e-Informatica Software Engineering Journal, Volume 11, Issue 1, 2017, pages: 61–78, DOI 10.5277/e-Inf170103

Systematic Literature Review on Search Based Mutation Testing

Nishtha Jatana[a], Bharti Suri[b], Shweta Rani[b]

[a] *Research Scholar, USICT and Assistant Professor, Department of Computer Science and Engineering, MSIT, New Delhi, India*

[b] *USICT, GGS Indraprastha University, New Delhi, India*

nishtha.jatana@gmail.com, bhartisuri@gmail.com, shweta2610@gmail.com

Abstract

Search based techniques have been widely applied in the domain of software testing. This Systematic Literature Review aims to present the research carried out in the field of search based approaches applied particularly to mutation testing. During the course of literature review, renowned databases were searched for the relevant publications in the field to include relevant studies up to the year 2014. Few studies for the year 2015–16, gathered by performing snowball search, have also been included. For reviewing the literature in the field, 43 studies were evaluated, out of which 18 studies were thoroughly studied and analysed. The result of this SLR shows that search based techniques were applied to mutation testing primarily for two purposes, either for mutant optimisation or for test case optimisation. The future directions of this SLR suggests the application of search based techniques for other issues related to mutation testing, like, solution to equivalents mutants, generation of non-trivial mutants, multi-objective test data generation and non-functional testing.

Keywords: software testing, analysis and verification, systematic reviews and mapping studies

1. Introduction

Software testing [1] is a rigorous activity which must be done to find the errors, quality assessment and to gain insight into the state of the system. The key challenge in the process of software testing is to reduce costs and maximize benefits. Since it is extremely time-consuming and requires a lot of effort; the overall testing process needs to be optimized for testing practices. Software testing involves test planning, design, execution and evaluation, reporting and closure activities. Test design is an activity that entails the major chunk of software test activities. It includes reviewing the test basis, identifying test conditions, designing tests, evaluating them and thereby designing the test environment setup. One of the critical tasks in testing is the generation of test data. Research on test case generation has become quite prevalent in the last two decades [2–5].

Search based approaches have been applied to several optimisation problems [6]. Software test design portrayed as a well formed optimisation problem has been solved using meta-heuristic techniques [7]. The generation of test data can be automated using meta-heuristics or search based techniques using a specific fitness function to guide the search towards a potentially good solution within a search space [8].

Search based mutation testing (SBMT) works by formulating the test data generation/optimization and mutant optimization problems as search problems and applying meta-heuristics techniques to solve them. Bottaci [9] introduced the fitness function to apply optimization algorithms to kill the mutants or faulty programs. This fitness function was based on three conditions: reachability, sufficiency and necessity. These three conditions served as a base for the foundation of SBMT and are are still used for

this purpose [10]. Optimization techniques, such as Genetic Algorithm (GA), Hill Climbing (HC), Ant Colony Optimization (ACO), Bacteriological Algorithm (BA) and Immune Inspired Algorithm (IIA) were used together with mutation testing by researchers as an art of SBMT [11].

The aim of this systematic review is to address the search based approaches applied to mutation testing either for test case generation or for mutant optimization. The systematic literature review (SLR) is the foremost essential step in the research process conducted to rigorously collect, analyse and report the current literature in the field. An extensive survey by Jia and Harman [12] comprehensively addressed developments of mutation testing. [13] lists the various advancements in the areas of SBMT since the year 2009. A recent systematic mapping by Souza et al. [11] mentioned related work in the context of test data generation for mutation testing, whereas whereas this SLR focuses particularly on the search based techniques smeared with mutation testing for test data generation/-optimization and mutant optimization. A similar study on SBMT has been recently published by Silva et al. [14]. The authors explored the use of meta-heuristic in the context of mutation testing. The study is very rigorous as well as detailed and reviews the publications up to the year 2014. The studies included in this SLR were also been in the systematic review by Silva et al. This SLR primarily aims to review the studies in the field of SBMT up to the year 2014, however, a few relevant studies of the year 2015 and 2016 were also cited. These recent papers were collected using the snowballing approach [15] to find the papers published in the last two years which cited the studies included in present SLR. The study and analysis of various papers collected from a number of sources involves a great deal of research effort and time as it also encompasses the review and modification of the paper entailing s time after it is communicated to a journal. Review and modification of the paper also entails time after it is communicated to a journal. Therefore, in this paper the relevant studies up to the year 2014 were thoroughly analysed and included only few studies published in the year 2015–16. The primary studies selected for review are the ones which proposed a new technique for SBMT up to the year 2014. This SLR contains many figures and ttables to offer easy access to comprehensive knowledge on the topic. The studies selected for review were used immensely by the researchers working in the field of SBMT.

The rest of the paper is systematized as follows: Section 2 differentiates between Literature Review (LR), Systematic Mapping (SM) and Systematic Literature Review (SLR). Section 3 shows the research method followed for this SLR highlighting the research question and the used search strategy. Section 4 analyses the studies in SBMT. Section 5 presents the results and then Section 6 presents conclusion.

2. Differentiating LR, SM and SLR

Reviewing the literature in the field is a fundamental process prior to any worthwhile research. This section compares and contrasts the implications of LR, SM and SLR.

Literature review [16] – "A literature review is an objective study done to thoroughly summarize and critically analyse the available relevant research, and to enable the researcher to gather up to date information, to gain insight into the current literature that forms a basis of a goal to be achieved and also justifies the future research in that area."

Systematic mapping [17] – "A systematic mapping study provides a structure of the type of research reports and results that have been published by categorizing them. It often gives a visual summary, the map, of its results. It requires less effort while providing a more coarse-grained overview."

Systematic Literature Review [18] – "A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Individual studies contributing to a systematic review are called primary studies; a systematic review is a form a secondary study."

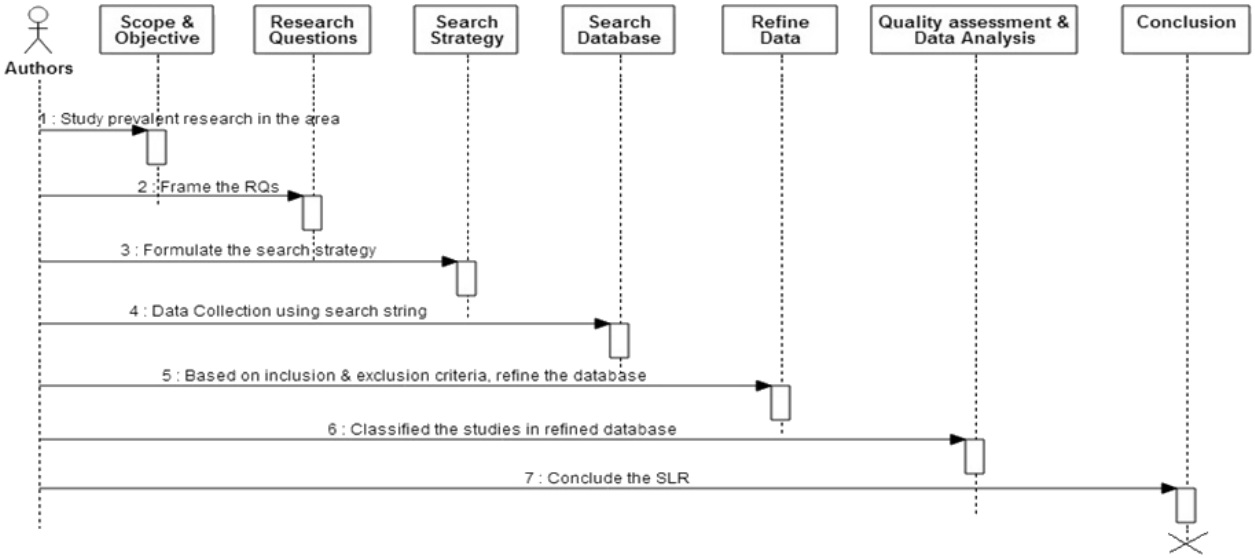

Figure 1. Undertaken course of action for SLR

SLR entails an exhaustive and comprehensive search process which adheres to the guidelines on the conduct of a review and inclusion/exclusion criteria, it is determined by the quality assessment process. However, in the case of LR the process may not be comprehensive and may or may not include the quality assessment. SM, on the other hand, has a complete search process and identifies primary and secondary research but may not include formal quality assessment process [19].

3. Research Method

The aim of this study is to offer insight into various types of research carried out in the area of search based mutation testing (SBMT). The authors followed the guidelines given by Kitchenham [18]. The undertaken course of action is demonstrated using a sequence diagram (Fig. 1).

Initially, primary knowledge about the subject of the study was collected. As a result of an inquisitive study of the research conducted in the field, a set of research questions (RQs) were formulated. Search strategy was then designed that forms the base for collecting the relevant research material from the data repositories. The most vital step was to refine the data gathered by defining an inclusion/exclusion criterion. After this stage, 43 relevant studies were categorized into 18 primary and 25 secondary studies (discussed in this section). The method used to extract the primary and secondary studies is explained in [20]. In order to answer the RQs, collected and segregated material was critically examined; thus, reaching the conclusion.

The steps trailed in the SLR for test data generation/optimization and mutant optimization using search based approaches were conducted in the way elaborated in the following subsections.

3.1. Research Questions (RQs)

As the RQs form the foundation of the SLR, the PICOC criteria [20] (given in Table 1) were used to define the research questions. The PICOC criteria were defined as follows:

- **Population** – people, or an application area, or a group of companies or any such community affected by the research
- **Intervention** – software methodologies/-tools/technologies/procedures which are required to solve a particular issue.
- **Comparison** – software methodologies/-tools/technologies/procedures with which intervention can be compared.
- **Outcomes** – factors of significance which are relevant in the study.
- **Context** – environment in which the comparison takes place.

The aim of this work is to summarize the current state of art of research in SBMT by

Table 1. PICOC criteria applied to SLR

Population	Search based Mutation Testing
Intervention	Search based/meta-heuristic techniques
Comparison	Approaches which are not meta-heuristic
Outcomes	Test data generation/optimization and mutant optimization techniques involved SBMT
Context	Within the domain of SBMT with a focus on empirical studies

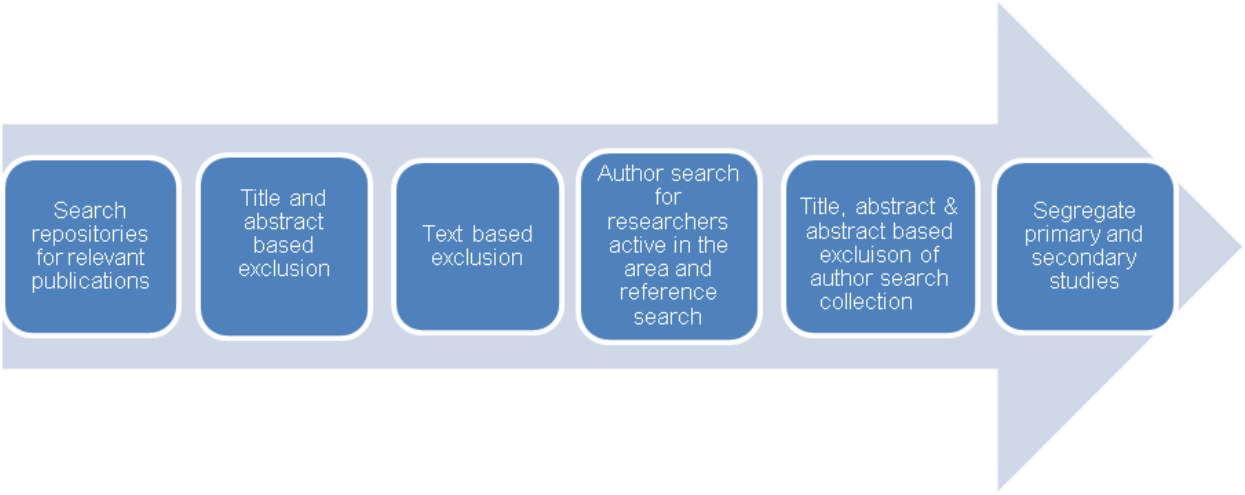

Figure 2. Inclusion/exclusion criteria

proposing answers to the set of the following research questions(RQs).
RQ1: Which search based approaches were used in collaboration with mutation testing?
RQ2: What are the areas of application in which search based approaches were applied for mutation testing?
RQ3: What are the findings from the comparison studies of the techniques used in SBMT?
RQ4: What are the major challenges faced by the researchers in the field of SBMT?

3.2. Search Process

This section describes the strategy used to mine the databases containing the research material and to extract the relevant information related to the conducted research.

3.2.1. Data Sources

According to the guidelines provided by Kitchenham [20], several data sources were searched to encompass the maximum possible information. The following data repositories were explored to retrieve the relevant publications from conference proceedings, workshop proceedings, journal articles, books and theses.

- ACM digital library (www.dl.acm.org)
- IEEE Xplore Digital Library (www.ieeexplore.ieee.org)
- Science Direct (www.sciencedirect.com)
- Springer (www.springerlink.com)
- Citeseer (scholar.google.com)
- Wiley Online Library (onlinelibrary.wiley.com)

A few papers were available in more than one searched repositories and in such cases the duplicate copies were removed manually.

3.3. Inclusion and Exclusion Criteria

The overall process depicted in Figure 2 was followed to obtain and segregate the relevant primary and secondary studies.

3.3.1. Search Strategy

Various research publication repositories provide different search options for data search. An

Table 2. Search Strings for selected Data Sources

Data Source	Search String
IEEE	((((Evolutionary OR heuristic OR search based OR search-based OR nature inspired OR nature-inspired OR optimization OR selection OR minimization OR prioritization)) AND (“Mutation Testing” OR “Mutation Analysis” OR “mutation operator testing” OR “fault injection” OR “fault based testing”))) in command search tab (under advanced search)
ACM	(Abstract:Mutation and Abstract:testing) and (Abstract:Evolutionary or Abstract:heuristic or Abstract:metaheuristic or Abstract:search based or Abstract:search-based or Abstract:nature or Abstract:inspired or Abstract:nature-inspired or Abstract:optimization or Abstract:optimisation or Abstract:selection or Abstract:minimization or Abstract:minimisation or Abstract:prioritisation or Abstract:prioritization)
Wiley	(Evolutionary OR heuristic OR metaheuristic OR search based OR search-based OR nature inspired OR nature-inspired OR optimization OR optimisation OR selection OR minimization OR minimisation OR prioritisation OR prioritization) in Abstract AND “Mutation Testing" OR “Mutation Analysis” OR “Mutants testing” OR “mutation operator testing” OR “fault injection” OR “fault based testing” in Abstract)
Elsevier	(Evolutionary OR heuristic OR metaheuristic OR search based OR search-based OR nature inspired OR nature-inspired OR optimization OR optimisation OR selection OR minimization OR minimisation OR prioritisation OR prioritization) and TITLE-ABSTR-KEY (“Mutation Testing” OR “Mutation Analysis” OR “mutation operator testing” OR “fault injection” OR “fault based testing”)
Springer	Mutation AND testing AND (Evolutionary OR heuristic OR metaheuristic OR search based OR search-based OR nature OR inspired OR nature-inspired OR optimization OR optimisation OR selection OR minimization OR minimsation OR prioritisation OR prioritization)
citeseer	((((Evolutionary OR heuristic OR metaheuristic OR search based OR search-based OR nature inspired OR nature-inspired OR optimization OR optimisation OR selection OR minimization OR minimisation OR prioritisation OR prioritization)) AND (“Mutation Testing” OR “Mutation Analysis” OR “Mutants testing” OR “mutation operator testing” OR “fault injection” OR “fault based testing”)))

advanced search strategy was used for mining the data sources to locate the pertinent publications. The Boolean operators “AND”, “OR” and “NOT” are used to arrange the keywords for forming the search string. As the various databases provide different search capabilities, as stated in a recent SLR [21, 22], conceptually similar search strings were used for each of the data sources listed in Table 2. To ensure the maximum retrieval of significant material, the search strategy was applied on to a title, an abstract and keywords. The earliest research in the field of search based on testing was published in the year 1976, and thus the start date for the search was established to be January, 1976 and the end date was set to December, 2014.

Initially using the search string as specified in section 3.2.2, a total of 4314 papers were found. However, many of these papers pertained to mutation and biology which were not relevant to our field. Thereafter, a multi-step process was followed to remove the irrelevant publications. The number of papers shown by the respective repositories using the aforesaid search string is listed in Table 3.

In order to exclude the irrelevant papers, title and abstract based exclusion was performed manually and then the full text was read to store the appropriate papers in our repository. After analysing the papers thus found, a list of active researchers (as listed in Table 4) in the same field was maintained. Then, to ensure the completeness of the data, another search

Table 3. Initial number of papers obtained by searching the data repositories

Data source	Initial Count
IEEE	201
ACM	448
Wiley	1844
Elsevier	20
Springer	1484
Citeseer	317

was accomplished to locate the leftover papers of the researchers working in the field. The reference section of the primary studies was also checked to extract any relevant publication that was missing in our collection. A total of 10 new papers were located out of which 6 were found by reference search and 4 were found by the author search. The total number of relevant papers thus collected is listed in Table 5.

Table 5. Number of papers after exclusion

Data source	Count after exclusion
IEEE	16
ACM	6
Wiley	1
Elsevier	4
Springer	10
Citeseer	1
Others	5

The totally pertinent publication includes all those papers which presented a new technique(s), compares existing techniques and empirically evaluated the techniques. These 43 studies were then thoroughly reviewed by each author to segregate them into primary and secondary studies. The papers with a significant contribution in the field in terms of research ideas and development were chosen as primary studies and the rest were marked as secondary ones. Eventually, 18 primary studies (as shown in Table 6) were then thoroughly, analysed separately by each author. It can be seen that 50% of the studies demonstrated their technique in a theoretical manner and the remaining 50% evaluated their technique empirically.

Table 4. List of authors actively working in the field of SBMT

Authors	Total publications in SBMT
P. May	4
K. Ayari	1
M. Harman	6
M. Papadakis	5
G. Fraser	3
B. Baudry	5
M. Rad	2

There are a few papers which were not been included in this SLR that were included in the recent SLR on SBMT [14]. The papers encompass [39–47]. These papers are relevant to the field of SBMT but their contribution to the field is not novel as the techniques they proposed are already used by researchers in the field of SBMT.

The papers published in 2015 and 2016 that are relevant to the field of SBMT were collected by snowballing. They are listed in the following section.

4. Analysis of studies in SBMT

This section presents the analysis, trend patterns and the discussion of the research done in the field of SBMT.

4.1. Trends in SBMT

Table 7 and Table 8 list the publication types for the selected studies and primary studies retrieved from the repositories. Figure 3 shows the publication trends observed from 1998 to 2014. The first publication was recorded in 1998 and after that the research was carried out continuously in this field. A decline in the research trends is observed during the year 2006–09.

4.2. Discussion on Primary Studies

Table 9 summarizes the contribution of the selected primary studies and search-based techniques evolved/used by them. Table 10 lists the subject programs, language and tools used by the selected primary studies. It is observed that

Table 6. Overview of the selected primary studies

Study ID	First Author	Year	Type (C/J/W/B)	Publisher	Research Category (E/T)	References
P1	L. Bottaci	2001	C	Others	T	[9]
P2	B. Baudry	2001	W	Springer	E	[23]
P3	B. Baudry	2002	C	IEEE	E	[24]
P4	P. May	2003	B	Springer	T	[25]
P5	M. C. F.P. Emer	2003	J	Others	E	[26]
P6	K. Adamopoulos	2004	C	Springer	T	[27]
P7	Md. M. Masud	2005	C	IEEE	T	[28]
P8	K. Ayari	2007	C	ACM	E	[29]
P9	Y. Jia	2008	C	IEEE	E	[30]
P10	K. K. Mishra	2010	C	IEEE	T	[31]
P11	B. Schwarz	2011	C	IEEE	T	[32]
P12	M. Harman	2011	C	ACM	T	[33]
P13	J.J. Dominguez-Jimenez	2011	J	Elsevier	E	[34]
P14	G. Fraser	2012	J	IEEE	E	[35]
P15	A. A. L. de Oliveira	2013	C	IEEE	E	[36]
P16	M. B. Bashir	2013	C	IEEE	E	[37]
P17	P. S. Yiasemis	2013	C	Springer	T	[38]
P18	M. Papadakis	2013	J	Springer	T	[10]

Type: C – Conference, W – Workshop, B – Book, J – Journal
Research Category: E – Experimental, T – Theoretical

Table 7. Publication type for 43 selected studies

Publication type	Number
Journal	13
Workshop	3
Thesis	1
Conference	23
Book	3

Table 8. Publication type for primary studies

Publication type	Number
Journal	4
Workshop	1
Book	1
Conference	12

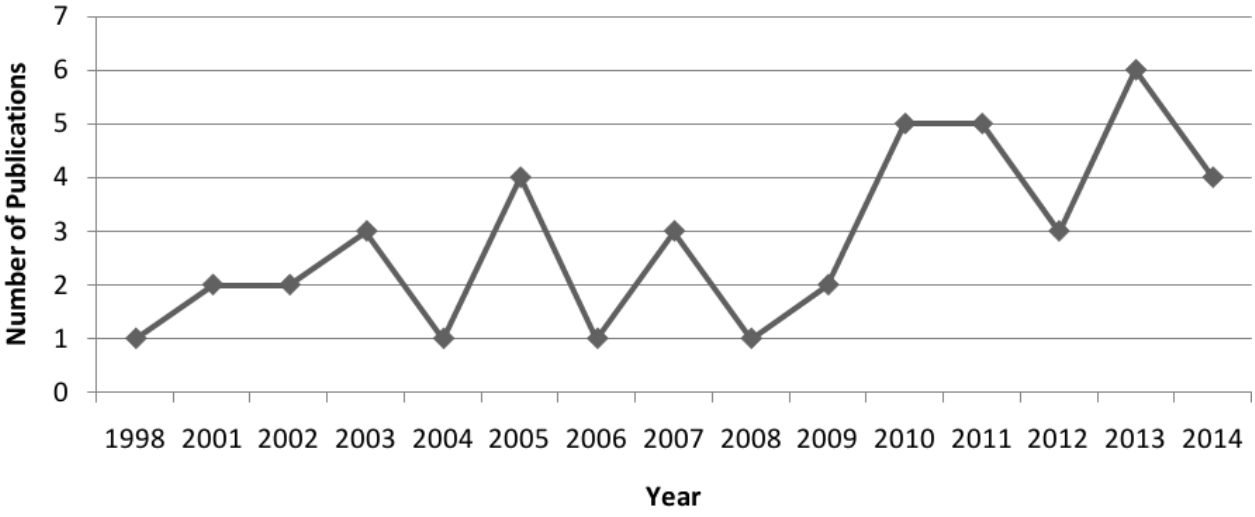

Figure 3. Publication trends of research in SBMT

Table 9. Contribution and techniques used by primary studies

Study no.	Contribution	Techniques	References
P1	Fitness function introduced based on GA for mutation testing	GA	[9]
P2	Automation of test case enhancement for object oriented software components	GA	[23]
P3	Introduced bacteriological algorithm and compared with GA	GA,BA	[24]
P4	Artificial Immune System applied for Mutation testing	AIS	[25]
P5	GP based procedure for selection and evaluation of test data	GA	[26]
P6	Dealt with equivalent mutant problem using GA	GA	[27]
P7	Test case generation for killing mutants in program units using GA	GA	[28]
P8	Test case generation using mutation testing with ACO	ACO	[29]
P9	Construction and evaluation of higher order mutants using GA, HC, Greedy	GA, HC, Greedy	[30]
P10	Elitist genetic algorithm applied with mutation testing	GA	[31]
P11	Generation of high impact mutants avoiding equivalent mutants using GA	GA	[32]
P12	Production of test input via strong higher order mutants	Search based	[33]
P13	Mutant optimization using GA for WS-BPEL	GA	[34]
P14	Test oracle generation using automated mutation testing, assertions and GA	GA	[35]
P15	Test case and mutant optimization using GA	GA	[36]
P16	Definition of a new fitness function aiming to produce test data	GA	[37]
P17	Automatically finding and correcting faults using code slices, GA and mutation testing	GA	[38]
P18	Improved fitness function for test generation using AVM	HC	[10]

Java programs are most popular for the research and there are different tools that are used by the researchers in the field of SBMT.

Figure 4 describes the yearly distribution of the techniques used by the selected primary studies. As is evident from the scatter chart, GA is the most prevalent and consistently used technique by the researchers working in the area of SBMT. Study P12 is not included in the chart, as it evaluates the application of search-based techniques in general (no specific technique) for higher-order mutation testing.

4.3. Relevant publications of 2015–2016

In order to collect publications in the domain of SBMT for the years 2015–16, the snowball approach was followed. The relevant studies collected asa result are cited here.

A few reviews relevant to the ones filed in SBMT have been published in the past two years. The literature review by Silva et al. [14] details the work carried out in the field before 2014. Other surveys include [13, 48–51].

Considerable work was done in the context of search based higher order mutation testing by eminent researchers in the field. The relevant publications include [52–56].

GA is still the most popular search algorithm used by the researchers of the domain [52, 57–59].

Hill Climbing [60] and PSO [61] have also been recently used by researchers for test data generation using SBMT. Other relevant publications include [62, 63].

5. Results (RQs)

This section presents answers to the RQs formulated above after analysing the 43 studies in SBMT, amongst which 18 studies that were identified as those stating a new technique, were thoroughly analysed.

Table 10. Language used, subject programs and tools used by (Experimental) primary studies

Study no.	Contribution	Techniques	References
P2	*	Pylon library	uSlayer
P3	C#	C# parser	*
P5	C++	cmm, fat, max, cmd	GPTesT
P8	Java	Triangle, Nextdate	muJava
P9	C	triangle, TCAS, schedule2, totinfo, printtokens, space	MILU
P13	WS-BPEL	WS-BPEL compositions (Travel Reservation, Service Extended Meta Search and Loan Approval Extended)	GAmera
P14	Java	commons CLI, commons codec, commons collections, commons logging, commons Math, commons Primitives, Google Collections, JGraphT, Joda Time, NanoXML	muTEST
P15	*	Bisect, Bub, Fourballs, Mid, trityp	*
P16	Object oriented	CGPA calculation	*
P17	Java	credit card validator, triangle classification, Base 64, Person sorted list, shapes, order set, Graph shortest Path, 3 Eclipse libraries	Kaveri/Indus
P18	Java	Triangle, Trityp, Triangle, Remainder, Calender, Fourballs, Cancel, Quadratic	*

* indicates the data not mentioned by the authors concerned in the study

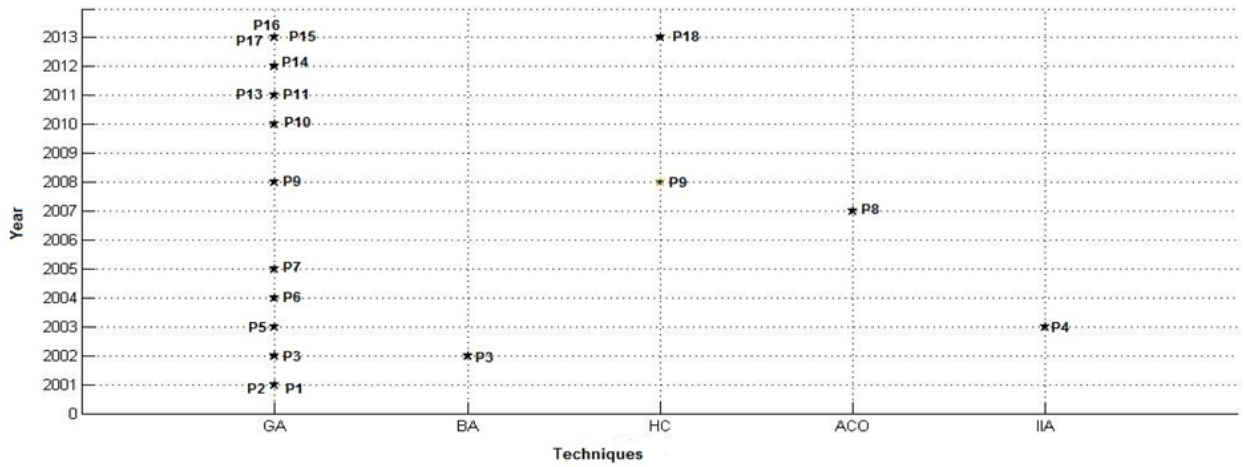

Figure 4. Corroboration of selected primary studies

5.1. Techniques of SBMT (RQ1)

In order to answer RQ1, the techniques used by the researchers working in the field are summarised below.

5.1.1. Hill Climbing (HC)

It is one of the meta-heuristic search based techniques that strives to improve the current solution by exploring all its neighbours in the search space. It includes an initialization stage, where the initial candidate solution is chosen randomly. Thereafter, all its neighbours are searched and evaluated until no improved solution can be found. The major drawback of this technique is local convergence as search includes the neighbourhood space only [64]. Jia and Harman [30] applied hill climbing with GA to deal with the explosion of a large number of higher order mutants created from lower order mutants. Papadakis and Malveris [10] used Alternating Variable Method, which is a variant of Hill Climbing, for test data generation using mutation testing by using an improved fitness function.

5.1.2. Ant Colony Optimisation (ACO)

ACO is a metaheuristic technique proposed by Dorigo in the late 1990s [65,66] for approximately solving the combinatorial problems which are otherwise hard to solve in a reasonable amount of time. It was inspired by Ant System [67] in 1991. ACO starts by mapping the problem under study such that it simulates the stigmergic behaviour of ants. The ants start at random first to locate a food source and then they lay a chemical compound known as pheromone on the most pertinent paths. The ants thereafter follow the pheromone trails to reach the desired destination. ACO was applied to obtain approximate solutions to many optimisation problems [68]. ACO finds its application in the domain of software testing [69]. Ayari et al. [29] proposed and applied ACO for automatic test data generation in the context of mutation testing and empirically proved that ACO gives better results as compared to Hill Climbing and the random approach.

5.1.3. Genetic Algorithm (GA)

The idea of GA was initially proposed by Fraser [70] and Bremermann [71] although it was popularised by Holland and his students because of applying its concepts in the field of computer science [72]. GA starts with randomly generated populations which are then evolved using genetic operators (mutation and crossover). The evolved population is evaluated for fitness and the best performing individuals are chosen. The process continues until stopping criteria are met. GA finds its application in the software testing domain for test data generation, mutant optimisation and minimisation of test data [73].

GA was used by various researchers working in the area of SBMT, but the conducted research is varied in nature. The paragraph below presents how GA was used for the optimisation of the process of search based mutation testing in various aspects.

Bottaci [9] gave a new fitness function for GA that works in collaboration with mutation testing. The fitness function was composed of three conditions: (i) Reachability: the path followed by the execution of test case must reach the mutated statement, (ii) Necessity: the condition stated by the mutated expression must be satisfied for it to be killed and (iii) Sufficiency: the difference between the subject program and the mutated program must be propagated to the final output. This fitness function was further used by Masud et al. [28], Ayari et al. [29]. Baudry et al. [23] used mutation analysis for integration testing by enhancing the test cases using GA. Baudry also applied GA for test case optimization for C# parser [24]. Emer et al. [26] developed a tool named GPTest for C++ programs for the selection and assessment of test data using fault based testing. GA was also used to address the problem of equivalent mutants in mutation testing by Adampolous et al. [27]. Masud et al. [28] proposed a model to apply GA for exposing faults by splitting the subject program into small units, hence reducing the cost and execution time of test cases. Jia and Harman [30] introduced the paradigm of higher order mutation testing and applied GA for optimization and selection amongst the huge number of higher order mutants (HOMs) from the lower order ones. Later, HOMs were extensively explored in [74]. Omar et al. [39, 40] introduced search algorithms to find subtle HOMs and concluded that local search performed better than GA and the random search in finding HOMs for Java and AspectJ subject programs. Elitist GA was used by Mishra [31] for the generation of efficient test data by selecting those test cases which have already killed a large set of mutants. Dominguez-Jimenez [34] used GA for a reduction in mutants which eventually derive test cases for improving the quality of initial test cases for WS-BPEL compositions. Fraser et al. [35] also optimised test cases using mutation analysis and GA and evaluated them on 10 open source libraries. GA has also been used by [36–38] for the optimisation of the process of mutation testing.

5.1.4. Bacteriological Algorithm (BA)

BA was proposed by Baudry et al. [24, 75, 76] and is inspired by the bacteriological adapta-

tion. This technique was particularly recommended for improving the test cases using mutation testing. It takes a set of test cases as input and makes small changes in them calling them bacteria. These bacteria are evaluated for fitness using a memorization function. The technique evolved from GA and it differs from it by two key points: introduces the memorization function for the evaluation of fitness and suppresses the crossover operator which was the cause of slow convergence in population evolution in GA.

5.1.5. Immune Inspired Algorithm (IIA)

Immune inspired algorithms are a class of intelligent algorithms which uses the primciples associated with the immune system of vertebrates. The characteristics exploited here are related to learning and memory and they are used to solve a problem. May et al. [25] applied these characteristics for optimization of test cases and mutants. May et al. further worked on this approach [77, 78] and successfully mapped the evolutionary approach to the process of the optimised generation of test cases and mutants. The process starts by choosing initial test cases called antibodies which are evolved through multiple iterations by seeking those which are capable of killing more mutants that are referred to as antigens. The mutation score is denoted by Affinity. The following chart shows the search-based techniques used by researchers in SBMT. It clearly shows that the Genetic Algorithm is the one which is the most frequently used by the authors.

5.2. Applications of SBMT (RQ2)

After the analysis of the 18 primary studies selected for the research, it was observed that SBMT found its application as shown below in Figure 6. Amongst the 18 primary studies, 9 were identified for their work in test case generation, 3 worked on test case optimization, 5 worked on mutant optimisation and the 2 remaining ones contributed to both test case and mutant optimisation.

5.2.1. Test Data Generation (TDG)

The process of TDG [2,5] using mutation testing begins with the generation of a set of mutants and the execution of the test suite on them. Then the required test suite then iteratively collects those test cases which can kill the mutants. Since this process is computationally expensive and time consuming, search based approaches [7, 8] were applied to automate this process. As is evident from Figure 6, nine studies representing TDG were conducted. Out of these nine studies, six [9, 23, 28, 31, 35, 37] are based on GA. Ayari et al. [29] applied ACO and Papadakis et al. [10] used a variant of HC for generating test data using the process of mutation testing. The symbolic execution and concolic testing were also suggested for test data generation by making use of weak mutation testing [79]. Harman et al. [33] evaluates the application of the search-based technique in general (not any specific technique) for higher-order mutation testing.

5.2.2. Test case optimization/minimization/prioritization (TCO)

The search based techniques were used for the optimized reduction or prioritization of test cases using mutation testing wherein the test cases that are capable of killing maximum mutants are considered at first hand, so that the maximum number of faults is covered with the minimum number of test cases. The search based techniques guide this search process in promising directions yielding optimum results in less computational time. As shown in Figure 6, three studies [23, 24, 26] contributing towards TCO were identified. GA were used in [23, 26] for test case optimisation and [76] for the comparison and evaluation of GA with BA for purpose of the optimisation of test cases.

5.2.3. Mutant Optimization (MTO)

The execution of a large number of mutants requires enormous computational efforts. Owing to this, a large number of program mutants

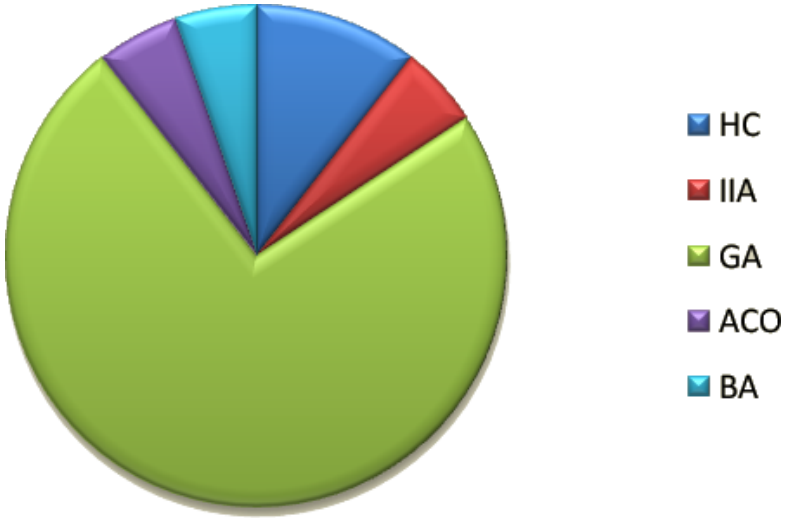

Figure 5. Usage of Search based techniques

needs to be optimised to obtain a reduced set of mutants to that they are capable of detecting the majority of significant plausible faults in the program under test. Figure 6 depicts five studies [27, 30, 32, 34, 38] found to implement the techniques for MTO. GA was predominantly adopted to obtain the optimised set of mutants [27, 32, 34, 38]. Jia et al. [30] evaluated GA with HC and the Greedy Approach for creation of higher order mutants.

5.2.4. Mutant and Test Case Optimisation (TCO_MTO)

Some researchers proposed approaches to optimise mutants as well as test cases using search based techniques with mutation testing. May et al. [25] proposed the AIS technique and Oliveira et al. [36] used GA for the purpose of mutant and test case optimisation. The figure below depicts the categorization of primary studies in the domain of SBMT.

Figure 6 shows that despite the fact that test case generation is a domain containing more studies, most approaches in this domain were concentrated on applying the Genetic Algorithm (6 studies). Mutant optimization is another prominent area in which researchers were specially interested in, and here also GA was actively applied (6 studies). Basically, SBMT relies on the fitness function which searches for candidate solutions and on mutation testing which is mainly a quality assessment technique. Both the fitness function and the quality assessment criteria determine the level to which the approach satisfies the analysed problem at hand. Thus test case generation is the most worked upon area as it is test case generation is the prior requirement in the field of software testing.

5.3. Findings from Comparison studies (RQ3)

Ayari et. al [29] proposed ACO and compared it with GA,HC and Random Search algorithms. Their preliminary results on two small programs (Triangle and NextDate) show that ACO outperformed the other algorithms in terms of mutation score and the convergence factor. Baudry et. al [23, 24, 76] introduced BA which works on a similar principle as GA, but differs from it by memorizing the efficient test cases across generations. The authors compared their work [75] with GA on 32 classes of C#. They state that BA is more stable than GA and converges faster. Jia and Harman [80] generated subsuming higher order mutants using search based techniques (GA, HC) and the greedy approach. The subsuming higher order mutants are those which are difficult to kill (detect). The authors applied the techniques (GA, HC and Greedy Approach) on 10 programs and stated that GA is the most efficient among them for the generation of subsuming higher order mutants. May et. al [77] proposed

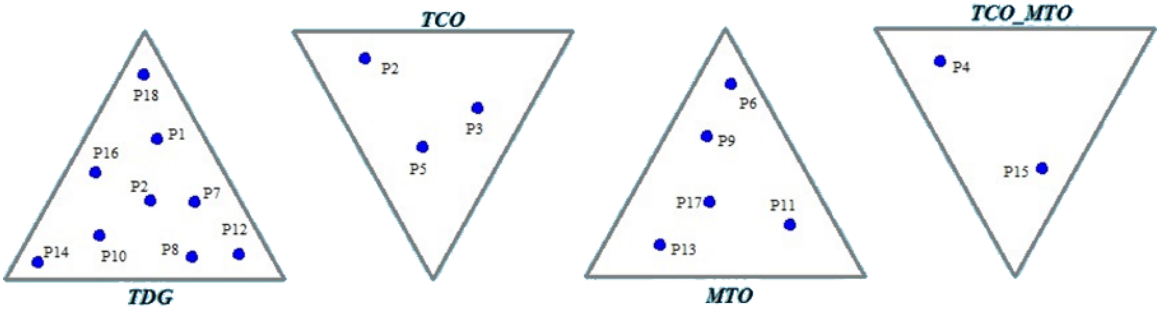

Figure 6. Categorization of primary studies according to SBMT application

a new approach (Immune Inspired Algorithm) for evolution of test data using mutation testing adequacy criteria and evaluated the efficiency of their approach on 4 programs compared with the Genetic Algorithm. They state that their approach gives better results in comparison with the others. Researchers working in the area of SBMT proposed various approaches and most of them evaluated their approach against GA.

5.4. Major challenges faced by researchers (RQ4)

The challenges faced by researchers in the field of SBMT are significantly related to those faced by mutation testing and those in SBST in general. The challenges (i)–(iv) as listed in [81, 82] are those dealing with mutation testing in general. The challenges (v)–(vii) are in context with search based software testing [83] which is applicable to SBMT as well. The challenges (viii)–(xi) are the findings of this SLR. The following points address RQ4.

(i) Computational cost of executing a large number of mutants is very high.

(ii) Mutants generated from the traditional mutant operators may be trivial (easily killable).

(iii) Detection of equivalent mutants is a cumbersome task.

(iv) Checking the output of each test case for every mutant with that of the original program takes a significant amount of time.

(v) Work on SBMT has mainly focussed on single objective optimisation, multi-objective test data generation is yet to be achieved.

(vi) SBMT focuses on unit structural test data generation. Other areas (non-functional, etc.) of software testing remains unexplored in this field.

(vii) Tools that qualify FiFiVerify (Find Fix and Verify) challenge [83] are missing in the domain of SBMT.

(viii) Most researchers of SBMT worked on artifacts (benchmark programs) of small size and several of those are artificial examples.

(ix) Most researchers worked independently, trying to find new SBMT techniques rather than extending or enhancing the proposed approaches of other researchers.

(x) Search techniques used in SBMT were applied mostly to single order mutants, however, less work has been carried out for higher order mutants.

(xi) Updates a few tools (like MuClipse) which are still being used by researchers of SBMT are not being updated with the optimisation standards which are used nowadays in mutation testing as per latest research.

6. Conclusion

In this paper, the results from a systematic review of search-based mutation testing are presented. The following are the findings in regard to the research questions:

RQ1: In the 18 primary studies identified in this work, search-based techniques namely HC, ACO, GA, BA and IIA were empirically evaluated in the area of mutation testing. The other search-based techniques have not yet been applied in this field.

RQ2: Search-based techniques were applied to mutation testing for test data generation, selection, minimization and optimisation and also for mutant optimization.

RQ3: Most researchers who proposed a new

SBMT technique empirically evaluated their technique with GA. However, there is a lot of variance in the uniqueness of the identified search-based techniques. Some techniques may be treated as novel at the time of their publications, while others may be considered as slight variations of already existing techniques. No technique can be said to be distinctly better than another as the programs used for empirical evaluation may not be considered as strong evidence to prove the superiority.

RQ4: The challenges prevailing in the area of mutation testing are pertinent to the area of SBMT along with a few more challenges. The major ones include the effort and cost entailed in mutation testing, and thus limit its application to testing real world programs. Most techniques given by researchers are either presented in a general manner or are not sufficiently empirically evaluated to serve as a base for enabling a practitioner to choose a specific SBMT technique for given software.

The analysis of the studies collected for the 2015–16 shows that the Genetic Algorithm is still being used most frequently by the researchers in the field of SBMT. The inclination of the interest of researchers towards Higher Order Mutation testing can also be observed. Alongwith other metaheuristics, PSO is also used for test data generation using SBMT.

The findings in the area of search based mutation testing, wrapping its application domain and the used techniques are covered here. This work identified research in the field since its evolution in the year 1976. The results of this SLR show that there is a significant research gap in the area of SBMT. Despite the fact that the research has been conducted in this area for more than 40 years, only 18 studies were segregated as primary studies out of 43 relevant studies. In addition to this, the problems in the area are still prevalent and not much work has been carried out to resolve them. Few techniques applied in the area of SBMT have been applied and tested on small programs and thus may not be scalable to the industrial needs and real software. Researchers worked independently rather than working collaboratively towards the elimination of open problems in the area. As a result, a lot of research can be carried out in the area including the work on the feasibility analysis of SBMT, applying SBMT to other programming languages, effective approaches for test data generation using SBMT, complexity analysis of approaches used in SBMT and reduction in the overall cost of SBMT.

References

[1] G.J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 2011.

[2] J. Edvardsson, "A survey on automatic test data generation," in *Proceedings of the 2nd Conference on Computer Science and Engineering*, 1999, pp. 21–28.

[3] M. Prasanna, S. Sivanandam, R. Venkatesan, and R. Sundarrajan, "A survey on automatic test case generation," *Academic Open Internet Journal*, Vol. 15, No. part 6, 2005.

[4] H. Tahbildar and B. Kalita, "Automated software test data generation: Direction of research," *International Journal of Computer Science and Engineering Survey*, Vol. 2, No. 1, 2011, pp. 99–120.

[5] S. Anand, E.K. Burke, T.Y. Chen, J. Clark, M.B. Cohen, W. Grieskamp, M. Harman, M.J. Harrold, P. McMinn *et al.*, "An orchestrated survey of methodologies for automated software test case generation," *Journal of Systems and Software*, Vol. 86, No. 8, 2013, pp. 1978–2001.

[6] C.A.C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information systems*, Vol. 1, No. 3, 1999, pp. 269–308.

[7] P. McMinn, "Search-based software test data generation: A survey," *Software Testing Verification and Reliability*, Vol. 14, No. 2, 2004, pp. 105–156.

[8] P. McMinn, "Search-based software testing: Past, present and future," in *IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2011, pp. 153–163.

[9] L. Bottaci, "A genetic algorithm fitness function for mutation testing," in *Proceedings of the SEMINALL-workshop at the 23rd International Conference on Software Engineering*, Toronto, Canada, 2001.

[10] M. Papadakis and N. Malevris, "Searching and generating test inputs for mutation testing," *SpringerPlus*, Vol. 2, No. 1, 2013, p. 1.

[11] F. Souza, M. Papadakis, V.H. Durelli, and M.E. Delamaro, "Test data generation techniques for mutation testing: A systematic mapping," *Proceedings of the 11th ESELAW*, 2014, pp. 1–14.

[12] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Transactions on Software Engineering*, Vol. 37, No. 5, 2011, pp. 649–678.

[13] N. Jatana, S. Rani, and B. Suri, "State of art in the field of search-based mutation testing," in *4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*. IEEE, 2015, pp. 1–6.

[14] R.A. Silva, S. do Rocio Senger de Souza, and P.S.L. de Souza, "A systematic review on search based mutation testing," *Information and Software Technology*, 2016.

[15] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014, p. 38.

[16] P. Cronin, F. Ryan, and M. Coughlan, "Undertaking a literature review: A step-by-step approach," *British Journal of Nursing*, Vol. 17, No. 1, 2008, p. 38.

[17] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering*, Vol. 17, No. 1. sn, 2008.

[18] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Keele University, Keele, Staffs, UK, Joint Technical Report TR/SE-0401, 2004. [Online]. http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/0/715071a8011d4c2f482577a700386d3a/$FILE/10.1.1.122.3308[1].pdf

[19] M.J. Grant and A. Booth, "A typology of reviews: An analysis of 14 review types and associated methodologies," *Health Information and Libraries Journal*, Vol. 26, No. 2, 2009, pp. 91–108.

[20] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University & University of Durham, EBSE Technical Report EBSE 2007-01, 2007.

[21] W. Orzeszyna, L. Madeyski, and R. Torkar, Protocol for a systematic literature review of methods dealing with equivalent mutant problem. [Online]. http://madeyski.e-informatyka.pl/download/slr/EquivalentMutantsSLRProtocol.pdf

[22] L. Madeyski, W. Orzeszyna, R. Torkar, and M. Jozala, "Overcoming the equivalent mutant problem: A systematic literature review and a comparative experiment of second order mutation," *IEEE Transactions on Software Engineering*, Vol. 40, No. 1, 2014, pp. 23–42.

[23] B. Baudry, V. Le Hanh, J.M. Jézéquel, and Y. Le Traon, "Trustable components: Yet another mutation-based approach," in *Mutation testing for the new century*. Springer, 2001, pp. 47–54.

[24] B. Baudry, F. Fleurey, J.M. Jézéquel, and Y. Le Traon, "Genes and bacteria for automatic test cases optimization in the .NET environment," in *13th International Symposium on Software Reliability Engineering*. IEEE, 2002, pp. 195–206.

[25] P. May, K. Mander, and J. Timmis, "Software vaccination: An artificial immune system approach to mutation testing," in *International Conference on Artificial Immune Systems*. Springer, 2003, pp. 81–92.

[26] M.C.F. Emer and S.R. Vergilio, "Selection and evaluation of test data based on genetic programming," *Software Quality Journal*, Vol. 11, No. 2, 2003, pp. 167–186.

[27] K. Adamopoulos, M. Harman, and R.M. Hierons, "How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution," in *Genetic and evolutionary computation conference*. Springer, 2004, pp. 1338–1349.

[28] M. Masud, A. Nayak, M. Zaman, and N. Bansal, "Strategy for mutation testing using genetic algorithms," in *Canadian Conference on Electrical and Computer Engineering*. IEEE, 2005, pp. 1049–1052.

[29] K. Ayari, S. Bouktif, and G. Antoniol, "Automatic mutation test input data generation via ant colony," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 1074–1081.

[30] Y. Jia and M. Harman, "Constructing subtle faults using higher order mutation testing," in *Eighth IEEE International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2008, pp. 249–258.

[31] K. Mishra, S. Tiwari, A. Kumar, and A. Misra, "An approach for mutation testing using elitist genetic algorithm," in *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, Vol. 5. IEEE, 2010, pp. 426–429.

[32] B. Schwarz, D. Schuler, and A. Zeller, "Breeding high-impact mutations," in *IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2011, pp. 382–387.

[33] M. Harman, Y. Jia, and W.B. Langdon, "Strong higher order mutation-based test data generation," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. ACM, 2011, pp. 212–222.

[34] J.J. Domínguez-Jiménez, A. Estero-Botaro, A. García-Domínguez, and I. Medina-Bulo, "Evolutionary mutation testing," *Information and Software Technology*, Vol. 53, No. 10, 2011, pp. 1108–1123.

[35] G. Fraser and A. Zeller, "Mutation-driven generation of unit tests and oracles," *IEEE Transactions on Software Engineering*, Vol. 38, No. 2, 2012, pp. 278–292.

[36] A.A.L. de Oliveira, C.G. Camilo-Junior, and A.M. Vincenzi, "A coevolutionary algorithm to automatic test case selection and mutant in mutation testing," in *IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 829–836.

[37] M.B. Bashir and A. Nadeem, "A fitness function for evolutionary mutation testing of object-oriented programs," in *9th International Conference on Emerging Technologies (ICET)*. IEEE, 2013, pp. 1–6.

[38] P.S. Yiasemis and A.S. Andreou, "Locating and correcting software faults in executable code slices via evolutionary mutation testing," in *International Conference on Enterprise Information Systems*. Springer, 2012, pp. 207–227.

[39] E. Omar, S. Ghosh, and D. Whitley, "Comparing search techniques for finding subtle higher order mutants," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1271–1278.

[40] E. Omar, S. Ghosh, and D. Whitley, "Constructing subtle higher order mutants for Java and AspectJ programs," in *IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2013, pp. 340–349.

[41] Y.M.B. Ali and F. Benmaiza, "Generating test case for object-oriented software using genetic algorithm and mutation testing method," *International Journal of Applied Metaheuristic Computing (IJAMC)*, Vol. 3, No. 1, 2012, pp. 15–23.

[42] A.S. Banzi, T. Nobre, G.B. Pinheiro, J.C.G. Árias, A. Pozo, and S.R. Vergilio, "Selecting mutation operators with a multiobjective approach," *Expert Systems with Applications*, Vol. 39, No. 15, 2012, pp. 12 131–12 142.

[43] B. Baudry, V. Le Hanh, J.M. Jézéquel, and Y. Le Traon, "Building trust into oo components using a genetic analogy," in *11th International Symposium on Software Reliability Engineering*. IEEE, 2000, pp. 4–14.

[44] S. Subramanian and R. Natarajan, "A tool for generation and minimization of test suite by mutant gene algorithm," *Journal of Computer Sciences*, Vol. 7, No. 10, 2011, pp. 1581–1589.

[45] K.T. Le Thi My Hanh and N.T.B. Tung, "Mutation-based test data generation for simulink models using genetic algorithm and simulated annealing," *International Journal of Computer and Information Technology*, Vol. 3, No. 04, 2014, pp. 763–771.

[46] N.T. Binh, K.T. Tung *et al.*, "A novel test data generation approach based upon mutation testing by using artificial immune system for Simulink models," in *Knowledge and Systems Engineering*. Springer, 2015, pp. 169–181.

[47] L.T.M. Hanh, N.T. Binh, and K.T. Tung, "Applying the meta-heuristic algorithms for mutation-based test data generation for Simulink models," in *Proceedings of the Fifth Symposium on Information and Communication Technology*. ACM, 2014, pp. 102–109.

[48] B.N. Thanh and T.K. Thanh, "Survey on mutation-based test data generation," *International Journal of Electrical and Computer Engineering*, Vol. 5, No. 5, 2015.

[49] M. Patrick, "Metaheuristic optimisation and mutation-driven test data generation," in *Computational Intelligence and Quantitative Software Engineering*. Springer, 2016, pp. 89–115.

[50] F. Popentiu-Vladicescu and G. Albeanu, "Nature-inspired approaches in software faults identification and debugging," *Procedia Computer Science*, Vol. 92, 2016, pp. 6–12.

[51] M. Dave and R. Agrawal, "Search based techniques and mutation analysis in automatic test case generation: A survey," in *IEEE International Advance Computing Conference (IACC)*. IEEE, 2015, pp. 795–799.

[52] Y. Jia, F. Wu, M. Harman, and J. Krinke, "Genetic improvement using higher order mutation," in *Proceedings of the Companion Publication of the Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 803–804.

[53] Q.V. Nguyen and L. Madeyski, "Searching for strongly subsuming higher order mutants by applying multi-objective optimization algorithm," in *Advanced Computational Methods for Knowledge Engineering*. Springer, 2015, pp. 391–402.

[54] Q.V. Nguyen and L. Madeyski, "Higher order mutation testing to drive development of new test cases: An empirical comparison of three strategies," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2016, pp. 235–244.

[55] Q.V. Nguyen and L. Madeyski, "Empirical evaluation of multiobjective optimization algorithms searching for higher order mutants," *Cybernetics and Systems*, 2016.

[56] F. Wu, M. Harman, Y. Jia, and J. Krinke, "HOMI: Searching higher order mutants for software improvement," in *International Symposium on Search Based Software Engineering*. Springer, 2016, pp. 18–33.

[57] A. Estero-Botaro, A. García-Domínguez, J.J. Domínguez-Jiménez, F. Palomo-Lozano, and I. Medina-Bulo, "A framework for genetic test-case generation for ws-bpel compositions," in *IFIP International Conference on Testing Software and Systems*. Springer, 2014, pp. 1–16.

[58] C.P. Rao and P. Govindarajulu, "Genetic algorithm for automatic generation of representative test suite for mutation testing," *International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 15, No. 2, 2015, p. 11.

[59] S. Rani and B. Suri, "An approach for test data generation based on genetic algorithm and delete mutation operators," in *Second International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, 2015, pp. 714–718.

[60] F.C.M. Souza, M. Papadakis, Y. Le Traon, and M.E. Delamaro, "Strong mutation-based test data generation using hill climbing," in *Proceedings of the 9th International Workshop on Search-Based Software Testing*. ACM, 2016, pp. 45–54.

[61] N. Jatana, B. Suri, S. Misra, P. Kumar, and A.R. Choudhury, "Particle swarm based evolution and generation of test data using mutation testing," in *International Conference on Computational Science and its Applications*. Springer, 2016, pp. 585–594.

[62] N.T. Binh, K.T. Tung *et al.*, "A novel fitness function of metaheuristic algorithms for test data generation for Simulink models based on mutation analysis," *Journal of Systems and Software*, Vol. 120, 2016, pp. 17–30.

[63] N. Jatana, B. Suri, P. Kumar, and B. Wadhwa, "Test suite reduction by mutation testing mapped to set cover problem," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. ACM, 2016, p. 36.

[64] P. McMinn, M. Harman, K. Lakhotia, Y. Hassoun, and J. Wegener, "Input domain reduction through irrelevant variable removal and its effect on local, global, and hybrid search-based structural test data generation," *IEEE Transactions on Software Engineering*, Vol. 38, No. 2, 2012, pp. 453–477.

[65] M. Dorigo and G.D. Caro, *New ideas in optimization*. McGraw-Hill Ltd., 1999, ch. The ant colony optimization meta-heuristic, pp. 11–32.

[66] M. Dorigo, G. Di Caro, and L.M. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, Vol. 5, No. 2, 1999, pp. 137–172.

[67] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[68] T. Stützle, M. López-Ibáñez, and M. Dorigo, "A concise overview of applications of ant colony optimization," *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[69] B. Suri and S. Singhal, "Literature survey of ant colony optimization in software testing," in *CSI Sixth International Conference on Software Engineering (CONSEG)*. IEEE, 2012, pp. 1–7.

[70] A.S. Fraser, "Simulation of genetic systems by automatic digital computers VI. Epistasis," *Australian Journal of Biological Sciences*, Vol. 13, No. 2, 1960, pp. 150–162.

[71] H.J. Bremermann, *The evolution of intelligence: The nervous system as a model of its environment*. University of Washington, Department of Mathematics, 1958.

[72] J.H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975.

[73] C. Sharma, S. Sabharwal, and R. Sibal, "A survey on software testing techniques using genetic algorithm," *CoRR*, 2014. [Online]. https://arxiv.org/abs/1411.1154

[74] E. Omar and S. Ghosh, "An exploratory study of higher order mutation testing in aspect-oriented programming," in *IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, 2012, pp. 1–10.

[75] B. Baudry, F. Fleurey, J.M. Jézéquel, and Y. Le Traon, "Automatic test case optimization using a bacteriological adaptation model: Application to .NET components," in *17th IEEE International Conference on Automated Software Engineering*. IEEE, 2002, pp. 253–256.

[76] B. Baudry, F. Fleurey, J.M. Jézéquel, and Y. Le Traon, "From genetic to bacteriological

algorithms for mutation-based testing," *Software Testing, Verification and Reliability*, Vol. 15, No. 2, 2005, pp. 73–96.

[77] P. May, J. Timmis, and K. Mander, "Immune and evolutionary approaches to software mutation testing," in *Artificial Immune Systems*. Springer, 2007, pp. 336–347.

[78] P. May, K. Mander, and J. Timmis, "Mutation testing: An artificial immune system approach," in *UK-Softest. UK Software Testing Workshop*. Citeseer, 2003.

[79] M. Papadakis and N. Malevris, "Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing," *Software Quality Journal*, Vol. 19, No. 4, 2011, pp. 691–723.

[80] Y. Jia and M. Harman, "Higher order mutation testing," *Information and Software Technology*, Vol. 51, No. 10, 2009, pp. 1379–1393.

[81] Y. Jia, "Higher order mutation testing," Ph.D. dissertation, University College London, 2013. [Online]. http://discovery.ucl.ac.uk/1401264/1/YuePhDFinal2013.pdf

[82] Q.V. Nguyen and L. Madeyski, "Problems of mutation testing and higher order mutation testing," in *Advanced computational methods for knowledge engineering*. Springer, 2014, pp. 157–172.

[83] M. Harman, Y. Jia, and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–12.