs) are based on biological behavior or evolution of population [27-28]. This algorithm is based on the principle of survival ofthe fittest and models some natural phenomena like genetic inheritance and Darwinianstrife for survival, constitute an interesting category of modern heuristic search [29-30].A strategy has been developed, to greatly reduce the necessary time and computational costs to achieve maximum benefits in the form of soft computing techniques like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony Optimization (ABCO) etc. [30-31].

2.3.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) is an evolutionary algorithm, which was developed by John Holland in 1975 [27-28]. GA has emerged as a practical, robust optimization technique and search method. It is inspired by the way nature evolves species using natural selection of the fittest individuals [29].

In the process of solving the problem, GA comes across a set of feasible solutions, called as the search space, where the individual solution is marked by its fitness value or score towards the problem. The fitness value or score of the individuals is determined through a predefined fitness function. This value defines the fitness of individual solution towards the given problem and facilitates the decision making regarding, which solution is to be included and which is to be discarded for next generation.It always aims for the optimal solution using some extreme value like searching for a minimum or maximum in the search space[30-31]. The algorithm provides the global optimum solution by employing its different operators, such as selection, crossover, mutation and elitism [32-34].

It is an 8-tupled expression defined in (2) [25], [35-36].

$$\boldsymbol{GA} = (\mathbf{Co}, \mathbf{F}, \mathbf{Po}, \mathbf{N}, \mathbf{S}, \mathbf{C}, \mathbf{M}, \mathbf{T}) \tag{2}$$

Where,*Co = Individual coding method,F =Individual fitness evaluation, Po= Initial population*
N= Population scale, S= Preferred selection operator, C =Preferred crossover operator
M=Preferred Mutation operator, T =Suspension of operation algebra.

2.3.2 Particle Swarm Optimization (PSO)

PSO is an evolutionary computation technique developed by kennedy and eberhart in 1995, that studies the social behavior of bird flocking or fish schooling. It begins with a group of randomly generated individuals called as initial population. The best solution can be found by a number of particles constituting a swarm, moving around in a particular real valued N-dimensional search space and adjusting their flying according to own and other's flying experience [37]. A fitness is defined to evaluate each particle from the population. In this process each particle is asigned a coordinates in the form of location and velocity, which are associated with best solution. At each steps of this process the velocity of each particle is changed to achieve best fitness(pbest) than the overall best(gbest) value obtained by any particle in the population. The particles are called as potential solution. Acceleration is weighted by a random term w called as weight inertia. The velocity and new particles can be updated by using (3) and (4).So different new population are generated for acceleration towards pbest and gbest locations. PSO has a premature convergence problem ie. It convergence to the local best solution [38-39].

$$v_i^{(t+1)} = wv_i + c_1r_1(p_i^{\ t} - x^i) + c_2r_2(g_i^{\ t}) \tag{3}$$

$$x_i^{\ (t+1)} = v_i^{\ (t+1)} + x_i^{\ t} \tag{4}$$

2.3.3 Ant Colony Optimization (ACO)

ACO is a distributed meta-heuristic algorithm, inspired by biological behaviors of real-world ants mainly used to solve many optimization problems. The Aunt System Algorithm was first proposed by Marco Dorigo et al. in 1991 to solve combinatorial discrete optimization problems [40]. In this algorithm, the optimization problem is represented as a graph and the artificial ants move around the paths of the graph repeatedly to find the best solution. During food searching the ants leave a chemical level called as pheromone on the randomly travelled paths so that other ants can coordinates with each other. Ants select their paths according to the higher pheromone levels of the graph edges. After some traversals the pheromon level of shortest path becomes higher than the others because the pheromon evaporation is more in longer paths.For each iteration possible solutions are created and finally evaluate the best quality solution by using a heuristic measure [41-42].

2.3.4 Artificial Bee Colony Optimization (ABCO)

ABCO is a population based process in which the independent and parallel of the scout bees, employed bees onlooker bees finds the global optimum solution faster. It is a non pheromenon based approach so no need of updation [43]. In this process the computational overhead and memory limit problems are balanced. Here some dedicated scout bees are appointed to explore flower patches in the sorrounding environment at random. The fitness value of a perticular flower patch is defined by taking the nectar amount, the distance and the direction of the designated flower patch from the bee hive. Scout bees gives the information to the onlooker bees in form of waggle dance and then the onlooker bees determine the fitness value and the probability value of the food source. The food souce with maximum profitability is selected for the exploration [44].

3. RELATED WORK

EAs are frequently used for path coverage based test data generation and optimization to achieve maximum path coverage. In this section, a few related research works on software path testing using variants of EAs has been discussed.

3.1. Test Case Generation and Optimization using GA

Hermadi et al. [19] developed a GA based approach to cover multiple path at a time. They have applied different fitness on several bench mark problems. The fitness was designed by combining the features of path traversal, neighborhood influence and normalization. They found the new GA based multiple path test data generator gives better results than the previous method. Cao et al. [13] developed an approach to generate test data, that covers only one specific path for a SUT. GA is used to increase the path coverage of a SUT for achieving the goals like better quality and reliability and the fitness is designed by taking the Overlapped Path Similarity (OPS) between the executed path and target path. The most popular program, TCP is taken for their experiments and found that GA based OPS can generate a huge number of valid test data with less consumption of time. ***Garg et al***. [26] proposed a new fitness named as Extended Level Branch (EXLB) for basic path testing using simple GA by using hill climbing method with selection operator, but the proposed method could not cover all paths. ***Zhu et al***. [34], 2017 proposed an improved GA to balance the load of each calculation resources in target paths of a SUT. A grouping strategy of target paths is defined by taking the common constraints of the target paths, to reduce the search space of test data. Symbolic execution tool along with GA is used to accelerate the convergence of search process, which leads to improve the efficiency of SBST. The proposed approach is implemented with four different programs such as bubble sort, insertion sort, select sort, and shell sort. The experimental results shows a very good performance in terms of both efficiency and load balancing of generated test data.

3.2. Test Case Generation and Optimization using PSO

Latiu et al. [37] used three different evolutionary algorithms to generate test data automatically for path testing and found evolutionary testing strategies are very well suited to generate test data for a software program. They have used GA, PSO, SA (Simulated Annealing) for their experiments. ***Huang et al***. [38] proposed a method SAF-GPSO (Swarm Activity Feedback-Gauss Particle Swarm Optimization), based on Improved PSO for multipath test case generation and found their method gives better result in comparison to GA and PSO. ***Han et al***. [21] proposed a modified multiple path test data generator using PSO. Authors have taken some bench mark problems and found their proposed approach is more effective and efficient for complicated and large path sets. They have taken different population size and from the experimental result, it is observed that the average iteration needed to cover all feasible paths, is decreases when population size increases.

3.3. Test case generation and optimization using ACO

Biswas et al. [40] proposed an ACO based approach that guarantees full software coverage with minimum redundancy. The proposed approach can generate optimal path in a prioritized order and also generates test data sequence within the domain to use as inputs of the generated paths. ***Mann et al*** [42] proposed an ACO based path prioritization to generate maximum path coverage test data. The algorithm is named as PP-ACO, which is used to generate optimal path sequence in DD graph for a SUT. The most popular search based program as TCP is taken for the experiment and the reported result shows that, the proposed technique can generates test data for full path coverage as well as prioritizes those test data according to the path strength.

3.4. Test case generation and optimization using ABC

Lam et al. [43] presented an approach for automatic generation of feasible independent test path by using edge coverage criteria. They have used ABC optimization technique to optimize test suite and show the efficiency of their proposed method by comparing with previous related approaches, but the proposed approach could not eliminate the duplicate test data in the final test suite. ***Khari et al***. [44] developed an automated testing tool for test suite generation and optimization to test a software using ABC. Their proposed method is able to provide a set of minimal test case with maximum path coverage. Authors have compared their result with CSA (Cuckoo Search Algorithm) and found the ABC based method offers better result than CSA in terms of path coverage. The reported results show that 90.3% path coverage is achieved for ABC whereas only 75.4% path coverage is achieved for CSA.

4. CONCLUSION

This paper briefly reviewed some of the related research work on path coverage based testing, one of the white box testing technique using different EAs viz. GA, PSO, ACO, ABCO. In path testing test data is generated to cover the basic path of a specific SUT. It is observed that different EAs are frequently used for test case generation and optimization to cover the basic path. However, as we move to higher path coverage based test suite with more complex software, more efficient methods are needed. Researchers have employed many different approaches, to achieve maximum coverage. However, it's very difficult to achieve 100% path coverage in complex software i.e. in terms of LOC and a large number of test data is required towards achieving a maximum. As a result, numerous algorithms have been proposed, implemented and applied to achieve highest coverage over the past decades. Presence of critical paths is also one main issues in achieving full path coverage. So detecting and generating the test data for a specific critical path is a very challenging issue during path testing.

In future it is planned, to develop an efficient EA based algorithm which, generates an optimized test suite to satisfy maximum path coverage for any SUT and simultaneously, validate the effectiveness and efficiency of the proposed algorithm in covering the most critical paths. It is also planned to design a real coded GA to generate and optimize the test data with maximum path coverage and minimum test data generation count.

REFERENCES

[1] Chauhan, N., "Software Testing: Principles and Practices", Oxford University Press, 2010.

[2] Mishra, D.B., Bilgaiyan, S., Mishra, R., Acharya, A.A. and Mishra, S., 2017. “A Review of Random Test Case Generation using Genetic Algorithm”. *Indian Journal of Science and Technology*, 10(30).
[3] Mishra, D.B., Mishra, R., Acharya, A.A. and Das, K.N., 2019. “Test Data Generation for Mutation Testing Using Genetic Algorithm”. In *Soft Computing for Problem Solving* (pp. 857-867). Springer, Singapore.
[4] Mishra, D.B., Mishra, R., Das, K.N. and Acharya, A.A., 2017. “A Systematic Review of Software Testing Using Evolutionary Techniques”. In *Proceedings of Sixth International Conference on Soft Computing for Problem Solving* (pp. 174-184). Springer, Singapore.
[5] Bhuyan, M.K., Mohapatra, D.P. and Sethi, S., 2016. “Software Reliability Prediction using Fuzzy Min-Max Algorithm and Recurrent Neural Network Approach”. *International Journal of Electrical and Computer Engineering (IJECE)*, *6*(4), pp.1929-1938.
[6] Srivastava, P.R. and Kim, T.H., 2009. “Application of Genetic Algorithm in Software Testing”. *International Journal of software Engineering and its Applications*, 3(4), pp.87-96.
[7] Ahmed, M.A. and Hermadi, I., 2008. “GA-Based Multiple Paths Test Data Generator”. *Computers & Operations Research*, 35(10), pp.3107-3124.
[8] Zhang, S., Zhang, Y., Zhou, H. and He, Q., 2010, October. “Automatic path test data generation based on GA-PSO”. In *Intelligent Computing and Intelligent Systems (ICIS)*, 2010 IEEE International Conference on (Vol. 1, pp. 142-146). IEEE.
[9] Zhang, Y. and Gong, D., 2014. “Generating Test Data for Both Paths Coverage and Faults Detection Using Genetic Algorithms: Multi-Path Case”. *Frontiers of Computer Science*, 8(5), pp.726-740.
[10] Mohi-Aldeen, S.M., Mohamad, R. and Deris, S., 2016. “Application of Negative Selection Algorithm (NSA) for Test Data Generation of Path Testing”. *Applied Soft Computing*, 49, pp.1118-1128.
[11] Manikumar, T., Kumar, A.J.S. and Maruthamuthu, R., 2016. “Automated Test Data Generation for Branch Testing Using Incremental Genetic Algorithm”. Sādhanā, 41(9), pp.959-976.
[12] Sharma, A., Rishon, P. and Aggarwal, A., 2016. “Software Testing Using Genetic Algorithms”. *Int. J. Comput. Sci. Eng. Surv.(IJCSES)*, 7(2), pp.21-33.
[13] Cao, Y., Hu, C. and Li, L., 2009, July. “An Approach to Generate Software Test Data for A Specific Path Automatically with Genetic Algorithm”. In *Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on* (pp. 888-892). IEEE.
[14] Torkamani, M.A., 2014. “Metric Suite to Evaluate Reusability of Software Product Line”. *International Journal of Electrical and Computer Engineering (IJECE)*, *4*(2), pp.285-294.
[15] Alshraideh, M., Mahafzah, B.A. and Al-Sharaeh, S., 2011. “A Multiple-Population Genetic Algorithm for Branch Coverage Test Data Generation”. *Software Quality Journal*, 19(3), pp.489-513.
[16] Khari, M. and Kumar, P., 2017. “An Extensive Evaluation of Search-Based Software Testing: A Review”. *Soft Computing*, pp.1-14.
[17] Mansour, N. and Salame, M., 2004. “Data generation for Path Testing”. *Software Quality Journal*, 12(2), pp.121-136.
[18] Gupta, M. and Gupta, G., 2012. “Effective Test Data Generation Using Genetic Algorithms”. *Journal of Engineering, Computers & Applied Sciences*, 1(2), pp.17-21.
[19] Hermadi, I., Lokan, C. and Sarker, R., 2010, December. “Genetic Algorithm Based Path Testing: Challenges and Key Parameters”. In *Software Engineering (WCSE), 2010 Second World Congress on* (Vol. 2, pp. 241-244). IEEE.
[20] Zapata, F., Akundi, A., Pineda, R. and Smith, E., 2013. “Basis Path Analysis for Testing Complex System of Systems”. *Procedia Computer Science*, 20, pp.256-261.
[21] Han, X., Lei, H. and Wang, Y.S., 2016. “Multiple Paths Test Data Generation Based on Particle Swarm Optimization”. *IET Software*, 11(2), pp.41-47.
[22] Boopathi, M., Sujatha, R., Kumar, C.S. and Narasimman, S., 2014, October. “The Mathematics of Software Testing Using Genetic Algorithm”. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference* on (pp. 1-6). IEEE.
[23] Zhonglin, Z. and Lingxia, M., 2010, August. “An Improved Method of Acquiring Basis Path for Software Testing”. In *Computer Science and Education (ICCSE), 2010 5th International Conference* on (pp. 1891-1894). IEEE.
[24] Thi, D.N., Hieu, V.D. and Ha, N.V., 2016, November. “A Technique for Generating Test Data Using Genetic Algorithm”. In *Advanced Computing and Applications (ACOMP)*, 2016 International Conference on (pp. 67-73). IEEE.
[25] Shimin, L. and Zhangang, W., 2011. “Genetic Algorithm and its Application in the Path-Oriented Test Data Automatic Generation”. *Procedia Engineering*, 15, pp.1186-1190.
[26] Garg, D. and Garg, P., 2015. “Basis Path Testing Using SGA & HGA with ExLB Fitness Function”. *Procedia Computer Science*, 70, pp.593-602.
[27] Shahbazi, A. and Miller, J., 2016. “Black-Box String Test Case Generation Through A Multi-Objective Optimization”. *IEEE Transactions on Software Engineering*, 42(4), pp.361-378.
[28] Yan, J. and Zhang, J., 2008. “An Efficient Method To Generate Feasible Paths For Basis Path Testing”. *Information Processing Letters*, 107(3-4), pp.87-92.
[29] Mishra, D.B., Mishra, R., Das, K.N. and Acharya, A.A., 2019. “Test Case Generation and Optimization for Critical Path Testing Using Genetic Algorithm”. In *Soft Computing for Problem Solving* (pp. 67-80). Springer, Singapore.
[30] Hermadi, I. and Ahmed, M.A., 2003, December. “Genetic Algorithm Based Test Data Generator”. In *Evolutionary Computation*, 2003. CEC'03. The 2003 Congress on (Vol. 1, pp. 85-91). IEEE.
[31] Deb, K., 2012. “Optimization For Engineering Design: Algorithms and Examples”. PHI Learning Pvt. Ltd.

[32] Jena, T. and Mohanty, J.R., 2016. “Disaster Recovery Services in Intercloud Using Genetic Algorithm Load Balancer”. *International Journal of Electrical and Computer Engineering(IJECE)*, 6(4), p.1828.
[33] Mishra, D.B., Mishra, R., Acharya, A.A. and Das, K.N., 2019. “Test Case Optimization and Prioritization Based on Multi-Objective Genetic Algorithm”. In *Harmony Search and Nature Inspired Optimization Algorithms* (pp. 371-381). Springer, Singapore.
[34] Zhu, Z., Xu, X. and Jiao, L., 2017, June. “Improved Evolutionary Generation of Test Data for Multiple Paths in Search-Based Software Testing”. In *Evolutionary Computation (CEC)*, 2017 IEEE Congress on (pp. 612-620). IEEE.
[35] Malhotra, R. and Kumar, N., 2016, September. “Automatic Test Data Generator: A Tool Based on Search-Based Techniques”. In *Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2016 5th International Conference on* (pp. 570-576). IEEE.
[36] R. Khan, M. Amjad and A. K. Srivastava, "Optimization of Automatic Generated Test Cases for Path Testing Using Genetic Algorithm," *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, 2016, pp. 32-36.
[37] Latiu, G.I., Cret, O.A. and Vacariu, L., 2012, September. “Automatic Test Data Generation for Software Path Testing Using Evolutionary Algorithms”. In *Emerging Intelligent Data and Web Technologies (EIDWT)*, 2012 Third International Conference on (pp. 1-8). IEEE.
[38] Huang, M., Zhang, C. and Liang, X., 2014, December. “Software Test Cases Generation Based on Improved Particle Swarm Optimization”. In *Information Technology and Electronic Commerce (ICITEC)*, 2014 2nd International Conference on(pp. 52-55). IEEE.
[39] Saravanan, C. and Panneerselvam, M.A., 2013. “A Comprehensive Analysis For Extracting Single Diode PV Model Parameters By Hybrid GA-PSO Algorithm”. *International Journal of Computer Applications*, *78*(8), pp.16-19.
[40] Biswas, S., Kaiser, M.S. and Mamun, S.A., 2015, May. “Applying Ant Colony Optimization in Software Testing to Generate Prioritized Optimal Path and Test Data”. In *Electrical Engineering and Information Communication Technology (ICEEICT)*, 2015 International Conference on (pp. 1-6). IEEE.
[41] Mohapatra, S.K. and Prasad, S., 2015. “Test Case Reduction Using Ant Colony Optimization for Object Oriented Program”. *International Journal of Electrical and Computer Engineering (IJECE)*, *5*(6), pp.1424-1432.
[42] Mann, M., 2015. “Generating and Prioritizing Optimal Paths Using Ant Colony Optimization”. *Computational Ecology and Software*, *5*(1), p.1.
[43] Lam, S.S.B., Raju, M.H.P., Ch, S. and Srivastav, P.R., 2012. “Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony”. *Procedia Engineering*, 30, pp.191-200.
[44] Khari, M., Kumar, P., Burgos, D. and Crespo, R.G., 2017. “Optimized Test Suites for Automated Testing Using Different Optimization Techniques”. *Soft Computing*, pp.1-12.

BIOGRAPHIES OF AUTHORS

Deepti Bala Mishra received her M.Tech degree in Computer Science and Engineering from BPUT in 2013. Currently, she is working as a full time research scholar in the School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India. Her research interests include Software Engineering, Soft Computing, Cloud Computing, and Data Analytics. She is a member of SCRS.

Dr. Arup Abhinna Acharya received his Ph.D. from Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar. He joined the School of Computer Engineering at Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar in 2006, where he is now Associate Professor and Program Head of Information Technology. His research interests include software engineering, Object-Oriented Systems and data analytics. He has published more than forty papers in these fields. He can be reached at acharyafcs@kiit.ac.in

Dr. Rajashree Mishra, currently working as Assistant Professor in Department of Mathematics, School of Applied Sciences, KIIT Deemed to be University, Bhubaneswar, Odisha, India. She received her Ph. D Degree from KIIT Deemed to be University in 2014. Her areas of research interest are Evolutionary Computing which specifically includes (Genetic Algorithm and Bacterial Foraging Optimization), Fuzzy probabilistic Programming and Multi-objective nonlinear optimization. She is having publications in reputed journals. She is also the Reviewer to many International Conferences. She is a member to many research societies.