International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 4 Issue 3 September 2013

Optimized Test Case Generation Using Genetic Algorithm

Amit Kumar Sharma
Assistant Professor, Department of Computer Science & Engineering,
NIET, NIMS University, Jaipur

Abstract – **Today, Automated generation of test cases to test software product is very essential as it can decrease the time and cost of testing process. While manual testing is very time consuming and costly, Software companies are now turning towards automated testing tools and techniques. In this paper we explore how Genetic Algorithm, improves the quality and reliability of the software by generating optimized test cases. A genetic algorithm (GA) is a search algorithm based on principles from natural selection and genetic reproduction, adopted by Darwin. Three basic steps responsible for GA are (1) selection of initial chromosomes from the population, (2) performing crossover by exchanging the information between selected chromosomes and (3) performing mutation operation on the selected genes. This paper also presents the automated software testing architecture. At last, we are putting some amount of light on pitfalls of automated testing technique.**

Keywords: **Genetic Algorithms, Automated Testing, Chromosomes, Test Cases, Testing Tools**

I. INTRODUCTION

Testing is a very crucial task among all the phases of software development life cycle. According to a survey, it consumes 50% of the total cost involved in software development. [astumtbts]. Software testing normally involves the stages of test case specification, test case generation, test execution, test adequacy evaluation, and regression testing. There are two types of testing styles in the discipline of software engineering today, (1) Manual testing and (2) Automated testing. Manual test generation involves the construction of test cases in a general purpose programming language or a test case specification language with human interaction while automated generation of test data attempts random, symbolic and dynamic approaches using testing tools. Automated software testing technology is now emerging which overcome many of the disadvantages inherent in manual testing. It offers greater efficiency and quality product than manual software testing. [2]

There are different levels of testing used in the testing process, each level of testing aims to test different aspects
of the system. There are different levels of testing (unit, integration and system testing) that can use any of the test design methods. Unit testing is process in which each unit
is separately tested, and changes are done to remove the defects found. Since each unit is relatively small and can be tested independently, they can be exercised much more thoroughly than a large program. During integration testing, the units are gradually assembled and partially assembled subsystems are tested. Testing subsystems allows the interface among modules to be tested. By incrementally adding units to a subsystem, the unit responsible for a failure can be identified more easily. The system as a whole is exercised during system testing. Debugging is continued until some exit criterion is satisfied. The objective of this phase is to find defects as fast as possible. [2]

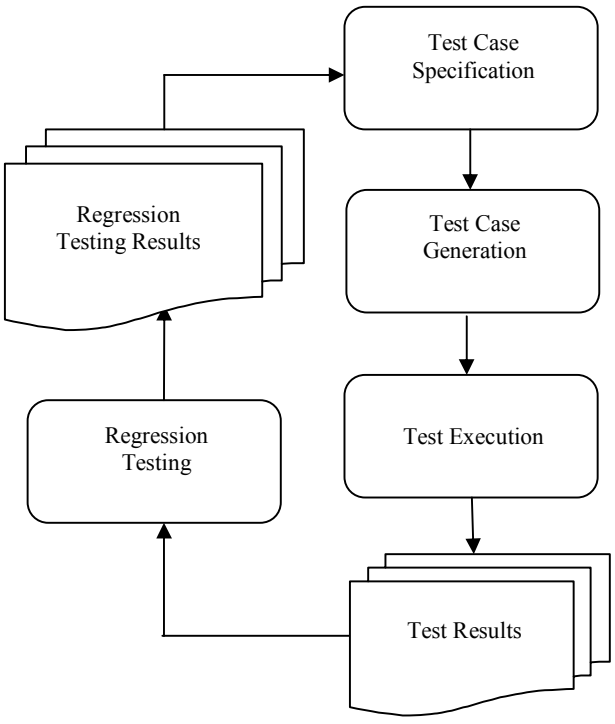

Fig. 1. Testing activities

To test a program, a number of inputs are applied called test cases. The test generation approaches can be divided into the classes: Black-box and White-box. Black-box (behavioral) test case generation is done by only considering the input/output description of the software, nothing about the implementation of the software is assumed to be known. This is the most common form of testing whereas White-box (structural) test design approach is the actual implementation to generate the tests. While test generation using the white-box approach is not common, evaluation of test

International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 4 Issue 3 September 2013

effectiveness often requires use of structural information.[tools2] The third type of testing technique combines the features of both is known as "gray-box" or "translucent-box" test design.

The most critical component of the testing is the generation of test cases. To do testing much faster and to save time we use genetic algorithm for automated testing. [3, 5]

II. MANUAL TESTING V/S AUTOMATED TESTING

Testing is the only tool in a development's arsenal which reduces defects. Planning, design and QA can reduce the number of defects which enter a product, but they can't eliminate any that are already there. Discovering most of the defects is a difficult task. Much of testing is still being done manually. Manual software testing is very repetitive, and thus very unattractive to software engineers. Use of appropriate tools will allow testers to use their skills at a much higher level. [2] The Quality Assurance Institute conducted a benchmark study comparing manual and automated testing in 1995 involving 1,750 test cases and 700 defects. The results are shown below in Table 1.

Test Step	Manual Testing	Automated Testing	% Improvement
Test Plan Development	32	40	-25%
Test Case Development	262	117	55%
Test Execution	466	23	95%
Test Result Analyses	117	58	50%
Defect Tracking	117	23	80%
Report Creation	96	16	83%
Total Hours	1090	277	75%

Table 1. Manual v/s automated testing

The basic purpose of automated test generation is that we reduce the amount of manual work. [tbp]. Manual test care generation is a slow and labor intensive process and may be insufficient if not done carefully. Arbitrarily generated tests can find defects with high testability relatively easily; however, these tests can become ineffective as testing progresses. Automatic test case generation can be an extremely important part of achieving high reliability software. [2] It is slowly becoming a necessity as the demand for low turnaround time and high quality. There are many automated software testing tools and techniques existing and being developed by huge number of testing professional. Three most important tools are: a). test case generator tool, b). test execution tool and c). test evaluation tool.

Goodness of automated testing

Automated testing is good at:-

Load and performance testing – automated tests are a prerequisite of conducting load and performance testing. It is not feasible to have 300 users manually test a system simultaneously, it must be automated

Smoke testing – a quick and dirty test to confirm that the system 'basically' works. A system which fails a smoke test is automatically sent back to the previous stage before work is conducted, saving time and effort.

Regression testing – testing functionality that should not have changed in a current release of code. Existing automated tests can be run and they will highlight changes in the functionality they have been designed to test. [10]

III. GENETIC ALGORITHM

Genetic Algorithm (GA) is implemented to simulate processes in natural systems necessary for evolution, specifically those that follow the principles of survival of the fittest. It is generally used in situations where the search space is relatively large and cannot be traversed efficiently by classical search methods.

The general factors to be considered in genetic algorithms are:

- Population Initialization
- The Fitness Function
- Selection/Operations used

Basic Steps of GA

Following steps are involved in genetic algorithms (as shown in Fig. 2) [9]:

1 [Start] Generate random population of n chromosomes (suitable solutions for the problem)

2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population

3. [New population] Create a new population by repeating following steps until the new population is complete

a. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

b. [Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

c. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

d. [Accepting] Place new offspring in the new population

4. [Replace] Use new generated population for a further run of the algorithm

5. [Test] If the end condition is satisfied, stop, and return the best solution in current population

International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 4 Issue 3 September 2013

6. [Loop] Go to step 2

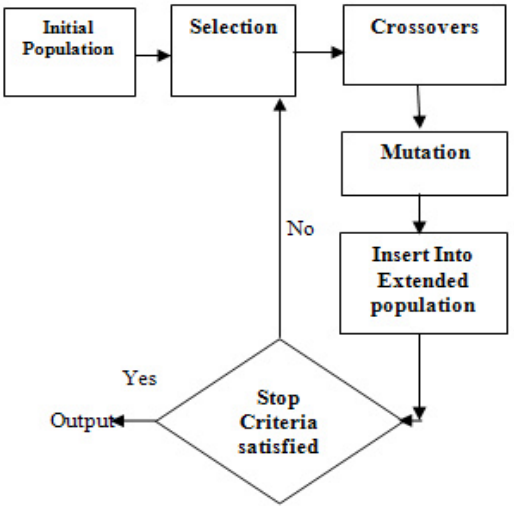

Fig. 2. Flowchart for the basic steps of GA

IV. TEST DATA OPTIMIZATION

A. Description

In the above steps, a population is like a list of several guesses (database) and it consists of information about the individuals. In order to start the optimisation method, the first population has to be generated. It can be seeded with a set of parameter values that can influence the search through the space. It can help to speed up the location of an optimum. Normally the seeding is performed by random selection, which means that random data are generated. There are different methods of representing a chromosome in the genetic algorithm, e.g. using binary, Gray, integer or floating data types.

Chromosomes are selected from the population to be parents for crossover. The problem is how to select these chromosomes. According to Darwin's theory of evolution the best ones survive to create new offspring.

B. Crossover and Mutation (Operators of GA)

Crossover and mutation are two basic operators of GA. Performance of GA depend on them very much. The type and implementation of operators depends on the encoding and also on the problem. Some of the encoding types are Binary Encoding, Permutation Encoding, Value Encoding, Tree Encoding etc.

In mutation a test suit (parent) is selected randomly. Then randomly select some test cases of the parent test suit and replace them with the new test cases generated randomly that are not present in the parent test suit earlier and the new childe test suit gets created. As shown in figure 3, some of the test cases (green, purple and white) of the parent are replaced with newly generated test cases (blue, red and grey) to create a new child test suit.

C. Crossover and Mutation Probability

There are two basic parameters of GA - crossover probability and mutation probability.

Crossover probability: how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population that survives to next generation.

Mutation probability: how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search.

V. TEST AUTOMATION: A DARK SIDE

A. *Various tools used during development are not easy to integrate:* it is quite possible that a different tool is used for each phase of the lifecycle i.e. a business modelling tool during the business analysis phase, a requirements management tool during the requirements phase, a design tool during the design phase, a test management tool during the testing phase, and so on. Because each of the tools used for the various phases may be from a different vendor, they don't easily integrate.

B. *Redundant information can be in multiple repositories:* software development teams may be purchase more than one tool for test management, requirement management or automated testing. By this, duplicate information can be kept in multiple places. It is very difficult to maintain and less productive.

C. *The automated testing tool drove the testing effort:* Often when a new tool is used for the first time on a testing program, more time is spent on automating test scripts than on actual testing. Test engineers may be keen to

automate sophisticated scripts, but may lose sight of the real goal, which is to test the application.

D. *Tools may be selected and purchased before an undefined system engineering environment:* tools that are evaluated and purchased without having system architecture in place can cause problems. When the decision for the architecture is being made, many compatibility issues can surface between the tools already purchased and the suggested architecture.

E. *Automated test script creation is cumbersome:* test engineers with manual test backgrounds are involved in creating the automated scripts.

F. *Various tool versions are in use:* To test a project that has numerous tool licenses, we have various tool versions in use. That means that scripts created in one version of the tool is not compatible in another version, causing considerable compatibility problems and requiring many workaround solutions.

G. *Reports produced by the tool are useless:* The test engineering staff on one project spend much time setting up elaborate customized reports, which is part of the automated testing tool. The reports are never used, since the data required for the report is never accumulated in the tool. [11]

VI. CONCLUSION

Testing is the practice of building confidence of the programmer that shows, the software does what it is intended to do, which in turn improves the consistency of the software. Software testing is the integral, costly, and time consuming activity in the software development life cycle. Automation of software testing process helps in achieving it with reduced cost and time. As it is impossible to test the software exhaustively, so this technique is very useful in selecting the best set of test cases (test suit). The selection is based on three factors that evaluate the test case whether it is good or bad. Genetic algorithms are used for this purpose, GAs are found to be the best technique for selection purpose when there is very large population. The GAs gives good results and their power lies in the good adaptation to the various and fast changing environments. The primary objective of this paper is to propose a GA-based software test data generator and to demonstrate its feasibility. It also shows how this can be achieved. GAs show good results in searching the input domain for the required test sets. GAs may not be the answer to the approach of software testing, but do provide an effective strategy.

RFERENCES

[1] L. A. Clarke. “A system to generate test data and symbolically execute programs”. IEEE Transactions on Software Engineering, vol. SE-2, no. 3, pp. 215- 222, Sept. 1976.

[2] “Automated Software Inspection: A New Approach to increased software Quality and Productivity” by Reasoning LLC, P.O.Box 478, Menlo Park, CA.
Malaiya K. Yashwant, “Automatic Test Software”

[3] B. Korel, "Automated Software Test Data Generation," IEEE Transactions on Software Engineering, vol. 16, no. 8, pp. 870-879, Aug. 1990.

[4] D. L. Bird and C. U. Munoz. “Automatic generation of random self-checking test cases”. IBM Systems Journal, vol. 22, no. 3, pp. 228–245, Mar. 1983.

[5] R. P. Pargas, M. J. Harrold, and R. Peck. “Test-data generation using genetic algorithms”. Software Testing, Verification & Reliability, vol. 9, no. 4, pp. 263–282, Dec. 1999.

[6] A. S. Ghiduk, M. J. Harrold, M. R. Girgis, “Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage;” Software Engineering Conference, APSEC 2007. 14th Asia-Pacific, pp. 41 – 48, Dec. 2007.

[7] Debasis Mohapatra, Prachet Bhuyan, Durga P. Mohapatra, "Automated Test Case Generation and Its Optimization for Path Testing Using Genetic Algorithm and Sampling," WASE International Conference on Information Engineering, ICIE, vol. 1, pp.643-646, Jul. 2009.

[8] P. R. Srivastava, T. h. Kim, “Application of Genetic Algorithm in Software Testing” International Journal of Software Engineering and Its Applications, vol. 3, no.4, pp. 87-95, Oct. 2009.

[9] K. Singh, R. Kumar, “Optimization of Functional Testing using Genetic Algorithms” International Journal of Innovation, Management and Technology, vol. 1,no. 1, pp. 43-46, ISSN: 2010-0248, Apr. 2010.

[10] Jenkins Nick, “A Software Testing Primer: An Introduction to Software Testing” 2008.

[11] Dustin Elfriled, “A Manager’s Guide to Avoiding Pitfall when Automated Testing” Sep-Oct, 1999.