Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 07, July - 2017

Automated Test Path Generation using Genetic Algorithm

Niveth Vijay K
Computer Science and Engineering
Govt. Engineering College
Thrissur, Kerala, India

Vipin Kumar K S
Computer Science and Engineering
Govt. Engineering College
Thrissur, Kerala, India

***Abstract*— Software testing is the process of finding bugs in the software by executing the program or application. Basis path testing or structured testing is a white box method which analyses the control flow graph to find the set of linear independent path. Path testing is an approach which ensures that each independent path through the program has been executed at least once. Source program is converted to control flow graph and the test path are extracted from this graph. In this paper, we propose an new fitness function for basis test path generation using Genetic algorithm.**

Keywords*— Software testing, Test path generation, Control flow graph (CFG), Genetic algorithm

I. INTRODUCTION

Software testing is an important and complicated phase of software development life cycle. Different types of testing which has been done in the industries before delivering the product to customer. There are mainly two approaches in software testing: (a) Black box testing, which tests the results of the given inputs without knowing the internal structure like program code of the software and (b) White box testing that is aimed to test the internal structure of the software under consideration completely. Both approaches are employed in an automatic test case generation. *Fig. 1* shows the types of software testing.

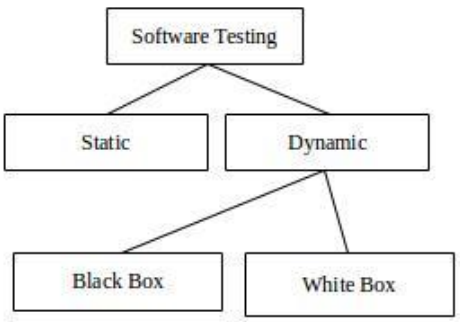

Fig. 1. Types of Software testing.

Path testing is an approach to testing which ensures that every path through a program has been executed at least once[1]. The starting point for path testing is a program flow graph. This is a skeletal model of all paths through the program. A flow graph consists of nodes representing statements and edges showing flow of control. The flow graph is constructed by replacing program control statements by equivalent diagrams. *Fig. 2* shows the types of structural testing.

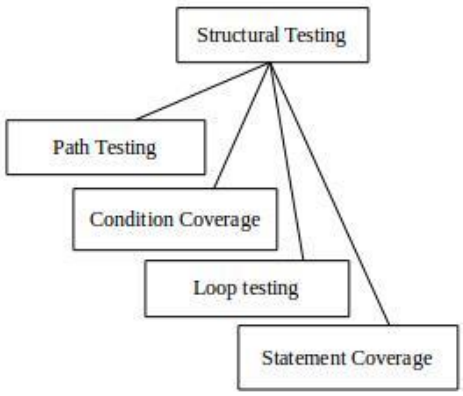

Fig. 2. Types of structural Testing.

Control flow graphs or program graphs that represent the control flow of programs are widely used in the analysis of software. The nodes of a control flow graph are statements of the program and the edges represent the control flow between the statements. Note that the control flow graph is a graph where each node (except entry and exit nodes) corresponds to one statement in the program code.

An independent path is a path of a program, where at least one edge of this path never appears in any other path in the control-flow graph (CFG)[2]. A basis set of paths is a set of paths; every path in this set should satisfy then next three conditions:

- Every path should be an independent path.
- All edges in a CFG should be covered by all paths in the basis set.
- Every path not contained in this basis set of paths can be constructed by performing linear operations among paths in this set.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 07, July - 2017

The basic concepts of genetic algorithm were developed by Holland [4]. The genetic algorithm start by creating an initial population of individuals, each represented by a randomly generated binary string called chromosome. The elements in this binary string of chromosome are called genes. The first step is the evaluation in which the fitness of each individual is determined using a fitness function. The selection step is used to find pairs of individuals that will be contribute to the next generation. The next step is crossover where the chunks of genes between selected chromosomes are swapped. Mutation introduces slight changes into a small proportion of population and is representative of an evolutionary step. The algorithm will iterate until the population has evolved to form a solution to the problem, or until a termination condition is satisfied.

Remaining section of this paper is organized as follows: Section II discusses and compares related works in test path generation, test case production and applying genetic algorithm. In Section III, detailed description of proposed system for test path generation from CFG is shown, and finally Section IV concludes the paper.

II. RELATED WORKS

Bahare Hoseini and Saeed Jalili [1] have proposed a model to generate test paths from UML sequence diagram using genetic algorithm. This model achieves prime path coverage, which is the strongest graph based coverage criteria. Using genetic algorithm has an impressive effect on reduction of test cases required for software testing because it results in minimum test paths. Unlike other attempts in white box testing, this model creates test paths from sequence diagram rather than source code. The advantage of using sequence diagram is that it speeds up test path generation process.

Poole [3] discussed a basis paths generation method based on the depth first search in CFG. It uses a recursive search in the CFG. This method fails in case of selection of the successor of multiple-successors node in a CFG to build a basis path during the construction of the basis set of paths. The basis set of paths obtained will be less than that of the cyclomatic complexity of the code. The major limitation is that the method is unreliable on unstructured code.

A. Ghiduk[2] introduced a new variable length genetic algorithm for automatically generating set of basis test paths which can be used as testing paths in any path testing technique. The length of each chromosome varies from iteration to iteration in the proposed genetic algorithm according to the change in the length of the path. This algorithm introduced a new fitness function to evaluate the generated paths. The disadvantage is that it uses a tool named Sugar for checking the feasibility of the generated paths. The fitness function used to compute needs an extra edge to get the maximum fitness value.

III. PROPOSED SYSTEM

Proposed System consists of following steps:

- Analysis module.
- Control flow graph generation module.
- Test path generation module.

Fig. 3 shows the proposed system architecture.

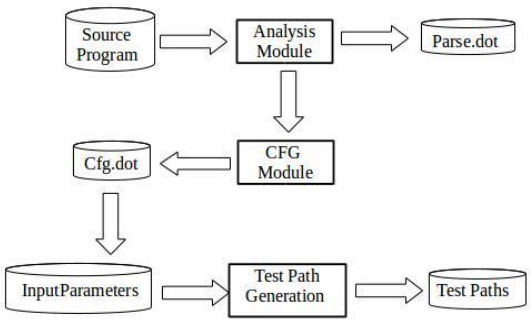

Fig. 3. Proposed System Architecture.

A. *Analysis Module*

The source code is given as the input in the form of text file. The input text file contains the function of the program that is to be tested. The code is converted in to a parse tree based on the grammar specified using lex and yacc programs.

```
void large(int a, int b)
{
  int c = 4,l;
  if(a>b)
  {
    if(a>c)
      l=a;
  }
  else
  {
    if (b>c)
    {
      l=b;
    }
    else
    l=c;
  }
}
```

Fig. 4 shows the sample c code function which determines the largest among three variables.

Fig. 4. Sample function in c code

B. *Control flow graph generation module.*

The source code is get classified based on the program statements and reformatted in to a directed graphical representation called Control Flow Graph (CFG).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 07, July - 2017

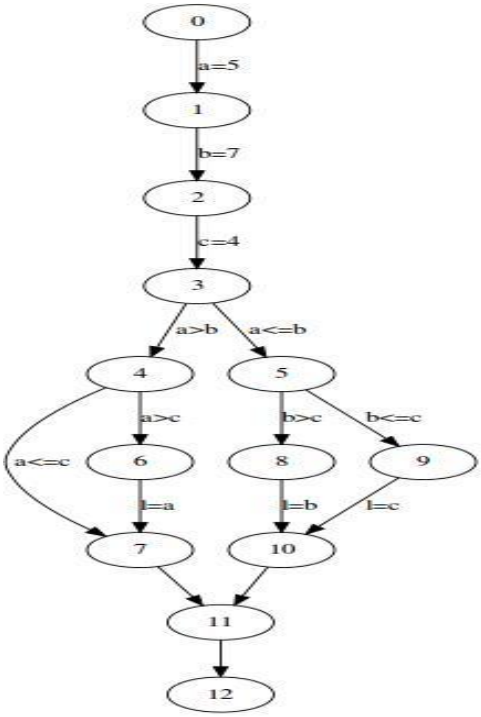

Fig. 5 shows the CFG of the above sample c function code.

Fig. 5. Control flow graph of above sample c code.

C. *Test Path Generation Module*

This section describes the proposed GA for automatic test path generation. The algorithm uses a vector as a chromosome to represent the nodes in the control flow graph of the program under test. The maximum length of the vector equals to the number of the nodes in the CFG of the program under test. Initially, the length of the vector would be of size two containing entry and exit nodes. The desired solution which resides in the search space is derived using genetic algorithm. Each point in the search space represents one possible solution. Genes are the elements in the chromosome which represents the nodes in the CFG. Suppose that, for the above CFG the input domain of new genetic algorithm is D, where n_0 and n_k are the entry and exit nodes, respectively, of node n, and k is the total number of nodes in CFG – 1.

$$D = \{\forall\ n \mid n \in CFG\}$$

Considering the CFG in the figure 3 the search space consist of the set of nodes {0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9, 10, 11, 12}. Each individual in the initial population will consist of entry and exit nodes {0, 12}. The fitness value of each chromosome v_i is calculated as follows:

$$\text{ft(vi)} = \frac{\text{Sum of Adjacent edges} + 1}{N}$$

The fitness value of each chromosome is calculated by using a adjacency matrix of CFG. For a simple graph with vertex set V, the adjacency matrix is a square |V| × |V| matrix A such that its element A_{ij} is one when there is an edge from vertex i to j, and zero when there is no edge. The elements in the chromosome are divided in to pair of nodes and check whether these pairs have an edge between them. Hence the number of adjacent nodes can be calculated.

For example, chromosome v_1 =(0, 1, 2, 3, 4, 12) v_2 =(0, 1, 2, 3, 4, 7, 11, 12) fitness value of v_1 and v_2 is calculated by dividing the chromosome in to adjacent element pairs ((0, 1), (1,2), (2,3), (3, 4), (4, 12)) and ((0, 1), (1,2), (2, 3), (3, 4), (4, 7), (7, !1), (11, 12)) respectively. The corresponding value of each pair in the adjacent matrix is taken and added together. For chromosome v_i the sum of adjacent element pairs will be 4, since the pair (4, 12) does not have any edge between them. The sum of adjacent element pair of v_2 is 7. Total number of genes in chromosome v_1 and v_2 equals 6 and 8. Therefore ft(v_i) = (4+1) / 6 = 0.83 and ft(v_2) = (7+1) / 8 = 1.

For every iteration, the fitness of each chromosome in the current population is computed. After that the algorithm selects parents of the new population from current population using tournament selection method. In tournament selection method r individuals are selected at random and the best out of these chromosomes are chosen.

The proposed algorithm uses uniform crossover for interchanging the genes between selected chromosomes. The genes are exchanged by selecting a random number which is greater than 1 and less than current chromosome length. After that mutation is performed on the same chromosomes, if there exist any sibling for genes at random position.

After performing crossover and mutation operations the length of the current population is incremented by inserting a new gene on the random position. The inserting gene should be a successor node of the gene at random position. This procedure is repeated until a complete path of nodes between n_0 and n_k is obtained. This complete path obtained is checked to find whether it is independent path or not. The path obtained is added to the path list if it is independent from other paths that have been obtained previously. The algorithm will stop the searching process if any of the following two cases reached. The first case is when the generated test paths satisfy the conditions of the basis set of paths. The second case when the number of generations reaches the maximum number of generations.

IV. CONCLUSION

In this paper, we have proposed a new fitness function for genetic algorithm to automate test path generation. Our model achieves basis path coverage, which is the strongest graph based coverage criteria. The length of each chromosome incremented by one gene from iteration to iteration in the proposed genetic algorithm until any of the stop conditions reached.

Using genetic algorithm has an impressive effect on reducing the number of test cases required for software testing because it results in basis set of test paths. Unlike other attempts in white box testing, our method creates test paths from Control flow graph. The advantage of proposed algorithm is that it detects the smaller paths first since the length of chromosome increments on each iteration.

REFERENCES

[1] Bahare Hoseini and Saeed Jalili, "Automatic Test Path Generation from Sequence Diagram Using Genetic Algorithm", 7 th International Symposium on Telecommunications (IST 2016).

[2] A. Ghiduk, “Automatic generation of basis test paths using variable length genetic algorithm”, Information Processing Letters(2014) 114. 304–316.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 07, July - 2017

[3] J. Poole, "A Method to Determine a Basis Set of Paths to Perform Program Testing" (NISTIR 5737), Technical report, U.S. Department of Commerce, National Institute of Standards and Technology, Nov.1995.

[4] J. Holland, "Adaptation in Natural and Artificial Systems, University of Michigan Press", Ann Arbor, MI, ISBN 0472084607 (1975).

[5] Ahmed S. Ghiduk, Mary Jean Harrold, Moheb R. Girgis, "Using genetic algorithms to aid test-data generation for data flow coverage", in: Proceeding of 14th Asia-Pacific Software Engineering Conference (APSEC 2007), December 5–7, 2007, pp. 41–48.

[6] C. Doungsa-ard, K. Dahal, A. Hossain, and T. Suwannasart, "GA-based automatic test data generation for UML state diagrams with parallel paths," In Advanced Design and Manufacture to Gain a Competitive Edge, pp. 147-156, Springer London, 2008.

[7] T. Yano, E. Martins, F.L. de Sousa, "Generating feasible test paths from an executable model using a multi-objective approach", in: 3th IEEE International Conference on Software Testing, Verification, and Validation Workshops, 2010, pp. 236–239.