International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 3, March 2015
ISSN: 2394 - 6598
www.ijetie.org

AN ANALYSIS ON SEARCH BASED OPTIMIZATION IN SOFTWARE ENGINEERING

G.PANDIYAN[1]
Research Scholar,
Rathnavel Subramaniam College of Arts and Science,
Sulur, Coimbatore, Tamilnadu, India – 641402,
ganeshanpandiyan@gmail.com

Dr.P.KRISHNAKUMARI[2]
Director, Department of Computer Applications,
Rathnavel Subramaniam College of Arts and Science,
Sulur, Coimbatore, Tamilnadu, India – 641402,
kkjagadeesh@yahoo.com

Abstract

Finding an alternative with the most cost effective or highest achievable performance under the given constraints, by maximizing desired factors and minimizing undesired ones. Optimization algorithms are used to address problems in Software Engineering, they are very powerful in the software development life cycle, and the reviews some of the benefits that can be expected to come from further development of the field of search based Software Engineering. SBSE has been applied to problems throughout the Software Engineering lifecycle, from requirements and project planning to maintenance and re-engineering. This paper describes the uses of optimization in software engineering.

Keywords:
Software Engineering; Optimization; Search Based; Software Quality.

I. Introduction

In Search Based Software Engineering, the term 'search' is used to refer to the metaheuristic search-based optimization techniques that are used. SBSE seeks to reformulate Software Engineering problems as search-based optimization problems (or 'search problems' for short). This is not to be confused with textual or hyper textual searching. Rather, for Search Based Software Engineering, a search problem is one in which optimal or near optimal solutions are sought in a search space of candidate solutions, guided by a fitness function that distinguishes between better and worse solutions. Maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation. The function allows comparison of the different choices for determining which might be "best."

II. Background

Although interest in SBSE has witnessed a recent dramatic rise, its origins can be traced back to early work on optimization in software engineering in the 1970s. One of the earliest attempts to apply optimization to a Software Engineering problem was reported by Miller and Spooner [6] in 1976 in the area of software testing. To the best of our knowledge, Xanthakis et al. [7] were the first apply a metaheuristic search technique to a software engineering problem in 1992.

The term SBSE was first used by Harman and Jones [8] in 2001. This was the first time in the literature that it was suggested that search based optimization could be applied right across the spectrum of Software Engineering activity.

The position paper acted as a form of 'manifesto' for SBSE. However, it should also be noted that, much earlier, Carl Chang has also used his IEEE Software editorial to promote the more widespread use of Evolutionary Computation in Software Engineering in 1994 [9].

III Search Based Optimization

A. Generality and Applicability

One of the striking features of the SBSE research programme that emerges from this survey is the wide variety of different Software Engineering problems to which SBSE has been applied. Clearly, testing remains the predominant 'killer application', with 59% of all SBSE papers targeting various aspects of testing. However, as the survey reveals, there are few areas of Software Engineering activity to which search based optimization remains unapplied.

This generality and applicability comes for the very nature of Software Engineering. The two primary tasks that have to be undertaken before a search based approach can be applied to a Software Engineering problem are the definition of a representation of the problem and the fitness function that captures the objective or objectives to be optimized.

Once these two tasks are accomplished it is possible to begin to get results from the application of many search based optimization techniques. In other engineering disciplines, it may not be easy to represent a problem; the physical properties of the engineering artifact may mean that simulation is the only economical option. This puts the optimization algorithm at one stage removed from the engineering problem at hand.

Furthermore, for other engineering disciplines, it may not be obvious how to measure the properties of the engineering artifact to be optimized. Even where the measurements required may be obvious, it may not be easy to collect the readings; once again the physical properties of the engineering materials may be a barrier to the application of optimization techniques. However, software has no physical manifestation. Therefore, there are fewer problems with the representation of a software artifact, since almost all software artifacts are, by their very nature, based on intangible 'materials' such as information, processes and logic.

This intangibility has made many problems for Software Engineering. However, by contrast, within the realm of SBSE, it is a significant advantage. There are few Software Engineering problems for which there will be no representation, and the readily available representations are often ready to use 'out of the box' for SBSE.

Furthermore, measurement is highly prevalent in Software Engineering, with a whole field of research in software metrics that has spawned many conferences and journals. Therefore, it is also unlikely that the would-be search based software engineer will find him or herself bereft of any putative fitness function. Indeed, it has been argued that all metrics are also fitness functions, waiting to be applied in SBSE [1].

For these reasons, it is likely that there will be a rapid growth in the breadth of SBSE research. In this survey is very likely to continue and authors are likely to continue to find ways to bring new Software Engineering subareas within the remit of SBSE.

B. Scalability

One of the biggest problems facing software engineers is that of scalability of results. Many approaches that are attractive and elegant in the laboratory turn out to be inapplicable in the field, because they lack scalability. One of the attractions of the search based model of optimization is that it is naturally parallelizable. Hill climbing can be performed in parallel, with each climb starting at a different point [2].

Genetic algorithms, being population based, are also naturally parallel; the fitness of each individual can be computed in parallel, with minimal overheads. Search algorithms in general and SBSE in particular, therefore offer a 'killer application' for the emergent paradigm of ubiquitous user-level parallel computing. Notwithstanding a breakthrough in quantum computation technology, it seems likely that future improvements in processing speed are likely to be based on increasing parallelism.

Already, limited parallelism is widely used in desktop computing and the importance of the drive toward superfast parallel computers is recognized at the highest levels. This trend towards greater parallelism, the need for scalable Software Engineering and the natural parallelism of many SBSE techniques all point to a likely significant development of parallel SBSE to address the issue of Software Engineering scalability.

C. Robustness

In some Software Engineering applications, solution robustness may be as important as solution functionality. For example, it may be better to locate an area of the search space that is rich in fit solutions, rather than identifying an even fitter solution that is surrounded by a set of far less fit solutions. In this way, the search seeks stable and fruitful areas of the landscape, such that near neighbors of the proposed solution are also highly fit according to the fitness function. This would have advantages where the solution needs to be not merely 'good enough' but also 'strong enough' to withstand small changes in problem character [3].

Hitherto, research on SBSE has tended to focus on the production of the fittest possible results. However, many application areas require solutions in a search space that may be subject to change. This makes robustness a natural property to which the research community could and should turn its attention.

D. Feedback and Insight

False intuition is often the cause of major error in software engineering, leading to misunderstood specifications, poor communication of requirements and implicit assumptions in designs. SBSE can address this problem. Unlike human-based search, automated search techniques carry with them no bias. They automatically scour the search space for the solutions that best fit the (stated) human assumptions in the fitness function. This is one of the central strengths of the search based approach.

It has been widely observed that search techniques are good at producing unexpected answers. For example, evolutionary algorithms have led to patented designs for digital filters [4] and the discovery of patented antenna designs [5]. Automated search techniques will effectively work in tandem with the human, in an iterative process of refinement, leading to better fitness functions and thereby, better encapsulation of human assumptions and intuition.

IV Conclusion

This paper has provided a detailed survey and review of the area of Software Engineering activity that has come to be known as Search Based Software Engineering (SBSE). As the survey shows, a dramatic increase in SBSE activity, with many new applications being addressed. The paper has identified trends in SBSE research, providing data to highlight the growth in papers and the predominance of Software Testing research.

The future of SBSE is bright; there are many areas to which the techniques associated with SBSE surely apply, yet have yet to be considered. In existing areas of application the results are strongly encouraging. New developments emanating from the optimization community will present exciting new possibilities, while developments in Software Engineering will present interesting new challenges. If Software Engineering really is an engineering discipline, then surely SBSE is a very natural consequence of the very existence of such an engineering discipline.

References:

[1] Harman, M. and Clark, J. A. (2004). Metrics Are Fitness Functions Too. In Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS ’04), pages 58–69, Chicago, USA. IEEE Computer Society.

[2] Mahdavi, K., Harman, M., and Hierons, R. M. (2003a). A Multiple Hill Climbing Approach to Software Module Clustering. In Proceedings of the International Conference on Software Maintenance (ICSM ’03), pages 315–324, Amsterdam, Holland. IEEE Computer Society.

[3] Beyer, H. G. and Sendhoff, B. (2007). Robustness optimization — a comprehensive survey. Computer Methods in Applied Mechanics and Engineering, 196, 3190–3218.

[4] Schnier, T., Yao, X., and Liu, P. (2004). Digital filter design using multiple pareto fronts. Soft Computing, 8(5), 332–343.

[5] Linden, D. S. (2002). Innovative antenna design using genetic algorithms. In D. W. Corne and P. J. Bentley, editors, Creative Evolutionary Systems, chapter 20. Elsevier, Amsterdam, The Netherland.

[6] Miller, W. and Spooner, D. L. (1976). Automatic Generation of Floating-Point Test Data. IEEE Transactions on Software Engineering, 2(3), 223–226.

[7] Xanthakis, S., Ellis, C., Skourlas, C., Le Gall, A., Katsikas, S., and Karapoulios, K. (1992). Application of Genetic Algorithms to Software Testing. In Proceedings of the 5th International Conference on Software Engineering and Applications, pages 625–636, Toulouse, France.

[8] Harman, M. and Jones, B. F. (2001a). Search-based Software Engineering. Information & Software Technology, 43(14), 833–839.

[9] Chang, C. K. (1994). Changing Face of Software Engineering. IEEE Software,11(1).

[10] Shepperd, M. (2007). Software economics. In L. Briand and A. Wolf, editors, Future of Software Engineering 2007, Los Alamitos, California, USA. IEEE Computer Society Press.