Information and Software Technology 81 (2017) 19–35

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# A systematic review on search based mutation testing

Rodolfo Adamshuk Silva*, Simone do Rocio Senger de Souza, Paulo Sérgio Lopes de Souza

*Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos, SP 13566-590, Brazil*

ARTICLE INFO

*Article history:*
Received 4 July 2015
Revised 23 December 2015
Accepted 28 January 2016
Available online 6 February 2016

*Keywords:*
Mutation testing
Search based software testing
Meta-heuristic

ABSTRACT

*Context:* Search Based Software Testing refers to the use of meta-heuristics for the optimization of a task in the context of software testing. Meta-heuristics can solve complex problems in which an optimum solution must be found among a large amount of possibilities. The use of meta-heuristics in testing activities is promising because of the high number of inputs that should be tested. Previous studies on search based software testing have focused on the application of meta-heuristics for the optimization of structural and functional criteria. Recently, some researchers have proposed the use of SBST for mutation testing and explored solutions for the cost of application of this testing criterion.
*Objective:* The objective is to identify how SBST has been explored in the context of mutation testing, how fitness functions are defined and the challenges and research opportunities in the application of meta-heuristic search techniques.
*Method:* A systematic review involving 263 papers published between 1996 and 2014 examined the studies on the use of meta-heuristic search techniques for the optimization of mutation testing.
*Results:* The results show meta-heuristic search techniques have been applied for the optimization of test data generation, mutant generation and selection of effective mutation operators. Five meta-heuristic techniques, namely Genetic Algorithm, Ant Colony, Bacteriological Algorithm, Hill Climbing and Simulated Annealing have been used in search based mutation testing. The review addressed different fitness functions used to guide the search.
*Conclusion:* Search based mutation testing is a field of interest, however, some issues remain unexplored. For instance, the use of meta-heuristics for the selection of effective mutation operators was identified in only one study. The results have pointed a range of possibilities for new studies to be developed, i.e., identification of equivalent mutants, experimental studies and application to different domains, such as concurrent programs.



## 1. Introduction

Glover [1] was the first author to introduce the term meta-heuristic (also called generic heuristics), defining it as a set of search algorithms to be used in different problems. The search process for the optimization of problems is inspired in different contexts, as Simulated Annealing technique [2], which uses concepts of metallurgy, and Genetic Algorithm [3], which uses fundamentals of biology and genetics (e.g. crossover, mutation and evolution). The use of meta-heuristics can be justified when the internal complexity of a problem hampers the use of an exhaustive technique due to the large number of possible solutions [4].

Glover and Kochenberger [5] addressed the general process of execution of a meta-heuristic in which a search for solutions is guided by a fitness function. A fitness function is a mathematical function that assigns values to each solution present in the search space. When an exhaustive technique is applied, all solutions are visited and the best solution is returned. On the other hand, when a meta-heuristic is applied, only some solutions are visited during the search for the optimal solution. Therefore, each meta-heuristic follows a specific search process, but hopefully in a clever way, so that good solutions can be found. Since only a few solutions are visited during the search, there is no guarantee the best solution of the search space will be returned.

Studies in the area of Search Based Software Engineering (SBSE) have shown the application of meta-heuristics is promising in the software testing context. Harman et al. [6] provided data on the increase in the number of publications on SBST. According to the authors, if the trend continues, over 1700 SBST papers will have been published before the end of this decade. Some issues addressed by

* Corresponding author. Tel.: +5516981221838.
*E-mail addresses:* adamshuk@icmc.usp.br, rodolfoadamshuk@gmail.com (R.A. Silva), srocio@icmc.usp.br (S.d.R. Senger de Souza), pssouza@icmc.usp.br (P.S. Lopes de Souza).

http://dx.doi.org/10.1016/j.infsof.2016.01.017
0950-5849/

SBST are test data generation [7–11], test case selection [12–15], test case prioritization [14,16–19], functional testing [20] and non-functional testing [21–23].

Mutation testing is the most widely known criterion of the fault injection testing technique [24]. It evaluates and improves the quality of a test case set. Therefore, small syntactic modifications are inserted in the program under test and for each modification made, a modified version of the program, called mutant, is created. Modifications represent possible mistakes programmers may commit while coding a program. This criterion aims at supporting the generation of a test case set that indicates the mistakes inserted in mutants are not present in the original program, which enhances the reliability of the program. For the application of this criterion, first the original program is executed with the initial test case set. Mutants are then generated and executed with the same test set. Those that behave differently than the original program are considered dead and are no longer used in the test. The set of alive mutants is analyzed and equivalent mutants are identified. A mutant is considered equivalent when, for all test cases, it displays exactly the same behavior of the program under test. Finally, new test cases are created to kill the alive mutants. Despite the benefits of mutation testing in terms of effectiveness, some problems such as high number of mutants generated, computational cost required for their execution and high effort necessary for the identification of equivalent mutants are raised [25].

This manuscript addresses applications of meta-heuristic search techniques to mutation testing. The focus is on the identification of potential characteristics of mutation testing to which search based techniques can be applied. Each technique and the fitness functions used are detailed and possible limitations are highlighted.

The remainder of the paper is organized as follows: Section 2 provides an overview of related works in the context of mutation testing and SBST; Section 3 addresses the plan for the systematic review, including its objective and the study selection criteria; Section 4 describes the review; Section 5 discusses the results, synthesis of findings and threats to validity; the conclusions are summarized in Section 6.

## 2. Related works

The concept of Search-Based Software Testing is related to the use of a meta-heuristic optimizing search technique, as Genetic Algorithm, for the automation or partial automation a testing task [26]. The aim is the obtaining of good solutions to a problem in an acceptable time in comparison with the application of a random search, for instance. Some techniques have attempted to solve problems of mutation testing criterion. In their application, the major computational cost is related to the execution and analysis of mutants due to their large number [27]. Many techniques have been proposed for reductions in the costs and can be classified into three categories: do fewer (for reducing the number of mutants without loss of performance), do faster (for executing the mutant as fast as possible) and do smart (for splitting computational cost/avoiding complete execution/saving state information of the execution) [28]. Do fewer techniques aim at reductions in the number of mutants. The first technique proposed was Mutant Sampling [29], which generates all mutants and randomly selects a certain percentage to execute the test and discards unused ones. A disadvantage lies in the fact random mutation does not consider characteristics of the operators regarding their efficacy in revealing defects, since mutants are randomly selected.

Mathur [30] defined the Constrained Mutation approach, in which only a few mutation operators are selected for application. The objective is the obtaining of a small set of mutation operators for the generation of a subset of all possible mutants with no significant loss of efficacy. In Selective Mutation [31,32], the selection of operators to be used is related to the number of mutants generated by the operator. Operators that generate a large number of mutants are not used in the test. The concept of Sufficient Mutation Operators is addressed in [33,34]. The objective is the determination of an essential subset of mutation operators that minimizes the number of mutants and maximizes their appropriateness in relation to the total number of operators.

Hussain [35] applied the concept of clustering to mutation testing. Clustering is an unsupervised classification technique used for grouping data through recognition based on standards and similar characteristics. Mutants are generated and the clustering algorithm classifies them into different clusters according to the test cases that kill them. Each mutant that belongs to a certain cluster is killed by a similar set of test cases. In Higher Order Mutation Testing, a Higher Order Mutant (HOM) is created by the union of two or more First Order Mutants. For instance, two different mutation operators are applied to one mutant. Jia and Harman [36] defined a subsuming HOM as a mutant difficult to be killed in comparison with first order mutants that compose it.

Untch [37] employed an approach called one-op mutation in which only one mutation operator is applied. The operator chosen was SSDL (deletion operator), which removes each statement from the program, so that the tester must design test cases that make the mutant (created without a statement) behave differently from the original program. SSDL generated between 1.3% and 4.3% of mutants likely to be created, which represents performance gains (fewer mutants to be executed). Ammann et al. [38] addressed the concept of Minimal Sets of Mutants. Such sets do not contain redundant mutants (those that do not contribute to increases in the quality of the test case set). The minimal set of mutants achieves the same improvement in the quality of the test case set in comparison with the execution of a complete set.

In general, the activity of software testing requires great efforts from the tester, as selection of test data for the maximization of code coverage, prioritization of aspects during the test (for example, the unit to be tested first, the order of execution of test cases, test requirements to be initially covered, and test cases to be run during regression testing), and identification of non-executable requirements and equivalent mutants.

Ideally, such test activities should be completely automated. In this sense, the concept of Search Based Software Testing (SBST) has emerged to describe the use of meta-heuristic search techniques for the partial or total automation of the testing activity [26]. Meta-heuristic search techniques are high-level frameworks that use heuristics to find solutions to problems involving combinations of results at an acceptable computational cost [39]. They are not complete algorithms, but strategies to be adapted to specific problems. Therefore, techniques, such as Genetic Algorithm, Hill Climbing, Ant Colony and NSGA-II have been explored. Studies in this area have shown the application of SBST to the context of test data generation [40,41], functional tests [42,43], prioritization of test cases [44] and regression testing [45] is promising.

Some systematic reviews have been conducted for the identification of studies on SBST. Ali et al. [46] analyzed 68 papers in the context of test data generation. Afzal et al. [47] considered 35 papers on SBST for non functional tests (between 1996 and 2007). A more general systematic review is presented in [48], in which SBST is applied to test data generation [7–11], non-functional testing [21–23], selection of test cases [12–15], prioritization of test cases [14,16–19] and functional tests [20].