International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Test Data Generation for Basis Path Testing Using Genetic Algorithm and Clonal Selection Algorithm

Poonam Saini[1], Sanjay Tyagi[2]

[1]M. Tech scholar, Department of Computer Science and Applications, Kurukshetra University Kurukshetra, India

[2]Assistant Professor, Department of Computer Science and Applications, Kurukshetra University Kurukshetra, India

Abstract: ***Test data is needed for testing the software which can be generated automatically and manually. Manual generation of test data involves a lot of efforts. Therefore automated test data generation methods are used. To find the suitable test data for a program, optimization should be applied on test data. In this paper, two optimization techniques, Genetic Algorithm (GA) and clonal selection algorithm have been used. This paper presents how these optimization tools generate the optimized test data that satisfy the basis path testing criteria. The paper also presents the results conducted on a set of programs that evaluate effectiveness of the techniques compared to the random-testing technique.***

Keywords: Basis Path Testing, Clonal Selection Algorithm, Genetic Algorithm, Software Testing, Test Data Generation.

1. Introduction

Software Testing is done to detect presence of faults, which cause software failure. Therefore, testing process becomes complex and time consuming task as the software size increases, which accounts for approximately 50% of the cost of a software system development [1].

Generally, the goal of Software Testing is to design a set of minimal number of test cases such that it reveals as many faults as possible. Absolutely, automation of Software Testing reduces the development cost of a program and also the execution time. 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. 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. Basis path testing is the testing technique of selecting the paths providing a basis set of execution paths through the program. Automation of the testing process includes a number of steps, such as Test Data Generation, test case execution and analysis of test results. Test-data generation is a process of identifying a set of program input data, which satisfies given testing criterion. The traditional methods used for generation of test data are random method, symbols implementation of laws, procedures & instrumentation and iterative relaxation method etc [2]. However, their practicality and the efficiency in large-scale projects are relatively poor. So, a technique should be used that can be efficient for automatic generation of test data. Search-based optimization techniques (e.g., hill climbing, simulated annealing, and Genetic Algorithms, clonal selection algorithm) have been applied to a wide variety of software engineering activities including cost estimation, next release problem, and test-data generation [3].

The Test Data Generation problem can be transformed to the test data search optimization problem. Several search based test-data generation techniques have been developed [4]-[9]. Some of these techniques had focused on finding test data to satisfy a wide range of control-flow testing criteria [7], [8], [10] and the other techniques had concentrated on generating test-data for covering a number of data-flow testing criteria [11], [12]. For example, Pargas et al. [4] presented a Genetic Algorithm directed by the control-dependence graph of the program under test to search for test data to satisfy all-nodes and all-branches criteria. Michael et al. [5] discussed the use of GAs for automatic test-data generation to satisfy condition-decision test-coverage criterion. Wegener et al. [6] presented a test environment for automatic generation of test data for statement and branch testing. These techniques evolve a set of test data using genetic operations (selection and recombination) to find the required test data. All GA-based test-data generation techniques except Bottaci [7] and Girgis [8] provided techniques only for control-flow coverage criteria such as statements, branches, path, and conditions, whereas Bottaci [7] provided a technique for mutation testing and Girgis [8] presented a technique for data-flow test-coverage criteria. Srivastava and Kim[1] had worked on Control Flow Graph (CFG) for path coverage such that every path should be traversed. They have demonstrated that Genetic Algorithm technique finds the most critical paths for improving Software Testing efficiency. Sthamer [9] focused on generating test data by using structural test coverage using Genetic Algorithms. Euclidean distance is used by Korel [10] to quantify the distance between two paths of the control flow graph. Lin and Yeh [11] discussed about automatic Test Data Generation using path testing criteria and Genetic Algorithms. Ahmed and Hermadi [12] attempted to generate test data for multiple paths using Genetic Algorithm. Ghiduk et. al. [13] proposed an approach to generate test data using du (definition use) paths coverage. Xu et. al. [16] presented a clonal selection algorithm to generate the path-oriented test data.

Many GA based test data generators adopted statement or branch coverage as their objectives, however, by nature, path coverage criteria covers statement and branch coverage. In other words, if testing can be designed to force execution of all paths, every statement in the program will have been guaranteed to be executed at least one time and every condition will have been executed in its true and false sides

[14], which make it the highest coverage. The main objective of using search based techniques lies in the ability to handle input data which may be complex in nature. Thus, the problem of Test Data Generation is treated entirely as an optimization problem. The benefit of using GA and CSA is that through the search and optimization process, test data sets are improved in a manner that they are at or close to the input domain.

This paper presents the utility and implementation of both algorithms to automatically generate the test data to ensure the complete coverage of the target path. This target path is given to the algorithm as an input and the test data are generated using different set of operators. In this paper, a comparison is also shown between the performances of these algorithms.

2. Basis Path Testing

It is one of the oldest structural testing techniques [15]. This technique is based on the control structure of the program. On the basis of that control structure, a flow graph is developed and it is assumed that all possible paths can be covered at least once during testing. Here, modified version of path coverage criteria is used, which is the most general criteria, when compared to other logic coverage criteria. The problem with the path coverage is that program that contains loops can have an infinite number of possible paths and it's impractical to test all those paths.
Practically, it is possible to apply path testing for a specific subset of paths in the control flow graph. This mechanism aims to compute the logical complexity of a procedural design and defines a set of execution paths. Test data are generated in such a way that they will execute every statement at least once. Genetic Algorithms could be applied to path testing if the target paths are clearly defined and an appropriate fitness function related to this goal is built.
Steps for basis path testing are:
1) Draw the CFG for a program.
2) Calculate the cyclomatic complexity which provides the number of independent paths.
3) Determine the basis set of independent paths.
4) Based on the independent paths, choose appropriate test data for execution.

3. Genetic Algorithm

Genetic Algorithms (GA) are direct, parallel and stochastic method for global search and optimization, which is used to find approximate or exact solutions based on the principles of natural evolution, described by Charles Darwin. GA is an evolutionary algorithm so can be applied to many optimization problems for generating the test plan for Software Testing and in many other areas. GA is suitable when an optimized function cannot be solved with more accurate deterministic methods [16]. The Genetic Algorithm uses the three main principles of the natural evolution: reproduction, selection and diversity of the species, maintained by the differences of each generation with the previous. Genetic Algorithms work with a set of individuals which are the strings of chromosomes, representing possible solutions of the task. Each chromosome has a fitness value associated with and this fitness determines the probability of survival of an individual chromosome in the next generation. The population is iteratively recombined and mutated to generate successive new populations [9].
Outline of a basic Genetic Algorithm:

1. [START] Initially the population is randomly generated of n chromosomes.
2. [FITNESS] Evaluation of fitness value for each chromosome.
3. [NEW POPULATION] Genetic operators like Selection, Crossover, and Mutation are applied for creating new generation.
4. [REPLACE] Old population is replaced by newly generated population.
5. [TEST] If the specified condition is satisfied, stop and return the solution.

In this paper, algorithm presented is implemented in MATLAB R2009a to generate the optimized set of test data. Here the problems are taken and coded as m-file and by using the Genetic Algorithm toolbox to generate test data, which helps to obtain optimized data. GA requires the number of variables, which are needed to be set and then size of the population also needed to be set. Population size has great effect on the GA speed to obtain an optimum solution. Mutation and Crossover operators help the GA to generate the solution which is in the range of local optima [4].

4. Clonal Selection Algorithm

The Clonal Selection Algorithm (CSA) [17] is also an optimization algorithm based on biological immune system, in which the antigen corresponds to the problem to be solved and the antibody corresponds to a solution to the problem.

This algorithm can be used in basis path testing by encoding random test data (input values) as antibodies. The test data are evaluated based on affinity function (referred to as fitness function in GA). This affinity function is a description of how best the individual test data perform in code coverage. The antibody clone each population to produce their own clones based on affinity value. In CSA, the process of crossover is overcome by performing hyper-mutation operation [18]. This step helps in achieving diversified test data. The process of cloning and hyper-mutation continue till the stopping criterion is encountered. Each antibody clones a clonal population according to the affinity, where better members clone more antibodies. Diversification of the antibodies is achieved through mutation and selection process. The process of cloning, hyper-mutation and selection continues until the termination condition is encountered.

5. Proposed Work

The aim of the work is to improve the fitness function as well as to generate the optimal test data. For improving the fitness function of branch predicates, Korel's Distance Function [10] is used. In Korel's distance function, branch predicates are used in the form of relational expression. Using this function, branch predicates are evaluated, as basis path testing includes both statement testing and branch testing. In this paper, both techniques Genetic Algorithm and clonal selection algorithm are used for the generation of the

optimized test data. The system for generating automated test data for feasible basis paths using GA and CSA has been coded in MATLAB. The basic outline for both algorithms is:

a) Test Data Generation using GA:
   Input: Randomly generated numbers (initial population act as test data) based on the target path to be covered.
   Output: Test data for the target path.
   i. Gen = 0
   ii. While Gen < 100
   iii. Do
   iv. Evaluate the fitness value of each chromosome based on the objective function.
   v. Use roulette wheel as selection operator, to select the individuals to enter into the mating pool.
   vi. Perform two-point cross over on the individuals in the mating pool, to generate the new population.
   vii. Perform bitwise Mutation on chromosomes of the new population
   viii. Gen = Gen +1
   ix. go to Step iii.
   x. End
   xi. Select the chromosome having the best fitness value as the desired result (test data for target path).

b) Test Data Generation using CSA [17]:
   i. Gen = 0
   ii. Initialize random population A_0
   iii. Evaluate Affinity Function A_n
   iv. if Gen > 100 then
   v. output= test data
   vi. Exit
   vii. Else
   viii. Clone A_n to A_n'
   ix. Hyper-mutate A_n' to A_n''
   x. Evaluate and Select A_n''
   xi. Destroy and renew to construct a new population A_n
   xii. Gen++
   xiii. end if
   xiv. goto Step iii.

6. Experiments and Results

Experiment has been done in MATLAB using optimization tool by taking a number of programs as case study for automated Test Data Generation. The experimental settings and results obtained in generating optimal test data are shown below:

Table 1: Experimental setup

Genetic Algorithm	Clonal selection algorithm
Coding: Binary string, Chromosome length:20 bits	Coding: Binary string, Antigen length: 15 bits
Population size (N): 20	Population size (N): 20
Selection method: roulette wheel, Two-point cross over(pc): 0.8, Mutation probability(pm): 0.1	Selection method: roulette wheel, Hyper Mutation (pm): 0.15 Nr = N/2; Ns = Nr/10; Ns-Worst antibodies; N= no. of generation Nr-Renewed antibodies.
Stopping criteria: # of generations = 100	Stopping criteria: # of generations = 100

The implementation of code coverage and Test Data Generation for a target path using the GA and CSA were carried out and the results are shown in table 2 by applying the experimental settings shown in table 1.

Table 2: Comparative results of number of test data generated by randomly, GA and CSA

Programs	Random	GA	CSA
Square	49.51	56.67	60.01
Square root	35.63	42.50	56.67
Quadratic equation	43.24	47.22	54.28
Trigonometric function	41.37	42.50	44.28
Linear equation(2 variables)	34.51	42.50	50.01
Linear equation(3 variables)	32.77	34.48	56.68
Area of rectangle	37.10	37.78	48.57
Triangle classification	31.71	56.62	58.34

7. 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. Since manual generation of test data consumes much of the computational time, the process of Test Data Generation has been automated. Software Testing is also an optimization problem with the objective that the efforts consumed should be minimized. Therefore, the search based optimization techniques Genetic Algorithm and clonal selection algorithm are used. To generate suitable data, methods were traversed to cover each node. Test data values were selected based on fitness/affinity values of antibodies which satisfy the predicate node. Based on the predicate node condition, both algorithms were applied and optimal test data was generated. Also, both techniques are compared with random testing to show that the test data generated by search based techniques are better than random testing as the number of test data generated for random testing is less optimal than GA, CSA.

References

[1] P. R. Srivastava and T. Kim, “Application of Genetic Algorithm in software testing”, International Journal of software Engineering and its Applications, 3(4), pp.87 – 96, 2009,
[2] S. Zhang, Y. Zhang, H. Zhou, Q. He, “Automatic Path Test Data Generation Based on GA-PSO”, Proceedings of International Conference on Intelligent Computing and Intelligent Systems, pp. 142-146, 2010.
[3] M. Harman, "The current state and future of search based software engineering", Proceedings of the International Conference on Future of Software Engineering, pp. 342-357, IEEE Press, May 2007.
[4] R. P. Pargas, M. J. Harrold, and R. R. Peck, “Test Data Generation using Genetic Algorithms”, Journal of Software Testing, Verifications, and Reliability, vol. 9, pp. 263-282, 1999.
[5] C. C. Michael, G. E. McGraw, M. A. Schatz, “Generating software test data by evolution”, IEEE Transactions on Software Engineering, vol.27, no.12, pp. 1085-1110, 2001.
[6] J. Wegener, A. Baresel, H. Sthamer, “Evolutionary test environment for automatic structural testing”, Journal of Information and Software Technology, vol. 43, pp. 841-854, 2001.

[7] L. Bottaci, "A Genetic Algorithm fitness function for mutation testing", Seminal: Software Engineering Using Metaheuristic Innovative Algorithms, 2001.
[8] M. R. Girgis, "Automatic Test Data Generation for data flow testing using a Genetic Algorithm", Journal of Universal computer Science, vol. 11, no. 5, pp. 898-915, 2005.
[9] H. Sthamer, "The Automatic Generation of Software Test Data Using Genetic Algorithms", PhD thesis, Great Britain, 1996.
[10] B. Korel, "Automated software test generation", IEEE Trans. On Software Engineering 16(8): 870–879, 1990.
[11] J.C. Lin, and P.L. Yeh, "Automatic Test Data Generation for path testing using GAs", Information Sc., vol.131, pp.47-64, 2001.
[12] M.A. Ahmed, and I. Hermadi, "GA based multiple paths test data generator", Computers and Operations Research, 2007.
[13] A.S. Ghiduk, M.J. Harrold, and M.R. Girgis, "Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage", 14th Asia-Pacific Software Engineering Conference, IEEE, pp.41-48, 2007.
[14] C. C. Michael, G. E. McGraw, M. A. Schatz and C. C. Walton, "Genetic Algorithm for Dynamic Test Data Generation" Proceedings of the 12th International Conference of Automated Software Engineering, vol.1, issue 5, pp: 307-308, Nov.1997.
[15] N. Chauhan, Software Testing - Principles and Practices, Oxford University Press, 2011.
[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addision-Wesley, Reading, MA, 1989.
[17] X. Xu, Y. Chen, X. Li, and D. Guo, "A Path-Oriented Test Data Generation Approach for Automatic Software Testing", Proceedings of 2nd International Conference on Anti-counterfeiting, Security and Identification, IEEE, pp. 63-66, 2008.
[18] J. Brownlee, "Clonal Selection Algorithms", CIS Technical Report 070209A, February 2007.

Author Profile

Dr. Sanjay Tyagi obtained his Master's degree (Master of Computer Applications) and PhD (Computer Science & Applications) from Kurukshetra University, Kurukshetra. Currently, he is Assistant Professor in the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. His research interests are in Genetic Algorithm and Software Testing. He has presented 30 papers in National & International Conferences.

Ms Poonam Saini obtained her B. Tech (Computer Science and Engg.) degree from Kurukshetra University. Currently, she is pursuing M.Tech (Computer Science & Applications) from the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. Her research interests are Genetic Algorithms, Soft Computing methods and Software Engineering.