Empir Software Eng (2013) 18:547–549
DOI 10.1007/s10664-013-9257-9

EDITORIAL

Introduction to the special issue on search based software engineering

Mel Ó Cinnéide · Myra B. Cohen

Published online: 17 May 2013

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, we face an increasing number of tasks that are intractable to perform through manual or even automated exhaustive means. 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 refactorings, and reverse engineering a program's module structure.

In SBSE, heuristic algorithms such as hill climbing, or meta-heuristic algorithms such as simulated annealing, tabu search, genetic algorithms, and ant colony optimisation are used to efficiently explore the solution space. They are guided by a *fitness function*, 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 almost 160).

While this jump in the use of SBSE is an exciting development, 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. Experiments must include a well thought

[1] http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/

M. Ó Cinnéide (✉)
School of Computer Science & Informatics,
University College Dublin, Dublin, Ireland
e-mail: mel.ocinneide@ucd.ie

M. B. Cohen
Department of Computer Science & Engineering,
University of Nebraska-Lincoln, Lincoln, NE, USA
e-mail: myra@cse.unl.edu

out design, control for variance and exposure of potential validity threats, which leads to a need for strong empirical evidence in the SBSE literature. This close connection between SBSE and empirical software engineering has provided us with the impetus for this special edition. We hope that both communities can benefit from its content.

In this special edition, we include two papers that are extended versions of papers that first appeared in the Third International Symposium on Search Based Software Engineering, 2011, held in Szeged, Hungary. In "GPGPU Test Suite Minimisation: Search Based Software Engineering Performance Improvement Using Graphics Cards" by Shin Yoo, Mark Harman and Shmuel Ur, the authors present the first work that uses GPGPUs (General-Purpose Graphical Programming Units) to perform SBSE. While SBSE naturally lends itself to parallel computation, there has been remarkably little research in this area. Using cheap and readily-available GPGPUs to achieve parallelism is a very appealing idea. In this work, three popular evolutionary algorithms (NSGA-II, SPEA2 and the Two-Archive Evolutionary Algorithm) are re-implemented for GPGPUs using the OpenCL library. Their performance is evaluated on a multi-objective formalisation of the well-known Test Suite Minimisation problem, i.e. the problem of calculating the minimum set of tests that are required to satisfy the given test requirements.

The results show that GPGPU-based optimisation can achieve a speed-up of up to a factor of 25 when compared to a single-threaded version of the same algorithm executed on a single CPU. Subsequent statistical analysis reveals that the speedup correlates to the logarithm of the problem size. This finding indicates that the parallel approach to the Test Suite Minimisation problem should scale well to larger problem sizes. A notable feature of the results is that in the case of the large-scale industrial example studied, the GPGPU-based approach was able to discover a workable solution within the actual time constraints imposed by the company practice, while the single CPU-based approach was simply unable to do this. This shows that in practice the parallel approach can do more than just speed up the minimisation process, it can find a solution where the traditional approach would simply have failed.

The second paper "Parameter Tuning or Default Values? An Empirical Investigation in Search-Based Software Engineering" by Andrea Arcuri and Gordon Fraser, conducts a large empirical evaluation in the context of test generation. It both quantifies the impact of parameter tuning in this context, and evaluates its relative benefit in practice (given the potentially expensive nature of tuning). The authors use the EvoSuite test generation tool for this study. EvoSuite employs a genetic algorithm to generate test cases that aim to satisfy a particular coverage criterion, such as to cover all of the branches or all of the statements in a program.

Search algorithms usually depend on multiple parameters such as population size, mutation rate and crossover rate, and the choice of these parameters can be set to a range of values. It has been proven in past work, that it is impossible to tune a search algorithm so that the parameter values are optimal for all possible problems to which it is applied. However, the impact on SBSE of the lack of problem specific tuning has not been explored. The results of this paper show that tuning does impact the search performance (as expected). Contrary to expectations, through the use of the response surface methodology, the paper also shows that in this particular setting, default tunings work relatively well. The experiments conclude that tuning parameter settings for each specific problem instance fails to statistically outperform the default

values that have been presented in the literature; i.e. improvement through tuning is hard to achieve. This suggests that in practice, using default values may be reasonable and justified, and that parameter tuning may not be that worthwhile (at least in this particular SBSE domain).

The guest editors would like to express their thanks to Lionel Briand for his support to the SBSE community by providing the opportunity for this special issue, and to both the authors and referees for their work both on the conference and journal versions of these papers.