A Cross-Platform Assessment of Energy Consumption in Evolutionary Algorithms

Towards Energy-Aware Bioinspired Algorithms

F. Fernández de Vega[1], F. Chávez[1], J. Díaz[1], J.A. García[1], P.A. Castillo[2], Juan J. Merelo[2], and C. Cotta[3]

[1] Universidad de Extremadura
{fcofdez,fchavez,mjdiaz,jangelgm}@unex.es
[2] ETSI Informática, Universidad de Granada
{pacv,jmerelo}@ugr.es
[3] ETSI Informática, Campus de Teatinos, Universidad de Málaga
ccottap@lcc.uma.es

Abstract. Energy consumption is a matter of paramount importance in nowadays environmentally conscious society. It is also bound to be a crucial issue in light of the emergent computational environments arising from the pervasive use of networked handheld devices and wearables. Evolutionary algorithms (EAs) are ideally suited for this kind of environments due to their intrinsic flexibility and adaptiveness, provided they operate on viable energy terms. In this work we analyze the energy requirements of EAs, and particularly one of their main flavours, genetic programming (GP), on several computational platforms and study the impact that parametrisation has on these requirements, paving the way for a future generation of energy-aware EAs. As experimentally demonstrated, handheld devices and tiny computer models mainly used for educational purposes may be the most energy efficient ones when looking for solutions by means of EAs.

Keywords: Green computing, energy-aware computing, performance measurements, evolutionary algorithms.

1 Introduction

In the analysis of single or multi-processor algorithm performance, an important feature is frequently forgotten: energy consumption, which largely correlates with performances provided by new processors. That is why, in an environment where raw processor speed is no longer doubling at an accelerated pace, reducing energy consumption and taking it into account when evaluating algorithms becomes an issue, to the point that latest HPC benchmarks also include this measurement in their reports and there are calls for *energy-proportional computing* [5] and *green computing* [10], a term that was born in the last decade to refer to problems associated to energy consumption in computing environments, particularly in large data centers. But this energy-aware and proportional point of view is equally applicable to desktop computers and any kind of algorithms that may be run.

When dealing with evolutionary algorithms (EAs), big efforts have been applied to improve performances while applying parallel and distributed systems [13]. Improvements have tried to analyze global quality of solutions when compared with time required to find them. But similarly as the traveler considering not only speed but also price when selecting means of transport, we should also consider energy consumption when running an algorithm, and not just the time to solution.

To the best of our knowledge, the influence of this important parameter has not been analyzed yet in the context of EAs, although its importance has already been recognized [6]. This is the main goal of this work, to make a preliminary analysis of the impact of energy consumption when running a well know EA, Genetic Programming, on different hardware architectures, so that we may in the future be aware of the importance, and even design energy-aware EAs; we will also measure the impact of a particular feature, population size, in the energy consumption, so that these parameters can be taken into account in an energy-aware design of evolutionary algorithms.

The rest of the paper is organized as follows: Section 2 describes previous works on the area; section 3 describes the experiments performed and section 4 shows the results obtained. Finally we summarize our conclusions in section 5.

2 Evolutionary Algorithms and Energy Consumption

Computer science took interest in energy efficiency a number of years ago, and a new research topic was born, *Green Computing* [10], together with the *energy-aware* [5,14] concept. Even processor makers offered new processors providing *dynamic frequency scaling*, which adapts energy consumption as well as heat dissipation to the need of the processes to be run [4,2,1].

On the other hand, EAs have already been applied as optimization algorithms in the context of energy management. We can thus find optimization problems associated to HVAC (Energy management of heating, ventilating and air-conditioning) [8], [12]. We can also find EAs applied to energy dispatch [7]. But any of the above referred problems are only tangentially connected to the problem we are interested in: how to include energy consumption as one of the main features of EAs to be considered when looking for solutions, and its relationship with the main parameters of the algorithm.

The main concept discussed in this paper is the capability of an EA to adapt to dynamic environments in which energy consumption is one of the main components to be optimized [6]. This capability, which is one of the self-$\star$ features of a given algorithm, including EAs [6], has already been considered by researchers in other kind of algorithms and computer architectures [3], in some cases an essential part of them [14]. Energy-awareness is considered a key component in infrastructures of any size, from large data centers to processor architectures for mobile devices where battery life must be optimized. Also in this context, EAs have been employed to design cache hierarchies reducing energy consumption and heat dissipation [16].

Yet, to the best of our knowledge, EAs have never been studied from the point of view of their own energy needs. Given their stochastic nature, the number of parameters regulating the way they perform the search process and the plethora of hardware platforms available to run them, we consider it of interest to study the energy consumed when looking for a solution, so that in the future they may become energy-aware and capable of self-regulating when progressing towards the solution of the problem faced. This is what we have set out to do in this paper.

3 Methodology

This preliminary study tries to measure energy consumption for the Genetic Programming (GP) algorithm. In the following subsections both the algorithm setting (3.1) and computational platforms (3.2) are presented.

3.1 Algorithmic Setting

Given the stochastic nature of this kind of algorithms, we firstly decided to run each of the experiment 30 times so that the average can be computed as an estimation of the algorithm behavior. In order to establish a fair comparison among the different hardware platforms considered, these runs are done with the same 30 random seeds so that all of the runs are exactly the same in every platform, when considering high level operations defined in the high level programming language.

On the other hand, and given the influence of computing time in the total amount of energy consumed by the algorithm, we configured the main loop of the algorithm to finish when the optimal solution is found. We are thus mainly interested in the average computing time for the 30 runs, together with the energy consumed along that time. The only differences that may arise are due to hardware differences: instruction set architecture, processor speed and operating system; features that are not the focus of this work. Nevertheless, these differences may influence future decision on the preferred hardware and operating system for the algorithms.

We must also mention the interest in studying some of the main parameters of the algorithm: they have a well-known impact on the time to find solutions, and may thus also directly, or indirectly influence the energy consumed to reach that solution. In this preliminary study we have focused on population size and have tested several values for the problem selected. Although we are working with GP and a well-known problem, this first analysis will be helpful to see that energy consumption is an important issue when working with EAs.

In order to ease the compilation processes in every hardware platform, we have selected a well known implementation of GP in the C programming lan-

Table 1: Main GP Parameters for multiplexer 6.

Max number of generations	500
Population sizes	100, 200, 400, 500, 1000
Crossover probability	0.9
Mutation probability	0.1

Table 2: Devices

Device	Processor	Cores	RAM	OS
raspberry pi	Cortex-A7 900 MHz	4	1GB	Raspbian GNU/Linux 7
tablet	Samsung Galaxy Tab 3 SM-T311, Exynos 4212 1.5 GHz	2	1.5 GB	Android 4.4.2 (kernel 3.0.31)
laptop	Intel(R) Core(TM) i5-2450M 2.5GHz	4	8GB	Ubuntu 12.04.5 LTS
iMac	Intel(R) Core(TM) i5 2.7GHz	4	4GB	OSX 10.11.4
blade	Virtual Machine (on IBM 8CPUs x 2GHz Intel Xeon CPUE5504 @ 2.00GHz (x2), 16Gb RAM)	4	4GB	Debian 6.0, 64 bits

guage: lilgp [4]. Regarding the problem selected for the experimental stage, we have selected one of the test problems provided by lilgp: the multiplexer problem. To be precise, we have set up to work with 6 bits. The main parameters of the algorithm are described in Table 1. Function and terminal sets are the standard ones as described by Koza [11].

3.2 Computational Platforms

Several computational platforms have been tested, i.e., raspberry pi, tablet, laptop, iMac and blade. Table 2 provides the details for the hardware architectures and operating systems used. Given the differences among hardware devices, we have employed different ways for measuring energy consumed by each of the algorithms.

Laptop and Raspberry Pi. Regarding the laptop and raspberry pi, we have employed a multimeter for measuring total power delivered to the device in two different scenarios: (i) when the algorithm is not running and (ii) when the algorithm is running. Starting with an initial measurement at rest (first scenario) in both cases, our multimeter is able to measure the watts delivered, which remains constant in the case of raspberry pi while the algorithm is running. So we can obtain watts delivered by the algorithm (second scenario) by simply subtracting both values. In the laptop the watts delivered vary continuously, so

[4] `http://garage.cse.msu.edu/software/lil-gp/`

we record a video in order to register all possible values and how long it lasted each one. Once all data are collected, we analyze how long each value stays with respect to the total execution time. Thus, we can accurately compute the average power delivered and, finally by subtracting the initial value, we get the power delivered by the algorithm.

Tablet. In order to collect data about energy consumption to Android devices, such as smartphones or tablets, *PowerTutor*[5] [15] has been used. This app is a diagnostic tool for analyzing system and app energy consumption. In order to obtain energy measurements of the EA, PowerTutor runs in the background and logs data on energy utilization for each app, summarizing the info in an intuitive user interface (reporting the number of Joules the app has consumed during the run).

iMac. Data collection in the iMac was done using *HardwareMonitor*[6]. This application suite includes a command-line tool that provides readings of the internal hardware sensors built on the computer. In order to obtain power measurements, a shell script is run in parallel to each run of the EA. This script gathers sensor data periodically (we use a sampling frequency of 1s) and goes to sleep state between measurements. During the experimentation, no other application is run, apart from background processes under OS control. To gauge the data, the same data-collection process is run for 100s before each batch of runs, thus providing an indication of the system basal consumption at that moment which is in turn used to compute the excess power delivered due to the EA (and hence accounting for eventual hysteretic phenomena).

Blade. Ecosystems such as clusters are often used to process big data sets. This kind of systems allow us to optimize, by sharing, resources like Ethernet, storage devices, power supply, etc. The ecosystem we use employs VMWare Esxi 5.0[7] (see Table 2) whose hypervisor provides us with information about energy consumed by both the hardware platform as well as any of the virtual machines running on it. Thus we can obtain specific data for the virtual machine running the algorithm, and thus we can compute the difference between energy consumption when the algorithm is running and when it is not running.

4 Results

As described in the previous section, computational times and power delivered for each of the devices are reported in Table 3; algorithms tested using different

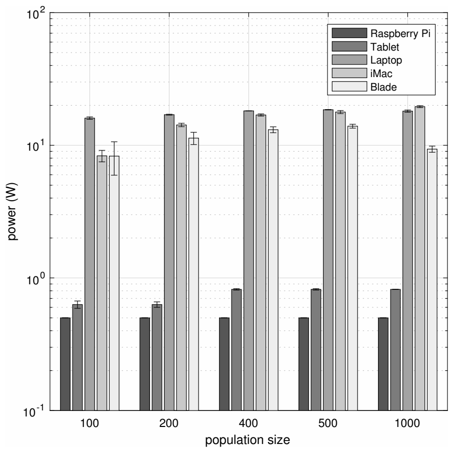
[5] `http://ziyang.eecs.umich.edu/projects/powertutor/documentation.html`

[6] `http://www.bresink.com/osx/HardwareMonitor.html`

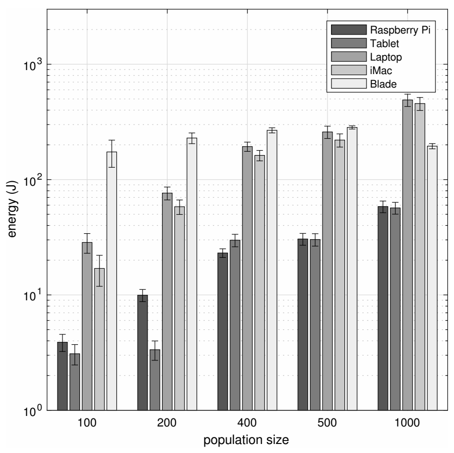
[7] `https://my.vmware.com/web/vmware/details?productId=229&downloadGroup=ESXI50`

Table 3: Time (in seconds) for lilgp-multiplexer-6 run on each system depending on the population size. The numbers denote the mean and the standard error of the mean for the 30 runs performed.

	population size				
System	100	200	400	500	1000
raspberry pi	7.77 ± 1.31	19.91 ± 2.41	46.22 ± 4.01	61.10 ± 7.19	116.80 ± 13.55
laptop	1.73 ± 0.31	4.43 ± 0.54	10.60 ± 0.97	13.89 ± 1.68	27.13 ± 3.36
iMac	1.38 ± 0.28	3.69 ± 0.48	8.98 ± 0.84	11.74 ± 1.44	22.95 ± 2.85
tablet	4.43 ± 0.75	4.85 ± 0.78	35.68 ± 4.15	36.17 ± 4.17	68.70 ± 7.89
blade	2.59 ± 0.53	6.88 ± 0.91	16.78 ± 1.59	22.28 ± 2.77	43.53 ± 5.48



(a)

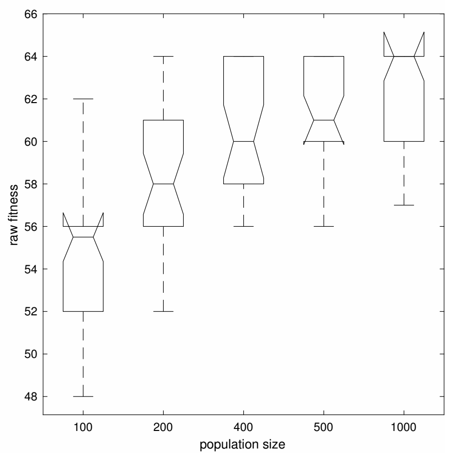


(b)

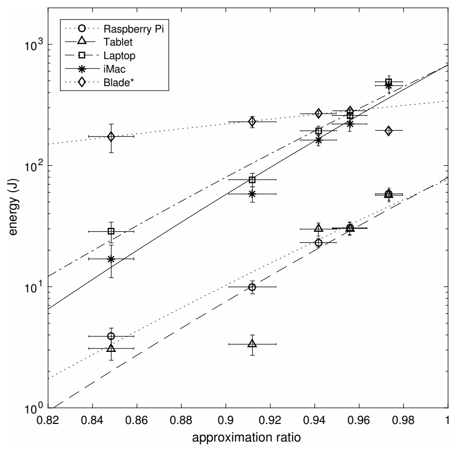
Fig. 1: (a) Average power delivered in each run. (b) Energy consumption per run. In both cases the bars (corresponding to raspberry pi, tablet, laptop, iMac and blade from left to right in each group) indicate mean values and the error bars span the standard error of the mean. Notice the logarithmic scale in the Y axis.

population sizes for each of the experiments are shown in Fig. 1a. The first thing that may be noticed is the difference in computing time among devices analyzed, which corresponds with expectations: small devices (raspberry pi and tablet) require quite a long time to reach solutions and this is typically the reason why they are not frequently used as the hardware platform to run EAs – although they are useful when non-standard distributed models are analyzed (such as pool-based models, [9]).

Nevertheless, we are considering a different point of view, and will not just focus on computing time. Different features can thus be analyzed: (i) device behavior when considering energy consumed or power delivered by the algorithm per unit of time; (ii) total energy required to find a solution and (iii) the way population size influences the energy needs when looking for a solution. If we



(a)



(b)

Fig. 2: (a) Distribution of fitness values attained for each population size (platform independent). (b) Trade-off between fitness attained and energy consumption. The data points mark mean values and the error bars indicate the standard error of the mean. A fit to a power-law $E \propto f^k$ is also included (in the case of the blade system, we have omitted the last point since it is an outlier).Notice the logarithmic scale in the Y axis.

focus firstly on the energy required by `lilgp` to be run in every device (see Fig. 1b), we notice that handheld devices (raspberry pi and tablet) are the least demanding ones, requiring an order of magnitude less energy to run the same algorithm when compared with more standard computers: iMac, laptop and blade system. Secondly, when considering the total energy required to reach the solution for the problem, the tablet running Android is the device that provides the *cheapest* solution, according to energy consumed, while blade and laptop are the most *expensive* ones in every case. Yet, if we only focus on computing time, the opposite is the case, and iMac, laptop and blade would be the preferred ones. But given that we are looking for energy-efficient ways of finding solutions by means of EAs, then raspberry pi and tablet might be preferred. Of course, this may look as a somewhat expected result a posteriori but it is nevertheless important to have obtained experimental confirmation of this fact, since different devices do not necessarily have to yield analogous trade-offs between energy consumption and speed.

In addition to absolute energy consumption values, it is also interesting to analyze how the energy requirements vary for a given device when the parameterization is changed. In this case, we have focused on population sizes for two reasons: firstly, it is an essential parameter that greatly influences the search process and can have an energy impact due to both the different behavior of the algorithm and memory management issues that might appear; secondly, its use in conjunction with a stopping condition based on generations influences the total work done as well as the quality of the solutions attained. This fact is used as a proxy to study energy consumption as a function of the attained quality of solutions in devices in which online energy measurements are not possible (and hence only the total energy consumed in a run can be measured).

We see in Fig. 1b that there is a general trend of increasing energy cost for increasing population sizes[8] which has a twofold cause: the longer computational time needed to complete the runs and the slightly larger power delivered in each case (i.e., energy spent per unit time is influenced by the population size). The latter effect can be due to issues related to memory management and is most remarkable in non-handheld devices (quite interestingly, no change is found for raspberry pi and quite small changes for the tablet). This increased energy toll does not always pay off as we can see by inspecting Fig. 2a; except for the largest population size, there does not seem to be a significant difference between the median fitness for a certain size and the immediately smaller size. A more focused perspective on this issue is shown in Fig. 2b in which we depict the energy required by each device to attain a certain fitness. The order of growth of this cost can be modeled as a power law as a first approximation. Such a power law is consistent with the superlinear cost of obtaining increasingly better approximations to the optimal solution and –while the fit can be obviously improved– it provides the means for a first comparison of these different devices. Thus, we can see how the general trends are not dramatically different for raspberry pi, tablet, laptop and iMac except for the offset of order of magnitude between the two former and the two latter. The blade system provides a more stable energy profile and can be preferred to laptop and iMac if a tight approximation to the optimum is sought. However, the smaller devices remain the best option in terms of absolute energy cost.

5 Conclusions

We have presented a preliminary study on the energy consumed by a well known EA: the Genetic Programming algorithm. We have analyzed the behavior of a benchmark GP problem, the multiplexer-6, and have run it using `lilgp` on different hardware devices running a number of operating systems, from blade systems using different Linux distributions to tablet devices running Android.

One of the first things we have learned is that although devices with better processors can run the algorithm faster, they spend larger amounts of energy, and the total energy required to find a solution is also larger. This means that although the standard preference for better hardware platforms and processors allows to find solutions more quickly, it incurs in waste of energy that should be considered: it is much more energy efficient to run the algorithm on a Raspberry

[8] In the case of the blade, we can observe that a population with 1000 individuals consumes less energy than with 500 individuals. This phenomenon is due to the processor frequency decreases because more memory is needed.

Pi or a tablet device. Of course, it is expected that some very computationally-expensive problems could not be fully solved in one of these devices. Then again, they can provide a valuable, energy-efficient contribution to the collective resolution of such problems when used within a larger ephemeral network of computing devices [6].

Secondly, we have seen the influence of one of the main parameters of the algorithm may have on the energy consumed: changing population sizes automatically produce a change in the amount of energy required to reach solutions, and this is a hint on future analysis. We should thus carefully consider how each of the parameters of the algorithm may also influence the amount of energy consumed when looking for solutions.

As a future line of work, it would be interesting to consider different devices in the same class exploring the space of solutions in a multiobjective way: which devices manage to find the solution faster for the least amount of energy? In principle the analysis is generic and does not rely on any feature that could be said to be GP-specific, so we believe the conclusions extracted are applicable to any EA. This said, it would be also interesting to further confirm this. We will thus expand the study to other evolutionary algorithms to check whether these energy profiles are exclusive of GP or there are variations among them. Energy profiling the algorithms will also allow us to find out where the energy expenses actually come from, allowing us to optimize the algorithm itself making it energy-aware.

Acknowledgements

We acknowledge support from Spanish Ministry of Economy and Competitiveness and European Regional Development Fund (FEDER) under project EphemeCH (TIN2014-56494-C4-{1,2,3}-P), from University of Granada, PROY-PP2015-06 (Plan Propio 2015 UGR), from Junta de Andalucía under project DNEMESIS (P10-TIC-6083), from Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech, from Junta de Extremadura FEDER, project GR15068 and FP7-PEOPLE-2013 IRSES Grant 612689 ACoBSEC.

References

1. Albers, S.: Energy-efficient algorithms. Communications of the ACM 53(5), 86–96 (2010)
2. Albers, S.: Algorithms for Dynamic Speed Scaling. In: Schwentick, T., Dürr, C. (eds.) 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011). Leibniz International Proceedings in Informatics (LIPIcs), vol. 9, pp. 1–11. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2011), `http://drops.dagstuhl.de/opus/volltexte/2011/2995`
3. Almeida, F., Blanco, V., Cabrera, A., Ruiz, J.: Modeling energy consumption for master–slave applications. The Journal of Supercomputing 65(3), 1137–1149 (2013)

4. Bansal, N., Kimbrel, T., Pruhs, K.: Dynamic speed scaling to manage energy and temperature. In: Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on. pp. 520–529 (Oct 2004)
5. Barroso, L.A., Hölzle, U.: The case for energy-proportional computing. Computer 12, 33–37 (2007)
6. Cotta, C., Fernández-Leiva, A., de Vega, F.F., Chávez, F., Merelo, J., Castillo, P., Bello, G., Camacho, D.: Ephemeral computing and bioinspired optimization – challenges and opportunities. In: 7th International Joint Conference on Evolutionary Computation Theory and Applications. pp. 319–324. Scitepress, Lisboa, Portugal (2015)
7. Fadaee, M., Radzi, M.: Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review. Renewable and Sustainable Energy Reviews 16(5), 3364 – 3369 (2012), `http://www.sciencedirect.com/science/article/pii/S1364032112001669`
8. Fong, K., Hanby, V., Chow, T.: {HVAC} system optimization for energy management by evolutionary programming. Energy and Buildings 38(3), 220 – 231 (2006), `http://www.sciencedirect.com/science/article/pii/S0378778805000939`
9. García-Valdez, M., Trujillo, L., Merelo, J.J., Fernández de Vega, F., Olague, G.: The evospace model for pool-based evolutionary algorithms. Journal of Grid Computing 13(3), 329–349 (2015), `http://dx.doi.org/10.1007/s10723-014-9319-2`
10. Hooper, A.: Green computing. Communications of the ACM 51(10), 11–13 (2008)
11. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
12. Lee, W.S., Chen, Y.T., Kao, Y.: Optimal chiller loading by differential evolution algorithm for reducing energy consumption. Energy and Buildings 43(2–3), 599 – 604 (2011), `http://www.sciencedirect.com/science/article/pii/S0378778810003804`
13. de Vega, F.F., Pérez, J.I.H., Lanchares, J.: Parallel Architectures and Bioinspired Algorithms, vol. 122. Springer (2012)
14. Yoo, C.M., Sungjoo, K.: Energy-Aware System Design. Springer Netherlands (2011)
15. Zhang, L., Tiwana, B., Dick, R.P., Qian, Z., Mao, Z.M., Wang, Z., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on. pp. 105–114 (Oct 2010)
16. Álvarez, J.D., Risco-Martín, J.L., Colmenar, J.M.: Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems. Journal of Systems and Software 111, 200 – 212 (2016), `http://www.sciencedirect.com/science/article/pii/S0164121215002241`