RESEARCH ARTICLE OPEN ACCESS

Search Based Software Engineering

Jaspreet Bedi, Kuljit Kaur
Department of Computer Science and Applications BBK DAV College for Women Amritsar, India
Department of Computer Science and Engineering, Guru Nanak Dev University, Amritsar, India

Abstract
This paper reviews the search based software engineering research and finds the major milestones in this direction. The SBSE approach has been the topic of several surveys and reviews. Search Based Software Engineering (SBSE) consists of the application of search-based optimization to software engineering. Using SBSE, a software engineering task is formulated as a search problem by defining a suitable candidate solution representation and a fitness function to differentiate between solution candidates. This paper gives an overview of major research studies undertaken in the domain.
Index Terms—GA, SBSE, MEOA, NSGA-II and SPEA2

I. INTRODUCTION

The term Search based software engineering was created by Harman and Jones in 2001.[7] The field of Search Based Software Engineering (SBSE) has grown around the need to find new ways of heuristically selecting solutions for software engineering problems. As software systems increase in size and complexity, an increasing number of tasks that are intractable to perform through manual or even automated exhaustive means are faced. In essence, the potential solution spaces for these problems are very large and often exponential in nature. This includes tasks such as generating test cases to cover specific branches or lines of code, finding optimal sequences of program refactoring, and reverse engineering a program's module structure.

Search-based software engineering (SBSE) deals mainly with application of metaheuristic search techniques like genetic algorithms, simulated annealing and tabu search to software engineering problems. Various activities in software engineering can be formulated as optimization problems. Due to the computational complexity of these problems, exact optimization techniques of operations research like linear programming or dynamic programming are mostly impractical for large scale software engineering problems. Because of this, researchers and practitioners have used metaheuristic search techniques to find near optimal or good-enough solutions.

Broadly speaking, SBSE problems can be divided into two types. The first is of the type black-box optimization, for example, assigning people to tasks (a typical combinatorial optimization problem). With this sort of problem domain, the underlying problem could have come from the software industry, but equally it could have originated from any domain where people are assigned to tasks. The second type are white-box problems where operations on source code need to be considered.

II. CHARACTERISTICS OF SBSE

Search-based software engineering (SBSE) is an approach to apply metaheuristic search techniques like genetic algorithms, simulated annealing and tabu search to software engineering problems. It is inspired by the observation that many activities in software engineering can be formulated as optimization problems. The characteristics include the following:

a. Search space: The search space consists of certain parameters which can be manipulated in order to make different candidate solutions is search space. The fitness function and search space are the two requirements for problem analysis when using SBSE.[7].
b. Multi objectiveness: The Search-Based Software Engineering (SBSE) community is increasingly recognizing the inherit "multiobjectiveness" in Software Engineering problems.[8].
c. Optimization: Historically, the field of Search-Based Software Engineering (SBSE) has seen a slow adoption of Pareto optimization techniques, generally known as multiobjective optimization techniques. SBSE consists of search-based optimization algorithms used in software engineering, with genetic algorithms, genetic programming, simulated annealing and hill climbing being the most widely used [1]. [9]
d. Types Of Algorithms: In SBSE, heuristic algorithms such as hill climbing or meta-heuristic algorithms such as simulated annealing, tabu search, genetic algorithms and ant colony optimization are used to efficiently explore the solution space.

e. Fitness Function: The problem analysis is guided by a fitness function in search space which is a measure of the quality of an individual solution. The idea is to converge on a solution that may not be optimal but that is good enough.

SBSE has been rapidly growing in its presence in the software engineering literature. According to the online SBSE repository [1] the annual number of publications in SBSE more than doubled between 2006 and 2010(from 60 to almost160). This jump in the use of SBSE is an exciting development as per authors and the stochastic nature of these algorithms means that individual runs of an algorithm may not be indicative of its effectiveness. In order to obtain reliable data and to be able to extrapolate results to valid and meaningful conclusions, it is essential to create carefully designed empirical evaluations. This involves intelligent choice of fitness function.

III. MAJOR MILESTONES

Mark Harman and bryan f. jones put forward basic introduction of concept.[1] while S. Kanmani and P. Maragathavalli [2]in search-based software test data generation using evolutionary testing techniques compared effectiveness of GA-based testing system with a Random testing system. They concluded that for simple programs both testing systems work fine but as the complexity of the program or the complexity of input domain grows, GA-based testing system significantly outperforms Random testing.

Mark Harman King's College London Strand in [3] concluded set of open problems, challenges and areas for future work while in second paper Mark Harman and Afshin Mansouri in [4] showed how SBSE can be used to help predict the performance characteristics of component-based system assemblies. The approach uses SBSE to build models using genetic programming, from which a behavioral model is formed. A combination of dynamic and static analysis is used to generate the required input for genetic programming.

Mark Harman and Afshin Mansouri, in [6] suggested that Software engineering is ideal for the application of meta heuristic search techniques. Goran Mauša et el. in article [7] presented an overview of search based software engineering (SBSE) and software defect prediction areas.

Abdel Salam Sayyad Hany Ammar [8] concluded that the SBSE field has seen a trend of adopting the Multiobjective Evolutionary Optimization Algorithms (MEOAs) that are widely used in other fields (such as NSGA-II and SPEA2) without much scrutiny into the reason why one algorithm should be preferred over the others. They also found that the majority of published work only tackled two-objective problems (or formulations of problems), leaving much to be desired in terms of exploiting the power of MEOAs to discover solutions to intractable problems characterized by many trade-offs and complex constraints. MelÓCinnéide and MyraB.Cohen in [9] concluded only introduction.

Gordon Fraser Saarland and Saarbr¨ucken in [10] presented report of the results of a large empirical analysis carried out on 20 Java projects (for a total of 1,752 public classes). Their experiments showed with strong statistical confidence that even for a testing tool that is already able to achieve high coverage, the use of appropriate seeding strategies can further improve performance. Abdel Salam Sayyad Tim Menzies Hany Ammar in [11] concluded that we need to change our methods for search based software engineering, particularly when studying complex decision spaces.

Jeremy S. Bradbury, David Kelk and Mark Green in [12] suggested open problems that may benefit from combining Search-Based Software Engineering (SBSE) techniques and software model checking. Márcio de Oliveira Barros Arilo Claudio Dias Neto in [13] discussed the threats to validity along with list of full papers published in the editions 2009 and 2010 of SBSE, a proposal of questionnaire to assess the proposed threats was included. They conducted an analysis of 23 SBSE papers using the proposed questionnaire and concluded that while conclusion threats are well addressed by current papers, the assessment of internal, external, and construct threats can be severely improved.

G. Mauša T. Galinac Grbac , B. Dalbelo Bašić in [14] used fitness function the closest thing to an artifact and optimized artifacts making SBSE. This property makes SBSE very attractive and potentially beneficial field. They concluded that search-based algorithms are attractive in software engineering also due to the fact that the data in software engineering are often inaccurate, over dispersed and incomplete, making some traditional optimization techniques inappropriate. Software testing exploited the search-based algorithms more than any other software engineering field. They showed the basic requirements and potential application for these algorithms in other optimization or multi-objective problems from various software engineering areas. Besides making use of obviously beneficial algorithms in other unexplored areas, there is also work to be done with the algorithms themselves. The emerging hybrid algorithms show very promising results and offer a potential scope for future research. To sum up, the potential of using SBSE is vast and still needs to be explored more thoroughly. They concluded that unlike other

engineering disciplines where search-based algorithms found application in, software engineering is the only discipline whose artifacts are solely virtual.

Bogdan Marculescu, Robert Feldt, and Richard Torkar in [15] proposed a search based software testing system designed to allow domain specialists with little software testing expertise to develop test cases for their applications. The value of such systems would be especially relevant for contexts were software testing experts are not available or where domain knowledge is the deciding factor in the success of the testing process. Guanzhou Lu, Rami Bahsoon and Xin Yao [16] carried out a case study to analyze the effectiveness of the proposed method, which is Sampling Hill Climbing (SHC) on a variant of the Next Release Problem (NRP). They found that if the objective function of a software engineering problem is linearly decomposable, it is possible to construct an elementary landscape and apply elementary properties to design a better algorithm for this problem. The experimental results show that SHC incorporated with elementary properties outperforms the initial algorithm. Therefore we could assume that the performance of algorithms for a particular problem could be improved by the application of the elementary landscape analysis. The future work will include exploiting other elementary properties that could be applied to the algorithms, and extending this method to more software engineering problems.

Arash Mehrmand in [17] "A Factorial Experiment on the Scalability of Search-Based Software Testing Master Thesis Software Engineering" reported on results from a factorial experiment, where the performance GA is compared to random testing in automation of software testing. In this experiment, even the process of generating SUTs were automated, which means use of GE to generate sample Java programs. He mentioned "To my knowledge, the presented results we had are not yet reported in the software test data generation area, because of having automatically generated programs, different levels of complexity and different kinds of coverage at the same time".

S. Kanmani & P. Maragathavalli [18] compared thirteen papers. They concluded that almost in all cases, the meta-heuristic search techniques have been implemented for the specific application for e.g., multi-objective NRP, Ajax web applications, triangle classification problem, software clustering problem, project resource allocation, signal generation, buffer overflow problem, network security, safety, R-T tasks and in fault prediction. In most of the combinatorial problems, they have got better results by implementing Evolutionary Algorithms such as GAs, SA, TS, GP and they compared their results with the local search such as RS, HC. The dataset used were taken from various sources. The main quality parameters considered are branch, path coverage, accuracy and fitness.

Mark Harman, Joachim Wegener [19] conducted experiments on various approaches to search based software engineering. Andrea Arcuri, Per Kristian Lehre and Xin Yao in [20] illustrated how runtime analysis can be applied in SBSE and they advocated its importance. Kiran Lakhotia, Mark Harman, Hamilton Gross[21] in AUSTIN:A tool for Search Based Software Testing for the C Language and its Evaluation on Deployed Automotive Systems developed a tool in C language.

Daniel Rodriguez, Israel Herraiz and Rachel Harrison in "On Software Engineering Repositories and Their Open Problems"[22] discussed the current data repositories that are available for Software Engineering research. They classified them and discussed some common problems faced when extracting information from them. They also discussed data related problems when applying machine learning techniques. Although some of the problems such as outliers or noise had been extensively studied in software engineering, others need further research, in particular, imbalance and data shifting from the machine learning point of view and replicability in general, providing not only the data but also the tools to replicate the empirical work.

Rakesh Roshan, Rabins Porwal and Chandra Mani Sharma in "Review of Search based Techniques in Software Testing" [23] reviewed the recent advancements in field of Search Based Software Testing. They concluded that area spawns an all new domain inning in the arena of modern Software Testing. They suggested that Search Based Software Test has many advantages including reduced efforts and improved reliability over state-of-the-art approaches of Software Testing. Gabriela Ochoa, in [24] described in the article various case studies by applying search methodologies to challenging problems in software engineering. The article also described a recent research initiative: Dynamic Adaptive Automated Software Engineering (DAASE), whose goal is to embed optimisation into deployed software to create self-optimising adaptive systems.

P. Maragathavalli in [25] provided an overview of the Search-Based Software Engineering and the Search-Based Software Testing used in test data generation. Phil McMinn[26] presented a preprint of an article accepted for publication in Software Testing. Verification and Reliability were discussed and results were obtained for various testing areas with many successful experiments undertaken using real-world examples drawn from industry.

Khalid Mahmood , Shahid Kamal , Hamid Masood Khan[27] concluded that search Based

Software Engineering (SBSE) is used for finding a near optimal solution for different software activities throughout software development life cycle. They suggested that SBSE has been applied to problems having static nature, but yet not applied to problems having dynamic nature. On the other hand Swarm Particle Intelligence techniques, such as Ant Colony Optimization (ACO) uses ants behavior and structure to find the real world problems, using AI technique to SBSE dynamic search problem will probably find the objective which yet not been achieved. Dynamic Network Routing problem using ACO will yield this objective. Results will be statistically and empirically tested and compared with other competitive studies to validate the research being proposed. Arthur I. Baars, Kiran Lakhotia, Tanja E.J. Vos and Joachim Wegene [28] presented an overview of Search–Based Testing and they discussed some of the open challenges remaining to make search–based techniques applicable to industry as well as the Future Internet.

IV. CONCLUSION

Survey of the research papers done so far suggests that there is a wide variety of the areas of software engineering which are influenced by SBSE but this is not the exhaustive list instead further research in this direction will be a boon for researchers and academicians

BIBLIOGRAPHY

[1] Search Based Test Data Generation Using Evolutionary Testing Techniques, International Journal of Software Engineering (IJSE), Volume (1) : Issue (5) , S. Kanmani , P. Maragathavalli.

[2] "Search based software engineering", www.elsevier.com/locate/infsof, Mark Harman,bryan f. jones

[3] "The Current State and Future of Search Based Software Engineering", Future of Software Engineering(FOSE'07) 0-7695-2829-5/07 $20.00 © 2007,IEEE,Mark Harman King's College London Strand, London,WC2R 2LS United Kingdom

[4] "Search Based Software Engineering: Introduction to the Special Issue of the IEEE Transactions on Software Engineering", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. 6, NOVEMBER/DECEMBER 2010. Mark Harman and Afshin Mansouri.

[5] "Search based software engineering Software engineering using meta heuristic innovative search algorithms.", Mark Harman and Afshin Mansouri.

[6] " A Survey on Search-Based Software Design". SERIES OF PUBLICATIONS DNET PUB LICATIONS D- 2009- 1, MARCH 2009 O uti Räihä , University of Tampere, Finland.

[7] "Search Based Software Engineering and Software Defect Prediction", Goran Mauša, mag. ing. el. University of Rijeka - Faculty of Engineering, Vukovarska 58, HR-51000 Rijeka, Croatia.

[8] Pareto-Optimal Search-Based Software Engineering (POSBSE): A Literature Survey, "Abdel Salam Sayyad Hany Ammar", Lane Department of Computer Science and Electrical Engineering West Virginia University Morgantown, WV, USA 978-1-4673-6437-9/13 2013 IEEE.

[9] "Introduction to the special issue on search based Software engineering", Springer Science+Business Media New York 2013 MelÓCinnéide ·MyraB.Cohen.

[10] "The Seed is Strong: Seeding Strategies in Search-Based Software Testing", Gordon Fraser Saarland University – Computer Science Saarbr¨ucken, Germany Andrea Arcuri.

[11] "On the Value of User Preferences in Search-Based Software Engineering: A Case Study in Software Product Lines" 978-1-4673-3076-3/13 2013 IEEE, "Abdel Salam Sayyad Tim Menzies Hany Ammar"

[12] "Effectively using Search-Based Software Engineering Techniques within Model Checking and Its Applications" 978-1-4673-6284-9/13 2013 IEEE "Jeremy S. Bradbury, David Kelk and Mark Green",

[13] "Threats to Validity in Search-based Software Engineering Empirical Studies", CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA," Márcio de Oliveira Barros Arilo Claudio Dias Neto"

[14] "Overview of search-based optimization algorithms used in software engineering", International Conference on Innovative Technologies, IN-TECH 2012, Rijeka, 26 - 29.09.2012, G. Mauša , T. Galinac Grbac , B. Dalbelo Bašić.

[15] "A Concept for an Interactive Search-Based Software Testing System", A Concept for an ISBST System, Bogdan Marculescu, Robert Feldt, and Richard Torkar.

[16] "Applying Elementary Landscape Analysis to Search-Based Software Engineering"," Guanzhou Lu, Rami Bahsoon, Xin Yao".

[17] A Factorial Experiment on the Scalability of Search-Based Software Testing Master Thesis Software Engineering Thesis Number: MSE-2009:20 June 2009 "Arash Mehrmand"

[18] Search Based Software Test Data Generation Using Evolutionary Testing Techniques International Journal of Software Engineering (IJSE), Volume (1): Issue (5), S. Kanmani & P. Maragathavalli. Thirteen papers are compared.

[19] "Getting Results from Search–Based Approaches to Software Engineering" Mark Harman, Joachim Wegener Proceedings of the 26th International Conference on Software Engineering (ICSE'04) 0270-5257/04 $20.00 © 2004 IEEE

[20] " Theoretical Runtime Analysis in Search Based Software Engineering" , Andrea Arcuri, Per Kristian Lehre and Xin Yao.

[21] AUSTIN:A tool for Search Based Software Testing for the C Language and its Evaluation on Deployed Automotive Systems Kiran Lakhotia, Mark Harman, Hamilton Gross.

[22] "On Software Engineering Repositories and Their Open Problems", http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/ Daniel Rodriguez, Israel Herraiz Rachel Harrison.

[23] "Review of Search based Techniques in Software Testing" Rakesh Roshan, Rabins Porwal,Chandra Mani Sharma International Journal of Computer Applications (0975 – 8887) Volume 51– No.6, August 2012.

[24] "Search Methodologies in Real-world Software Engineering", Gabriela Ochoa, GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands. Copyright 2013 ACM 978-1-4503-1964-5/13/07

[25] "SEARCH-BASED SOFTWARE TEST DATA GENERATION USING EVOLUTIONARY COMPUTATION", International Journal of Computer Science & Information Technology (IJCSIT), Vol 3, No 1, Feb 2011 DOI: 10.5121/ijcsit.2011. 3115 213. P. Maragathavalli

[26] "Search-based Software Test Data Generation: A Survey" Phil McMinn. A preprint of an article accepted for publication in Software Testing Verification and Reliability, copyright (c) Wiley 2004.

[27] "Dynamic Optimization of Network Routing Problem through Ant Colony Optimization (ACO)". Khalid Mahmood , Shahid Kamal , Hamid Masood Khan, Computer Engineering and Intelligent Systems www.iiste.org ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online) Vol 3, No.8, 201260.

[28] "Search–Based Testing, the Underlying Engine of Future Internet Testing",Arthur I. Baars, Kiran Lakhotia, Tanja E.J. Vos and Joachim Wegene, Proceedings of the Federated Conference on Computer Science and Information Systems pp. 917–923 (ISBN 978-83-60810-22-4).