A Survey Paper on Test Case Generation and Optimization: Cuckoo Search and Firefly Algorithm

[1]Kavita Choudhary, [2]Yogita Gigras, [3]Shilpa, [4]Payal Rani, [5]Akshath Grover
[1]Assoc. Professor, [2]Assistant Professor, [3]Student
[1]Department of CSE/IT,
[1]ITM University Gurgaon, Haryana, India

***Abstract* - The optimization problems that are encountered are solved by implementing some metaheuristic techniques. In this paper, two of the meta-heuristic techniques are undertaken for checking the optimality of the set of solutions. One of the techniques is cuckoo search i.e. able to generate the set of test cases and optimizes them. Another algorithm is firefly algorithm is used here for checking the optimality of set of test cases that are generated from the cuckoo search. The paper reviews the concepts of test case generation and optimization and produces the results of maximum code coverage in the execution of optimal set of solutions**

***Index Terms* – Meta-heuristic techniques, Test data generation, Optimization, Cuckoo Search, Firefly Algorithm**

I. INTRODUCTION

While designing a software, several steps are initiated which are transformed to Software Development Life Cycle. Every phase of this methodology is equally essential for development of software but the most critical among these is Software Testing. The software testing can be done in two ways- either manually or automated testing is performed to save time. The software testing phase ensures the quality of software by identifying bugs in a system. The phase generates set of input data which checks the proper functionality of the system and tests whether requirements are fulfilled or not. As manual testing is very time consuming, automated testing is preferred which is further classified as random techniques and data oriented techniques. Random technique as name suggests selects random set of inputs for testing rather than concerning test requirements. The limitation seems to be of this technique is not effective and sometimes fails to output desired test data. The second technique, i.e., path oriented technique uses the control flow graph to generate the test cases. The technique is further classified as static and dynamic techniques. Static technique based on symbolic execution, generates test data without actually executing the system being tested whereas dynamic technique executes the system for obtaining results. Static approach undergoes many problems such as indefinite arrays, pointers, loops, etc. which can be overcome by dynamic approach. Approach which presents the testing problem as optimization problem is called as meta heuristic approach.
These approaches are used for optimized and automated generation of test data. Cuckoo search and genetic search are some such approaches. Cuckoo search is based on the behavior of cuckoo bird. It is more optimized and cost effective solution for test generation as compared to other approaches. In this paper, Cuckoo search is discussed to generate test cases. The algorithm discussed requires control flow graph of the system under test. After the test cases are generated, another algorithm is discussed to check the optimality of the generated test cases. The algorithm is firefly algorithm. The algorithm is also a Meta heuristic approach which is influenced by the flashing behavior of fireflies. Firefly algorithm uses the set of idealized rules. The fireflies attract other fireflies' inconsideration of their sex as they are unisex. Fireflies are attracted towards the attractive brightness value. Brightness value is determined by a defined fitness function. The algorithm discussed can be used to check the optimality of any set of test cases generated.

II. LITERATURE SURVEY

Srivastava et. al. [1] presented the use of meta heuristic approach Tabu search and Cuckoo Search for test case generation which combined the strength of Cuckoo Search of converging to the solution with less pssible time and mechanism of Tabu search of backtracking the local optima by Levy flight. Srivastava et. al.[2] discussed the meta heuristic optimization technique Cuckoo Search to analyze complete software. The search is opted where the quasi-random manner is followed. State Transition Diagram is used to generate optimal test case solutions which proved that the algorithm is more efficient than other Meta heuristic approaches such as Particle Swarm optimization and Genetic Algorithm. Srivastava et. al. [3] discussed the test effort estimation and factors affecting the estimation and proposed the model on the basis of Cuckoo Search for the estimation of Test Effort. Various weights are assigned to different involved factors in the model which is used in predicting the project's test effort for similar kind of project. Srivastava et. al.[4] discussed the method to increase the efficiency of software testing with identification of the optimal set of test cases through state machine diagram. The method used Cuckoo search which is a meta heuristic approach for investigation of the paths in state machine diagram. A technique is provided so as to ensure that all the paths and transitions are covered at least once with minimum number of repeating states and transitions. The objective function of algorithm is maximized to focus on the critical parts of program which are prone to error and are tested first. Akram et. al.[5] aimed on the investigated on the importance of software's reliability which is done using Software Reliability Growth Models by estimating required parameters. The failure data is generated and required parameters are estimated with Cuckoo Search. The idea behind using Cuckoo Search is its properties such as robustness, efficiency and effectiveness. Srivastava et. al.[6] discussed the importance and demand of structural testing in software testing for code-based criteria and presented an efficient method to generate all the paths possible in a Control Flow Graph required for Path Testing. The optimal paths are identified through Cuckoo Search algorithm which is equal to the cyclomatic complexity of program under test. Iqbal et. al.[7] considered the problem of test case generation as a problem of multi objective optimization which is having two objectives which has to be fulfilled simultaneously. The two discussed objectives are Path Priority and Oracle Cost. For effective Testing to take place in real time many constraints have to be fulfilled. For the satisfaction of the generation of test cases the multiple objectives reduces the overall efforts of testing. MOFA technique which is Multi Objective Firefly Algorithm is used to solve such kind of problems. Sudhir et. al.[8] reduced the regression testing cost by scheduling the test cases according to the test case prioritization techniques and maximized some objective function. Test Case prioritization takes place according to the test cases importance in some criteria and prioritized test cases are first tested in regression Testing. The applicability of objective function are on the basis of the rapid discovered faults during the process of testing. Evaluation of the techniques for test case prioritization is discussed with Average Percentage of Faults Detected) metric. Srivatsava et.al.[9] presented an approach on Firefly Algorithm which is a meta heuristic technique for the generation of optimal paths. The algorithm is modified by the different definition of objective function and introduction of guidance matrix for path traversal in the graph.

III. Problem Identification

Firstly the problem is selected and then the problem is identified. After the problem identification, the control flow graph is constructed. Then using cuckoo search test case generation is performed and the different sets of test cases are generated. After that these test cases are optimized using firefly algorithm. Firefly algorithm checks the solutions optimality which is generated through the cuckoo search.

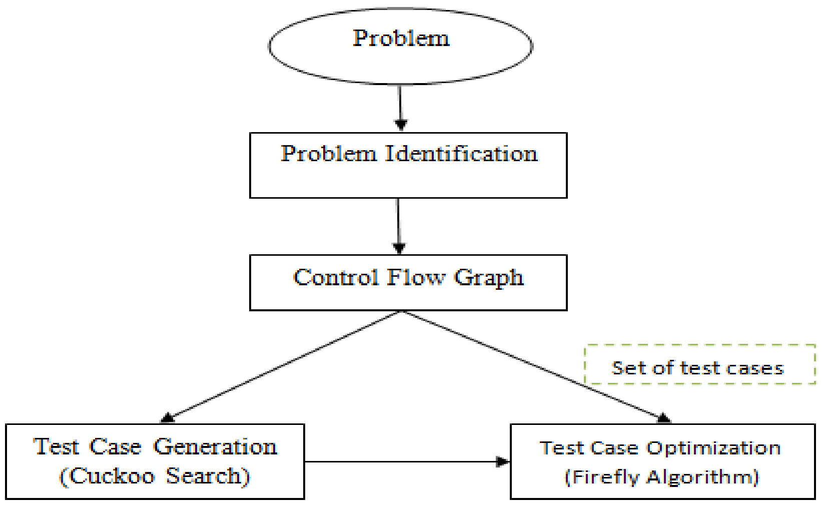

Problem
Problem Identification
Control Flow Graph
Set of test cases
Test Case Generation (Cuckoo Search)
Test Case Optimization (Firefly Algorithm)

Fig 1. Problem Identification

The **Triangle classification problem** is considered for solving the optimization problem for the test data generation and optimization. For test data generation and optimization, two approaches are considered such as Cuckoo Search and Firefly techniques. Through these techniques the objective is to check the maximum code coverage and get the best optimum results. The **Triangle classification problem** is as follows:-

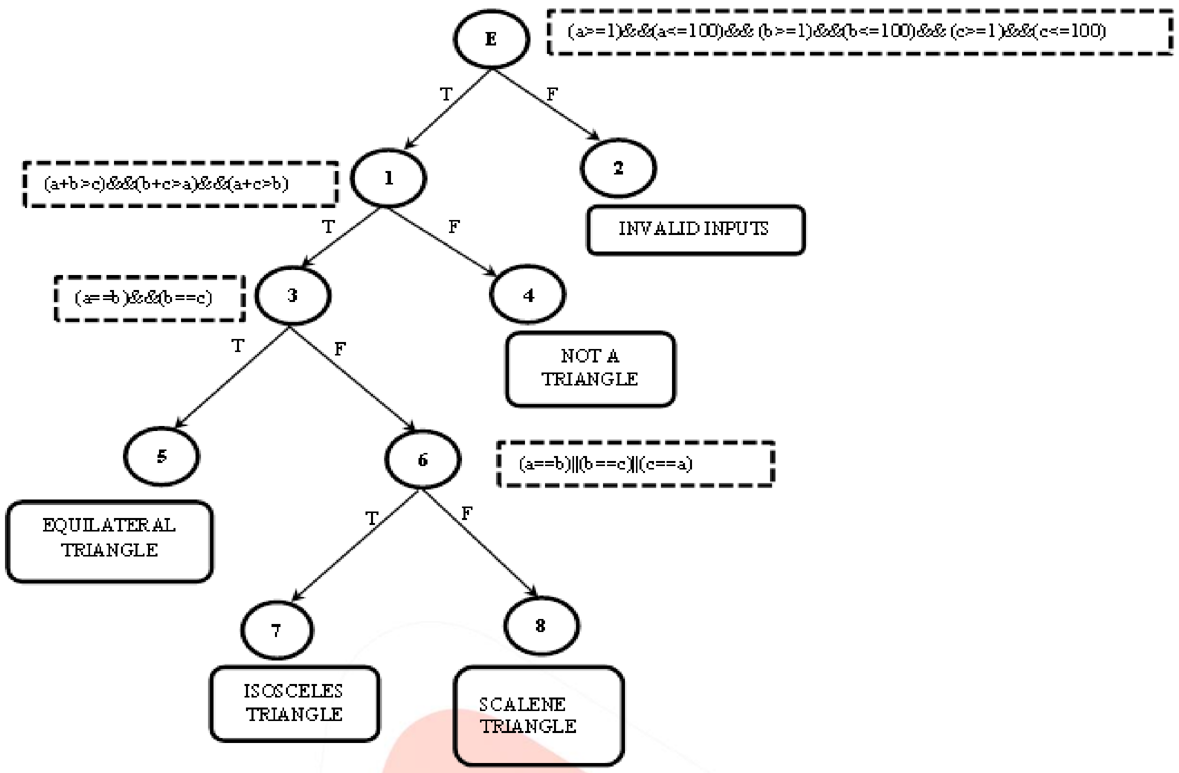


Fig 2. Triangle Classification Problem

(A) Test Case Generation

Cuckoo algorithm works under the exploration phase i.e. by using levy flights it generates the different nests and identifies fitness of these nests, after that exploitation has been done i.e. a fraction of nests is replaced and then evaluation and ranking of the fitness are performed one more time. Here in the proposed work, the cuckoo search algorithm is used for generating the optimal set of test cases for current cuckoo nest. Then the fitness value is evaluated for every test case solution regarding to each target node that is equal to the common branches considered between the solution and the path carried from the root to the target node. The fitness with highest value will be considered as an optimal solution and rest of the solutions goes for the subsequent population of the current cuckoo nest in next iteration. By the proposed algorithm, the evaluations are consumed per iterations rather by evaluating through the original algorithm. The flow graph demonstrates the implementation of the cuckoo search algorithm by using these steps. The steps are as follows:-

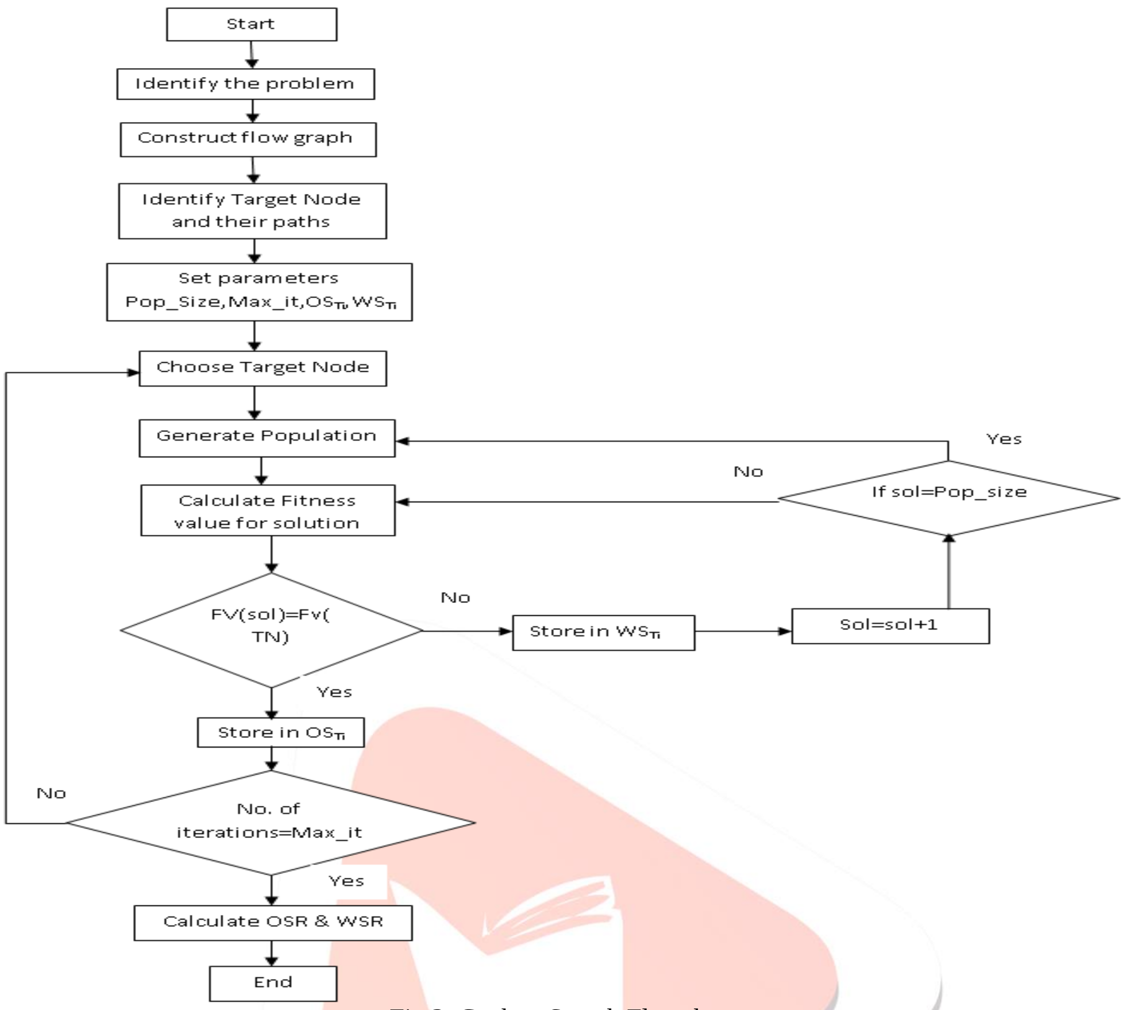


Fig 3. Cuckoo Search Flowchart

Firstly, the problem is identified and on the basis of problem identification, the control flow graph is constructed. Then the target nodes and their path is identified that will be covered in each iteration one by one. After that variables are defined Max_it, Pop_Size, OS_{TI} and WS_{TI}. The Max_it will be equal to the no. of target node and the Pop_Size calculated through ceil of (Max_it/2). OS_{TI} and WS_{TI} are used for storing the good and bad solutions. Then the first target node is selected on the basis of lowest height and the population is generated equal to the number of pop_Size. Then the fitness value is calculated for the solution that is that is equal to the number of common branches between the solution and the path carried from the root to the target node. If the fitness value of the solution is equal to the fitness value of the target node then that solution will be stored in OS_{TI} otherwise goes into the WS_{TI}. The stored worst solutions in WS_{Ti} will be the subsequent population of cuckoo nest for the next iteration and randomly test case generated is added to that set of subsequent population. Then, the process continues to check for the optimal solution for next iteration. Then on the basis of fitness value, the optimum solutions calculated for cuckoo search are stored in the OSR (Optimal Solution Repository) i.e. union of all OS_{TI} and rest of the solutions are stored in WSR (Worst solution repository).

(B) Test Case Optimization

Another algorithm used here is the **Firefly algorithm** that is based on the attractiveness and the brightness of the fireflies. Fireflies use flashing phenomenon for attracting the other fireflies. At random locations, the fireflies are placed. Then the objective function is evaluated for each of the location that is acquired by the fireflies. The highest value of the objective function results in the greatest brightness value. After a firefly has been moved towards all brighter fireflies, its brightness is updated by evaluating the objective function in the new position. If this position is better than this becomes the best otherwise the process continues. In proposed work, the firefly algorithm is used for the evaluation of optimality of solutions that are generated from the cuckoo search. The set of optimal test cases considered as input in this method and checks that solutions whether they covers every node or not in the control flow graph for the given problem. The main objective is to execute the optimal set of solutions generated from the cuckoo search for getting the maximum code coverage. The implementation of the firefly algorithm discussed here through the flow graph. The flow graph includes theses steps:-

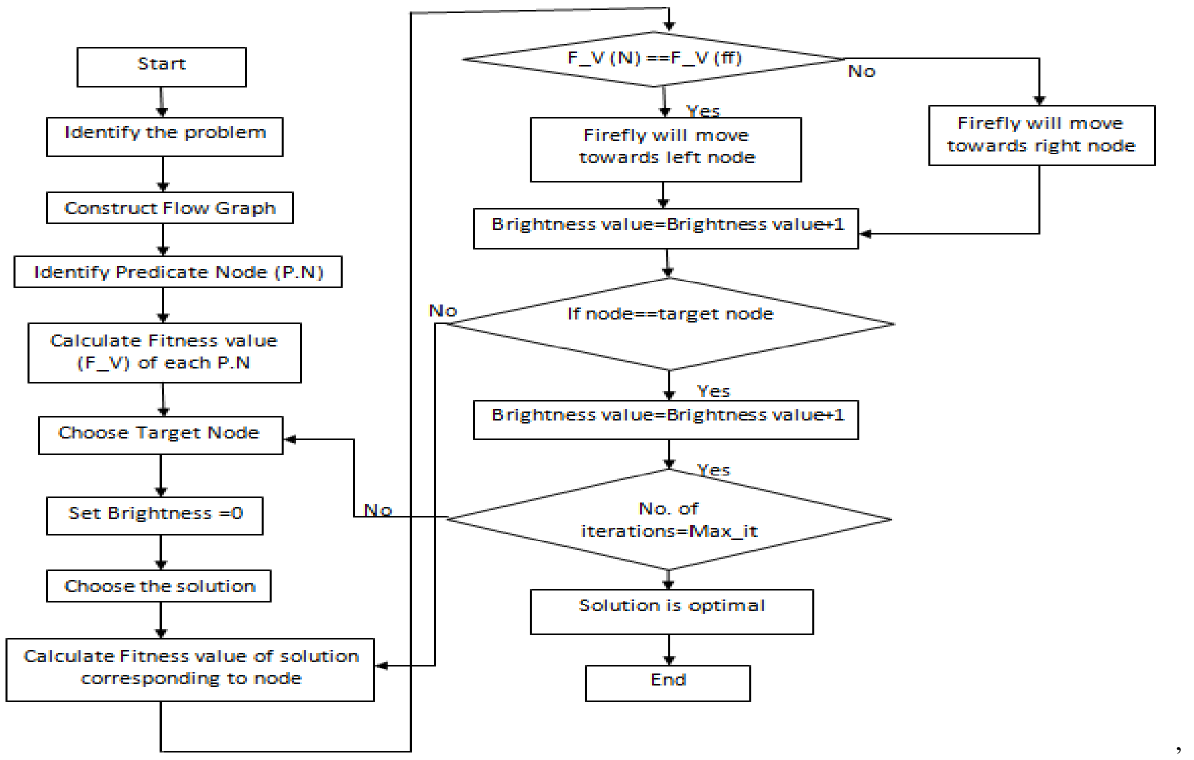

Start
Identify the problem
Construct Flow Graph
Identify Predicate Node (P.N)
Calculate Fitness value (F_V) of each P.N
Choose Target Node
Set Brightness =0
Choose the solution
Calculate Fitness value of solution corresponding to node
F_V (N) ==F_V (ff)
No
Yes
Firefly will move towards left node
Firefly will move towards right node
Brightness value=Brightness value+1
If node==target node
No
Yes
Brightness value=Brightness value+1
Yes
No. of iterations=Max_it
No
Solution is optimal
End

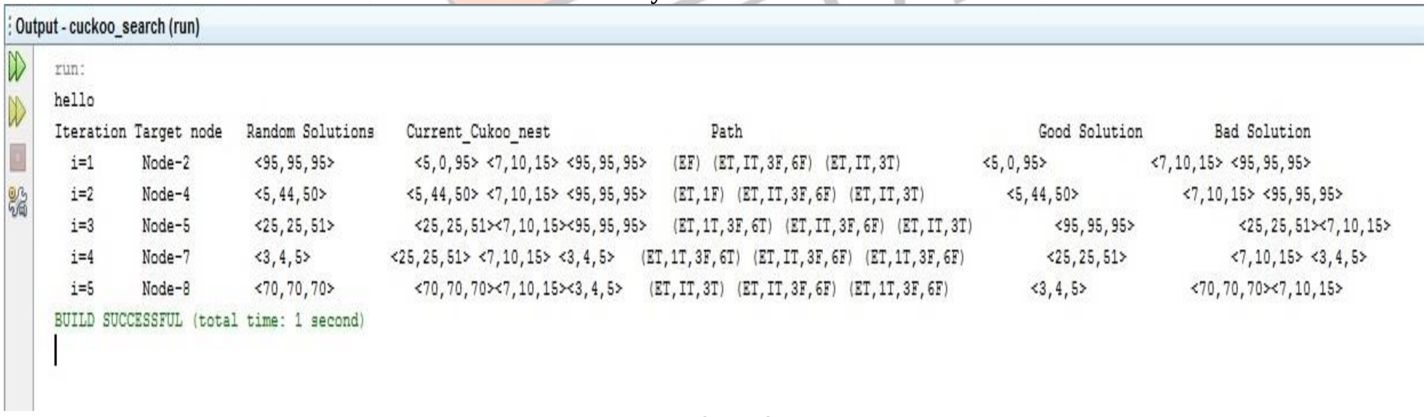
Fig 4. Firefly algorithm Flowchart

The steps involved in the flow graph describe the problem identification and then control flow graph is constructed. After that Predicate nodes is identified and fitness function is calculated for each predicate node that is number of conditions in predicate node. Next, the target node is selected and brightness of each node set to zero. Then the solutions are selected and calculate the fitness value of that solution corresponding to that node. Fitness value of solution is calculated trough the matched condition of the predicate node. Then fitness value of predicate node is compared with Fitness value of firefly. If they are equal, then firefly will tends to move left of the control flow graph and brightness value of every target node corresponding to that predicate node will be incremented by 1. If they are not equal then it will tends to move towards right direction. If traversed node is target node then Brightness value is incremented by 1 and the firefly achieved its destination.

and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

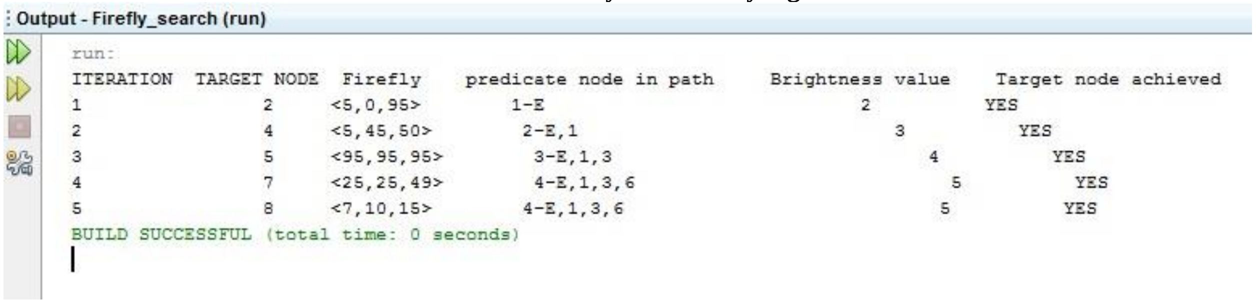
IV. Simulation Results

Table 1: Analysis of Cuckoo search


Output - cuckoo_search (run)

```
run:
hello
Iteration Target node   Random Solutions    Current_Cukoo_nest                Path                                       Good Solution       Bad Solution
  i=1       Node-2       <95,95,95>          <5,0,95> <7,10,15> <95,95,95>    (EF) (ET,IT,3F,6F) (ET,IT,3T)          <5,0,95>           <7,10,15> <95,95,95>
  i=2       Node-4       <5,44,50>          <5,44,50> <7,10,15> <95,95,95>    (ET,1F) (ET,IT,3F,6F) (ET,IT,3T)        <5,44,50>           <7,10,15> <95,95,95>
  i=3       Node-5       <25,25,51>          <25,25,51><7,10,15><95,95,95>    (ET,1T,3F,6T) (ET,IT,3F,6F) (ET,IT,3T)      <95,95,95>           <25,25,51><7,10,15>
  i=4       Node-7       <3,4,5>           <25,25,51> <7,10,15> <3,4,5>   (ET,1T,3F,6T) (ET,IT,3F,6F) (ET,1T,3F,6F)     <25,25,51>           <7,10,15> <3,4,5>
  i=5       Node-8       <70,70,70>          <70,70,70><7,10,15><3,4,5>    (ET,IT,3T) (ET,IT,3F,6F) (ET,1T,3F,6F)       <3,4,5>           <70,70,70><7,10,15>
BUILD SUCCESSFUL (total time: 1 second)
```

Table 2: Analysis of Firefly algorithm


Output - Firefly_search (run)

```
run:
ITERATION  TARGET NODE   Firefly    predicate node in path     Brightness value    Target node achieved
1                2      <5,0,95>        1-E                            2          YES
2                4      <5,45,50>        2-E,1                             3          YES
3                5      <95,95,95>        3-E,1,3                             4          YES
4                7      <25,25,49>        4-E,1,3,6                             5          YES
5                8      <7,10,15>        4-E,1,3,6                            5          YES
BUILD SUCCESSFUL (total time: 0 seconds)
```

V. Conclusion

The Meta heuristic approaches Cuckoo Search and Firefly Algorithm have proved to be successful for Test Case Generation and optimization. Table 1 shows the results obtained for Software Testing. The test cases generated are optimal set of test cases and covers each and every node of control Flow Graph of problem under test. To verify the optimality of test cases generated, Table 2 shows that the test cases generated are optimal set of solutions which covers every node and path in control flow graph for the problem given.

References

[1]. P. R. Srivastava, R. Khandelwal, S. Khandelwal, S. Kumar and S. S. Ranganatha, “Automated Test Data Generation Using Cuckoo Search and Tabu Search (CSTS) Algorithm”- http://www.researchgate.net/profile/Dr_Praveen_Srivastava/publication/235799431_Automated_Test_Data_Generation_Using_Cuckoo_Search_and_Tabu_Search_(CSTS)_Algorithm/links/02bfe513b28346f11e000000.pdf

[2]. P. R. Srivastava , “Software Analysis Using Cuckoo Search”, Springer International Publishing, Advances in Intelligent Systems and Computing, 2015.

[3]. P. R. Srivastava, A. Varshney, P. Nama, X. S. Yang, “Software test effort estimation: a model based on cuckoo search”, International Journal of Bio-Inspired Computation, 2012.

[4]. P. R. Srivastava, A. K. Singh, H. Kumhar, M. Jain, “Optimal Test Sequence Generation in State Based Testing Using Cuckoo Search”, International Journal of Applied Evolutionary Computation, Volume 3 Issue 3, July 2012.

[5]. N. Akram, M. A. AlKareem, “The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models”, International Journal of Computer Science and Information Security(IJCSIS), Vol. 11, No. 6, June 2013.

[6]. P. R. Srivastava, M. Chis, S.Deb, X.S. Yang, “An Efficient Optimization Algorithm for Structural Software Testing”, International journal of artificial intelligence, 2012.

[7]. Iqbal, Zafar, Zyad, “Multi-objective optimization of test sequence generation using multi-objective firefly algorithm (MOFA)”, Robotics and Emerging Allied Technologies in Engineering (iCREATE), 2014.

[8]. Sudhir, “Performance Evaluation of Regression Test Suite Prioritization Techniques”, International Journal of Advanced Engineering and Global Technology Vol-2, Issue-10, October 2014.

[9]. P. Srivatsava, B. Mallikarjun, X.Yang,“ Optimal test sequence generation using firefly algorithm”- Swarm and Evolutionary Computation, Volume 8, February 2013, pp. 44-53.