IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. IV (Nov – Dec. 2015), PP 20-26
www.iosrjournals.org

Generation of Search Based Test Data on Acceptability Testing Principle

Dr.I.Surya Prabha, Dr.Chinthagunta Mukundha
Professor Institute of Aeronautical Engineering, Dundugal,Hyd-500043,India.
AssociateProfessor Sreenidhi Institute of Science and Technology Ghatakesar,Hyd-501301,India

Abstract: *The main objective of this paper is to acquire the basic concepts related to automated search based test data generation. The use of Metaheuristic searching techniques for the automatic and generation of search based test data has been a burgeoning interest for many researchers in recent years. Metaheuristic searching techniques are much guaranted in regard to these problems. Metaheuristic searching techniques are high-level tools, which utilize heuristics to seek solutions for combinatorial problems at a considarable computational cost. Metaheuristic search techniques have been applied to automate search based test data generation for structural and functional testing. Evolutionary testing designates the use of metaheuristic search methods for test case generation.*

The search space is the input domain of the test object, with each individual or potential solution, being an encoded set of inputs to that test object. The fitness function is tailored to find search based test data for the type of testing that is being considered. Evolutionary Testing (ET) uses optimizing search techniques such as evolutionary algorithms to generate search based test data. The efficiency of GA-based testing system is compared with a Random testing system. For easy programs both testing systems work fine, but as the complexity of the program or the complexity of input domain grows, GA-based testing systems. The results suggest that our acceptability based algorithm is better than the reliability based path testing and condition testing techniques in both of these categories. Thus this algorithm may significantly reduce the time of search based test data generation significantly outperforms Random testing.

Keywords*: Automated, Metaheuristic, Framework, Potential, Random.*

I. Introduction

The use of metaheuristic search techniques for the automatic generation of search based test data has been a burgeoning interest for many researchers in recent years. In industry, search based test data selection is generally a manual process - the responsibility for which usually falls on the tester.

The bugs in software can cause major loss in IT organization if they are not removed before delivery. Software testing is important parameter developing software that is free from bugs and defects. Software testing is performed to support quality assurance. A good quality software can be made by using an efficient test method. Statistics say that 50% of the total cost of software development is devoted to software testing even it is more in case of critical software. Depending on time, scale and performing methods we can classify testing as unit testing, integration testing, system testing, alpha testing, beta testing, acceptance testing, regression testing, mutation testing, performance testing, stress testing etc.

Finding a set of search based test data to achieve identified coverage criteria is typically a labour-intensive activity consuming a good part of the resources of the software development process. Automation of this process can greatly reduce the cost of testing and hence the overall cost of the system. Many automated search based test data generation techniques have been proposed by researchers. We can broadly classify these techniques into three categories: random, static and dynamic .Random approaches generate test input vectors with elements randomly chosen from appropriate domains. Input vectors are generated until some identified criterion has been satisfied. Random testing may be an effective means of gaining an adequate test set but may simply fail to generate appropriate data in any reasonable time-frame for more complex software.

Recently Search Based Software Engineering has evolved as a major research field in the software engineering community. Search Based Software Engineering has been applied successfully to many software engineering activities ranging from requirements engineering to software maintenance and quality assessment. One major area where Search Based Software Engineering has seen intense activity is software testing. Active research is underway to improve the existing search based test data generation techniques and propose novel approaches to solve the test generation problem. However, despite much research, there are still limitations that have hampered the wide acceptance of these techniques. Also many areas are under-explored, and there are distinct possibilities for the successful use of search based approaches.

A search based test data generation algorithm that generates search based test data using adequacy based testing criteria and genetic algorithms. In this paper, we mainly focus on providing the algorithm in a

formalized manner and on evaluating the algorithm by comparing it with other search based test data generation techniques. The main aim is to prove the effectiveness of our proposed algorithm based on adequacy based testing criteria. Our algorithm applies mutation analysis to generate an adequate search based test data set.

Search Based Software Engineering research has attracted much attention in recent years as part of a general interest in search based software engineering approaches. The growing interest in search based software testing can be attributed to the fact that there is a need for automatic generation of test data, since it is well known that exhaustive testing is infeasible and the fact that software test data
Generation is considered NP-hard problem.

II. Literature Review

Software has become an intrinsic part of human life and it is important that it should perform its intended function. Otherwise it can cause frustration, loss of resources and even loss of life. The main activity that attempts to prevent this and verify software quality and reliability is software testing. Testing is a dynamic activity, as it requires execution of program on some finite set of input data. Nevertheless there are other methods such as static analysis and formal proof of correctness. However, only testing can be used to gain confidence in the correct functioning of the software in its intended environment. We cannot perform exhaustive testing because the

Domain of program inputs is usually too large and there are too many possible input paths. Therefore, the software is tested against suitably selected test cases.

Evolutionary testing makes use of meta-heuristic search techniques for test case generation .Evolutionary Testing is a sub-field of Search Based Testing in which Evolutionary Algorithms are used to guide the search. The test aim is transformed into an optimization problem. The input domain of the test object forms the search space. The test object searches for search based test data that fulfils the respective test aim in the search space. A numeric representation of the test aim is necessary for this search. This numeric representation is used to define objective functions suitable for the evaluation of the generated search based test data. Depending on the test aim pursued, different heuristic functions emerge for test data evaluation. Due to the non-linearity of software the conversion of test aim to optimization problems mostly leads to complex, discontinuous and non-linear search spaces. Therefore neighborhood search methods are not recommended. Instead, meta-heuristic search methods are employed, e.g. evolutionary algorithms, simulated annealing or tabu search. Evolutionary Algorithms have proved a powerful optimization algorithm for the successful solution of software testing.

The main activities in software testing are test case generation, executing program using these generated test cases and evaluating the results. A test case is a set of test input data and the expected results. The test data is a set of input values to the program, which may be generated from the code or usually derived from program specifications. Program specifications also help in determining the expected results.

Meta-heuristic techniques have also been applied to testing problems in a field known as Search Based Software Testing a sub-area of Search Based Software Engineering .Evolutionary algorithms are one of the most popular meta-heuristic search algorithms and are widely used to solve a variety of problems.
The local Search techniques generally used are
i. Hill Climbing
ii. Simulated Annealing
iii. Tabu Search

Hill Climbing

In hill climbing, the search proceeds from randomly chosen point by considering the neighbors of the point. Once a neighbor is found to be fitter then this becomes the current point in the search space and the process is repeated. If there is no fitter neighbor, then the search terminates and a maximum has been found. However, HC is a simple technique which is easy to implement and robust in the software engineering applications of modularization and cost estimation.

Simulated Annealing

Simulated annealing is a local search method. It samples the whole domain and improves the solution by recombination in some form. In simulated annealing a value x1, is chosen for the solution, x, and the solution which has the minimal cost function, E, is chosen. Cost functions define the relative and desirability of particular solutions. Minimizing the objective function is usually referred to as a cost function; whereas, maximizing is usually referred to as fitness function.

Tabu Search

Tabu search is a metaheuristic algorithm that can be used for solving combinatorial optimization problems, such as the travelling salesman problem. Tabu search uses a local or neighbourhood search procedure to iteratively move from a solution *x* to a solution *x*' in the neighbourhood of *x*, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure, tabu search modifies the neighbourhood structure of each solution as the search progresses.

Evolutionary Search Using Genetic Algorithms

Genetic Algorithms forms a method of adaptive search in the sense that they modify the data in order to optimize a fitness function. A search space is defined, and the Genetic Algorithm search probe for the global optimum. A Genetic Algorithms starts with guesses and attempts to improve the guesses by evolution. A Genetic Algorithms will typically have five parts: (1) a representation of a guess called a chromosome, (2) an initial pool of chromosomes, (3) a fitness function, (4) a selection function and (5) a crossover operator and a mutation operator. A chromosome can be a binary string or a more elaborate data structure. The initial pool of chromosomes can be randomly produced or manually created. The fitness function measures the suitability of a chromosome to meet a specified objective: for coverage based ATG, a chromosome is fitter if it corresponds to greater coverage. The selection function decides which chromosomes will participate in the evolution stage of the genetic algorithm made up by the crossover and mutation operators. The crossover operator exchanges genes from two chromosomes and creates two new chromosomes. The mutation operator changes a gene in a chromosome and creates one new chromosome.

III. Generation Of Search Based

Test Data

Genetic programming results in a program, which gives the solution of a particular problem. The fitness function is defined in terms of how close the program comes to solving the problem. The operators for mutation and mating are defined in terms of the program's abstract syntax tree. Because these operators are applied to trees rather than sequences, their definition is typically less straight forward than those applied to Genetic Algorithms GP can be used to find fits to software engineering data, such as project estimation data.

In order to apply metaheuristics to software engineering problems the following steps should therefore be considered:

i. Ask: Is this a suitable problem?
That is, "is the search space sufficiently large to make exhaustive search impractical?"
ii. Define a representation for the possible solutions.
iii. Define the fitness function.
iv. Select an appropriate metaheuristic technique for the problem.
v. Start with the simple local search and consider other genetic approaches.

The testing requirements satisfied by the generated test data is the measurement of coverage in terms of statement, condition, path, branch, decision etc.

Statement coverage

Statement coverage measures the number of executable statements in the code that are executed by a test suite. 100% statement coverage is achieved when every statement in the code is executed.

Decision coverage

Decision coverage, also known as branch coverage, measures the extent to which all outcomes of branch statements are covered by test cases. To achieve decision coverage, two test data *I*1 and *I*2 need to be generated corresponding to each decision *di* in the program such that *di* evaluates to true when the code is executed with input *I*1 and evaluates to false when code is executed with input *I*2. For example, to cover the decision at line 70 in Fig.1, we require two test data such that the 'if' condition evaluates to true in one case and false in the other.

```
10: int inp1 ,inp2 ; //inputs given
20: int test()     // function
30: {
40:      int 1Var=0 ,retVal=0;
50:      if(inp1 > 15)
60:     1Var=1;
70:     if(1Var && inp2)
```

```
80:    retVal=1;
90:    return retVal;
100: }
```

Fig 1:Sample C Code

Condition coverage

Condition coverage is similar to decision coverage with the only difference being that for condition coverage, two test data *I*1 and *I*2 are needed for each condition in a decision.

3.1 Automated test data generation (ATDG)

Most of the work on Software Testing has concerned the problem of generating inputs that provide a test suite that meets a test adequacy criterion. The schematic representation is presented in Fig.2. Often this problem of generating search based test inputs is called 'Automated Test Data Generation (ATDG)' though, strictly speaking, without an oracle, only the input is generated.Fig.2 illustrates the generic form of the most common approach in the literature, in which search based test inputs are generated according to a test adequacy criteria. The test adequacy criterion is the human input to the process. It determines the goal of testing.

The adequacy criteria can be almost any form of testing goal that can be defined and assessed numerically. For instance, it can be structural functional, temporal etc. This generic nature of Search-Based Testing has been a considerable advantage and has been one of the reasons why many authors have been able to adapt the Search-Based Testing approach different formulations.

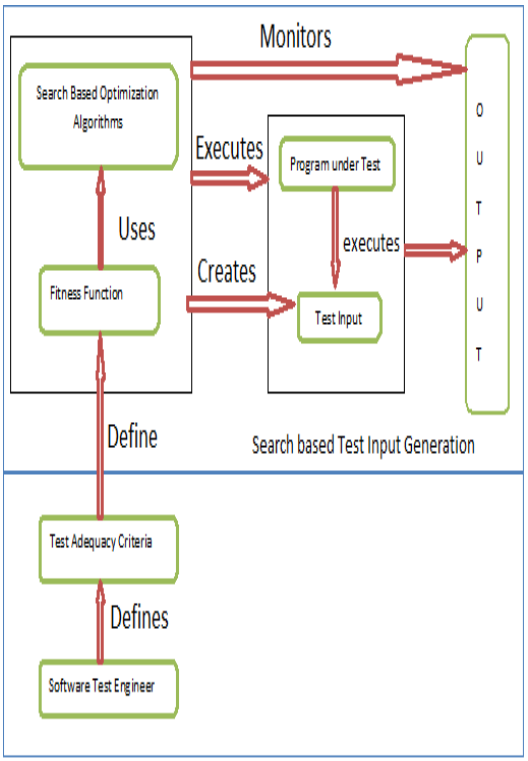

Figure2 : A generic search-based test input generation scheme

3.2 Evolutionary Algorithms

Evolutionary Algorithms use simulated evolution as a search strategy to evolve candidate solutions, using operators inspired by genetics and natural selection. For Genetic Algorithms, the search is primarily driven by the use of recombination - a mechanism of exchange of information between solutions to \breed" new ones - whereas Evolution Strategies principally use mutation - a process of randomly modifying solutions.

```
Select a starting Solution s ϵ  S
Select an Initial Temperature t > 0
Repeat
      It ← 0
      Repeat
```

```
    Select   s^I ϵ N(s) at Random
   Δe ← obj(s^I) – obj(s)
  If Δe < 0
      S ← s^I
  Else
     Generate random Number r , 0 ≤ r < 1
     If r < e − δ/t   then s ← S^I
  End if
  It ← it+1
 Until it=num_solns
 Decrease t according to cooling schedule
Until Stopping Condition Reached
```

Fig: 3 Evolutionary Algorithm

IV. Pragmatic Data Collection

Evaluating the performance of any technique requires selecting certain subject programs which forms the basis for evaluation. To evaluate the performance of our proposed algorithm and to compare it with other techniques, we have selected fifty real time programs written in C language. The subject programs we have chosen are described in Table 1. The programs range from 35 to 350 lines of source code.

We have selected a large program base that Contains programs ranging from very basic such as computing the grade of student, finding the biggest of three numbers to very complex such as implementing the binary search tree and finding the intersection of two linked lists. We have chosen a diversified range of programs including mathematical problems such as finding roots of quadratic equation, triangle classification problem, computing the median of the triangle; general
logical problems such as checking for the Armstrong number, magic number, palindrome number; business problem such as payroll system, commission problem, credit risk analysis; data structures such as linked list, sorting (insertion sort, selection sort, bubble sort, merge sort, heap sort, quick sort, shell sort), searching (linear search, binary search) etc. All the programs are written in standard C language that makes it easier to work with these programs.

Subject Program	Description
SP1	To find grade of a student
SP2	To find bigger of three students
Sp3	Payroll System
SP4	Primality Checking
SP5	To guess a number and then check whether it is prime or not
SP6	Calculator Implementation
SP7	Leap Year Detection
SP8	To find Number of Odd days in a given year
SP9	Reverse of a Number
SP10	To read a number and calculate its square root if it is a perfect square
SP11	To display a three digit number that is 3 time the sum of its digits
SP12	To display a 3 digit number after adding 1 to each of its digits
SP13	To Check for a Palindrome number
SP14	To check for 3 digit Armstrong number
SP15	To read two numbers and check they are co-prime
SP16	To guess a number and check whether it is single digit number
SP17	Credit risk Analysis : To estimate the amount of risk involved in granting a loan to a person given his income and savings
SP18	To guess a number and then check whether it is the magic number
SP19	Linear Search
SP20	Binary Search
SP21	Insertion Sort
SP22	Selection Sort
SP23	Bubble Sort

SP24	To find smallest and biggest number from a list of numbers
SP25	To find number of dates between two dates
SP26	To read a number and find the sum of its digits
Sp27	To find area of an equilateral triangle
SP28	Previous Date Problem : To find the previous date of a given date
SP29	To find roots of a quadratic equation
SP30	Triangle classification problem
SP31	To compute the medium of the triangle
SP32	Commission problem
SP33	To compute GCD of two numbers
SP34	Single Linked list implementation
SP35	Double linked List implementation
SP36	To find the intersection of Two linked list
SP37	Quick Sort
SP38	Merge Sort
SP39	Heap Sort
SP40	Shell Sort
SP41	Binary Search Tree
SP42	Stack Implementation using array
SP43	Queue implementation using array
SP44	To read a number and displayit in words
SP45	Breadth First Search
SP46	Depth First Search
SP47	Transposition Cipher Implementation
SP48	Play-Fair Cipher Implementation
SP49	Implementation of 8-puzzle problem
SP50	Implementation of perception

Table 1 : Subject Programs Description

V. Conclusion

This paper has provided an overview of the Search-Based Software Engineering and the Search-Based Software Testing used in test data generation. The main goal is to make a study of the use of search-based optimization techniques to automate the evolution of solutions for software engineering problems.

In this paper, we considered the time required for test data generation and the percentage of branch coverage; other parameters can also be considered for future work. Test data has been generated in numerals; similarly for character, string, arrays and pointers can also be tried with the genetic algorithm.

The main aim is to compare the adequacy based testing criteria with the reliability based testing criteria and to show the effectiveness of former over the latter. In order to do this, we evaluate our algorithm using fifty real time programs written in C language. We compare our algorithm with path testing and condition testing techniques for these fifty programs, in two categories viz. number of test cases and time taken to generate test cases.

References

[1] Mark Harman, "The Current State and Future of SBSE", Future of Software Engineering (FOSE'07), IEEE Computer Society, 2007, pp. 1-16.

[2] Phil McMinn, "Search-Based Software Test Data Generation: A Survey", Ph.D. Thesis,Software Testing, Verification and Reliability, 2004.

[3] Maha Alzabidi, Ajay Kumar, and A.D. Shaligram, "Automatic Software Structural Testing by Using Evolutionary Algorithms for Test Data Generations", IJCSNS International Journal of Computer Science and Network Security, VOL.9, No.4, April 2009.

[4] Kamran Ghani and John A. Clark, "Automatic Test Data Generation for Multiple Condition and MCDC Coverage", 2009 Fourth International Conference on Software Engineering Advances, 2009, pp. 152-157.

[5] Mark Harman, S. Afshin Mansouri and Yuanyuan Zhang, "Search Based Software Engineering: AComprehensive Analysis and Review of Trends, Techniques and Applications", April 9, 2009.

[6] J. Clark, J. J. Dolado, M. Harman, R. M. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper,and M. Shepperd, "Reformulating Software Engineering as a Search Problem," IEE Proceedings - Software, vol. 150,no. 3, 2003, pp. 161–175.

[7] André Baresel, Harmen Sthamer and Michael Schmidt, "Fitness Function Design To Improve Evolutionary Structural Testing," Proceedings Of The Genetic And Evolutionary Computation Conference, 2002.

[8] Mark Harman, "Automated Test Data Generation using Search Based Software Engineering", Second International Workshop on Automation of Software Test (AST'07), IEEE Computer Society, 2007, pp. 1-2.

[9] M. Harman and B.F. Jones, "Search Based Software Engineering", Information and Software Technology, Dec. 2001, 43(41): pp. 833-839.
[10] Praveen Ranjan Srivastava and Tai-hoon Kim, "Application of Genetic Algorithm in Software Testing", International Journal of Software Engineering and Its Applications, Vol.3, No.4, Oct'2009, pp. 87-95.
[11] Mitchell B.S., "A Heuristic Search Approach to Solving the Software Clustering Problem", PhD Thesis, Drexel University, Philadelphia, PA, Jan'2002.
[12] N.J. Tracey, "A search-Based Automated Test-data Generation Framework for Safety-Critical Systems", DPhil University of York, 2000.
[13] Y. Zhan, John A. Clark, "A Search-Based Framework for Automatic Testing of MATLAB/Simulink Models", The Journal of Systems and Software 81 (2008), pp. 262-285.
[14] P. McMinn, M. Harman, D. Binkley and Paolo Tonella, "The Species per Path Approach to Search-Based Test Data Generation", International Symposium on Software Testing and Analysis (ISSTA'06), July 17-20, USA, 2006, pp. 1-11.
[15] Yuanyuan Zhang, "Multi-Objective Search - based Requirements Selection and Optimisation", Ph.D Thesis, King's College, University of London, February 2010, pp. 1-276.

Authors Profile

Dr.I.Surya Prabha ,Professor,Department of Information Technology ,Institute of AeronauticalEngineering,HYD-500043 , AP, India.
E-mail-ipsurya17@gmail.com

Dr.Chinthagunta Mukundha, Associate Professor, Department of Information Technology, Sreenidhi Institute of Science and Technology, HYD-501301, AP, India.
E-mail-mukundhach@gmail.com.