International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8, Issue-1, May 2019

Random Test Data Generation for Critical Path in Software Path Testing

Deepti Bala Mishra, Arup Abhinna Acharya, Rajashree Mishra

Abstract: ***Path coverage based testing is the strongest coverage criteria among all white box testing techniques. Through this testing around 65% of defects present in the Software Under Test (SUT) can be detected. In path testing each and every linearly independent path is executed at least once. 100% statement and branch coverage can be achieved by the help of path testing i.e. path coverage based test data can be used to achieve 100% statement and branch coverage. This paper presents a random based test data generation technique which can generate a single test data at a time. The proposed method can also, generate test data to cover the most critical path present in a specific SUT. Multiple hamming distance method is used along with one to one correspondence to map the test data with corresponding path and control flow graph of SUT is considered for basic path generation. Further, the recorded result is compared with other previous work. The experimental results ensures that the proposed random based method can generate a set of test data to cover maximum basic paths present in SUT.***

Index Terms: Control Flow Graph, Path Testing, RBST, Hamming Distance, Test Data Generation

I. INTRODUCTION

Software development follows a series of steps, known as Software Development Life Cycle (SDLC) shown in Figure 1. Software testing is an important phase of SDLC, as it is a major quality control measure used in software development and the quality of software is ensured through rigorous testing [1]. So qualitative, robust and trust worthy software are ensured by efficient testing [2]. It ensures, whether the software meets the performance requirement, reliability, flexibility, correctness etc. or not. To check whether a system or a component of the whole system, satisfies the specific requirements, or not testing is done. The differences between the required and the actual results of software can only be identified by efficient testing [1]. It is also a process of executing a software to find the errors present in it. In other words, testing is the activity of executing the system in order to detect the failure. It aims to quality assurance, validation, verification, and reliability estimation [3].

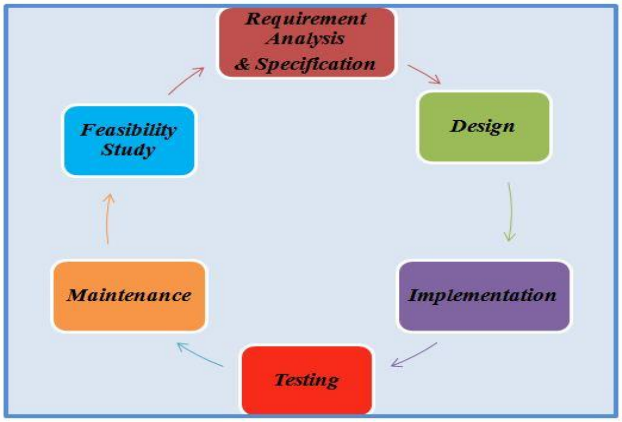

Fig. 1 Basic Phases of SDLC

During testing process two major works are done as effective test case generation and test case execution which requires a lot of efforts. 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 [4]. It is very time consuming, less reliable, incomplete coverage and risky process. But, these issues can be addressed by automatic testing. Through automated process the cost and time of testing process can also be reduced. Automatic testing also helps in increasing the reliability and efficiency. So, it is the most important aspect of software testing. In recent days each and every software industries give the priority on generating of high quality test data in an automated process [5, 6]. Researchers have shown the suitability of using different techniques in testing environment and also developed random based test data generators. In this paper a random based technique is developed to generate a test data to cover a particular target path [7].

II. BASIC CONCEPTS

This section presents some relevant basic concepts which are used in the rest of this paper.

A.Software testing

Qualitative, robust and trust worthy software are ensured by efficient testing [1]. 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 case [8]. A test case is represented in a triplet form as [I, S, O], where I indicates the input data, S is the state of the system when data is input and O is the output [1]. The goal of testing is to cover as many faults as possible with a set of test cases [9].

Revised Manuscript Received on 30 May 2019.
* Correspondence Author
Deepti Bala Mishra*, School of Computer Engineering, KIIT University, Bhubaneswar, India-751024.
Arup Abhinna Acharya, School of Computer Engineering, KIIT University, Bhubaneswar, India-751024.
Rajashree Mishra, School of Applied Sciences, KIIT University, Bhubaneswar, India-751024.

The software testing techniques can be classified as: (i) unit testing, which is done after a module or component of the software product has been coded and reviewed; (ii) integration testing , which check whether the modules of the software, interface with each other properly or not; (iii) system testing, where fully developed system is tested to assure that it fulfils all the requirements and specifications; (iv) acceptance testing, which is done to confirm that the software meets its business requirements and specifications; (v) regression testing, where an old test suite is to be executed along with the new test cases designed for updated system [7]. The testing strategy can be developed in the form of Verification and Validation (V&V), where verification means building the software product correctly, and validation is the building of correct product. V&V are the building blocks of the testing process and shows the paths for various types of testing techniques. Early testing process can be planned by following the steps of V&V diagram, shown in Figure 2 as it supports parallelism in the activities of developers and testers. Automatic test data generation process is categorized into two different types as random based approach and search based approach [8, 9].

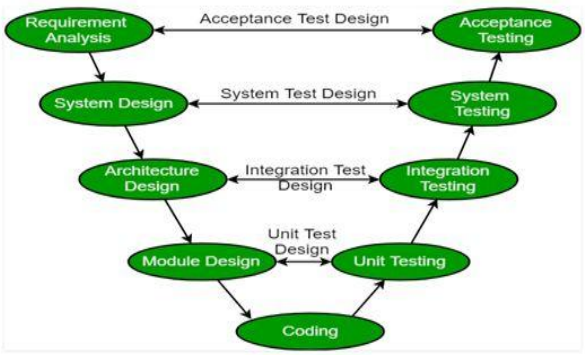

Fig. 2 V& V Model

B. Random Based Software Testing

Random based testing 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 [7]. However, achieving the target data to cover the critical path is a big challenge for researchers. To address this particular issue, an efficient random based method is designed in this paper.

C. Search Based Software Testing (SBST)

In this technique, the problem of generating test data is converted to an optimization problem [8]. To deal with the optimization problem, so many heuristic search algorithms are used in SBST [2, 7]. The solution to the problem can be found in domain of possible input data. This process starts with a random generation and during the process the target is reached to a closer optimum solution and the same process is repeated until either the algorithm in the process converges or the target solution is achieved [6].

D. Path testing

Among all kinds of software testing techniques, white box testing is the base of all other types of testing [1]. It is done in the coding stage and if it is not done properly, the cost and time for other testing will increase. So, the software quality can be maintained through unit testing [3]. In unit testing, test cases are generated by two different ways as white box approach (glass box or structural approach) and black box approach (functional approach) [2].In white box testing technique a unit or a module is tested and is very important for software developer. To test a unit or a module, different coverage based testing techniques are used. 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 [9, 10]. 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 [2]. In path coverage based testing, all linearly independent or basic paths present in SUT should be executed. The basic path can be found through the Control Flow Graph (CFG) of a SUT [7, 9]. McCabe's Cyclomatic Complexity (CC) gives the upper bound value of the linearly independent paths present in a program. The CC of a program can be found by using Eq.(1).

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

In path coverage based testing each and every possible paths of the SUT are executed so that maximum errors can be detected. This testing technique is involved with the execution of all feasible paths in the program as there may be infinite numbers of paths due to presence of loops. Path coverage based testing is the strongest among all as it satisfies all other type of coverage based testing techniques, by covering all feasible paths [9].

E. Hamming Distance

The Hamming distance of two binary code is defined as the number of points at which the values are different i.e. the number of differences between the corresponding bits. The two lines of binary code should be of same length. Data strings can be measured by this metric [11].

The Hamming distance can be found by applying the XOR ($\oplus$) operation on the two binary strings, defined in Eq. (2) and count the number of 1's in the result, which is a value greater than zero. One example is given below.

$$D(X, Y) = X \oplus Y \qquad \textbf{(2)}$$

F. One to One Correspondence

In one-to-one correspondence the elements of two different sets are exactly paired with other elements. There are no unpaired elements present in both the sets [12]. Figure 3 shows one to one correspondence, $f : A \to B$, where A and B are two finite sets. $A = \{1,2,3,4,5\}$, $B = \{U,V,W,X,Y\}$ and $f(1) = U$, f(2)=V, f(3)=W, f(4)=X, and f(5)=y.

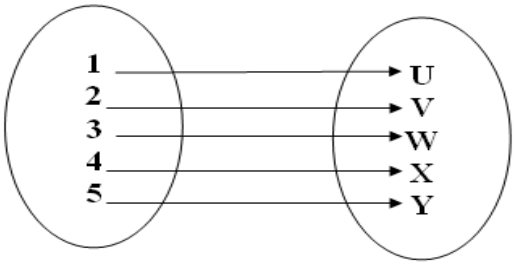

Fig. 3 One to One Correspondence

III. RELATED WORK

A lot of work have been done on the domain of test data generation for path testing. Different techniques are developed for both random and search based test data generation to achieve maximum path coverage. In this section, a few related research works on software path testing are presented, which includes test case generation using random base methods, static methods, dynamic methods.

Yan et al. [5] proposed an efficient method for feasible paths generation. Authors have taken different real world C programs for their experiments. The realizable complexity is taken to construct a test set that covers the feasible basic paths automatically. The proposed method can generate test paths when the program size is less than its realizable complexity. ***Zapata et al. [6]*** proposed a method to show how to generate a test suite for a System of System (SoS) architecture by using basic path testing analysis. They have computed a set of test cases to traverse each individual path by computing *CC* of SoS design flow graph. ***Aldeen et al. [3]*** proposed a Negative Selection based method (NSA) to generate test data automatically. They have taken thirteen number of bench mark programs for their experiments. Their reported result confirms that the total number of test data generation needed for maximum path coverage is reduced by the proposed method. The efficiency of test data generation process is also increases. Authors have proposed a random based method to evaluate the proposed NSA based method. ***Gotlieb et al. [10]*** proposed a divide-and-conquer based method to generate random test data. Test data can be generated in two phases. At first, symbolic execution method is used to derive path conditions. Secondly, the test data is generated through the uniform random test data generator which is designed from the approximated sub domain. The experimental results showed the proposed path oriented random testing method improves the time of a two order magnitude CPU during path extraction.

IV. PROPOSED APPROACH FOR RANDOM TEST CASE GENERATION

In literature, it is seen that most of methods are developed on the basis of search based software testing and a few focus has been given to random based software testing. So, considering path coverage as the test adequacy criteria, an efficient random based method is proposed, where test data is generated to cover a single path including the most difficult path present in software. The test data generation for path coverage based testing can be stated as follows:

Given:
Let P be the PUT
S is the set of all input values
TP is the target path

Problem:
Find *x*, where x ϵ S, so that *P(x)* traverse the path TP.
If all paths of a SUT are executed during testing, then the path control flow coverage will be 100%.
The proposed random based method can generate test data with the help of CFG of the SUT, multiple hamming distance and one to one correspondence function. The most popular problem, as Triangle Classifier Problem (TCP) [2, 3] is taken for the experiments.

A. Algorithm for Test Data Generation

The proposed random based algorithm generate test data after calculating the similarity distance between the candidates and covers as many paths as possible. In the proposed approach the stopping criterion is set as the maximum number of test data is generated or maximum number of paths have been covered. The algorithm for random test data generation is outlined in Algorithm 2.

Algorithm for Proposed Random based Test Data Generation:

Input: The SUT, Set of Basic Paths S, Input Variables $x = x_1, x_2, x_3 \ldots x_n$
Max=10 (No. of Iteration)
Output: The test data x traversing the path p, p ϵ S
Step 1: Generate random data
Step 2: Calculate the similarity of x_i with every x_j using Eq.(1)
$f(x_i, x_j) = x_i \oplus x_j$
Step 3: Check the distance $f(x_i, x_j)$
Step 4: If $(P(f(x_i, x_j)))$ ϵ S then add the test data
Step 5: else
Step 6: Repeat step 1 to step 4 until number of iteration >= Max or maximum number of path traversed
Step 7: end if
Step 8: end

V. EXPERIMENTAL SETUP

The proposed random based method is implemented on TCP, taken from literature. The details of TCP is described in Table 1. The table lists a short description for TCP, including LOC, number of basic paths. This program has been widely used as a major case study by researchers in the field of RBST. The program for TCP is written in Dev-C++, shown in Figure 4, and the corresponding CFG is shown in Fig.5. The CC can be calculated by using Eq.(1), and for TCP the value of CC is 5.

Table 1 Details of TCP

Program Name	Description	LOC	Number of Basic Path
TCP [2, 3]	Check the type of	28	5

```
1.  int main()
2.  {
3.  int a,b,c;
4.  cout<<"\nEnter the sides of a triangle";
5.  cin>>a>>b>>c;
6.  if((a>0) && (b>0) && (c>0))
7.  if((a+b>c)&&(b+c>a)&&(c+a>b))
8.  if((a==b)&&(b==c))
9.  cout<<"\nThe triangle is equilateral";
10. else
11. if((a==b) || (b==c) || (c==a))
12. cout<<"\nThe triangle is isosceles";
13. else
14. cout<<"\nThe triangle is scalene";
15. else
16. cout<<"Not a triangle";
17. else
18. cout<<"Invalid Input";
19. return 0;
20. }
```

Fig. 4 Source Code for TCP

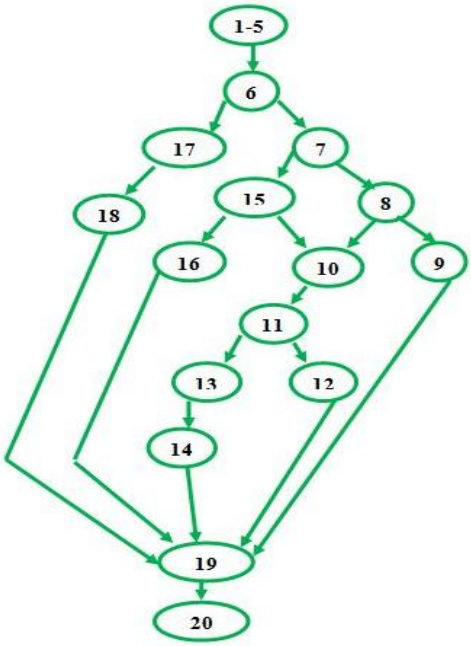

Fig. 5 CFG of TCP

The linearly independent paths are shown below:

- 1-5, 6, 17, 18, 19, 20 (Invalid Input)
- 1-5, 6, 7, 15, 16, 19, 20 (Not a triangle)
- 1-5, 6,7,8,10,11,13,14,19, 20 (Scalene)
- 1-5, 6,7,8,10,11,12,19,20 (Isosceles)
- 1-5, 6, 7,8,9,19,20 (Equilateral)

A. Procedure for the Proposed Random Based Method

-The proposed random based algorithm selects the test data, when the result of hamming distance matches with the specific path. The random generation process is continued until the required criterion has been achieved. The criterion is set as the number of iteration exceeds or maximum path of the SUT is detected. In this work, the test data is generated randomly, and then we have applied hamming distance in a multi way manner. Multiple concept is used due to multiple test data generation. The proposed algorithm calculate the hamming distance with each and every randomly generated data. To validate the effect of the proposed random based algorithm, the results of the random based method is compared with other previous random method [3]. The procedure of random based method is shown in Figure 6.

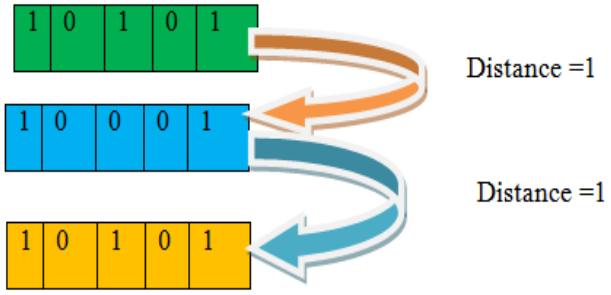

Fig. 6 Multiple Hamming Distance

Figure 6 represents that the test data for path one is generated. In the same manner all other paths for TCP can be covered i.e. test data for paths are generated.

B. Result Discussion

The proposed method was executed in Dev-C++ for several times. Although, TCP has five paths that we have calculated through CC but, in the proposed method we have considered as four depending the multiple distance values. The corresponding paths are listed as follows:

- Equilateral - Path 1
- Isosceles - Path 2
- Scalene - Path 3
- Not a Triangle - Path 4

For the proposed method, the maximum number of iteration is taken as 100. The algorithm has been ran in C++ environment for number of times and the results have been recorded i.e. the number of test data required to cover each path, which is shown in Table 2. The random process has been ran for several times and the best result found in ten iteration has been recorded. Table 2 lists the total number of test data generated to cover different paths including the most critical path. Figure 7 shows the number of test data generated for each path and Figure 8 shows the average test data generated to cover each path for TCP. From the recorded results, it is observed that the path for equilateral could not be covered in easy manner, and it is very difficult to generate the test data to cover that particular path, which is also called as critical path in the most famous TCP. But, the proposed random based method can generate the test data to cover critical path present in the software. However, the number of generated test data to cover the path for equilateral is very less.

Table 2 Test Data Generation for Different Paths

(P1-Equilateral, P2-Isoscales, P3- Scalene, P4-Invalid)

Generation	P1	P2	P3	P4
1	0	10	20	70
2	0	27	33	40
3	0	8	12	80
4	0	20	28	52
5	0	10	20	70
6	0	12	48	40
7	0	25	28	47
8	**1**	17	16	66
9	0	12	10	78
10	0	14	16	70
Average	0.1	15.5	23.1	61.3

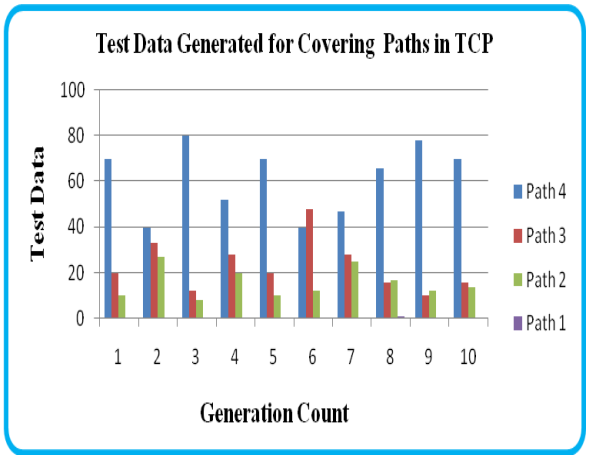

Fig. 7 Test Data Generated for Different Path

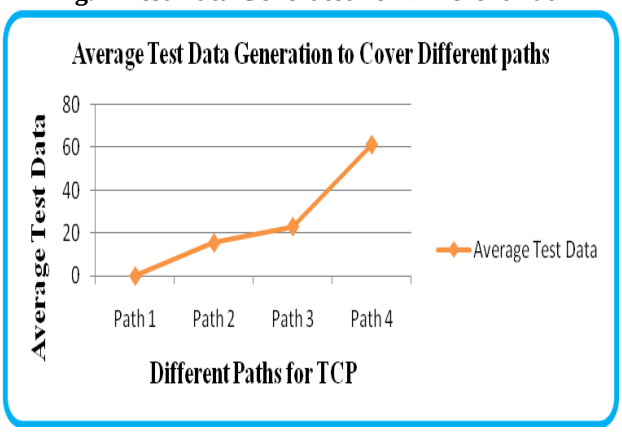

Fig. 8 Average Test Data Generation for Different Path

C. Comparative Study

The result of the proposed random based method is compared with the previous related work [3]. The only SUT as TCP is taken for comparison and the performance result is shown in Table 3 in terms of average test data required to cover the most critical path i.e. the path for equilateral triangle. However, the proposed method can generate a very less number of test data to cover the critical path present in TCP.

Table 3 Average Test Data Generated to Cover the Path for Equilateral Triangle

Program	Authors	Technique	Total path	Average test data for Equilateral path
TCP	*Aldeen et al.[3], Proposed Random Method*	Random	4	0
TCP		Random	4	**0.1**

VI. THREATS TO VALIDITY

The proposed algorithm is implemented on single programs written in C++ language and the target paths are found from CFG, which is independent of the language. The program taken for the experiments is small scale as the flow graphs and the paths have been hand coded. So, the approach requires automatic tools that support the task of flow graphs and path generation. Also, the proposed method is able to generate a single test data at a time.

VII. Conclusion and Future Work

In recent days to avoid software failure, software industries are trying to generate and execute effective test data. In order to generate test data different test data generators are employed. In this paper, a random based algorithm is proposed for automatic test data generation taking the CFG of the SUT. The multiple hamming distance and one to one correspondence function is used to address the test data with corresponding path. The proposed random method can generate a single test data at a time. From the experimental results it is found that a very few test data can be generated to cover critical paths through the proposed random method. Studying other bench mark programs is a matter for future work. The proposed random based method can be enhanced by using EAs to cover multiple path coverage at a time.

REFERENCES:

1. Gong, D. and Yao, X., 2010. Automatic detection of infeasible paths in software testing. *IET software*, *4*(5), pp.361-370.
2. 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.
3. 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.
4. Mansour, N. and Salame, M., 2004. Data generation for path testing. Software Quality Journal, 12(2), pp.121-136.
5. 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.

6. 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.
7. 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.
8. Yao, X., Gong, D. and Wang, W., 2015. Test data generation for multiple paths based on local evolution. Chinese Journal of Electronics, 24(1), pp.46-51.
9. 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.
10. Gotlieb, A. and Petit, M., 2010. A uniform random test data generator for path testing. *Journal of Systems and Software*, *83*(12), pp.2618-2626.
11. 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.
12. Bartle, R.G. and Bartle, R.G., 1964. The elements of real analysis (Vol. 2). New York: Wiley.

AUTHORS PROFILE

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.

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.