Search Based Software Engineering: Introduction to the Special Issue of the *IEEE Transactions on Software Engineering*

Mark Harman and Afshin Mansouri

1 Introduction

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 [11].

The candidate solution representation to the problem defines the search space in which the search takes place. In order to guide the search-based optimization process, a fitness function (or cost function) is required. This function determines the better of two candidate solutions, imbuing the search algorithm with the ability to differentiate between solutions and to measure progress.

The most widely used algorithms for SBSE have been genetic algorithms, genetic programming, simulated annealing, and hill climbing (gradient descent) [12]. However, many other optimization techniques have also been applied, including greedy-based approaches and traditional operations research techniques through to more recently developed metaheuristic optimization techniques such as particle swarm optimization and ant colony optimization [12].

In this special issue, the authors have also used a variety of search-based optimization algorithms, most notably single and multi-objective genetic algorithms, genetic programming, and hill climbing. The most important attribute that these applications share is the search based formulation; the problem is thought of as a *search* among a large space of candidate solutions. It is from this search based approach that Search Based Software Engineering derives its name.

The term SBSE was coined by Harman and Jones [11] in 2001. However, there were many other authors who had previously applied search-based optimization to aspects of software engineering (for example, in work on search-based optimization for software project management [5] and testing [19], [22], [20]). The theory and practice of SBSE as a coherent approach to software engineering optimization has been developed by many other authors [8]. The SBSE approach has been the topic of several surveys and reviews [1], [12], [14], [17], to which this special issue also adds a significant new contribution in the form of the systematic review of the literature on Search Based Software Testing (SBST) by Ali, Briand, Hemmati, and Panesar-Walawege.

SBSE is important in software engineering because it provides a way to attack hard, highly constrained problems that involve many (potentially competing and conflicting) objectives using an automated approach. It is relatively easy to apply because representations and fitness functions are readily available in software engineering [10].

In other engineering disciplines, such as mechanical, materials, chemical, electric, and electronic engineering, search-based optimization has been applied for many years [9]. It is only recently that software engineering has started to catch up with this trend. There is every reason to think that this growing interest will continue. In some ways, the advent of SBSE is merely a reflection of the evolution of software development into a mature and fully fledged engineering discipline. However, software engineering, more than any other engineering discipline, is redolent with search based optimization application potential.

The very nature of software makes it even better suited to search based optimization than traditional engineering artifacts. In traditional engineering optimization, the artifact to be optimized is often simulated. This is typically necessary precisely because the artifact to be optimized is a physical entity (such as a processing plant, aircraft engine, or circuit layout). Search-based optimization requires repeated fitness computation, which is impractical if a physical solution has to be constructed for each fitness evaluation.

Engineers seeking to apply search-based optimization to traditional physical engineering artifacts therefore have to content themselves with an approach that is one step removed from reality; optimizing not the artifact itself, but some simulation or representation of it. The fitness thus computed is not the fitness of the final product, resulting in an additional layer of potential inaccuracy and cost.

By contrast, software has no physical existence. It exists only in the virtual world of discrete logic and conception. This virtual existence is perhaps the single most important property that makes software unique among engineering

- *M. Harman is with the Centre for Research on Evolution, Search and Testing (CREST), Department of Computer Science, University College London, Malet Place, London WC1E 6BT, UK. E-mail: Mark.Harman@cs.ucl.ac.uk.*
- *A. Mansouri is with the Brunel Business School, Brunel University, Uxbredge, Middlesex UB8 3PH, UK. E-mail: Afshin.Mansouri@brunel.ac.uk.*

For information on obtaining reprints of this article, please send e-mail to: tse@computer.org.

0098-5589/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

artifacts. Software derives much of its complexity and consequent engineering challenge from this property. However, in the realm of SBSE it becomes one of software engineering's primary advantages. It means that, for software engineering, in contrast with all other engineering disciplines, the application of search-based optimization can often be completely direct. SBSE can be used to construct optimal and/or near optimal instances of test cases, designs, and logical structures, not merely optimizations of their corresponding simulations.

SBSE is not merely applicable to program code, but all of the artifacts which, in the broadest sense can be considered part of software and relevant to software engineering. Already SBSE has been successfully applied to the construction of optimal test cases [18], clustering of modules and heaps [6], [15], requirements sets [4], management [2], and refactorings [16]. The papers in this special issue also reflect this wide application diversity, applying SBSE to construction of many different software artifacts including designs, test cases, and behavioral and quality models.

Not only is SBSE widely applicable to many very different software engineering problems, it also brings to these problems many generic advantages that make it an attractive solution technique in many cases:

1. **SBSE is highly robust**. Search based optimization techniques are stochastic algorithms that can often be tuned using a selection of parameters. This can initially dissuade software engineers, particularly those more comfortable with formal approaches to software development. However, it should be noted that optimization algorithms are extremely robust and that often the solutions required need only lie within some specified tolerance. Indeed, part of the appeal of SBSE derives from the way in which it moves us away from thinking that there should be a single perfect solution to a more "engineering centric" world view in which we seek a solution that falls within a suitable tolerance. This is a world view well adapted to the emergent engineering challenges that come with nonfunctional properties, massive scales, and complex emergent behavior interactions. This issue of robustness is an important property for investigation. For example, in this special issue, the paper by Garousi addresses questions of robustness in the application of SBSE to stress testing.
2. **SBSE has attractive scalability potential**. Search based optimization techniques are known as "embarrassingly parallel" because of their potential for scalability through parallel execution of fitness computations. Recent work has shown how this parallelism can be exploited on General Purpose Graphical Processing devices (GPGPUs). For instance, Langdon and Banzhaf reported results that achieved multiple orders of magnitude scale up with hundreds of millions of GP operations performed per second using relatively cheap and widely available GPGPU devices [13].
3. **SBSE creates new links.** SBSE creates links between otherwise apparently unconnected software engineering disciplines. For instance, regression testing and requirements engineering are two software engineering subareas that are typically regarded as existing at opposite ends of the software development process. They have largely nonintersecting communities of researchers, each with their own separate conferences and journals. However, regression and requirements optimization problems share common formulations. Viewed through the SBSE lens, they are essentially selection and prioritization problems and can be solved with similar search-based approaches [12].
4. **SBSE offers a source of insight**. SBSE is not just a way to construct software artifacts such as test cases and designs. The process of search can be used to reveal insight into the structure of the problem to be optimized. For example, in this special issue, the papers on modeling and classification by Krogmann et al. and Yi et al., respectively, seek to give insight into models of performance and quality.

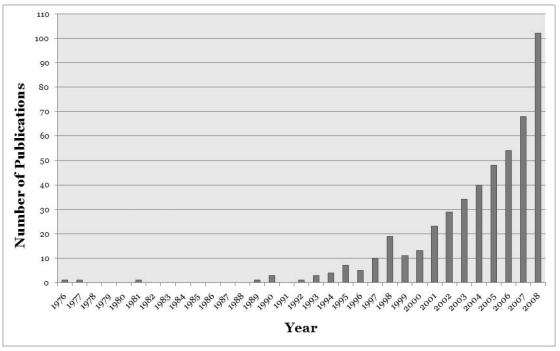

Fig. 1. The trend of publications on SBSE.

These attributes have made SBSE increasingly popular among researchers. Fig. 1 presents evidence for this increasing popularity. It is constructed from the data available in the SBSE repository.[1] As can be seen from this figure, the growth rate is dramatic.

Undoubtedly the first area of software engineering to receive significant attention from SBSE researchers was software testing and, as Fig. 2 shows, testing remains the area of software engineering most widely covered by SBSE. Software test objectives are natural candidates for SBSE. Software testing applications are easy to formulate as SBSE problems because the search space is simply the space of possible inputs to the system under test, while the fitness function that guides the search is directly translatable from test objective metrics.

The field of SBST research is now reaching a level of maturity. Evidence for this comes from the fact that a systematic review of the literature like that presented by Ali et al. in this special issue is possible and necessary. Also, there is now an established workshop on SBST, which is

1. The SBSE repository is a publicly available resource at http://www.sebase.org/sbse/publications/. The graph in Fig. 1 was constructed from the data available on 20 November 2009. The guest editors wish to thank Dr. Yuanyuan Zhang, the repository maintainer, for constructing Figs. 1 and 2 presented in this editorial.

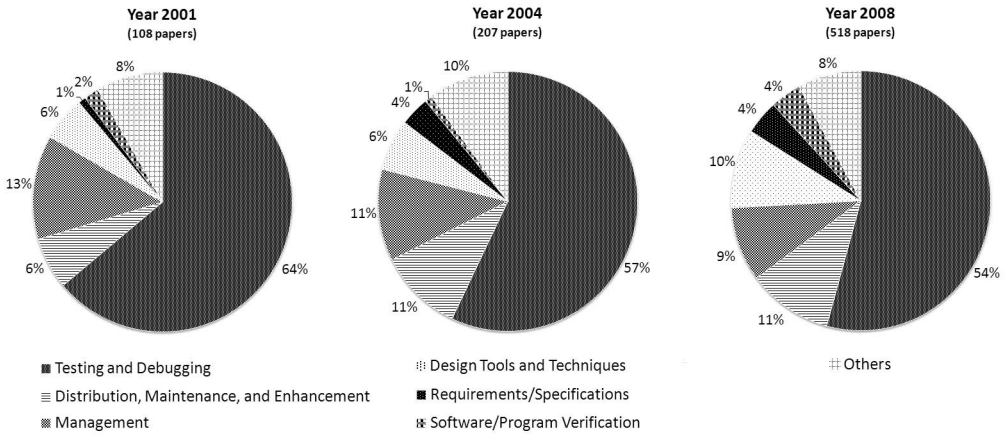

Fig. 2. The increasingly diversified trend of software engineering topics covered by SBSE since 2001.

held annually in colocation with the IEEE International Conference on Software Testing (ICST).

Nonetheless, it is notable that of the eight papers in this special issue, five concern topics other than testing. This does not mean that SBST is a topic in decline—on the contrary—it continues to grow. However, this observation reflects a growing trend. SBSE researchers are reaching out to other software engineering domains with the result that new software engineering topic areas are starting to fall within the SBSE catchment area.

Fig. 2 shows the development of SBST as a topic within SBSE. As can be seen, topics such as requirements, management, design, and maintenance are all growing in their share of the work on SBSE. The fall in the proportion of work on software testing does not indicate that there is diminishing interest in this topic. Rather, interest in all SBSE topics, including testing, is growing, as shown by Fig. 1. However, it is interesting to see how the application of search-based optimization techniques is spreading from its original application site of software testing throughout the overall spectrum of software engineering activities. Based on the trends revealed in these figures and the historical development of the subject, it seems likely that within the next five years we may see tutorials, workshops, and other dedicated events and special issues focusing on search-based requirements optimization, search-based software design, search-based software project management, and search-based software maintenance and reengineering. Already, there are nascent communities developing in all of these areas and many others.

The field of SBSE continues to develop and grow, with many exciting new application areas within software engineering continuing to emerge. As well as this special issue of the *IEEE Transactions on Software Engineering*, there have also been special issues in the journals *Information and Software Technology* (*IST*), *Software Maintenance and Evolution* (*JSME*), *Computers and Operations Research* (*COR*), *Empirical Software Engineering* (*EMSE*), and *Software Practice and Experience* (*SPE*). This reflects the wide range of application areas within software engineering as well as uptake of software engineering problems within the operations research and metaheuristic search communities. There is now an established Symposium on Search Based Software Engineering (SSBSE) and a dedicated track of the Genetic and Evolutionary Computation COference (GECCO) on search based software engineering.

The papers in this special issue make a strong contribution to this growing body of literature. The call for papers attracted the submission of 31 papers, from which the eight papers in this special issue were selected for publication. The submissions went through *TSE*'s normal comprehensive and thorough refereeing process. Additionally, for each paper, at least one referee was chosen to have expertise in SBSE, while at least one was chosen from outside the SBSE community. This referee process sought to ensure that the papers ultimately selected for the special issue attain a level of interest to the general software engineer as well as to those actively involved in the SBSE community.

The special issue guest editors would like to thank the authors for their contributions to this special issue and the reviwers for their time and expertise. The guest editors sincerely hope and believe that the papers within this special issue of the *IEEE Transactions on Software Engineering* will give a sense of the vibrancy and diversity of the growing and promising area of research and practice that is Search Based Software Engineering.

2 Summary of the Papers in This Special Issue

The eight papers in this special issue cover topics in Search Based Software Testing (SBST) and SBSE for design, modeling, and prediction.

2.1 SBSE for Software Testing

As mentioned in the introduction to this editorial, Search Based Software Testing (SBST) was the first area of software engineering to which search based optimization techniques were applied and it remains that most widely studied area. The survy by Ali et al. in this special issue answers questions about the empirical results available in the literature, using the mechanism of a systematic review to collect, select, and extract from the literature statements of what can be claimed for SBST. The authors also present recommendations for the construction of future empirical studies in SBST that will be an invaluable guide for future research. Many of these recommnedations also apply more

widely to the field of SBSE as a whole, not merely to the subarea of SBST.

SBSE has been applied to many other software testing applications, including structural testing, model-based testing, mutation testing, temporal testing, exception testing, regression testing, integration testing, configuration, and interaction testing [12]. However, hitherto, there has been little work in search-based approaches to statistical testing. The paper by Poulding and Clark addresses this problem. the authors use the hill climbing approach to derive optimal and near optimal probability distributions for statistical testing, with evidence to show that this produces better fault finding abilities by specializing the distribution (compared to uniform random distributions). Also, Poulding and Clark present empirical results that support the claim that the distributions of test cases they derive using SBSE outperform the fault finding abilities of traditional structural testing techniques.

The paper by Garousi also concerns SBST. It addresses the problem of stress testing. Previous work by Garousi and others has resulted in SBSE approaches to the generation of test cases that stress the system under test. The approach can be useful for identification of particularly challenging operating environments and scenarios. In the paper in the issue, Garousi empirically studies the problem of search-based stress testing scalability and robustness, using a tool that he has also made publicly available.

2.2 SBSE for Software Design

Software design is one of the recent areas of growing SBSE research interest. Software design presents the engineer with a naturally complex design space in which many competing and conflicting objectives must be optimized and balances between different concerns must be found. As such, the software design space is a natural candidate for SBSE research. Indeed, design level SBSE has been the subject of a recent survey reflecting its growing importance [17].

In this special issue, the reader will find two contributions to the problem of software design using SBSE. Simons and Parmee address the problem from a user centered perspective. Until recently, SBSE research focused on fully automated approaches to fitness computation in which the human input to the overall design of good fitness functions is crucial to the success of SBSE. However, there has been little previous work on the direct involvement of the human software engineer in the search itself [12].

The paper by Simons and Parmee involves the human directly as a part of the evolutionary computation, using a technique known in the evolutionary computation community as "interactive evolution." This is a natural approach to design since many design judgments ultimately rely upon human intuition. As a result of similar intuition-guided design properties in other engineering disciplines, interactive evolution can also be applied to SBSE. Simons and Parmee combine software and human agents interactively to guide the search process.

The paper by Bowman, Briand, and Labiche also focuses on software design and uses SBSE tehcniques to guide the decision maker in their design process. Unlike the Simons and Parmee approach, the paper does not directly use interactive evolution, but involves the designer in the overall process of allocating class responsibility. The paper also uses multi-objective optimization, an increasing trend in SBSE research [12]. Using multi-objective optimization, and SBSE approach can seek to find a balance between several competing objectives. These different objectives may be in competition with one another. also, it may be impossible to determine, a priori, the relative importance of each objective. In this situation, researchers have found Pareto optimal approaches to be very attractive. Bowman et al. use a Pareto optimal approach to seek class responsibility assignments that balance the outcomes of four different cohesion and coupling metrics.

The paper by White, Dougherty, and Schmidt is also concerned with design, but not merely software design. The paper presents an approach, ASCENT, that uses SBSE to develop designs for hardware and software in tandem. This approach illustrates the wide applicability of SBSE research, indicating that it has contributions to make in software systems engineering as well as purely within software engineering.

2.3 SBSE for Software Modeling and Prediction

Modeling and prediction are also topics for which SBSE is well adapted because it can cater to multiple, potentially conflicting goals and constraints and can be used to interpolate a best fit to a set of data, using fit as a fitness function. The paper by Khoshgoftaar, Liu, and Seliya concerns the construction of software quality models using SBSE. They adddress search-based software quality modeling with multiple software data repositories. The authors show that optimizing software quality models can improve predictive capability, employing genetic programming to build optimized models from multiple data sets.

The paper by Krogmann, Kuperberg, and Reussner is also concerned with SBSE as a means of building predictive models. The paper shows 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
Afshin Mansouri
Guest Editors

REFERENCES

[1] W. Afzal, R. Torkar, and R. Feldt, "A Systematic Review of Search-Based Testing for Non-Functional System Properties," *Information and Software Technology*, vol. 51, no. 6, pp. 957-976, 2009.

[2] E. Alba and F. Chicano, "Software Project Management with Gas," *Information Sciences*, vol. 177, no. 11, pp. 2380-2401, June 2007.

[3] A. Arcuri, "On the Automation of Fixing Software Bugs," *Proc. Doctoral Symp. IEEE Int'l Conf. Software Eng.*, pp. 1003-1006, May 2008.

[4] A.J. Bagnall, V.J. Rayward-Smith, and I.M. Whittley, "The Next Release Problem," *Information and Software Technology*, vol. 43, no. 14, pp. 883-890, 2001.

[5] C.K. Chang, "Changing Face of Software Engineering," *IEEE Software*, vol. 11, no. 1, pp. 4-5, Jan.. 1994.

[6] M. Cohen, S.B. Kooi, and W. Srisa-an, "Clustering the Heap in Multi-Threaded Applications for Improved Garbage Collection," *Proc. Eighth Ann. Conf. Genetic and Evolutionary Computation*, vol. 2, pp. 1901-1908, July 2006.

[7] P. Funes, E. Bonabeau, J. Herve, and Y. Morieux, "Interactive Multi-Participant Task Allocation," *Proc. 2004 IEEE Congress Evolutionary Computation*, pp. 1699-1705, June 2004.
[8] M. Harman, "The Current State and Future of Search Based Software Engineering," *Proc. Int'l Conf. Software Eng./Future of Software Eng.*, L. Briand and A. Wolf, eds., pp. 342-357, May 2007.
[9] M. Harman, "Why the Virtual Nature of Software Makes It Ideal for Search Based Optimization," *Proc. 13th Int'l Conf. Fundamental Approaches to Software Eng.*, 2010.
[10] M. Harman and J. Clark, "Metrics Are Fitness Functions Too," *Proc. 10th Int'l Software Metrics Symp.*, pp. 58-69, Sept. 2004.
[11] M. Harman and B.F. Jones, "Search-Based Software Engineering," *Information & Software Technology*, vol. 43, no. 14, pp. 833-839, Dec. 2001.
[12] M. Harman, A. Mansouri, and Y. Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications," Technical Report TR-09-03, Dept. of Computer Science, King's College London, Apr. 2009.
[13] W.B. Langdon and W. Banzhaf, "A SIMD Interpreter for Genetic Programming on GPU Graphics Cards," *Proc. 11th European Conf. Genetic Programming*, pp. 73-85, Mar. 2008.
[14] P. McMinn, "Search-Based Software Test Data Generation: A Survey," *Software Testing, Verification and Reliability*, vol. 14, no. 2, pp. 105-156, 2004.
[15] B.S. Mitchell and S. Mancoridis, "On the Automatic Modularization of Software Systems Using the Bunch Tool," *IEEE Trans. Software Eng.*, vol. 32, no. 3, pp. 193-208, Mar. 2006.
[16] M. O'Keeffe and M. Ó Cinnéide, "Search-Based Software Maintenance," *Proc. Conf, Software Maintenance and Reeng.*, pp. 249-260, Mar. 2006.
[17] O. Räihä, "A Survey on Search Based Software Design," Technical Report D-2009-1, Dept. of Computer Sciences, Univ. of Tampere, 2009.
[18] P. Tonella, "Evolutionary Testing of Classes," *Proc. 2004 ACM SIGSOFT Int'l Symp. Software Testing and Analysis*, pp. 119-128, July 2004.
[19] N. Tracey, J. Clark, and K. Mander, "Automated Program Flaw Finding Using Simulated Annealing," *Proc. 1998 ACM SIGSOFT Int'l Symp. Software Testing and Analysis*, pp. 73-81, Mar. 1998.
[20] J. Wegener, H. Sthamer, B.F. Jones, and D.E. Eyres, "Testing Real-Time Systems Using Genetic Algorithms," *Software Quality*, vol. 6, no. 2, pp. 127-135, June 1997.
[21] W. Weimer, T.V. Nguyen, C. Le Goues, and S. Forrest, "Automatically Finding Patches Using Genetic Programming," *Proc. Int'l Conf. Software Eng.*, pp. 364-374, 2009.
[22] S. Xanthakis, C. Ellis, C. Skourlas, A. LeGall, S. Katsikas, and K. Karapoulios, "Application of Genetic Algorithms to Software Testing," *Proc. Fifth Int'l Conf. Software Eng. and Applications*, pp. 625-636, Dec. 1992.

Mark Harman is a professor of software engineering in the Department of Computer Science at University College London. He is widely known for work on source code analysis and testing and he was instrumental in the founding of the field of Search Based Software Engineering, the topic of this special issue. He has given 14 keynote invited talks on SBSE and its applications in the past four years. Professor Harman is the author of more than 150 refereed publications, is on the editorial boards of seven international journals, and has served on 90 program committees. He is director of the CREST centre at University College London.

Afshin Mansouri received the MSc and PhD degrees in industrial engineering from Amirkabir University of Technology, Iran. He is a lecturer in operations management at Brunel University, United Kingdom. Prior to this position, he worked as a research associate in the CREST centre at King's College London and as a postdoctoral research fellow in the Laboratory of Computer Science at the University of Tours, France. He has published more than 20 papers in international journals and conference proceedings on applied operations research and management science, metaheuristic search techniques, and multi-objective optimization.