International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 8 (2013), pp. 825-832
© International Research Publications House
http://www. irphouse.com /ijict.htm

Testing Target Path by Automatic Generation of Test Data Using Genetic Algorithm

Shveta Parnami

The IIS University, Jaipur, Rajasthan.

Abstract

Software testing is the most important component of software development process. Path testing is a popular structural testing method that uses the source code of a program to find every possible executable path. The adequate and accurate test data are required to test each path. Test data generation is a key problem in software testing and its automation will improve the efficiency and effectiveness of software testing. Different test data generation methods like random test data generator, symbolic evaluator, function minimization method and metaheuristic search methods had been proposed in the literature. The paper explores the Genetic Algorithm approach to generate adequate and accurate test data for a target path. Genetic Algorithm is an adaptive heuristic search algorithm that premise on the evolutionary ideas of natural selection and genetic. The approach first converts the program into its corresponding Control Flow Graph, and then automatically generates the test data for the target path using different sets of GA operators. The paper also compares and demonstrates the effect on generated test data by changing the type of GA operator like crossover, by varying population size.

Keywords: Path Testing, Target Path, Genetic Algorithm, Automatic Test Data Generation.

1. Introduction

Path testing is a structural testing method that finds every possible executable path from the source code of a program. The method ensures that every path through a program has been executed at least once. One of the major difficulties in the automation of software testing is automatic generation of adequate set of test data that satisfies the complete path coverage of a given program. Test-data generation is a process of identifying a set of program input data, which satisfies a given testing criterion. Since it is impossible to cover all paths in software, the path testing method selects a subset of paths to execute and find test data to cover it. Many attempts were made to automate the test data generation process for path testing and suffered many limitations as the test data generation process is extensive and difficult process.

Many different test data generation methods like random test data generator, symbolic evaluator, function minimization method and metaheuristic search methods have been proposed in the literature. The paper focuses on the Genetic Algorithm approach which is an adaptive heuristic search algorithm that premise on the evolutionary ideas of natural selection and genetic, to automate test data generation for a target path. The related literature suggested that genetic encoding techniques and genetic operators have very important influence on the automatic generation of test data. This paper presents the utility and implementation of GA to automatically generate the test data to ensure the complete coverage of the target path.

This paper is organized as follows: Section II gives a backdrop of genetic algorithms. Section III gives a review of the related test-data generation techniques, especially techniques based on genetic algorithms. Section IV presents the methodology for test-data generation based on genetic algorithms. Section V describes the experimental set up for the given approach. Section VI presents the results and the discussions. Section VII presents the conclusions.

2. Genetic Algorithm

A Genetic Algorithm is a global search heuristics technique used in computing to find true or approximate solutions to optimization and search problems. The Genetic Algorithm concept is easy to understand; modular based and supports multi-objective optimization. Genetic algorithms are evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators which are selection, recombination and mutation. Selection replicates the most successful solutions found in a population at a rate proportional to their relative quality, Recombination decomposes two distinct solutions and then randomly mixes their parts to form novel solutions and Mutation randomly perturbs a candidate solution.

3. Simple Genetic Algorithm

Produce an initial population of individuals evaluate the fitness of all individuals while {termination condition not met} do

- Select fitter individuals for reproduction
- Recombine between individuals
- Mutate individuals
- Evaluate the fitness of the modified individuals

- Generate a new population
- End while

4. Application of Genetic Algorithm to Automate Test Data Generation

The various research works carried out in the field of test data generation, different researchers have used different coverage criteria while automating the test data generation. Genetic Algorithm approach is proposed by Michael et. al. (2001) for automated test data generation using branch coverage. Sthamer (1996) focused on generating test data by using several, structural test coverage using genetic algorithms. They identified all the test data generation technique focus on multiple test criteria instead of single test criteria. Duran et.al.(1984) generated the software test data using random testing technique using various coverage criteria such as segment coverage, branch coverage, and path coverage. Srivastava and Kim(2009) had worked on control flow graph (CFG) to ensure that all the independent paths for path coverage are moved along at least edge that has not been traversed before the path is defined. They have demonstrated that Genetic Algorithm technique finds the most critical paths for improving software testing efficiency. Genetic algorithm automatically generates test cases to test selected path by taking it as a target and executes sequences of operators iteratively for test cases to evolve. The evolved test case then leads the program execution to achieve the target path as shown by Nirpal and Kale (2010). The fitness function proposed by authors achieves path coverage that incorporates path traversal techniques, neighborhood influence, weighting, and normalization. Euclidean distance is used by Korel (1990) to quantify the distance between two paths of the control flow graph. Clake (1976) generated the software test data by using path coverage testing criteria. They selected target path, execute the path symbolically, identify constraints, and then generated the test case values such that the identified constraints are satisfied. Mansour and Salame (2004) and Lin and Yeh (2001) discussed about automatic test data generation using path testing criteria and genetic algorithms. Mansour and Salame (2004) used hamming distance as a fitness function operator in Genetic Algorithm. Ahmed and Hermadi (2007) attempted to generate test data for multiple paths using genetic algorithm. Ghiduk et. al. (2007) proposed an approach to generate test data using du (definition use) paths coverage. They focused upon generating the dominance tree from the control flow graph of the program. Gursaran (2012) employed the path prefix strategy as a branch ordering strategy and memory and elitism in addition to the usual genetic algorithm operators for test data generation for branch testing. Shen et. al. (2007) proposed the hybrid scheme of genetic algorithm and tabu search that came to known as GATS algorithm using function coverage as testing criteria.

Referring to the above mentioned related works it is observed that various operators of GA have been applied to automate the test data generation for different testing coverage criteria. The researchers are aiming to minimize the effort of generating the test data by using various GA operators on different coverage criteria.

5. Methodology

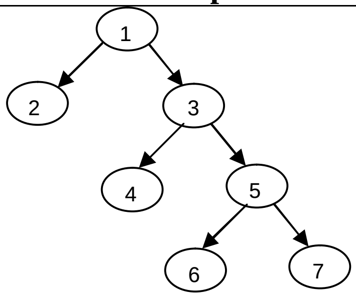
The paper proposes a Genetic Algorithm based method which automatically generates the test data for the target path of a program. The method consist the following steps:

1. Structuring Control Flow Graph: The program is transformed into the CFG. Each node in CFG represents a unique character so that every path can be represented by a string.
2. Identification of Target Path: A CFG has 2^n paths where n is number of branches. Traversing each path may be very time consuming therefore it is required to select a meaningful paths as target paths.
3. Selection of Appropriate GA Operators: The GA operators are Selection, Fitness Function, Crossover, Mutation and Reinsertion. Each of these operators plays important role in generation of test data.
4. Automatic Generation of Test Data: GA based tester automatically and regularly generates new test data to monitor whether the target paths are covered or not and accordingly update GA parameters to lead new population to traverse uncovered paths. Original test data are chosen from their domain at random and GA generates new test data in order to achieve the target path.
5. Satisfaction of the tester algorithm: A suitable test data that executes along the target paths are generated or no suitable test data may be found because of exceeding max generation.

It is difficult to generate the test data for every path therefore a target path is selected which is rarest of all. The proposed approach generates the test data automatically for the given target path to ensure the complete coverage. This target path is given to GA as an input and the test data are generated using different set of operators of GA. The approach also compares and shows how the change in GA operators affects the test data generation keeping the rest environment alike.

6. Experimental Set Up

The experiment on triangle classifier procedure is conducted in order to investigate the effectiveness of using GA for path testing. The paper follows two experiments. In first experiment the test data is generated by varying the population size from 20 to 100. In the second experiment the crossover operator is changed form one-point crossover method to two-point crossover method for population size ranging from 20 to 100. The fitness function is designed by computing approximation level between target path and execution path through comparing with the same path nodes the data traversing. The experiment compares the target path with the actual traversed path. The deviation degree of the paths that traversed by test data x and the target paths are measured. The higher fitness value to the individual are set if there are more nodes or edges same in sequence before the deviation. The triangle classifier procedure and its corresponding CFG are as follows. The nodes of the CFG are represented by the statements in the triangle classifier procedure.

Program Under Test	**Control Flow Graph**
1 If ((A+B<C)||(B+C<A)||(A+C<B)) then 2 "Not A Triangle" 3 else if((A= =B)&&(B= =C) 4 " Equilateral Triangle " 5 else if((A= =B)||(B= =C)||(A= = C)) 6 " Isosceles Triangle " else 7 " Scalene Triangle "	

Parameter Settings: SURVIVE_RATE: 0.5, CROSSOVER_RATE: 0.9, MUTATION_RATE: 0.01, POP_SIZE: 20-100, CHROMO_LENGTH: 15, GENE_LENGTH: 5, MAX_ALLOWABLE_GENERATIONS: 50

7. Results

Experiment 1: The test data for the target path <134> (Equilateral triangle) is been generated. The input to the program is the target path <134> which is difficult to obtain. The program is expected to generate the test data for the target path <134>. The program is executed for 15 times with different population size. The chromosome length is 15 which is three times gene length which is 5, as the number of input variables are three and are defined by gene. The binary encoding technique has been used and the initial population is randomly generated. According to Sthamer (1996) the population, where there are as many chromosomes as there are bits in the chromosome itself, is the best population. Therefore the experiment starts with the population size greater than chromosome length. The result in Figure 1 depict that the population size greater than thrice of chromosome length produces the better result for the target path <134>. The x axis represents the population size and y axis represents the test data set generated for the target path <134>by executing the program for 15 times for population size varying from 20 to 100. The result varies with the size of the population. The experiment showed that when the population size is thrice or four times the chromosome length then more set of test data for target path <134> are generated.

Experiment 2: An experiment is performed to test the effect of crossover type on the generation of test data for a target path in triangle classification problem. The test has been performed for one point crossover and two point crossover types. The variation in the results are examined and compared. The comparison is depicted in Figure 2 where two-point crossover has more probability of generating the test data for target path<134> (equilateral triangle) when compared with one point crossover for 15 runs with different population size.

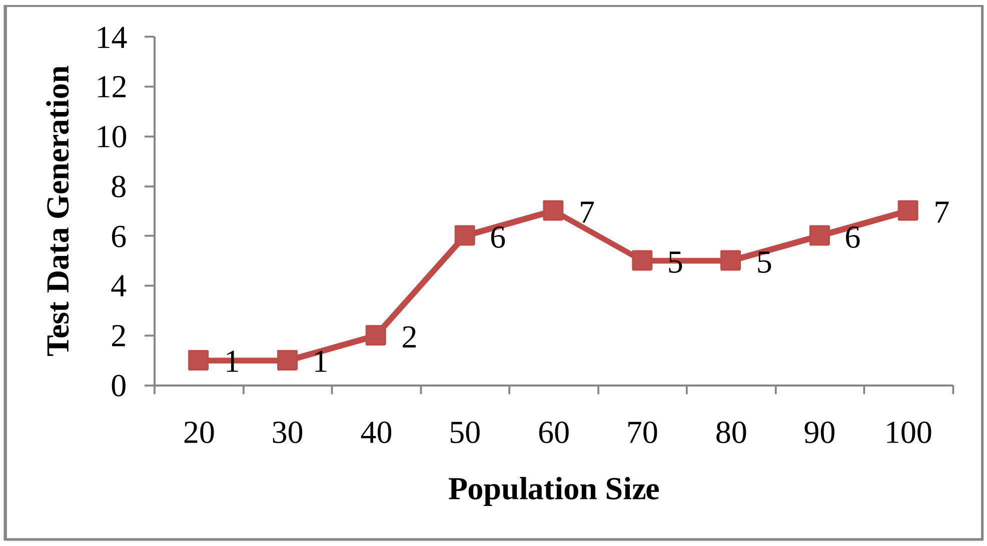


Figure 1

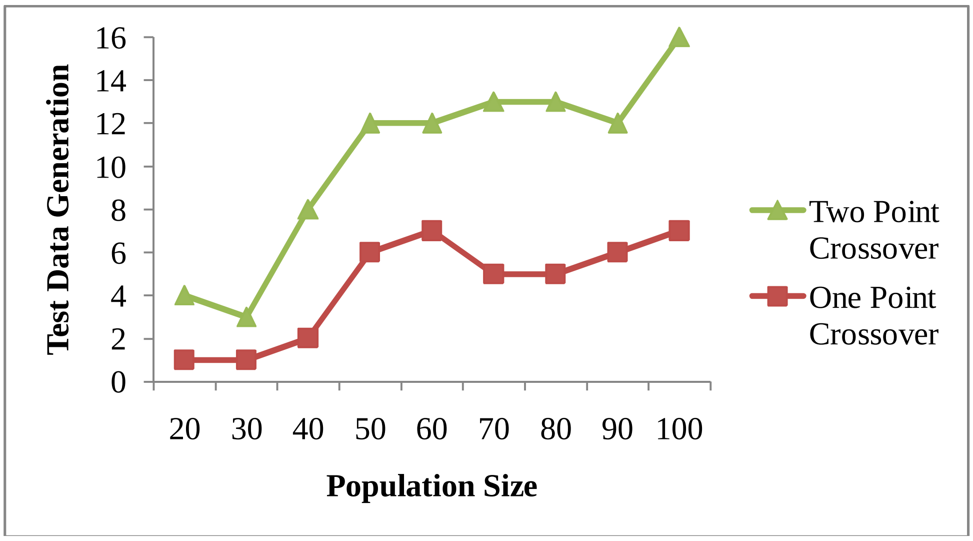


Figure 2

8. Conclusion

In software testing, the generation of testing data is one of the key steps which have a great effect on the automation of software testing. The paper discusses the algorithm that depends on the principles of genetic algorithms to generate test data that provide good coverage in terms of the paths it tests or visits within the application. The greatest merit of genetic algorithm in program testing is its simplicity. Genetic algorithms are often used for optimization problems in which the evolution of a population is a search for a satisfactory solution given a set of constraints. The proposed experimental sets have used different combinations of the GA operator to find the test data for a target path in the CFG of a program under test. The experiment on triangle classification problem suggests that the change in GA operators affects the test data generation keeping the rest environment alike. The two experiments are performed in which the first experiment shows that the population size effects the generation of test data and the second experiment suggests that varying the GA operator i.e. crossover from one point to two point will also vary the result. Given the same experimental environment the two point crossover generates more set of require test data as compared to one point crossover.

References

[1] "Automatic test data generation for path testing using a new stochastic algorithm", Abreu, B. T., Martins, E., and de Sousa, F. L, 2005. In Proc. of the 19th Brazilian Symp. on Software Engineering, vol19, pp247–262.

[2] "GA based multiple paths test data generator", Ahmed,M.A., Hermadi,I.,Computers and Operations Research (2007).

[3] "A system to generate test data and symbolically execute programs",Clarke,L.,1976, IEEE Trans. on S.E.,Vol.2, pp.215-. 222.

[4] "Advanced Topics in Computer Science: Testing Path Testing", H.Schliglof , M. Roggenbach, Luke Gregory 321512.

[5] "An evaluation of random testing", Duran J.W., Ntafos S.C.,IEEE Trans. on S.E., 1984, 10(4): 438–443.

[6] "Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage", Ghiduk,A.S., Harrold,M.J., GirgisM.R.,14th Asia-Pacific Software Engineering Conference, 1530-1362/07 © 2007 , IEEE, pp.41-48.

[7] "Program Test Data Generation For Branch Coverage With Genetic Algorithm: Comparative Evaluation Of A Maximization And Minimization Approach", Gursaran, Ankur Pachauriand,2012, IJSEA, Vol.3, No.1.

[8] "Automated software test generation", Korel, B, 1990, IEEE Trans. on Software Engineering 16(8): 870–879.

[9] "Automatic test data generation for path testing using GAs",Lin,J.C., Yeh,P.L., Information Sc.,vol.131,2001,pp.47-64.

[10] "Data Generation for Path Testing", Mansour, N, Salame,M, Software Quality Journal,12, 121–136, 2004.

[11] "Structured Testing:A Testing Methodology Using the Cyclomatic Complexity Metric", McCabe, Thomas, J., Watson, Arthur, H., 1996. NIST Special Publication 500-235.

[12] "Generating software test data by evolution",Michael C.,McGraw G.,Schatz M., IEEE Transactions on S.E.,2001,1085-1110.

[13] "Tutorial: Program Testing Techniques", Miller, E.F., COMPSAC '77 IEEE Computer Society, 1977.

[14] "Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm",Nirpal,Kale,IJCSET,2010,Vol.1.

[15] "Automatic Generation of Test Case based on GATS Algorithm" Shen, X.,Wang,Q., Wang,P., Bo Zhou,2007, AA04Z148.
[16] "Application of Genetic Algorithm in Software Testing", Srivastava P.R., Kim T, IJSEA,Vol.3,No. 4, 2009, pp.87-96.
[17] "The Automatic Generation of Software Test Data Using Genetic Algorithms",Sthamer, H.,PhD thesis, Great Britain, (1996).
[18] "The Limitation Of Genetic Algorithms In Software Testing", Sultan H. Aljahdali, Ahmed. S. Ghiduk, Mohammed El-Telbany, 2010, Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA).