Review of Search based Techniques in Software Testing

Rakesh Roshan
Research Scholar
Manav Bharti University
Solan(HP),INDIA

Rabins Porwal
ITS
Ghaziabad
UP,INDIA

Chandra Mani Sharma
ITS
Ghaziabad
UP, INDIA

ABSTRACT
The most effort seeking job in software testing is the generation of test cases. The success of testing pursuit highly depends on the effectiveness of the test cases. Various approaches have been proposed to ease the task of test case generation and to perform software testing. It has witnessed a paradigm shift from manual test case generation to automated test case generation in the recent time. Search Based Software Testing (SBST) has evolved as a new domain in software testing. This paper reviews the various Search Based Software Testing approaches, foresees trends in the research being conducted in this area and explores the new possibilities which future of the software testing envisages. This paper presents an exhaustive survey on Search Based Software Testing and also touches upon the other disciplines of modern day computing which seamlessly overlap with SBST.

General Terms
Survey, Review Article, Theoretical Computer Sciences

Keywords
Software Testing, Model Based Software Testing, Test Case Generation Approaches.

1. INTRODUCTION
Search-based software testing is the use of random or directed search techniques such as hill climbing, genetic algorithms etc. to address problems in the software testing and verification and validation domain. Search Based techniques are gaining more and more popularity in software testing, verification, and validation and is especially very useful in test data generation. A problem in the software testing and/or verification and validation domain can be addressed by using a search strategy such as random search, local search (e.g. hill climbing, simulated annealing and tabu search), evolutionary algorithms (e.g. genetic algorithms, evolution strategies and genetic programming), ant colony optimization and particle swarm optimization. Other modern buzzwords in the arena of Software Testing are- model-based testing, real-time testing, interaction testing, testing of service-oriented architectures, test case prioritization and generation of whole tests with data.

Testing is the important phase to ensure the quality of a software development. The main reason to test is to find bugs in the software. Even though no bugs found, testing cannot ensure that the software is bug-free [1]. Testing increases our confidence to software reliability. Also if we predict the defects, then we can save time as well as cost for software development.

Firstly testers check the correctness of software and there are various tools to check the correctness of the software. Automated generation of test cases reduces the testing effort and time. Failure of software occurs when any module or part of software does not work as we required and expected. So, software needs to be tested on a variety of input data to ensure its correct working. With the continuous growth of the software and system complexity, requirement of efficient testing mechanisms is also in demand. The automation of testing is a must to reduce efforts laid down by the tester. In recent time, the concept of search based software testing has started gaining popularity because of several reasons. Search based software testing (SBST) is the part of search based software engineering (SBSE) [2]. Search based software testing (SBST) naturally draws the attention of researchers because of the increasing span of search based software engineering (SBSE). Recent years have witnessed for the rise of Search Based Software Testing (SBST) and especially for techniques of generating the test data that meets a given criterion. McMinn et al, presents a survey on Search based software test generation, which shows the application of search based techniques for White Box testing, Black Box testing, Grey Box testing and for the verification of non-functional properties[1]. The Search Based Software Testing can be implemented using White Box approach as well as using Black Box approach.

2. SURVEY OF THE RELATED WORK
The term Search Based Software Testing was first used by Miller et al. in 1976. Their approach was brand new approach for that time, which proved to an effective technique for test data generation. This approach was efficiently applied for generating the floating point inputs. With the passage of time various concepts associated with Search Based Software testing such as the evolutionary approach. Genetic algorithms were primarily used for this purpose. Genetic A genetic test case generation evolves like a tree, in which each node is a function and its inputs are the children of that node. According to a problem specific fitness function, individuals are used to fill the next population, which is held at each generation. Antoniol et al. evaluate search based testing approaches to detect software vulnerabilities [3]. Their work presents some of the interesting open research problems in the arena of search based testing while the key thrust area being software vulnerability as paradoxically, the absence of software vulnerability does not reduce security risk [3] Also, security is not limited to a program in isolation, a given environment or network configuration. A recent NIST technical guide to information security testing and assessment [4], lists panorama of penetration testing[5]. Penetration testing mimics a real world attack and can be conducted in many different ways. Windisch et al. propose two novel approaches for generating input signals from within search-based testing techniques [6]. They apply this technique to generate the valid signals for continuous systems. These approaches are then shown to be very effective when experimentally applied to the problem of approximating a set of realistic signals. Approximating a signal and reaching a certain goal within search-based software testing are both based on fitness value feedback, thus being directly comparable. The results indicate that both approaches are, despite some minor issues, well-suited for signal generation and optimization when applied to software testing. Blanco et

al. present an approach based on the meta-heuristic technique 'Scatter Search' for the automatic test case generation of the BPEL business process[5]. A transition coverage is used as adequacy criterion. The approach presented by Blanco et al.[5] is an evolution of the TCSS-LS algorithm described in Blanco et al.; 2008, which generates test cases for the branch coverage criterion for programs written in C. Marchetto et al.; 2009 investigate a search-based algorithm for the exploration of the huge space of long interaction sequences, in order to select those that are most promising, based on a measure of test case diversity. In another paper, Marchetto et al. 2008 investigate the use of state-based testing for AJAX Web applications and particularly focus on the specific faults introduced by this technology. The technique is based on dynamic extraction of a finite state machine for an Ajax application and its analysis with the aim of identifying sets of test cases based on semantically interacting events. Empirical evidence shows the effectiveness of this kind of technique in finding faults. Automatically generated test cases are comparable to those obtained by careful functional testing, manually performed by expert testers. Unfortunately, one of the main drawbacks of the technique based on semantically interacting events is that it generates testing suites composed of a very large number of test cases and this can limit its usefulness [7]. Afzal et al.[2] apply experiments targeting fault prediction using genetic programming and compare the results with traditional approaches in terms of efficiency gains. They evaluate the use of genetic programming for predicting fault count data. Harman et al.; 2010 introduce three algorithms which do this without compromising the coverage achieved. Results of the empirical study of effectiveness of the three algorithms on five benchmark programs, containing non trivial search spaces for branch coverage, have also been presented by Harman et al.; 2010. The results indicate that it is possible to make reductions in the number of test cases, produced by search based testing, without loss of coverage. The branch coverage based approaches of search based testing re-formulate the automated test data generation problem. They introduce a new formulation of the search based structural test data generation problem in which the goal is to maximize coverage, while simultaneously minimizing the number of test cases, with a view to taking into account the human oracle cost effort involved in checking the behavior of the software under test for a given test suite. Zhao et al.; 2010 present an automated test data generation system for feasible transition paths (FTP) on Extended Finite State Machines (EFSM) models. They investigate the statistical properties of testing efficiency using statistical tests for correlation and formalization according to the test data generated by applying the system on four widely used EFSM models. An important and encouraging finding is a close positive correlation between test generation cost and the number of numerical equal operators in conditions (NNEOC) on a FTP. Li et al.; 2010 perform a simulation experiment to study five search algorithms for test case prioritization and compare the performance of these algorithms. The target of the study is to have an in-depth investigation and improve the generality of the comparison results. The simulation study provides two useful guidelines: (1) Use of efficient search algorithms such as Additional Greedy Algorithm (AGA) and Optimal Greedy Algorithm(OGA). These outperform the other three search algorithms in most of the cases. Hence, these two search algorithms are recommended to be used preferentially. (2) Ratio of Overlapping [8], is introduced to better investigate the performance of these five search algorithms. Results in paper [9], indicate that the performance of two best search algorithms, Additional Greedy Algorithm and Optimal Greedy Algorithm, is maximum when the Ratio of Overlapping (RO) is around 3 and it starts declining when RO increases.

Afzal et al.; 2010, evaluate the following five different techniques for predicting the number of faults slipping through unit, function, integration and system testing phases.

i. Particle swarm optimization based artificial neural networks (PSO-ANN),
ii. Artificial immune recognition systems (AIRS),
iii. Gene expression programming (GEP),
iv. Genetic programming (GP)
v. Multiple regression (MR),

The objective of the study in paper [10] is to quantify improvement potential in different testing phases by finding the right faults in the right phase. At the unit and function testing phases, AIRS and PSO-ANN performed better while GP performed better at integration and system testing phases. The study concludes that a variety of search-based techniques are applicable for predicting the improvement potential in different testing phases with GP showing more consistent performance across two of the four test phases. This study is different from the other software quality evaluation studies. First, the dependent variable of interest here is the number of faults slipping through to various testing phases with the aim of triggering corrective actions for avoiding unnecessary rework late in software testing. Second, Afzal et al.; 2010 make use of several independent variables at the project level, i.e. variables depicting work status, testing progress status and fault-inflow. Lindlar et al.; 2010 present an approach in which evolutionary functional testing is performed using an actual electronic control unit for test case evaluation. A test environment designed to be used for large-scale industrial systems is introduced. An extensive case study has been carried out to assess its capabilities. Results indicate that the approach proposed in this work is suitable for automated functional testing of embedded control systems within a Hardware-in-the-Loop test environment. De-Souza et al.; 2010 report a comprehensive experimental study regarding the human competitiveness of search based software engineering (SBSE). The experiments are performed over four well-known SBSE problem formulations: next release problem, multi-objective next release problem, workgroup formation problem and the multi-objective test case selection problem. For each of these problems, two instances, with increasing sizes, are synthetically generated and solved by both meta-heuristics and human subjects. De-Souza et al.;2010 conduct an experiment, in which 63 professional software engineers participated for solving some or all problem instances, producing together 128 responses. The comparison analysis strongly suggests that the results generated by search based software testing can be said to be human competitive. The research, reported in paper [11], addresses human competitiveness of search based software engineering results precisely under this criterion. Therefore, controlled experiments, as described later, designed to compare the automatically generated results with the results obtained from human subjects were performed and analyzed. In another search-based test data generation approach proposed by Romano et al.; 2011 identification of NPEs is done. The approach consists of two steps: (i) an inter-procedural data and control flow analysis—relying on existing technology—that identifies paths between input parameters

and potential NPEs, and (ii) a genetic algorithm that evolves a population of test data with the aim of covering such paths. The algorithm is able to deal with complex inputs containing arbitrary data structures.

The approach, proposed in paper [12], has been evaluated on to test class clusters from six Java open source systems, where NPE bugs have been artificially introduced. Results show that the approach is, indeed, able to identify the NPE bugs, and it outperforms random testing. Also, they show how the approach is able to identify real NPE bugs some of which are posted in the bug tracking system of the Apache libraries. Yano et al.; 2011 introduce a multi-objective evolutionary approach to test case generation from extended finite state machines (EFSM), named MOST. Testing from an EFSM generally involves executing various transition paths, until a given coverage criterion such as coverage of all transitions is met. As traditional test generation methods from FSM only consider the control aspects, they can produce many infeasible paths when applied to EFSMs, due to conflicts in guard conditions along a path.

3. FUTURE DIRECTIONS IN SEARCH BASED SOFTWARE TESTING

Although, the concept of Search Based Testing is much old and was introduced by **Miller et al.; 1976**, yet the progress in this field was almost negligible decades after. Here, we present the progress made in this direction during the last two decades. This can be envisaged from Table 1 that in from 1990 to 2000, the progress in the field of Search Based Software Testing has been very slow. Only 59 research papers appear during the whole decade on the topic. During the decade of 2001-2010, this area of research became more active and during the year 2008-09 only, a huge number of 120 research papers were published on Search Based Software Testing. In the days to come it can be projected that more and more researchers will be attracted to this field because of its increasing applications in software testing. As in the days to come, majority of the software applications would be web based. Various new technologies including SAS(Software As a Service), Cloud Computing, and Search Engine Optimization(SEO) will definitely provide a thrust to the growth of the Search Based Software Testing.

Table 1: Year-wise Research Publications on Search Based Software Testing

Year	1990-1991	1992-1993	1994-1995	1996-1997	1998-1999	2000-2001	2002-2003	2004-2005	2006-2007	2008-2009	2010-2011
Research Publications	2	3	7	11	18	18	35	50	77	120	62+

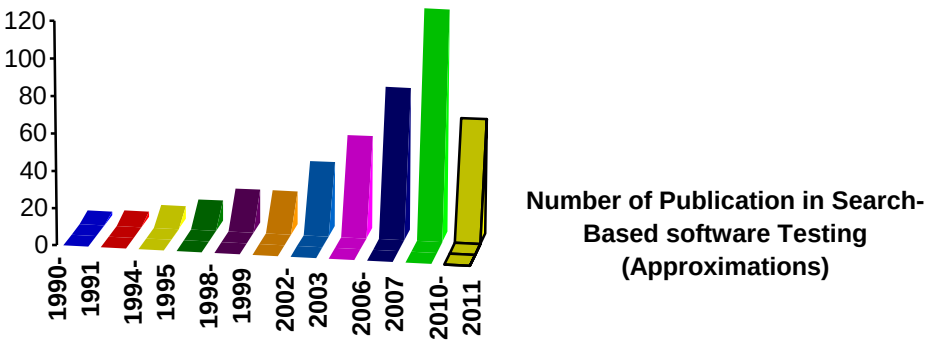

Figure 1: Growing Interest of Research Community Towards Search Based Software Testing

Figure1 depicts the increasing interest towards Search Based Software Testing. On horizontal axis, the corresponding years have been depicted, while vertical axis marks the number of publications in the respective years. Although initial growth in the field of Search Based Software Testing was moderate, yet the involvement of active research community has made growth of the filed ever increasing. The popularity of the domain can be measured from the fact that various International conferences are having the special sessions on SBST. In last five years, IEEE has been organizing the International workshops on Search Based Software Testing in conjunction with the International Conference on Software Testing, Verification and Validation (ICST). First IEEE workshop was held in 2008 at Lillehammer, Norway, second in 2009 at Denver, Colorado, USA, third in 2010 at Paris, France , fourth in 2011 at Berlin, Germany and fifth workshop on Search Based Software Testing (SBST 12) is scheduled to be held in April ,2012 at Montreal, Canada. The area of Search Based Software Testing promises of vast possibilities in the days to come.

4. CONCLUSIONS

This paper reviews the recent advancements in this field of Search Based Software Testing. The area spawns an all new domain ining the arena of modern Software Testing. Search Based Software Test has many advantages including reduced efforts and improved reliability over state-of-the-art approaches of Software Testing. Search Based Software

Testing can be implemented in various forms, including White Box and Black Box Testing. It can exploit the potential of other modern day computing approaches including evolutionary approaches of computing such as Genetic Algorithms and Finite State Machines (FSM). This paper also projects the increasing interest of research community towards this highly promising area of Software Testing in terms of increasing number of publications on Search Based Software Testing, year after year. .

5. ACKNOWLEDGMENTS

The first author of the paper is thankful to MBU, Solan for encouragement and support.

6. REFERENCES

[1] P. McMinn, "Search-based software test data generation: A survey," Journal of Software Testing Verification and Reliability, pp. 105-156, June, 2004.

[2] W. Afzal, R. Torkar and R. Feldt, "A systematic review of search based testing for non-functional system properties," Journal of Information and Software Technology, vol. 51, 2009.

[3] G. Antoniol, "Search Based Software Testing for Software Security: Breaking Code to Make it," in Proceedings of the International Conference on Software Testing Verification and Validation Workshops, pp 87-100, 2009.

[4] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Technical guide to information security testing and assessment," U.S. Dept. Of Commerce - National Institute of Standards and Technology, Technical Report, pp. 800-811, September 2008.

[5] R. Blanco, J. Tuya, B. Adenso-Díaz, "Automated test data generation using a scatter search approach", Journal of Information and Software Technology, vol. 51, pp.708-720,2008.

[6] A. Windisch and N.A.Moubayed, "Signal Generation for Search-Based Testing of Continuous Systems," in Proceedings of IEEE International Conference on Software Testing Verification and Validation, pp 121-130, 2009.

[7] X. Yuan and A. M. Memon, "Using GUI run-time state as feedback to generate test cases," in Proceedings of the 29th International Conference on Software Engineering, Washington, DC, USA, May 23–25, pp. 396–405, 2007.

[8] A.Marchetto and P.Tonella,"Search-Based Testing of AjaxWeb Applications," in Proceedings of IEEE International Symposium on Search Based Software Engineering, pp 3-12, 2009.

[9] S.Li, N.Bian, Z.Chen, D.You, Y.He, "A Simulation Study on Some Search Algorithms for Regression Test Case Prioritization" 10th IEEE International Conference on Quality Software, pp 72-81, 2010.

[10] W.Afzal, R.Torkar and R.Feldt,"Search-based prediction of fault-slip-through in large software projects", 2nd IEEE International Symposium on Search Based Software Engineering, pp 79-88, 2010.

[11] J.T. De-Souza, C. L. Maia, F. G. De-Freitas and D.P. Coutinho, "The Human Competitiveness of Search Based Software Engineering," 2nd IEEE International Symposium on Search Based Software Engineering, pp 143-152, 2010.

[12] D. Romano, M. Di Penta, G.Antoniol,"An Approach for Search Based Testing of Null Pointer Exceptions," Fourth IEEE International Conference on Software Testing, Verification and Validation, pp 160-169, 2011.