Test Data Generation Using Intelligent Geno-Neuro Technique

Obe O.O.

Abstract— Software testing is an integral part in software development which requires the use of test data to test software. During testing, the most difficult problem is the generation of test data. A test data generator is a tool which assists programmers or testers in generation of test data for a program which will thus reduce the time and cost of software testing. Some techniques have been proposed to develop the test data generator. Using geno-neuro technique, more rigorous test data will be generated. This approach will make use of the control flow graph to guide the test data generation. Control flow graph will be used to analyse the necessary paths in the program. The neural network will be trained with the genetic algorithm for the generation of test data. This approach (geno-neuro technique) has been shown to enhance the generation of more rigorous test data.

Index Terms-genetic algorithms,geno-neuro,machine learning,neural networks,software engineering,software teting,test data,chromosones

1 INTRODUCTION

Software bugs will almost always exist in any software module with moderate size: not because programmers are careless or irresponsible, but because the complexity of software is generally intractable and humans have only limited ability to manage complexity. Discovering the design defect in software is equally difficult for the same reason of complexity. This is so because software and any digital systems are not continuous, testing boundary values are not sufficient to guarantee correctness. Regardless of the limitations, testing is an integral part in software development. It is broadly deployed in every phase in the software development cycle.

Studies show that software testing already consumes up to 50% of software development costs. One of the most difficult and expensive technical problems of software testing is the actual generation of test data values. Automation is a good way to minimize the cost and time in software development. If software testing process could be automated, the cost of developing software would be reduced significantly. Of the problems involved in testing software; one of them is the problem of generating test data. A test data generator is a tool which assists programmer in generation of test data for a program.

2 BACKGROUND AND RELATED WORK

2.1 Review Stage

Test data generation is an important part of software testing which includes the process of identifying a set of data for testing the adequacy of software application. Test data generation involves identifying program input data which satisfy selected testing criterion. It may be the actual data that has been taken from previous operations or artificial data created for this purpose. Given a testing requirement, such as an input that will cause execution of a particular statement, test-data generation techniques attempt to find a program input that will satisfy the testing requirement.

Genetic algorithm is an optimization heuristic that mimics natural processes, such as selection and mutation in natural evolution, to evolve solutions to problems whose solution spaces are impractical for traditional search techniques, such branch-and-bound or optimization technique and linear programming. This algorithm encodes a potential solution to a specific problem on a simple chromosome-like data structure and applies recombination operators to these structures so as to preserve critical information. Genetic algorithm will be used to search for test cases that satisfy desired testing requirements.

In machine learning and cognitive science, Artificial neural network (ANN) is a family of models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Neural network requires training which can either use supervised or unsupervised learning method.

Geno-neuro technique is a hybrid technique which is based on Artificial Neural network and genetic algorithm. In geno-neuro technique, the neural network will be trained with the genetic algorithm which is an optimization technique used in generating data.

Jon E, 1999 released a publication on a survey on automatic test data generation in proceedings of the second conference on computer science and engineering in Linkoping. His objective is to describe the basic concepts and notions of test data generation as

well as how test data generator works. He made use of control flow-graph or flowgraph which is a directed graph G = (N, E, s, e) consisting of a set of nodes N and a set of edges E = {(n, m)|n, m ∈ N} connecting the nodes to show the graphical representation of program. In each flowgraph there are two special nodes: one entry- and one exit-node, s and e respectively. The nodes represent the statements of the program and the edges represent flow of control between the statements. He identified some possible challenges for further research on test data generation. Some of these challenges are Modules, Assertions, Path selection, Pointers and shapes, Object-oriented programs.

[17] Also published a journal on test-data generation using genetic algorithm. His objective is to present a technique that uses genetic algorithm for automatic test-data generation. He made use of control-dependence graphs and genetic algorithm to review the automatic test-data generation. Control-dependence for a program is defined in terms of the program's control-flow graph and the post dominance relation that exists among the nodes in the control-flow graph. In the control-flow graph, nodes represent statements, and edges represent the flow of control between statements. He made use of genetic algorithm to search for test data. He developed an algorithm for automatic generation of test data for a given program. This algorithm is called GenerateData This algorithm (GenerateData) is implemented and has been implemented on only six programs. Although the results are promising, more experimentation must be done before any conclusive statements can be made.

[13] Made a research on automated test data generation for programs with procedure. His objective is to present an approach for automated test data generation for programs with procedures. He made use of chaining approach in the research. The basic idea of chaining approach is to identify a sequence of nodes to be executed prior to execution of selected node. The chaining approach uses the data dependency concepts to identify such a sequence. The approach starts by executing a program for an arbitrary program input x. During program execution for each executed branch (p,q), the search process decides whether the execution should continue through this branch or whether an alternative branch should be taken. If an undesirable execution flow at the current branch (p,q) is observed, then a real-valued function is associated with this branch. Function minimization search algorithms are used to find automatically new input that will change the flow execution at this branch. Event sequences are used to guide the execution of the program during the search process. He performed an experiment to compare the random test data generation, path-oriented test data generation and the chaining approach method of test data generation. The results of the experiment indicate that the chaining approach may increase the chances of generating test data.

Chayanika et al, 2013 wrote a journal on a survey of software testing techniques using genetic algorithm. The objective is to present a survey of genetic algorithm approach for addressing the various issues encountered during software testing. Genetic algorithm approach is used to survey test data generation. Another test generation approach proposed by P.R Srivastava is based on path coverage testing. The test data is generated for Resource Request algorithm using Ant Colony Optimization algorithm (ACO) and GA. The ACO algorithm is inspired from behaviour of real ants where ants find closest possible route to a food source or destination. The ants generate chemical substance called pheromones which helps ants to follow the path. The pheromone content increases as more ants follow the trail. The possible paths of CFG are generated having maximum number of nodes. Using ACO, optimized path ensuring safety sequence in resource request algorithm is generated covering all edges of CFG. With GA, suitable test data set is generated which covers the need for each process. The backbone of genetic process is the fitness function which counts number of times a particular data enters and continues the resource algorithm. Higher the value of count, higher is chances avoiding a deadlock. The test data with higher values of count is taken and generic crossover and mutation is applied to yield better results. Simultaneously, poor test data is removed each time. The experimental results shows that success rate of ACO are much better than GA. In weighted CFG approach, experiments were done on small examples and need to be done on larger commercial examples. Moreover, method can be further improved.

[1] Also wrote a report on dynamic test case generation using neural networks. The objective is to explore the use of neural networks to find test cases that would be able to execute specific portion of the program. He made use of **c**ontrol dependence which is defined in terms of the program's control flow graph and the post-dominance relation that exists among the nodes in the control-flow graph. In a control-flow graph, nodes represent statements, and edges represent the flow of control between statements - an edge (X, Y) in a control-flow graph means that program control can flow from X to

Y . To facilitate analysis, a control-flow graph is augmented with unique entry and exit nodes. Given a code, static analysis can easily provide the control dependence graph. He mde use artificial neural network approach in dynamic test data generation. It is used to compute a function using example inputs and outputs. Neural networks have been used for a variety of applications, including pattern recognition, classification, and image understanding. Neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The system can also be trained using a whole set (batch) of inputs, reducing the total error. He developed a triangle classification program which was instrumented and executed with various random values and tried to model the branch functions using radial basis neural networks in matlab. The input values were real values constrained between 0 and 1, since this network is designed for values between 0 and 1.

2.2 PROPOSED TECHNIQUE

The proposed algorithm is geno-neuro technique has been devised to generate more rigorous test data. Since genetic algorithm is an optimization technique, it will be sufficient to train the neural network to generate rigorous test data.

2.3 PROPOSED GENO-NEURO TECHNIQUE

The proposed test data generator technique (geno-neuro technique) will consist of two parts which are the evolution part and the learning part:

i. The evolving part will use genetic algorithm to search for better test data
ii. The learning part whereby the artificial neural network will be trained using genetic algorithm and the test data generated by the genetic algorithm.

The three sample programs that the test data generator system is developed for are:

i. CGPA calculator program
ii. Triangle classification program
iii. Quadratic equation roots classification program.

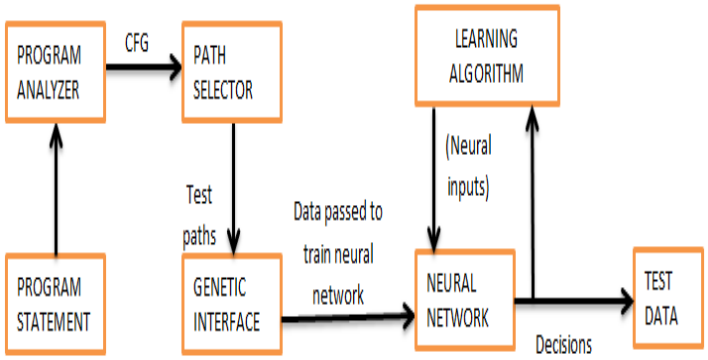

Figure 1: Architecture of geno-neuro test data generator

The geno-neuro test data generator system architecture follows three major steps. These are:

i. Construction of program control flow graph
ii. Selection of path
iii. Generating test data.

To generate test data for any program, the program must be analysed. In this research work, the programs are analysed manually using the control flow graph. With the control flow graph, all the feasible paths in the programs will be identified.
The control flow graph for the quadratic equation root classification program is given below:

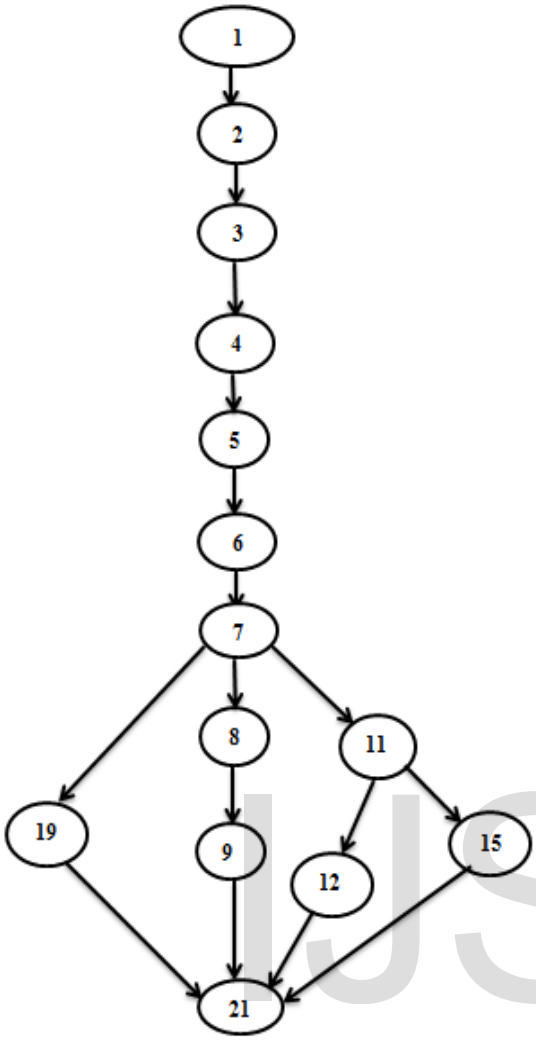

Figure 2: Control flow graph of the quadratic program

From the analysis made through the control flow graph, the flow of the program and the feasible paths of the programs will be identified easily. From the control flow graph above, the start node is 1 and the exit node is 21. This shows that any input to the program would have taken a feasible path if and only if it executes from the start node to the exit node.

For the quadratic program, it can be deduced from its control flow graph that its feasible paths are:

i. 1-2-3-4-5-6-7-8-9-21 (EQUAL ROOTS)
ii. 1-2-3-4-5-6-7-11-15-21 (COMPLEX ROOTS)
iii. 1-2-3-4-5-6-7-11-12-21 (REAL ROOTS)
iv. 1-2-3-4-5-6-7-19-21 (INVALID COEFFICIENTS)

Genetic algorithm is used twice in the development of the geno-neuro test data generator system. Genetic algorithm was first used to tune the data generated through the random numbers. These data are then used to train the neural network. The neural network makes use of a three layer network which includes an input layer, a hidden layer and an output layer.

At the input layer, the neural input is received by the network. This inputs come from the random data that are tuned by the genetic algorithm. The hidden layer serves as the link between the input layer and the output layer. Some processes are done on the input received at the hidden layer. At the output layer, test data will be generated.

The neural network is trained using the genetic algorithm. The input data to the neural network will serve as the initial population to the genetic algorithm. The genetic algorithm will encode the solution using a chromosome-like structure. For this quadratic program, each chromosome represents a set of data which is a potential solution consisting of three genes. These genes represent the coefficients of the quadratic equation which will be used in identifying the type of root a quadratic equation has which may be EQUAL ROOTS, REAL ROOTS, COMPLEX ROOTS or INVALID COEFFICIENTS.

Chromosome 1	**S1**	**S2**	**S3**
Chromosome 2	**S4**	**S5**	**S6**
Chromosome 3	**S7**	**S8**	**S9**
Chromosome 4	**S10**	**S11**	**S12**

Figure 3: Chromosomes representing potential solutions

In figure 3, Chromosomes 1, 2, 3 and 4 represent potential solutions. S1, S2, S3 ….. S12 are referred to as the genes which are numerically encoded. S1, S2 and S3 are the genes for the chromosome 1. These genes represent a test data. Each gene will have a numeric value which represents a coefficient of the quadratic equation.

In training the neural network with genetic algorithm, genetic operators is be used to generate test data. Fitness value is very important as it is useful when the selecting data for the next generation. Before selecting data or individuals for the next generation, each of the data will be passed to the program and the path taken will be noted. The fitness value is determined based on the path that the test data is to be generated for.

The fitness value will be determined based on the predicate of the paths. The individuals that follow the same predicate or almost the same predicate with the path to generate test data for are taken to be the best fit such that if the path to generate test data for is 1-2-3-4-5-6-7-8-9-21 then if the data follows path:

Chromosome 1	**S1**	**S2**	**S3**
Chromosome 2	**S4**	**S5**	**S6**
Chromosome 3	**S7**	**S8**	**S9**
	S10	**S11**	**S12**

i. 1-2-3-4-5-6-7-8-9-21, then fitness value = 4
ii. 1-2-3-4-5-6-7-11-12-21, then fitness value = 3
iii. 1-2-3-4-5-6-7-11-15-21, then fitness value = 2
iv. 1-2-3-4-5-6-7-19-21, then fitness value = 1
v. Infeasible path, then fitness value = 0

If the fitness of each individual is f_i, the probability p_i that a test case will be selected for the next generation is:

$$p_i = f_i / \sum_{j=1}^{n} f_j$$

where n is the number of test cases in the population.

Also,

$$\sum_{j=1}^{n} p_i = 1$$

In selecting individuals for the next generation, elitist selection is used which select the fittest members of each generation will be selected for the next generation. Genes from parent chromosomes will be combined to form new offspring chromosome (new population). Combining the genes of the parent chromosomes gives a better chance of generating better test data. In this research work, a two-point crossover is used for the crossover operation. Mutating the gene gives a better chance of searching for global optimal solutions and not getting stuck in local optimal solutions. The gene will be mutated by replacing the value of the gene with another value. The genetic algorithm terminates when the maximum number of iterations is reached. During mutation, the neural network will adjust the weight (test data) to satisfy the specific path to generate the test data for.

3 EXPERIMENTAL ANALYSIS

In this paper, comparison is made between genetic algorithm and geno-neuro technique in test data generation. The geno-neuro test data generator is developed and analysis is made between using only genetic algorithm technique and using geno-neuro (genetic algorithm and neural network) technique in developing the test data generator. The test data generator programs are run for thirty (30) iterations for each of the techniques and the summation of the fitness values is recorded for each of the iterations.

The graph below shows the relationship between each iteration and fitness values and the summation of the fitness values. The summation of the fitness values is on the y-axis and the iteration is on the x-axis of the graph. The graph shows that the geno-neuro technique generates data that are fitter than the data generated by the genetic algorithm.

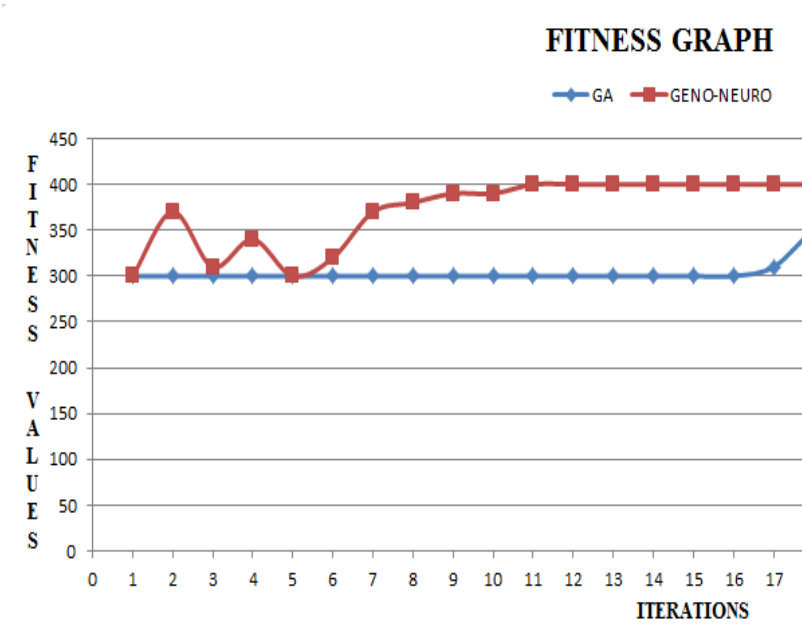

Figure 3: Fitness graph for genetic and geno-neuro techniques in test data generation

From the graph above, geno-neuro technique converges quickly (after iteration 11) compared to the genetic algorithm (that converges after iteration 26) during test data generation. Thus, geno-neuro technique enhances the generation of more rigorous test data during test data generation and can save half of the time of generation by genetic algorithm.

4 CONCLUSION

This project work has successfully developed a test data generator for three sample programs using geno-neuro technique. From the analysis of the results above, it can be concluded that using geno-neuro technique, the test data generator converges at a higher rate compared to test data generator that is developed using genetic algorithm. This is so because the data passed to the neural network for training are not randomly generated and the neural network is able to adjust data based on the path to generate test data for. Thus, geno-neuro technique enhances the generation of more rigorous test data.

REFERENCES

[1] *Abhas K., Dynamic Test Case Generation using Neural Networks, Indian Institute of Technology, Kanpur pp 1-3, 2013.*

[2] *Bogdan K., Automated Test Data Generation for Programs with Procedures, Departmrnt of computer science, Illinois Institute of Technology, pg 209, 1996.*

[3] *Bogdan K., Automated Test Data Generation for Programs with Procedures, Departmrnt of computer science, Illinois Institute of Technology, pg 209, 1996.*

[4] *Bogdan K., Automated Test Data Generation for Programs with Procedures, Departmrnt of computer science, Illinois Institute of Technology, pg 209, 1996.*

[5] *Christos S., Dimitrious S., Neural Networks, https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, viewed 12/03/2016.*

[6] *Christos S., Dimitrious S., Neural Networks, https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, viewed 12/03/2016.*

[7] *Christos S., Dimitrious S., Neural Networks, https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, viewed 12/03/2016.*

[8] *Christos S., Dimitrious S., Neural Networks, https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, viewed 12/03/2016.*

[9] *John E., Wachovia B., Charlotte N., Software testing fundamentals- concepts, roles, and terminology, SUGI 30 proceedings, pg 2, pg 8, 2005.*

[10] *Jiantao P., Software Testing, https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/, viewed 12/03/2016.*

[11] *Kamal Z., Nor A., Mohamed F., Siti N., A Tool for Automated Test Data Generation (and Execution) Based on Combinatorial Approach, International Journal of Software Engineering and Its Applications, Vol. 1, No 1, pp 19-33, 2007.*

[12] *Korel B., Automated Software Test Data Generation, IEEE Transactions on software engineer. No. 8, Vol. 16, 870-879, 1990.*

[13] *Korel B., Automated Test Data generation for programs with procedures, Department of computer science Illinois Institute of Technology Chicago, IL 6061, 209-214, 1996.*

[14] *Melanie M., Genetic Algorithms: An overview, Santa Fe Institute, pp 1-8, 1995.*

[15] *Michael N., Using neural network to recognize handwritten digits, http://neuralnetworksanddeeplearning.com, 12/03/2016, 2016.*

[16] *Mohammad R., Automatic Software Test Case Generation: An Analytical Classification Framework, International Journal of Software Engineering and Its Applications, Vol. 6, No. 4, pp, 2012.*

[17] *Roy P., Mary J., Robert R., Test-data generation using Genetic algorithm, Journal of software testing, verification and reliability, pp 1-4, 1999.*

[18] *Shagodoyin D., Obe O., Arnab R., Dlamani S., Tool support for systematic test data generation using genetic algorithm, pp 399-402, pp 2006.*

[19] *Swapan K., Hitesh T., Automated Test Data Generation Using Fuzzy Logic-Genetic Algorithm Hybridization System for Class Testing Of Object Oriented Programming, International Journal of Soft Computing and Engineering, Vol. 3, Iss. 5, pp 40-49, 2013.*