SOFTWARE RELIABILITY ASSESSMENT WITH GA BASED OPTIMAL TEST DATA GENERATION

CH MADHU BABU 1*, Dr. A VINAYA BABU 2*

1. *Assoc Prof, Dept of CSE, Padmasri Dr.B.V.Raju Institute of Technology.*
2. *Prof, Dept of CSE, JNTU, Hyderabad.*

Abstract: Software reliability assessment is a phase where the software is analyzed to detect the differences and differentiate it with earlier results with present. In order to achieve this one should be provided with adequate test data to test its quality and reliability. In this paper an evolutionary algorithm (GA-genetic algorithm) based test data generation approach is provided to evaluate quality of the software. The test data is generated by summing of all the weights in each path as the fitness function in an optimized manner.

Keyword: Software reliability, testing, genetic algorithm, control flow graph.

1. Introduction

Different phases that involve in development of software are termed as software development life cycle (SDLC) and it becomes very difficult to develop fault free software in a single iteration of the cycle. The software developed during this cycle process may contain bugs or errors or may produce outputs which are deviated from the client requirement. All these errors or bugs may occur due to the misinterpretation of the client requirement to the developers or due to some external factors. However there is a need to detect and rectify these errors to produce a fault free software meeting the client requirements. Software testing is the stage in SDLC where the software is analyzed to trace out the bugs and find the difference between the existing and required results, it means testing is the process which improvises and determines the quality of the software.

Software reliability is the probability of providing fault free software meeting the client requirements over a period of time under specified environment where as quality is defined to the design, function and non functional requirements for the give specifications. Software undergoes rigorous testing to ensure the best quality and reliability. In recent times, every organization is focusing on automated testing rather than relying on manual testing since it involves more time, effort and cost. In order to implement this, they require large amount of data for testing which is not possible manually, hence a strategy has to be developed for generating an optimized data sets used for testing [1] [2].

Alsmadi et.al in [3] generated test cases with good path coverage, representing them in binary format. The main intension of this work is to generate a new test case each time and to sole a fitness function to find error. Michael et.al in [4] extended the work of dynamic test data generation with a minimization function and incorporated it with genetic algorithm (GA) to minimize this fitness function. This approach of defining the fitness function seems to be inadequate as the condition contains a Boolean variable.

Srivastava et.al in [5] presented techniques for path testing using GA; this involves the testing of critical paths. It uses the weight of the paths as the fitness function and selects parents based on the highest fitness value.Donget.al in [6] improved the GA concept by modifying the basic GA by encoding and decoding processes. Encoding involves the solution of the targeted problem with a particular string and the decoding is the reverse of the process, they proved that the proposed modified GA is more superior to the traditional GA algorithm. In this paper the optimized data set for testing structured programs will be generated using evolutionary algorithm like genetic algorithm.

This paper is organized as follows, section explains about the role of GA in software testing and in generation of automated synthetic test data. Section II briefly presents the algorithm of GA and its process, section III presents the proposed approach and its objectives, section Iv presents the experimental approach followed in this approach and the obtained results with pictorial representation of CFG's[7].

2. Background

Genetic Algorithm

An evolutionary algorithm which was invented during 70's by John Holland, which is used for optimization solutions. This is more robust than the traditional artificial intelligence techniques which can be even used in noisy environmental conditions. GA can be implemented for multi dimensional problems and for large search spaces[8]. This algorithm includes few terms like population size, search space, operators, chromosomes, genes and fitness function [9] [10]. General process that involved in GA is depicted in below figure

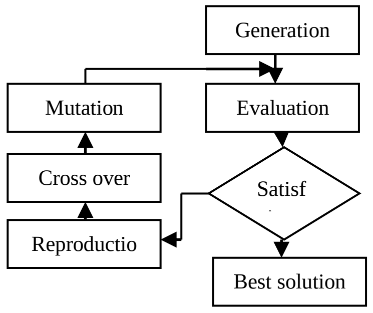

Figure 1: Genetic Algorithm flow diagram

Algorithm:

- *Generate the population randomly*
- *Calculate the fitness value of each individual population using fitness function and evaluate them*
- *Check the stopping condition criteria*
- *Select the best solutions from the population*

- *Apply crossover to the solutions at random points*
- *Apply mutation to the new solution according to mutation probability*
- *Go to the current generation of solutions*

The population size is known as the number of candidate solutions in one generation, the larger the population size and the larger the solution space, the more is the search. The main intension of GA is to reduce problem and optimize it which is defined by fitness function**.**

3. Proposed Approach

In the proposed approach path based coverage testing is adopted [2] as it retains the best coverage and used cyclomatic complexity for paths and tried to generate the test data that could cover all the paths. Control flow graph (CFG) is used as intermediate representation of the program lines.

Algorithm:

- *Write a program for experimental analysis*
- *Generate the control flow graph for the program*
- *Find all the basic path sequences in the program*
- *GA is used to generate the test data for the program*
- *Stop the iterative process when the stopping condition criterion is satisfied.*

The first step in the proposed approach is to write a program for test data generation, and generate the CFG for it. Each node in CFG represents a statement in the program and the edges represent the control flow between each node in the program. Assign weights to each edge of the graph, the edges after the decision node are given weights according to 80, 20 rule. The more important edge is given 80 % while the least is given 20% of the weights. If two edges converge at a node then the outgoing edge contains the sum of the weights at both the edges.

4. Implementation and Results

To implement the proposed approach a simple greatest common divisor (GCD) program is considered and the sample program is stated below

```
#include<stdio.h>
#include<conio.h>
int main ( )
{
int a; int b; int r;
If (a>b)
{
r=a;
a=b;
b=r;
}
r=a%b;
while(r!=0)
{
a=b;
b=r;
r=a%b;
}
return b;
}
```

The CGF for the above programme is depicted below

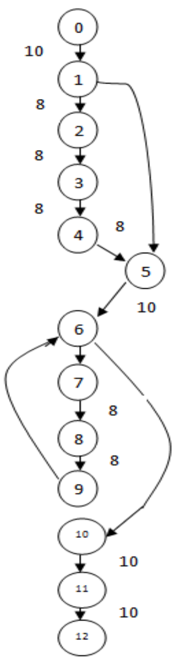

Figure 2: CFG for above program

Independent paths that were obtained are

- Path 1: 0-1-5-6-10-11-12
- Path 2: 0-1-5-6—8-9-6-10-11-12
- Path : 0-1-2-3-4-5-6-10-11-12
- Path 4: 0-1-2-3-4-5-6-7-8-9-10-11-12

Fitness function used for GA:

In this approach the paths were considered for fitness function i.e; $\boldsymbol{f(x) = \sum_{i=0}^{n} W_i}$ where W_i are the weights assigned for each path.

The following parameters were considered for GA

- *The chromosome length is represented by b*p where b is the number of bits and p are is the number of parameters*
- *Random numbers are generated between [0 1]*
- *Crossover Probability P_c=0.8*
- *Mutation Probability P_m=0.2*
- *Initial population size =4*

Initial population is given in Table I

Table 1: Initial population

Test Data	**Fitness Value**	**Random number**
(12,4)	44	0.256
(4,5)	106	0.125
(81,9)	44	0.545
(120,20)	44	0.l654

The stopping criteria in our approach are either it continues till the number of generation or if the fitness function values are 3 same values or more then it stops. As it can be observed that at the 6th generation a stop criterion is attained then the test data set will be as shown in table 2

Table 2: Population at the 6th generation

Test Data	**Fitness Value**	**Random number**
(80,21)	140	0.124
(80,21)	140	0.212
(80,21)	140	0.365
(80,21)	140	0.154

5. Conclusion

In this paper GA algorithm is used to generate the test for a simple program for finding the best solution path coverage based testing. The mutation score is considered as the fitness function here which is very adequate and reliable. Results endure that the proposed approach could attain the objective of finding the best path at an ease of generations. In future this work can be further extended by incorporating either a hybrid schemes along with GA and improvise the search algorithms.

References

[1] J. Hassl A. Mette. A guide to Advanced Software Testing. Artech House Publications, 2008.

[2] N. Chauhan. Software Testing Principles and Practices, Oxford University, Press, India, 2010.

[3] Izzat Alsmadi. Using genetic algorithms for test case generation and selection optimization. In 23rd Canadian Conference on Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on, pages 1-4, IEEE, 2010.

[4] Christoph C. Michael, Gary McGraw, and Michael A. Schatz. Generating Software Test Data by Evolution, volume 27. Dec 2001.

[5] Praveen Ranjan Srivastava and Tai-hoon Kim. Application of genetic algorithm in software testing. International Journal of software Engineering and its Applications, 3(4):87{96, 2009.

[6] Yuehua Dong and Jidong Peng. Automatic generation of software test cases based on improved genetic algorithm. In Multimedia Technology (ICMT), 2011 International Conference on, pages 227-230. IEEE, 2011

[7] Wang Xibo and Su Na. Automatic test data generation for path testing using genetic algorithms. In Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on, volume 1, pages 596-599. IEEE, 2011.

[8] Abdelaziz M Khamis, Moheb R Girgis, and Ahmed S Ghiduk. Automatic software test data generation for spanning sets coverage using genetic algorithms. Computing and Informatics, 26(4):383-401, 2012.

[9] Mark Harman and Phil McMinn. A theoretical and empirical study of search based testing: Local, global, and hybrid search. Software Engineering, IEEE Transactions on, 36(2):226{247, 2010.

[10] Roshni Rajkumari and BG Geetha. Automated test data generation and optimization scheme using genetic algorithm. In Proceedings of International Conference on Software and Computer Applications (ICSCA 2011),2011