Genetic Improvement of Software Efficiency: The Curse of Fitness Estimation

Mahmoud A. Bokhari
Optimisation and Logistics, School of Computer Science, The University of Adelaide, Australia
Computer Science Department, Taibah University, Kingdom of Saudi Arabia
mahmoud.bokhari@adelaide.edu.au

Markus Wagner
Optimisation and Logistics, School of Computer Science, The University of Adelaide, Australia
markus.wagner@adelaide.edu.au

Brad Alexander
Optimisation and Logistics, School of Computer Science, The University of Adelaide, Australia
bradley.alexander@adelaide.edu.au

ABSTRACT

Many challenges arise in the application of Genetic Improvement (GI) of Software to improve non-functional requirements of software such as energy use and run-time. These challenges are mainly centred around the complexity of the search space and the estimation of the desired fitness function. For example, such fitness function are expensive, noisy and estimating them is not a straightforward task. In this paper, we illustrate some of the challenges in computing such fitness functions and propose a synergy between *in-vivo* evaluation and machine learning approaches to overcome such issues.

CCS CONCEPTS

• **Hardware** → *Batteries*; • **Software and its engineering** → *Software maintenance tools.*

KEYWORDS

Genetic Improvement, Machine Learning, Non-Functional Properties, Energy Consumption, Mobile Applications, Android

ACM Reference Format:
Mahmoud A. Bokhari, Markus Wagner, and Brad Alexander. 2020. Genetic Improvement of Software Efficiency: The Curse of Fitness Estimation . In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3398109

1 INTRODUCTION

The challenges of optimising non-functional requirements of software are mainly centred around the complexity of the search space and the estimation of the desired fitness function [8]. A large number of fitness evaluations is required to explore the interesting regions of the search space and to find near optimal solution sets. However, real-world applications tend to have expensive fitness evaluations and non-monotonic search spaces [1, 2].

In addition, software can not be evaluated in a complete isolation, noise impacts the fitness evaluation. This means the exact fitness can not be determined. Re-sampling requires even more evaluations which affects the evaluation budget considerably. The rest of this paper discuses issues involving energy optimisation of software on smart-phones and proposes two solutions.

2 ISSUES IN COMPUTING FITNESS VALUES

Current solutions to compute software energy use are by the mean of Hardware (HW) meters which requires *in-vivo* evaluation [1] or by utilising models that abstract away the real environment of software [7]. The use of HW meters helps the optimiser to gain accurate measurements, evaluate the generated variants in the real environment and helps prevent overestimating variants at the expense of the evaluation budget. On the other hand, model-based fitness functions can speed up the evaluation process and reduce the effects of noisy environments. However, the behaviour of complex systems such as PCs and smart-phones can not effectively be reduced into a simple model that describes the environment in terms of only one dependent variable. The problem with the current models is their simplicity, which defies the main purpose of evaluating the performance (run-time and energy use) of real software in realistic scenarios. For example, Li et al. [7] used a simple model that describes the energy usage only in terms of displayed colours on OLED screens while visiting web-pages. During the validation process, the authors found loading and rendering of the optimised web-pages differed from the optimisation process, even though their approach did not involve optimising such activities. This indeed impacted the outcomes of the optimisation since their fitness functions did not account for the loading and rendering phases. It is worth mentioning that OLED screen energy use was also found to be non-linear with respect to each pixel's colour [4].

In addition, non-determinism in the host system in which the software is evaluated drastically affects the obtained fitness values [9]. This means a bad solution may sometimes outperform good ones because the current state of the system which neither of *in-vivo* and simple model-based fitness evaluation can handle. For example, Burles et al. [3] used a byte-code based model to estimate the energy use of Google Guava software. Interestingly, their results show disabling Just-in-Time compilation during optimisation resulted in a slight improvement compared to enabling it during the validation. This shows that simple models failed to show how well the created solutions performed. Moreover, other factors can cause

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3398109

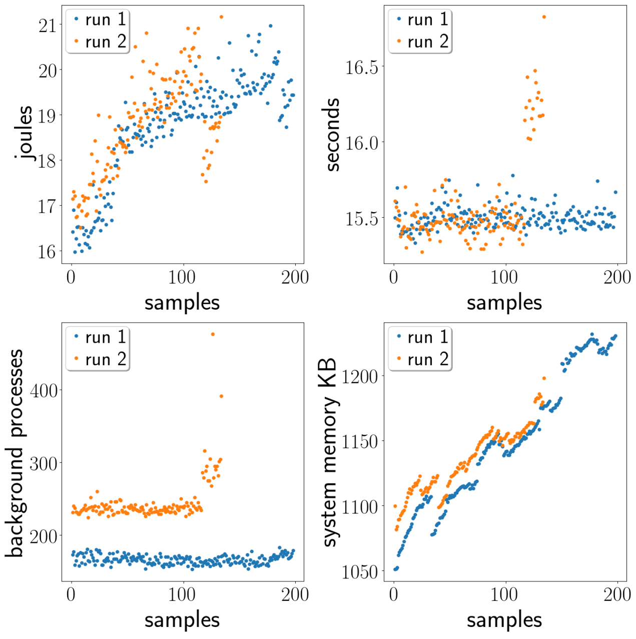

Figure 1: The system stats during running Rebound library on Nexus6 running Android 7

non-deterministic behaviours of the system such as the memory layout [9].

To illustrate how non-determinism greatly can mislead the search process on smart devices, we use a Nexus 6 platform running Android 7 and the Rebound animation library developed by Facebook as a case study. In our experiment, Rebound was executed repeatedly in two different runs. Between the runs the device was recharged and rebooted, and both runs had the same battery budget. Figure 1 shows (left to right) the energy use of the device, Rebound execution time, the number of background process, and the system memory during Rebound execution. As can be seen, there are serious issues that can drastically affect the optimiser. The initial state of the system varied in both runs which can give advantages to variants in the first run when optimising for energy or memory usage. One may argue that rebooting is not essential in optimisation. However, there exist other system states affecting the energy use in both runs. For example, at the end of the runs energy use plunged. The execution time was also impacted in the second run by a rapid increase. Interestingly, there is a significant difference between the number of processes in both runs, which affected (not primarily) the energy use and the execution time.

3 PROPOSED SOLUTIONS

The first proposed solution is to integrate machine learning approaches with GI. This helps learning complex models that take into consideration the complexity of the host environment and the non-linearity in energy usage. For example, the background processes have repeated patterns that ML can help GI to overcome their noise. Voltage drops and spikes can divert GI from interesting search spots, nevertheless, ML learns the impact of such variation and compensates for GI. Although memory usage can be argue that it has a second-order correlation with non-functional properties such as energy and run-time, its side effect reflected by garbage collection (GC) can be enormous. GC could stop the world in some virtual machines used in Android [11]. ML can also play an important role to alleviate such potential deficiencies in fitness estimation since it can effectively learn non-linearity of non-functional properties usages such as the energy use of CPU [10]. It is worth mentioning that the use of machine learning methods to model energy consumption and other non-functional properties has been used in the literature, our novelty is to employ it to optimise these properties.

Our second proposal is a synergy between learned models and *in-vivo* evaluations. The result of such a synergy is adaptive models that get re-calibrated as the optimisation proceeds. During the optimisation, the *in-vivo* evaluation takes in place to evaluate selected solutions and their samples are used to fine-tune the model. Solutions can be clustered based on their genetic materials. Representatives of those clusters are selected for the *in-vivo* evaluation. This step is important to maintain the accuracy of the fitness function since the search space can be non-monotonic [1, 2], and to mitigate uncertainties in the host system (i.e. smart-phones). In addition, this synergy does not completely eliminate the real behaviour of the system which can exhibit unseen states. It is worth mentioning that meta-models or surrogate functions have been used to solve computationally expensive optimisation problems [5, 6]. Our novelty is to utilise it in the domain of GI for software non-functional properties.

REFERENCES

[1] Mahmoud A. Bokhari, Brad Alexander, and Markus Wagner. 2018. In-vivo and Offline Optimisation of Energy Use in the Presence of Small Energy Signals: A Case Study on a Popular Android Library. In *15th EAI Int. Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. ACM.
[2] B. R. Bruce, J. Petke, M. Harman, and E. T. Barr. 2019. Approximate Oracles and Synergy in Software Energy Search Spaces. *IEEE Transactions on Software Engineering* 45, 11 (Nov 2019), 1150–1169. https://doi.org/10.1109/TSE.2018.2827066
[3] Nathan Burles, Edward Bowles, Alexander E. I. Brownlee, Zoltan A. Kocsis, Jerry Swan, and Nadarajen Veerapen. 2015. Object-Oriented Genetic Improvement for Improved Energy Consumption in Google Guava. In *Symposium on Search-Based Software Engineering (SSBSE)*. Springer, 255–261.
[4] Mian Dong, Yung-Seok Kevin Choi, and Lin Zhong. 2009. Power modeling of graphical user interfaces on OLED displays. In *2009 46th ACM/IEEE Design Automation Conference*. IEEE, 652–657.
[5] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.* 1, 2 (2011), 61–70. https://doi.org/10.1016/j.swevo.2011.05.001
[6] A. Montano L. Santana-Quintero and C. Coello. 2010. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In *Computational intelligence in expensive optimization problems*. Springer, 29–59.
[7] Ding Li, Angelica Huyen Tran, and William GJ Halfond. 2014. Making web applications more energy efficient for OLED smartphones. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 527–538.
[8] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward. 2018. Genetic Improvement of Software: A Comprehensive Survey. *IEEE Transactions on Evolutionary Computation* 22, 3 (June 2018), 415–432.
[9] M. Hauswirth P. Sweeney J. Amaral T. Brecht L. Bulej C. Click L. Eeckhout S. Fischmeister S. Blackburn, A. Diwan et al. 2016. The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 38, 4 (2016), 1–20.
[10] D. Da Cunha S. De Carvalho and A. Da Silva-Filho. 2017. Autonomous Power Management for Embedded Systems Using a Non-linear Power Predictor. In *2017 Euromicro Conference on Digital System Design (DSD)*. IEEE, 22–29.
[11] Doug Sillars. 2015. *High Performance Android Apps: Improve ratings with speed, optimizations, and testing.* O'Reilly Media, Inc.