Article

Software Testing Based on Research: A Road Map

Omar Salim Abdulla [1, *], Bahaa Abdul Qader Thabit [2], and Ghazwan Ahmed Al-Zaidi[2]

[1] Bilad Al-Rafidain university college, Diyala 32001, Iraq; dr.omersalim@bauc14.edu.iq
[2] Institute of Graduate studies and Research, Alexandria university, Egypt; igsr.BahaaThabet@alexu.edu.eg
[3] Faculty of Information Science & Technology, UKM, Malaysia; P105672@siswa.ukm.edu.my
[*] Correspondence: Tel.: +964-7740219350

Abstract: This paper serves as a guide for both researchers and students who are new to the research area of Search-Based Package Testing. The application of metaheuristic explore methods in this context specifically pertains to utilizing these algorithms for generating test data. In the realm of software engineering research, software testing emerges as a robust and fertile ground for exploration. The integration of AI methods into software program testing is an evolving research direction. Often, newcomers to this field face challenges due to limited knowledge about the interaction between software testing and artificial intelligence. This paper aims to provide a roadmap for these new researchers or students in the field.

Keywords: SBSE; AI; Software Testing

Citation: To be added by editorial staff during production.

Academic Editor: Dr. Mohd Hasan Alwan

Received: 24/10/2023
Revised: 27/11/2023
Accepted: 5/12/2023
Published: 10/12/2023

1. Introduction

Relatively, Search-based software engineering SBSE is one of the new exploration projects filed. Where, the first publication used this term published by Mark Harman in 2001 in [1]. SBSE terms consist of two parts as the prefix "S" represent the word "Search" which refers to the Artificial Intelligence (AI) and the suffix "BSE" refers to the software engineering part. Recently, AI used widely in most of the software engineering lifecycle as an automation process such as in the software testing which is our concern in this paper.

The application of AI methods in computer software challenging involves utilizing search algorithms to generate and optimize test data for testing purposes. Just as SBSE serves as a prefix for search-based software engineering, SBST denotes the use of search algorithms in software testing. SBST can be broadly categorized into two main directions. First, is the white-box software testing and second, the black-box software testing. In white-box and some researchers address it as structural testing, the tester tests the entire of the software component one by one as the conditions statement or the loop statement. Unlike the structural testing, the black box testing or the functional testing, tester don't care about the software components. The tester in functional testing care about the software functions only by giving an input and examine the output regardless how this output generated.

This paper represents a roadmap for all the new searchers or student in the SBST research field. The remaining sections of the paper are structured as trails, an introduction about the term Search Based Software Engineering "SBSE", is represented in section 1. The most well-known software testing approaches in section 2, Search-Based Software Testing SBST discussed in section 3 and evolutionary examination data creation techniques in section 4.

2. Materials and Methods

2.1 SEARCH BASED SOFTWARE ENGINEERING (SBSE)

The term of Search Based Software Engineering SBSE was first time initiated by Mark Harman [2]. Where the word "search" refers to use the metaheuristic search techniques. The term SBSE in generally mean reformulating the problem of software engineering as search problem (optimization problem). Software engineering generally like other filed were, the software engineer looking for near optimal solution for the software engineering problem with acceptable forbearance. Hence, the problem of software engineering can be perfect application for metaheuristic search algorithm in various software engineering problem. Search based software engineering SBSE addressed various types of engineering problem according to publication of which represent comprehensive and rich content about SBSE, there is concretion about different categories and application of software engineering problem tackled as search as an optimization problem. It has been highlighted these application examples as [2]:

- Testing and debugging: examples in [3, 4].
- Requirements engineering: [5]..
- Project planning and cost estimation: [1, 6].
- Automated maintenance: [7].
- Service oriented software engineering: [8]

In addition, there are extra software engineering applications used as case studies for search-based optimization techniques can be seen in [9, 10]. It has been explained the growth in publication over time in the in search-based software engineering [11].

Recent approaches tends to achieve higher levels in terms of Search based software testing where, some recent publication discussed testing the software in terms of many testing objectives in addition to code coverage level. It has been stated a research question "However, is high code coverage alone sufficient to detect bugs effectively?". They introduced a predictive many-objective sorting algorithm (PreMOSA) that combines defect prediction information with coverage details to determine optimal areas for enhancing test coverage within the class under test (CUT).. [12]

Another tends in the SBST field is the use of automated and manual fitness function design in order to guide the search exploration toward the software failure. It has been proposed an approach based on automated and manual fitness function design for SBST technique named as ATheNA testing framework. According to the experimental test, they stated that ATheNA-S generated more failure-revealing test cases than two baseline SBST frameworks [13].

Different software testing techniques reported in the literature in the recent years. In this research article, we refers the reader to the most important articles from our overview as [14-16].

Research manuscripts reporting large datasets that are deposited in a publicly available database should specify where the data have been deposited and provide the relevant accession numbers. If the accession numbers have not yet been obtained at the time of submission, please state that they will be provided during review. They must be provided prior to publication.

Interventionary studies involving animals or humans, and other studies that require ethical approval, must list the authority that provided approval and the corresponding ethical approval code.

2.2 SOFTWARE TESTING APPROACHES

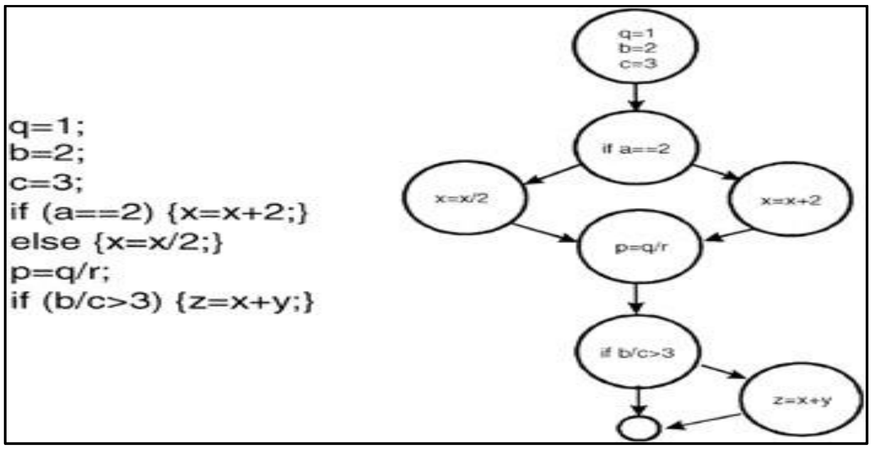
In software testing, there is some points have to be token in consideration, especially what associated with the input test cases. Where, the huge number of the possible test cases can be an input to test the software. That led the tester to decide how to test the software in order to reduce the testing cost by find minimum number of test cases satisfy the test goals. For this purpose, there is common methodologies applied in software testing. In this paper, we will address two of them as first is white box testing and the second as black box testing.

3.2.1 White (structural) Box Testing

In this approach, testing process is deriving test inputs from the internal structure of the software under test [17]. This approach treating the software as group of structure such as branches, paths and statement. Every structure considered as coverage criteria. Where, coverage criteria such as, branch coverage, statement coverage or any others established as testing goal to be satisfied. This approach concern on test a predetermined coverage criteria (branch, as example) through executing this structure wherever found.

i. Control flow graph (CFG)

Control flow graph is another representation for software where, it is describing the software as set of Nods (N) connected by Edges (E). Control flow graph for program P starts by unique starting node s and ends by another unique node as exit node x. Each node n ∈ N represent a statement in the program P. Also, each e = (ni , nj) ∈ E represent transfer between the node ni to nj . Branching nods represent the statements have decision in the program P such as (if statement, while statement...). Simple example for source code and its control flow graph shown in figure 1.



```
q=1;
b=2;
c=3;
if (a==2) {x=x+2;}
else {x=x/2;}
p=q/r;
if (b/c>3) {z=x+y;}
```

Figure 1. CFG example

ii. Coverage criteria

The idea behind identifying the coverage criteria in structural test data generation is to convert these criterions as an objective function. Mainly, the coverage criterions are statement, branch, and condition [18]. For example, if the tests goals are the branch coverage, the objective function will design to generate test data can cover as much as possible branches in the software under test.

2.3 Black (functional) Box Testing

In this approach of testing tester have no knowledge about the inner framework of the software being examined. Tester have two features can employ in testing, the test inputs and the output. Input is the sequence of test case, which can be valid, or not. The only knowledge about the validity of the test input can verified from the observation of the tester for the output result. Output of testing process (tester have the expected out form the specification of the software under test) will reflect the result of testing as shown in figure 2 [19].

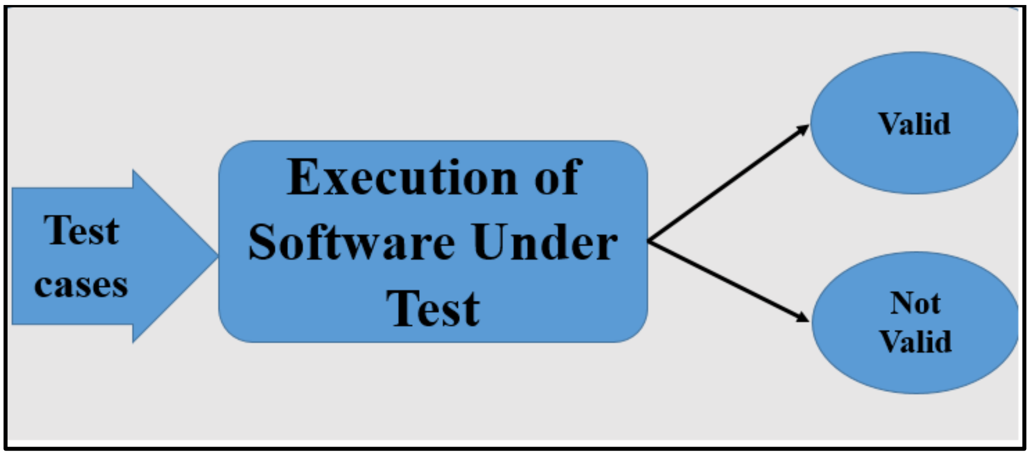


Figure 2. Black box testing scheme

4. SEARCH-BASED SOFTWARE TESTING SBST

Software testing and particularly test data generation is complex and undecidable problem especially when the software is large and complicated. Therefore, the automation of test generation is required. Search-Based Software Testing SBST is the term of using metaheuristic search algorithm in generating set of data as software test inputs. It is only an example of Search-Bases Software Engineering SBSE. The relation between the artificial intelligence search algorithms with the software testing shown in figure 3.

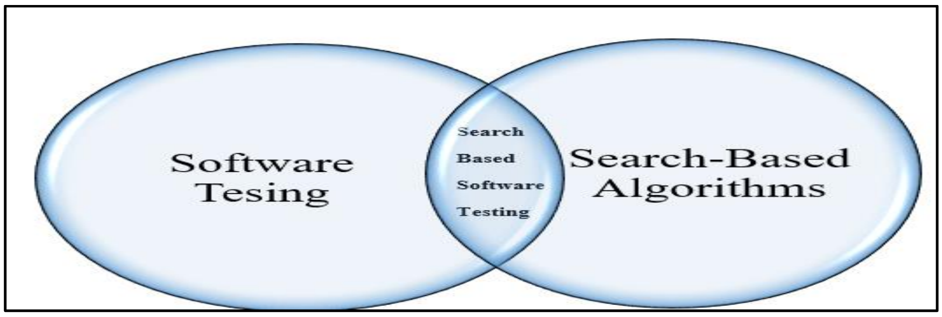


Figure 3. Relation between the artificial intelligence search algorithms with the software testing

3. Results

3.1 EVOLUTIONARY TEST DATA GENERATION TECHNIQUES

Recently, several approaches applied in order to automate the process of test data generation. An exhaustive description for the current techniques can be found in the works [20]. From one hand, our concentration in this work will be on the techniques, which consider the metaheuristic search algorithm to generate optimal set of test data in order to maximize structural code coverage. On the other hand, there are some classifications of the most widely applied according to objective functions used. In general, test generation techniques can be categorized into static and dynamic structural test data generation [21]. Figure 4 depicts the categorization of evolutionary structural test generation techniques.

5.1 Coverage-Oriented-Approach

In this approach, the idea is rewarding the solution according to the covered structures of the software under test. In [22], introduced this approach by rewarding the individuals which covering biggest number of program structures, which selected as coverage criterion. In addition, attempt to achieve full path coverage by penalizes the solutions that follow the path that already covered in order to cover the path uncovered yet.

This approach is not always efficient because it is lack for guidance because where, there is no guarantee to cover the structures, which not covered by chance [21].

a. Structure Oriented

With structure-oriented approaches, the main idea behind these approaches is to be dealing with the program structures required to cover in separates in order to obtain full coverage. The search separated for each uncovered structure independently. Based on the information utilized in this approach to guide the search through the objective function, the approach can be classified into branch-distance-oriented and control-oriented.

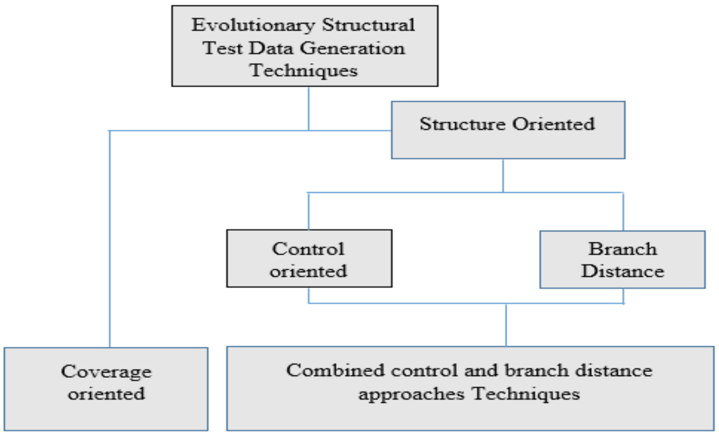

Evolutionary Structural Test Data Generation Techniques
Structure Oriented
Control oriented
Branch Distance
Coverage oriented
Combined control and branch distance approaches Techniques

Figure 4. Evolutionary Structural Test Data Generation Techniques Classification

b. Branch-Distance-Oriented

According to the information of branch predicates, objective function guides the search in order to generate set of data cover the branches uncovered yet. Where, the objective function value show how the input far from the required value which make the branch predicate true or false according to test goals. Branch distance objective function example for the branch predicate if (a<b) can be calculated by rearranging this predicate into another mathematic form (b-a). The work of [23] applied branch distance function in order to generate test data for testing software program and later many works follow the same method. The paper [24] gives explanation of similar works.

c. Control-oriented-Approach

In this approach, the objective function takes into consideration the execution of branching nodes. Branching nodes execution will executed in way that can led to execute the desired structures. The idea her is to generate set of data can exercise the nodes which control the structures which tester want to test. This approach applied in loop and branching nodes in the work [25].

d. Combined Control and Branch Distance

In this approach, the branch distance and the control information will be considering to the fitness function. In this approach, the control nodes determining the target structure are identified. The generated solution then selects one of the branch predicates in the critical branch, and the distance function is computed accordingly. The work of [26] used the objective function combined from the control nodes and branch distance to evaluate the generated test data.

5. Conclusions

Search-based software testing is a notable instance within the broader domain of search-based software engineering, representing a significant and crucial research area. In this paper, research roadmap introduced in order to guide the researchers and students in the search-based software testing research field. In the paper, essential terms and techniques highlighted to explain the main meaning of using the search algorithm in software testing activities. In the future work, we tend to be more specific in covering the research work on the structural software testing through explaining the main techniques used, the objective function and the components of the software need test.

Conflicts of Interest: “The authors declare no conflict of interest.”

References

1. Aguilar-Ruiz, J.S., et al., *An evolutionary approach to estimating software development projects.* Information and Software Technology, 2001. **43**(14): p. 875-882.
2. Harman, M. and B.F. Jones, *Search-based software engineering.* Information and software Technology, 2001. **43**(14): p. 833-839.
3. Harman, M., S.A. Mansouri, and Y. Zhang, *Search-based software engineering: Trends, techniques and applications.* ACM Computing Surveys (CSUR), 2012. **45**(1): p. 1-61.
4. Akhtar, M.F., K. Ali, and S. Sadaqat, *Factors influencing the profitability of Islamic banks of Pakistan.* International research journal of finance and economics, 2011. **66**(66): p. 1-8.
5. Bagnall, A.J., V.J. Rayward-Smith, and I.M. Whittley, *The next release problem.* Information and software technology, 2001. **43**(14): p. 883-890.
6. Kirsopp, C., M.J. Shepperd, and J. Hart, *Search heuristics, case-based reasoning and software project effort prediction.* 2002.
7. Mitchell, B.S. and S. Mancoridis, *On the evaluation of the bunch search-based software modularization algorithm.* Soft Computing, 2008. **12**: p. 77-93.
8. Canfora, G., et al. *An approach for QoS-aware service composition based on genetic algorithms*. in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. 2005.
9. Cohen, J.A., A.P. Mannarino, and V.R. Staron, *A pilot study of modified cognitive-behavioral therapy for childhood traumatic grief (CBT-CTG).* Journal of the American Academy of Child & Adolescent Psychiatry, 2006. **45**(12): p. 1465-1473.
10. Mitchison, H.M., et al., *Targeted disruption of the Cln3 gene provides a mouse model for Batten disease.* Neurobiology of disease, 1999. **6**(5): p. 321-334.
11. Harman, G., *Prince of networks: Bruno Latour and metaphysics*. 2009: re. press.
12. Vogel, T., C. Tran, and L. Grunske, *A comprehensive empirical evaluation of generating test suites for mobile applications with diversity.* Information and Software Technology, 2021. **130**: p. 106436.
13. Anand, A., et al., *Knowledge sharing, knowledge transfer and SMEs: evolution, antecedents, outcomes and directions.* Personnel review, 2021. **50**(9): p. 1873-1893.
14. Shioda, S., *Coupon subset collection problem with quotas.* Methodology and Computing in Applied Probability, 2021. **23**(4): p. 1203-1235.
15. Feldt, R. and S. Yoo. *Flexible probabilistic modeling for search based test data generation*. in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020.
16. Sarro, K.J., et al., *Seasonal variation of strength and power magnitude and asymmetry, and injury profile of Brazilian jiu-jitsu athletes.* Journal of Physical Education and Sport, 2022. **22**(6): p. 1346-1355.
17. Parry, O., et al. *Flake it'till you make it: Using automated repair to induce and fix latent test flakiness*. in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020.
18. Binkley, D., et al. *An Investigation into the Effect of Control and Data Dependence Paths on Predicate Testability*. in *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 2020. IEEE.
19. Dąbrowski, P., et al., *Photosynthetic efficiency of Microcystis ssp. under salt stress.* Environmental and Experimental Botany, 2021. **186**: p. 104459.
20. Anand, S., et al., *An orchestrated survey of methodologies for automated software test case generation.* Journal of systems and software, 2013. **86**(8): p. 1978-2001.
21. McMinn, P., *Search ‐based software test data generation: a survey.* Software testing, Verification and reliability, 2004. **14**(2): p. 105-156.
22. Roper, S., *Product innovation and small business growth: a comparison of the strategies of German, UK and Irish companies.* Small Business Economics, 1997. **9**: p. 523-537.

23. Miller, W. and D.L. Spooner, *Automatic generation of floating-point test data.* IEEE Transactions on Software Engineering, 1976(3): p. 223-226.

24. McMinn, P. *Search-based software testing: Past, present and future*. in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. 2011. IEEE.

25. Pargas, R.P., M.J. Harrold, and R.R. Peck, *Test‐data generation using genetic algorithms.* Software testing, verification and reliability, 1999. **9**(4): p. 263-282.

26. Tracey, I., et al., *Imaging attentional modulation of pain in the periaqueductal gray in humans.* Journal of Neuroscience, 2002. **22**(7): p. 2748-2752.