# Genetic Improvement of Energy Usage is only as Reliable as the Measurements are Accurate

Saemundur O. Haraldsson
University of Stirling
Stirling, Scotland, UK
soh@cs.stir.ac.uk

John R. Woodward
University of Stirling
Stirling, Scotland, UK
jrw@cs.stir.ac.uk

## ABSTRACT

Energy has recently become an objective for Genetic Improvement. Measuring software energy use is complicated which might tempt us to use simpler measurements. However if we base the GI on inaccurate measurements we can not expect improvements. This paper seeks to highlight important issues when evaluating energy use of programs.

## Keywords

Genetic Improvement (GI), Search Based Software Engineering (SBSE), Genetic Programming (GP), Energy Optimization

## Categories and Subject Descriptors

I.2.2 [**Automatic Programming**]: Program modification

## 1. INTRODUCTION

*"If you can not measure it, you can not improve it."*– Lord Kelvin

Recent awareness of climate change and the importance of conserving energy has not been lost on the computer science community. Efforts have mainly been focused on large scale computing and hardware in data centres [3] and on a smaller scale, i.e. in embedded, low resource systems [10]. Energy consumption of hardware can however only be optimized to a certain degree. Energy consumption of deleting a single bit has been shown to have a lower bound [1]. Therefore, hardware and consequently general computer optimization pertaining to energy will also have a lower bound. No matter how energy efficient the hardware we produce, if we do not develop software to follow suit, we can never hope to get close to reaching the limit of Landauer's principle [4]. Like a car is only as energy efficient as the person driving it, the same applies to computers; the hardware can only be as efficient as the software running on it.

The issue of energy optimisation is to know which changes to the software will lower the energy usage. Most developers are not aware of how to optimize for energy, although they would want to [9]. Despite willingness, developers can only be expected to tackle a limited number of objectives while coding. Many can probably code functionality while optimizing for one or two non-functional features, such as memory and speed. This is assuming those properties are simple to manually configure, e.g. shortening iterations where possible or allocating sufficient but minimal memory. However, what configurations reduce energy consumption? In all likelihood, that is a question few developers could answer confidently [9]. Recent developments in automatic programming [2] could however provide the answer. Genetic Improvement (GI), a general method of automatically improving software through modifications of source code, has the potential to handle complicated, multi-objective optimizations [10, 5]. GI opens up the possibilities for clever, subtle or unintuitive adjustments that most developers would not have thought of. However with added flexibility come additional pitfalls that developers need to be aware of. Of particular interest are the fitness evaluations for energy objectives, as poor measurements do not imply actual improvements.

Physically measuring the actual energy used would be ideal for accuracy but for programs it is complicated since they do not have a direct source of energy. Instead they control multiple hardware components that share an energy source among themselves. Is there a way we can estimate the energy usage of an application with a surrogate evaluation? As it turns out, that could be even more complicated and unpredictable, with inconsistency across platforms, environments [6] and refactorings [8]. Furthermore a surrogate evaluation might give inconsistent fitness readings from one generation to another while evolving a program, having inconvenient consequences for practical use and research conclusions. When applying GI to optimize for energy use, we must be careful when constructing the fitness evaluations.

## 2. ENERGY IN COMPUTATION

To measure the energy used by a computer we have to sample the electrical connection for current drawn and voltage over a specific time period. Then we can estimate the energy used by the computer for that time period.

One could argue that there are four levels to improving energy usage in computing. 1) Hardware optimisation, designing and producing hardware that conserves energy. 2) Optimizing the OS or kernel. 3) Minimizing the amount of computing used for a particular task. 4) End user specific energy conservations like turning off features that are not in use at all times such as network devices and positioning



*GECCO '15, July 11 - 15, 2015, Madrid, Spain*



ISBN 978-1-4503-3488-4/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739482.2768421

system.

In GI we are concerned with the second and third levels where we can directly apply it to improve certain parts of the OS or specific software. Although we can specialize software to be energy efficient in isolation, we cannot prevent other applications from interacting with it. However, GI gives developers the flexibility to specialize software between platforms, computers and hardware but caution should be taken when measuring the energy used.

Physically measuring the electrical consumption of the computer and each of its physical components such as hard disk drives and graphics processing units requires direct access to the hardware. Software, however does not have a physical connection that can be measured as it operates on a system of multiple hardware components. In theory, we could measure the energy use for each component in isolation, but in practice the components interact, further complicating the measurements. Also the application itself has no control over the native scheduling algorithm, so when should the sampling of current and voltage start and finish for each component? Apart from energy use varying for different type of memory (flash, cache, HDD) and where the data is stored [7], how one program uses memory or the graphics card might also have consequences for other programs, and vice versa. Two questions then arise: Do we need to measure the energy consumption of the system after the application has terminated? How does the state of the system before execution affect the measurements?

Knowing how complicated physical measurements of program's energy use are, how can we possibly come up with a function that estimates this property? It has to be accurate enough, but also simple enough so that it is useful in practice. It must take into account the whole system; hardware, firmware, and OS while also considering the changes of the energy profile of the program being improved.

Measuring the total energy consumption of a system while a program is executing makes it only possible to draw conclusions about the program on that particular system.

# 3. OPEN ISSUES FOR ENERGY IMPROVEMENTS WITH GI

GI is a young field within computer science and software engineering and it faces numerous challenges in general. Recent incentives towards "green computing" have added to those challenges. Some of the general issues for energy efficient computing are related to measuring the energy usage. Measuring energy usage accurately however is nearly an impossible task. Not only are physical measurements estimates through multiple sampling but they also take human effort and apply only to the systems being measured. Moreover, physical measurements are presently not viable for dynamically optimizing software after it has been launched. Until now the only reliable way to measure how much energy a software consumes has been through highly specialized or customized system. Similarly to cars being rated in test conditions for their fuel consumption, the rating never accurately translates to intricacies in the real world.

Surrogate objectives are necessary to achieve dynamically adaptive "green" software. A single point of measurement as a substitute such as CPU cycles is not enough since they provide no consistency across programs or platforms. We therefore can not rely on them to make scientific conclusions nor use them in practice. We would want alternative measurements such as simulations or modelling from multiple programmatical sources so that we can reliably use them in science or at the very least make energy optimisation practical.

Extending the quote from Lord Kelvin: If you can not measure it *accurately*, you can not improve it *reliably*.

# 4. REFERENCES


[1] Antoine Bérut, Artak Arakelyan, Artyom Petrosyan, Sergio Ciliberto, Raoul Dillenschneider, and Eric Lutz. Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483(7388):187–189, 2012.

[2] Saemundur O Haraldsson and John R Woodward. Automated Design of Algorithms and Genetic Improvement : Contrast and Commonalities. In *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp '14, pages 1373–1380. ACM, 2014.

[3] Stavros Harizopoulos, Mehul A. Shah, Justin Meza, and Parthasarathy Ranganathan. Energy Efficiency : The New Holy Grail of Data Management Systems Research. In *4th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2009.

[4] R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[5] William B. Langdon and Mark Harman. Optimising Existing Software with Genetic Programming. *IEEE Transactions on Evolutionary Computation*, PP(99):118–135, 2014.

[6] Irene Manotas, Cagri Sahin, James Clause, Lori Pollock, and Kristina Winbladh. Investigating the impacts of web servers on web application energy usage. In *Green and Sustainable Software (GREENS), 2013 2nd International Workshop on*, pages 16–23. IEEE, 2013.

[7] James Pallister, Kerstin Eder, Simon J. Hollis, and Jeremy Bennett. A High-level Model of Embedded Flash Energy Consumption. In *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '14, pages 1–9. ACM, 2014.

[8] Cagri Sahin, Furkan Cayci, Irene Lizeth Manotas Gutiérrez, James Clause, Fouad Kiamilev, Lori Pollock, and Kristina Winbladh. Initial explorations on design pattern energy usage. In *2012 First International Workshop on Green and Sustainable Software (GREENS)*, pages 55–61. IEEE, 2012.

[9] Cagri Sahin, Lori Pollock, and James Clause. How do code refactorings affect energy usage? In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, pages 1–10. ACM, 2014.

[10] David R. White, John Clark, Jeremy Jacob, and Simon Poulding. Searching for Resource-Efficient Programs : Low-Power Pseudorandom Number Generators. In Maarten Keijzer et al., editor, *GECCO'08, 10th annual conference on Genetic and evolutionary computation*, pages 1775–1782. ACM, 2008.