SEARCH BASED SOFTWARE ENGINEERING-A REVIEW

Arif Mohammad Lattoo *

ABSTRACT

This paper gives a brief report about search-based software engineering (SBSE) and the scope of the field, in research interest. The paper briefly reviews widely used optimization techniques and the important elements required for their successful application to software engineering. The paper also sets out the new techniques, which can be formulated to attain the success.

Key Words: Software Engineering, Simulate, Annealing, Genetic Algorithm, Test Case Sint.

1. Introduction

Search Based Software Engineering (SBSE), as perceived, is quite an advanced branch of computer science to evolve in future. It relies on the benchmark "every algorithm that is to be employed to devise a solution is actually search based". SBSE is the approach in which search based optimisation is applied to software engineering. SBSE consists of search-based optimization algorithms that are becoming incorporated in almost every area of software engineering. It aims to search for candidate solutions in a search space and examines each candidate solution for some specified metrics and then ultimately focuses on devising an optimal solution. The parameters that help in distinguishing between better & worst solutions are characterised by employing Fitness Function(s) [1].

SBSE starts with basic two components a) Representation & b) A Fitness Function. A Software Engineer, at the start, will have some representation of the problem area under consideration. The Software Engineer can thus easily devise a representation of the problem from the "Minimal Data" available in the beginning. It may rarely be the case that certain problems may not ensure a representation and hence hindering the formulation of converging towards an optimal solution.

The second ingredient is fitness function. Some metric(s) associated with the problem area can guide best in formulating a fitness function. The fitness function evaluates whether the algorithm has found a better solution than the one in previous step and guides the search to as optimal as possible solution [2].

2. Implementing Search Based Approach

SBSE consists of search-based optimization algorithms used in software engineering, with genetic algorithms, simulated annealing and hill climbing being the most widely used [2]. SBSE encompasses the concept of Randomized Algorithms for its implementation. One of the techniques in this regard is "Hill Climbing". This technique employs a "Converging Mechanism" with a heuristic policy to rendezvous with the final optimal solution. It starts with a random point in the search space and then inspects for some neighbourhood candidate solution points and then compares the metrics of the two for some aspects and if improved fitness is informed at this point, a 'Move' is made. Then the same mechanism is repeated for this point and so on until the neighbourhood point offers no better comparative fitness. There is a problem with

* DDE, University of Kashmir, Hazratbal, Srinagar, Jammu & Kashmir, India 190 006 arifmohammad.001@gmail.com

the hill climbing approach, the located hill by the algorithm may be local maxima and may be far poorer than global maxima in the search space [3].

2.1 Simulated Annealing: Simulated Annealing (SA) inspired from *annealing in metallurgy* the logic is like a variation of hill climbing that avoids the problem of local maxima by allowing moves to less fit individuals. The approach gives more chances to consider less fit individuals in the earlier stages of the exploration of the search space. This chance is gradually reduced until the approach becomes a traditional hill climb in the very final stages of the exploration of the search space.

2.2 Genetic Algorithm: Genetic algorithms are an inspiration from Darwin's theory about evolution. To start the Algorithm we take a set of solutions (Represented by chromosomes) known as population. New Populations is formed from the solutions of the previous population. The motivation here is driven by a hope, that the resulting population will be better than the old one. The solutions to be selected for the formation of a new solution (offspring) are selected on the basis of their fitness - the more fitting they are, more the chances of they having to produce again. This process continues until some condition (for example number of populations or improvement of the best solution) is satisfied.

3. Problem Statement

SBSE believes in overcoming the difficulty faced when a software engineer finds himself surrounded by questions that are fully immersed in probability; e.g., how many test cases suffice for the current problem?, what is the minimum requirement set for the current program? & so on. This will definitely serve as an endeavour in computer science as these sort of uncertain questions always pose obstructions in the life cycle of any project , and hence any problem.

Many problems are computationally demanding, and have motivated the necessity for developing novel optimization approaches. There are problems which can best be solved with combining approaches of optimization techniques [3].

In this approach of problem solving, the output of one optimization technique can be fed as input to another optimization technique. In order to look for near optimal solution we got to have better understanding of search landscapes, which may suggest the application of hybrid search techniques. In the hybrid approach the best aspects of existing search algorithms can be combined.

Combining optimization search algorithms/hybrid algorithms can be further used as a possible approach to achieve the optimization or near optimal solutions with following goals:

- To achieve better balance between exploration and exploitation.
- To yield better solutions in shorter time.
- To combine global and local strategies for solving problems.

One important lacuna that may arrive out of nowhere is, the "Size of the test case suite". Test case suite is the set of all tests available to a software engineer at any instance of time. At the very scratch the size of the problem under consideration is small; hence an engineer will find it easy to put the current set of instructions through the test suite. But as the instruction set increases or the software evolves, it is the responsibility of the engineer to ensure that the modifications don't alter the existing functionality of the software. In linear thought, a software engineer will again re-employ all the tests in the test suite. But as the

software evolves, the size of test suite also increases and it is not feasible to employ all the tests, the justifications being limited resources available [1]. Here comes a stage where the engineer has to select only certain tests and not all in the test suite.

4. Research Scope

There has been an increase of interest in the SBSE over the recent years. SBSE has many stake holders throughout the software engineering lifecycle paradigm, from requirement engineering and project planning to maintenance and re-engineering [1]. Several journal articles and papers that have discussed the possibility of SBSE as an alternate name to the software engineering have attracted the reader's interest. Since, SBSE claims that all the software engineering problems can be addressed as SBSE problems, and can thus use SBSE and software engineering interchangeably [1]. These articles and books about the SBSE provide sufficient information to get a complete review of literature and devise a work plan accordingly in order to carry out further research about the subject. The following two paragraphs highlight the use of these sources in the interest of research.

At one end the focus is on achieving better balance between exploitation and exploration. Exploration deals with understanding of the configuration space, regardless of the planning problem, while exploitation deals with the solution of the problem based on the information obtained by exploration. The experiments are to be performed to demonstrate the adaptive balancing of exploration and exploitation.

Further, such experiment would yield better solutions in shorter time. The evaluation is based on experimenting on optimization algorithms and on combining of several search techniques in order to find optimal solution in shorter time. The third goal is to combine global and local strategies for solving problems. The Global optimization is done in order to find the best solution globally, in the (possible or known) presence of multiple local optima. If the traditional local scope search methods are used for solving the problem, then the locally optimal solutions of varying quality will be found, showing dependence on the starting point of the search, we will often find. A global scope search effort is needed in order to find the globally optimal solution.

5. Conclusion

In SBSE the goal is to use Search Based Optimization Algorithms to automate the construction of solutions to software engineering problems. Hybrid optimization can serve as a valuable resource for researchers studying SBSE or Operations Research. The goal of using hybrid techniques is to include mechanism from different approaches and thus benefiting from their advantages while minimizing their drawbacks.

6. References

Harman, M., McMinn, P., De Souza, J. T., & Yoo, S. (2012). Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical software engineering and verification* (pp. 1-59). Springer Berlin Heidelberg.

Harman, M., Mansouri, S. A., & Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, King's College London, Tech. Rep. TR-09-03*.

Harman, M. (2007, May). The current state and future of search based software engineering. In *2007 Future of Software Engineering* (pp. 342-357). IEEE Computer Society.