Can we measure energy consumption in population-based metaheuristics?

JJ Merelo
Department of Computer Engineering, Automatics and Robotics
University of Granada
Email: jmerelo@ugr.es
CITIC, UGR, Spain
ORCID:0000-0002-1385-9741

Cecilia Merelo-Molina
Zenzorrito, Granada, Spain
ORCID: 0000-0002-5902-0159

***Abstract*—As we proceed with incorporating energy efficiency into the design of our algorithms, establishing a robust methodology for energy profiling and the eventual comparison of algorithms or implementations is essential for a solid foundation for any further work. The main issue to overcome is the lack of specific per-process measurement of energy consumption, and above that, the fact that we are going to be measuring a system that is actively trying to optimize energy consumption itself, and doing so for the whole system, above and beyond the workload we are trying to measure. In this paper, we propose a two-stage methodology for energy profiling in population-based metaheuristics. The first stage will consist of an experimental design, i.e., how to run a series of experiments to measure the energy consumption of a specific configuration and implementation of the algorithm. The second phase will consist of the statistical processing of these results to obtain a measure of energy consumption with a certain degree of certainty. Finally, we will apply these measurements to a type of evolutionary algorithm, called the Brave New Algorithm, written in the Julia language, to determine the impact of changes at different levels on energy consumption.**

***Index Terms*—Green computing, Energy profiling, Metaheuristics**

I. Introduction

For the last few years, metaheuristics and machine learning algorithms have been achieving new levels of performance thanks to the use of more powerful processors, more powerful architectures and a greater amount of on-board memory. We already know that there is no free lunch, and this has been done at the expenses of increasing energy consumption. One of the languages that is more widely used in machine learning and scientific computing in particular, Python, is known to be one of the most energy-consuming ones [1], [2], [3], but there are decisions that need to be taken at many different levels, from algorithm parametrization down to the language and architecture used in it, whose energy consumption needs to be assessed.

Whether you want to present the reduction of energy consumption in algorithms implementation as an engineering requirement, as part of a genetic improvement framework [4] or as an ethical objective [5], the need of a methodology for comparing different configurations so that a decision can be reached is quite clear. This methodology must take into account that, nowadays, scientists seldom devote exclusive resources to running their experiments, relying on personal systems in most cases. Any measurement methodology should work on these systems giving reliable or at least actionable results.

Additionally to this overhead that should be eliminated to consider only the workload we are interested in, we should take into account that modern computers rely on automatic power management techniques that are applied directly on the processor or the chipset. Their objective is on one hand help reduce energy consumption, but on the other it will make it more difficult to measure it, since it might be working in the same direction as the optimization, or maybe the opposite, if whatever micro-optimization is introduced happens to disconnect the heuristics the system itself is utilizing. Additionally, the system is trying to do several things at the same time: keeping the system temperature below a given safe level, and lowering power consumption without reducing performance [4]. This results in a very high variability.

Experimental design itself can add another source of variability to results. In most cases, we are interested in measuring the energy consumption of a particular workload, related to the algorithm itself. However, energy measurements will yield a quantity E, which can be decomposed as $E = E_{worlkoad} + E_{overheadS} + E_{overheadM} + E_{system}$, where the overheadS corresponds to the overhead introduced by the runtime system, that is, JIT-compiling or interpreting and loading the executable or script; overheadM corresponds to the measurement overhead, that is, the energy consumed by the measurement system itself. Since we are interested in the workload only, we try to take a baseline measurement $E_{baseline} = E_{overheadM} + E_{system}$ which is then subtracted from E to compute the workload. As it can be seen, the way we obtain this baseline is crucial to the accuracy and variability of these measurements. On the other hand, we will never be able to measure directly $E_{workload}$; it will always include $E_{overheadS}$. The key here is to use a tool that reduces that overhead to a minimum, but even so, to consider the published overhead of the tool when reporting energy consumption or comparing measurements by different tools that might have a different overhead.

In this paper, we will focus on how to measure $E_{workload}$

in a specific type of evolutionary algorithm called Brave New Algorithm [6], [7], although the developed methodology is extensive to all population-based algorithms. We will focus mainly on developing a experimental design methodology so that the workload measures can be extracted successfully from the total energy consumption, avoiding non-factual negative values, as well as reducing variability of measures so that they can be compared in an actionable way so as to take decisions at any level of the algorithm implementation. The main research question is in the title: is there a statistically sound way to measure energy consumption in population-based metaheuristics? We will try to answer this question by designing a methodology that will, effectively, extract these measures in a reliably way so that actionable decisions can be taken based on them.

The rest of the paper is organized as follows: the next Section describes the state of the art in the exploration of energy consumption of evolutionary algorithms and other metaheuristics. Section III describes the algorithm and the parametrization used in this paper. Since there is no fixed methodology for measuring energy consumption, we will show how it was done for this experiment and the trade-offs it implies, together with the results, in Section IV; these results will be discussed in the last Section V along with our conclusions.

II. State of the art

Despite the growing interest in green computing, the challenges of measuring with a certain degree of accuracy the energy consumption of a particular workload, although acknowledged [8] have not been addressed in a thorough and methodological way. Von Kistowski et al. [9] introduces a methodology, called "SPEC power", which takes into account the CPU load and enables to measure the power consumed by the workload under study. The framework is rather geared towards measuring whole systems, since it uses an external, calibrated, power meter. However, working on a real system, with other processes running, would require a totally different approach. Freina et al. [10] present a more comprehensive survey of the challenges involved in using hardware or in-system software APIs to measure the amount of energy consumed by a certain workload, showing all available options with their corresponding precision and overhead. Software solutions that use model-specific registries are shown to be widely available and with a convenient tool support; their error is shown to be inferior to 5% (citing [11]) at a 20 Hz sampling frequency; however, a sampling frequency of 10Hz is advised to achieve a better temporal granularity. The same paper mentions the overhead of around 1% for most tools.

However, in front of that array of tools, there is no specific methodology or best practices for making energy measurements of specific workloads in real conditions. Von Kistowski et al. [9] mention a series of principles: Reproducibility, fairness, verifiability and usability, but what they propose is in the context of whole computers or even data centers, and also focused on hardware power meters. These are, however, guiding principles that should be taken into account in our case too.

As a matter of fact, most methodological work has arrived in this precise area, evolutionary algorithms and metaheuristics in general; Cotta et al. [12] show that measurements have a *hysteretic*, with the system staying in a high-energy-consumption state after running the workload, and thus propose a *sleep* phase after every measurement. Although a very interesting step, since it is impossible to guarantee the state the rest of the processes have left the system, however, this will not yield the desirable reproducibility in a real environment.

It certainly is if online measurements want to be used for genetic optimization of algorithm implementations, as Bokhari et al proposed in [4]. Since energy measurements are one of the possible criteria for genetic improvement, these four principles need to be applied. However, what they observe is that at least reproducibility is a challenge, since there are big variations over long periods of time. They propose a series of approaches that essentially run a single program after rebooting the system with different battery load levels. What changes is the sequence of the different variants that are under test, finally settling on an approach called R3-validation (Robin and Rotate) that runs cycles of setup plus several runs of the different variants changing the order. The stated objective of this approach is to "mitigate the effect of changing system states", concluding that this approach mitigates the effects of background energy consumption (what we call E_{system}), proving that an experimental design approach can be a way to achieve the four principles mentioned above or at least reproducibility.

What we are proposing in this paper is a methodology for measuring energy consumption of population-based algorithms in real-world conditions. We are going to test in a stratified population evolutionary algorithm called Brave New Algorithm, which we will describe next.

III. A Brave New Algorithm

The Brave New Algorithm [6] is essentially an evolutionary algorithm, using the usual mechanism of mutation, crossover and selection to find the optimum of a given function, represented as a *chromosome*, in this case a vector of real numbers. However, how selection and reproduction take place is similar to how the novel Brave New World [13] organized society: in castes.

- α generates new members by coupling the best individuals among them, to then undergo mutation and crossover.
- β caste needs to reproduce a member of the α caste, and another member of its own caste, to then undergo mutation and crossover.
- δ and ϵ castes generate new individuals by mutation. The only difference between them is that those who reproduce are chosen among the other members of the population, and that mutation rates can be set separately.
- γ generates new members by mutation, but mutation might be applied several times while the new individual is better than the previous one.

The operators used are mutation, that will change 40% of the elements of the vector generating a new, random element, 2-point crossover, and selection via binary tournament [14]. After every generation the population is re-ranked again, with the first $\%\alpha$ of the population assigned to that caste, and so on; through generations every caste will hold the same number of individuals. Since the two *upper* castes are mainly devoted to exploitation and the rest is devoted to exploration, the proportion of individuals in them is the main parameter governing the balance. There are some restrictions in these percentages, due to the way new members are generated: the beta caste needs to have twice as many members as the alpha caste; the percentage of population in the α caste is, thus, the main parameter for this, since we can use it to generate the rest. In practice, what we have done is to vary the percentage of the γ caste, the "first" that performs local search, while keeping δ and ϵ fixed to a low value.

The implementation of the algorithm used is available under a free license at https://github.com/cecimerelo/BraveNewAlgorithm.jl.

IV. Experimental methodology and results

In order to carry out these initial experiments, we have used a methodology proposed in [15]. We employ pinpoint, [16], a command line tool that taps the RAPL interface [17] to measure energy consumption of a given process. The version we have used was compiled from commit `dfee658`. This tool was validated in [18] by comparing its output to other energy-measuring tools and finding that it was less error-prone than the rest, and gave measures that were consistent with the tools such as `perf` and `likwid` when they did not fail. In general, software-based power meters offer a free alternative to more precise hardware-based meters, can be used on any system, and have been experimentally validated repeatedly [19]. The combination of the RAPL API, which places estimates in a register, and the overhead by the tool cannot be avoided, however, which is why we have to apply a careful experimental design to mitigate adverse effects. The sampling frequency we use for the RAPL counters is 10 Hz, that is, 1 sample every 100 milliseconds.

We carry experiments in an AMD Ryzen 9 9950X 16-Core Processor, with Ubuntu Linux 25.04 with kernel version 6.14.0-29-generic. Please note that AMD implements a version of the RAPL API which only allows the measurement of a single value for the processor, PKG or package, excluding memory and different components within the package. Although useful, diving into memory consumption or the consumption of other parts of the processor would be more interesting for micro-optimization, or explaining the results achieved, than for overall assessment or reduction of energy consumption, which is what we are doing in this paper. Julia version has been 1.11.7 except when noted.[1]

[1]Julia has now changed the minor version to 1.12, with 1.12.1 being the latest released at the moment of writing this paper. However, there are some performance issues with this new version, which has prevented us for using it for the time being.

We have used the Sphere function [20], defined as sum of the squares of the distance to the center minus a fixed value, which is part of the BBOB benchmarks; as a matter of fact, the implementation of this function is taken from the BlackBoxOptimizationBenchmarking.jl, also written in Julia. This function is also lightweight enough to allow us to compare the energy profiles of different combinations of genetic operators, which will then have a bigger impact on overall consumption.

Our initial experimental design includes two problem sizes: vectors with 3, 5 and 10 dimensions, to check the impact of the parameters for different problem difficulties. This corresponds to the low end of the BBOB benchmark suites; remember that we are using this suit just for the purpose of profiling the energy for this language and algorithm, so for the time being we did not find it necessary to go into higher dimensions. The function itself is not as important as the fact that it will require several seconds to run, and thus will have enough resolution for the energy sampling to work properly.

- Population size: 200 or 400 individuals, which has an impact on the diversity of the population and thus the exploitation capabilities of the algorithm.
- Maximum number of generations without improvement: 10 or 25, which is the stopping criterion for the algorithm. In this paper we have made this specific choice for stopping criterion, because if the algorithm tends towards exploration, it will explore in fruitless directions generating (possibly) useless chromosomes that will not allow it to escape a local minimum; in particular, this will have the consequence that depending on the problem dimension the execution might fall well short of an optimum value. We have left it this way, however, due to its direct relationship with exploration.
- Percentage of population in the α caste: 10%, since we have proved in [21] that is obtains the best results. The rest of the parameters will be set accordingly: β percentage will double that value. γ caste will hold a proportion equal to the α and β percentages subtracted from 90. δ and ϵ castes will always have the same proportion, each equal to 5%.

By default, every parametrization runs 30 times sequentially in a single core. Results written to standard output are processed and are available from this paper's repository at https://github.com/JJ/brave-new-green-algorithm.

One of the approaches we have used to separate $E_{workload}$ from the total energy consumption E is to repeat baseline measurements for a number of times, ideally many times so that the result is well characterized statistically. The baseline runs for all experiments were run together, and before the workload measurements.

What is observed in 1, which represents two different baseline runs, is that there are different levels of background energy consumption; for group 1, up to 2000 seconds it is substantially the same, but then it changes to a higher level (totally unrelated to an actual change in parametrization) and then eventually to a slightly higher one, around 400 Joules,

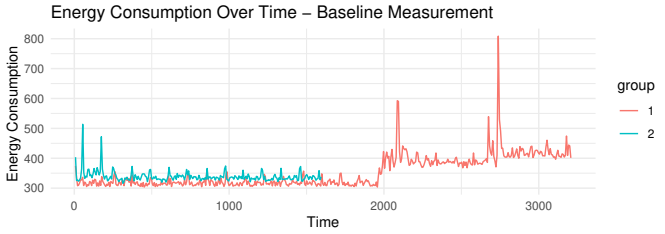

Fig. 1. PKG energy measured vs. time for a baseline measurement.

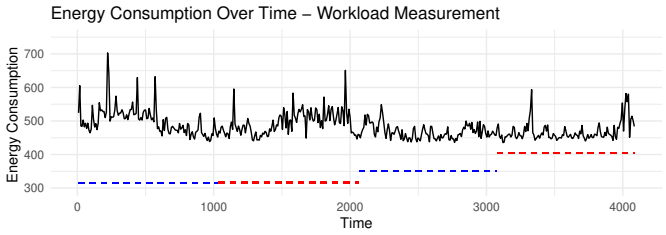

Fig. 2. PKG energy measured vs. time for workload measurements. Dashed lines correspond to baseline averages for that specific segment.

but there are peaks of more than 400 Joules. For group 2 consumption grows at the beginning, and then settles down with variations of up to 50 joules from one run to the next, with peaks of up to 150 joules; being substantially the same code, the average for group 2 is different from the one for group 1.

We can observe the effect of using this baseline in Figure 2, where the dashed lines represent the trimmed mean of the baseline for that specific configuration; these averages that will be subtracted from the workload have values that are more related to the state of the system when the baseline experiment was run, thus rendering $E_{workload}$ usable, but with a certain degree of uncertainty; in extreme cases this average will be higher than some workload energy measurement, resulting in "negative" energy averages for certain configurations [22]. Even if we use the trimmed mean for the baseline to mitigate the effect of outliers, there is still the issue that the system state is changing all the time, which means that effectively subtracting the baseline will not effectively eliminate it. But there are additional issues with this statistical treatment: it maintains the variability of the original measures, with $E_{workload}$ having the same variability as E itself and the same interquartile range. As you can see here, different measurements for the same configuration can have a difference of 200 Joules; subtracting a constant value from it will keep the same variability, thus having more or less the same reproducibility as the E measurements.

We can mitigate this effect of baseline and workload experiments finding the system in different states by using a *sequential mixed* strategy: one baseline measurement, one workload measurement with the same parameters. We are still keeping the same configuration of dimensions and population size

The graph shown in Figure 3 shows how baseline and workload measurements follow more or less the same pattern, thus mitigating the issue of the experiments finding the underlying system in different states. In this specific case, which was run for Julia version 1.11.8, the workload measurements are shown in Table I.

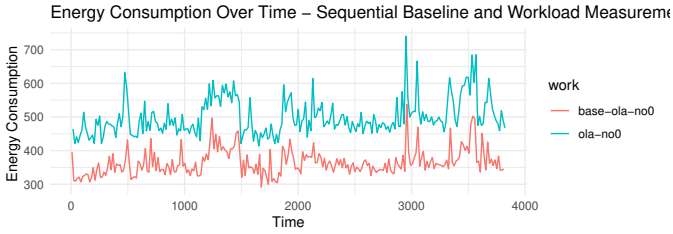

Fig. 3. PKG energy measured vs. time for sequential baseline and workload measurements. Dashed lines correspond to baseline averages for that specific segment.

Besides showing the $E_{workload}$ (represented as Delta PKG in the table), the most interesting columns are labeled IQR (for interquartile range). Two such columns are shown: for computed $E_{workload}$ and for E. Since we are subtracting variable values to compute $E_{workload}$, the IQRs are different. However, it is not always contracted: in some cases it becomes bigger (although not by much), while in some cases it is actually reduced (for instance, D=3, P=200, max gens = 25). We are increasing reproducibility by reducing this variability, but it will depend on the underlying conditions of the system.

Since the variability we observe has that saw-tooth effect, we can try and reduce additionally that variability by using a *sandwich strategy*: for every parameter, we will use two baseline measurements, subtracting the average of both from the workload. For every parameter, we will run 1 baseline experiment more than workload experiments. While we were doing 30 x 2 experiments for every parameter value, we will do now 30 x 2 +1, with every parameter combination run starting and ending with a baseline measurement.

In this case, we have also expanded the parameters we are testing to include 10 dimensions, showing the result in Table II. What we were looking for, a reduction in variability, is observed in the IQR columns, where in most cases it is cut almost in half. There are some exceptions, however. For D=3, P=400, max gens = 25 the standard deviation is very high and the initial IQR too, but it is slightly increased when the delta PKG, that is, $E_{workload}$ is computed. In general, for any run, it can happen that the variability in a segment of the experiment is too high, which will make impossible to reduce it via subtraction of two baselines.

In order to avoid this, we propose the repetition of experiments in different moments, including after rebooting the machine. We have performed 5 experimental runs of the whole set of experiments, with some reboots.

The experimental data summarized in Table III, which includes the run included in Table II, shows that, in all cases, the variability (which is obviously much higher in this case) is reduced almost 1/3rd of the original. As indicated in

TABLE I
SUMMARY STATISTICS FOR SEQUENTIAL MIXED BASELINE AND WORKLOAD MEASUREMENTS.

D	P	max gens	Delta PKG (J): Mean	Trimmed Mean	SD	IQR	PKG: IQR	Delta PKG: CI
3	200	10	146.1137	132.8572	71.76868	43.1350	31.3775	[119.31, 172.91]
3	200	25	150.8840	154.2333	43.51976	61.3175	99.2200	[134.63, 167.13]
3	400	10	130.4417	128.2961	37.33820	49.9975	45.3150	[116.5, 144.38]
3	400	25	127.2033	127.1322	20.42298	21.9475	19.6725	[119.58, 134.83]
5	200	10	141.3670	137.6856	41.70697	63.7950	96.2925	[125.79, 156.94]
5	200	25	118.9190	117.8006	37.38169	49.1825	42.0375	[104.96, 132.88]
5	400	10	131.1080	126.2894	41.66641	35.7225	40.6100	[115.55, 146.67]
5	400	25	126.1533	124.5600	35.46423	37.3650	43.2000	[112.91, 139.4]

TABLE II
SUMMARY STATISTICS FOR SEQUENTIAL MIXED BASELINE AND WORKLOAD MEASUREMENTS.

D	P	max gens	Delta PKG (J): Mean	Trimmed Mean	SD	IQR	PKG: IQR	Delta PKG: CI
3	200	10	187.8513	172.6453	114.784572	62.99125	129.2100	[144.99, 230.71]
3	200	25	195.5313	183.0269	84.498215	59.43625	113.4050	[163.98, 227.08]
3	400	10	140.8608	140.8792	4.365697	4.81500	4.4825	[139.23, 142.49]
3	400	25	186.8615	188.1747	71.574529	104.45750	102.2975	[160.14, 213.59]
5	200	10	156.4145	154.0931	29.603609	14.55500	13.2375	[145.36, 167.47]
5	200	25	186.2820	170.1056	70.430739	87.63375	155.0500	[159.98, 212.58]
5	400	10	196.5437	190.1200	91.755665	114.43875	108.0675	[162.28, 230.81]
5	400	25	205.6712	197.7369	78.235051	87.21875	119.1100	[176.46, 234.88]
10	200	10	162.1528	144.3683	123.757014	84.77250	118.1925	[115.94, 208.36]
10	200	25	187.3865	180.5606	86.651866	101.96250	122.3250	[155.03, 219.74]
10	400	10	160.5248	149.3500	61.303950	72.18750	136.0225	[137.63, 183.42]
10	400	25	138.1980	134.6769	32.901151	43.24500	59.0800	[125.91, 150.48]

TABLE III
SUMMARY STATISTICS FOR SANDWICH BASELINE AND WORKLOAD MEASUREMENTS INCLUDING REBOOTS.

D	P	max gens	Delta PKG (J): Mean	Trimmed Mean	SD	IQR	PKG: IQR	Delta PKG: CI
3	200	10	160.2228	151.2352	67.31370	85.43125	307.8625	[149.36, 171.08]
3	200	25	183.9833	176.5616	75.31423	90.09250	220.7025	[171.83, 196.13]
3	400	10	153.3072	139.4127	62.71797	55.21750	237.8225	[143.19, 163.43]
3	400	25	162.3287	155.9652	59.09720	90.90875	287.3475	[152.79, 171.86]
5	200	10	150.7041	145.7589	41.36536	51.23875	101.8050	[144.03, 157.38]
5	200	25	174.8418	148.2183	111.61211	49.10125	149.2900	[156.83, 192.85]
5	400	10	161.9803	149.1006	80.91981	69.11625	196.0825	[148.92, 175.04]
5	400	25	160.4335	147.9425	64.48569	62.02875	184.4250	[150.03, 170.84]
10	200	10	151.7899	137.1039	73.79687	53.37375	196.9925	[139.88, 163.7]
10	200	25	161.4610	140.3745	83.64334	63.22250	167.5950	[147.97, 174.96]
10	400	10	186.6787	173.3149	84.98141	110.81125	262.4575	[172.97, 200.39]
10	400	25	175.9202	161.3487	82.49049	99.63625	348.5825	[162.61, 189.23]

the introduction, due to the combination of different factors affecting E_{system}, it cannot be avoided; however, by trying to make different runs in different moments, and using robust indicators, such as the trimmed mean and 95% confidence intervals, we can finally approach a degree of reproducibility that allows us to compare different parameter configurations and take decisions based on them. In this case we can conclude that for every problem dimension using the smaller population and a maximum number of generations without change equal to 10 will consume the least energy for all problem dimensions examined. Since these measure were taken for the latest stable Julia version (1.12.4) and Linux kernel (`6.17.0-8`), we can use them as baseline for examining different versions, or to check the effect of other optimizations and genetic operator change.

V. CONCLUSION AND DISCUSSION

Creating a methodology that allows a more precise comparison of the energy consumption of different implementation of population based mechanisms open the gates to applying hyperheuristics such as irace [23]. Even if methods such as R3-validation [4] were promising first steps by trying to mitigate system variability via repetition and permutation of the configurations that are going to be compared, underlying states in the system are persistent and different enough to need further experimentation. In this paper we have proposed such a methodology for comparing energy consumption of different population-based algorithms parametrization which consists in using a *sandwich* method that puts workload experiments between two baseline measurements, and then repeating the

whole set of experiments several times, five in our case. With these combined measures we have proved that the variability is mitigated, reducing the confidence intervals, and thus yielding actionable information that can be used to take decisions on parametrizations, framework versions, or even genetic operator micro-optimizations. The issue here is that, due to the circumstances needed to run a single experiment, which even if it is not long needs machine reboots to ensure the machine being in different states, its use for online meta-optimization needs to follow a different approach, for instance considering energy a *noisy* fitness and dealing with it statistically, as proposed, for instance, in [24].

At any rate, we have answered here the main research question by proposing a statistically sound methodology, based on the reduction of variability and thus the improvement of reproducibility, for measuring energy consumption of different parametrization of population based algorithms, that focuses on the energy spent by the workload itself, eliminating to the the extent that is possible the effects of the rest of the system and the language or framework overhead. This methodology, in turn, will be used to craft the Brave New Algorithm at different levels (language, implementation, algorithm parametrization) so that it reduces its carbon footprint, at the same time it keeps or improves the fitness levels reached.

Acknowledgements

This work is supported by the Ministerio español de Economía y Competitividad (Spanish Ministry of Competitivity and Economy) under project PID2023-147409NB-C21.

References

[1] F. J. Luque-Hernández, S. Aquino-Britez, J. Díaz-Álvarez, and P. García-Sánchez, "A comparison of energy consumption and quality of solutions in evolutionary algorithms," *Algorithms*, vol. 18, no. 9, 2025. [Online]. Available: https://www.mdpi.com/1999-4893/18/9/593

[2] J. Díaz-Álvarez, M. García Arenas, A. Sánchez-Venegas, G. Romero López, F. F. de Vega, and P. A. C. Valdivieso, "Hardware and software influence on EAs power consumption," in *Advances in Computational Intelligence*, I. Rojas, G. Joya, and A. Catala, Eds. Cham: Springer Nature Switzerland, 2026, pp. 248–259.

[3] F. Nahrstedt, M. Karmouche, K. Bargieł, P. Banijamali, A. Nalini Pradeep Kumar, and I. Malavolta, "An empirical study on the energy usage and performance of Pandas and Polars data analysis Python libraries," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 2024, pp. 58–68.

[4] M. A. Bokhari, B. Alexander, and M. Wagner, "Towards rigorous validation of energy optimisation experiments," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1232–1240. [Online]. Available: https://doi.org/10.1145/3377930.3390245

[5] B. Wernaart and G. Schouten, *Moral Design and Green Technology*. Leiden, The Netherlands: Wageningen Academic, 2025. [Online]. Available: https://brill.com/view/title/71071

[6] C. Merelo, J. J. Merelo, and M. García-Valdez, "A brave new algorithm to maintain the exploration/exploitation balance," in *New Perspectives on Hybrid Intelligent System Design based on Fuzzy Logic, Neural Networks and Metaheuristics*. Springer, 2022, pp. 305–316.

[7] J. J. Merelo Guervos and C. Merelo-Molina, "Analyzing how the exploration/exploitation trade off in biologically-inspired algorithms affects energy consumption," University of Granada, 11 2025. [Online]. Available: https://hdl.handle.net/10481/107864

[8] M. F. Argerich and M. Patiño-Martínez, "Measuring and improving the energy efficiency of large language models inference," *IEEE Access*, vol. 12, pp. 80 194–80 207, 2024.

[9] J. von Kistowski, K.-D. Lange, J. A. Arnold, S. Sharma, J. Pais, and H. Block, "Measuring and benchmarking power consumption and energy efficiency," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 57–65.

[10] D. Freina, M. Jansen, and A. Trivedi, "A survey of energy measurement methodologies for computer systems," https://www.msjansen.com/assets/pdf/education/2024-dfreina-litsurvey.pdf, 2024.

[11] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 194–204.

[12] C. Cotta and J. Martínez-Cruz, "Evaluating the impact of hysteretic phenomena and implementation choices on energy consumption in evolutionary algorithms," in *Applications of Evolutionary Computation*, P. García-Sánchez, E. Hart, and S. L. Thomson, Eds. Cham: Springer Nature Switzerland, 2025, pp. 227–239.

[13] A. Huxley, *Brave new world*. DigiCat, 2022.

[14] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evolutionary Computation*, vol. 4, no. 4, pp. 361–394, 1996.

[15] J. J. Merelo, M. Garcia Valdez, and G. Romero López, "Energy consumption of evolutionary algorithms implemented in low-level languages," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 157–160. [Online]. Available: https://doi.org/10.1145/3672608.3707829

[16] S. Köhler, B. Herzog, T. Hönig, L. Wenzel, M. Plauth, J. Nolte, A. Polze, and W. Schröder-Preikschat, "Pinpoint the Joules: Unifying runtime-support for energy measurements on heterogeneous systems," in *2020 IEEE/ACM International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)*, 2020, pp. 31–40.

[17] J. A. Garcia, "Exploration of energy consumption using the Intel Running Average Power Limit interface," in *2019 IEEE Space Computing Conference (SCC)*, 2019, pp. 1–10.

[18] J. J. Merelo-Guervós, M. García-Valdez, and P. A. Castillo, "An analysis of energy consumption of JavaScript interpreters with evolutionary algorithm workloads," in *Proceedings of the 18th International Conference on Software Technologies, ICSOFT 2023, Rome, Italy, July 10-12, 2023*, H. Fill, F. J. D. Mayo, M. van Sinderen, and L. A. Maciaszek, Eds. SCITEPRESS, 2023, pp. 175–184. [Online]. Available: https://doi.org/10.5220/0012128100003538

[19] M. Jay, V. Ostapenco, L. Lefèvre, D. Trystram, A.-C. Orgerie, and B. Fichel, "An experimental comparison of software-based power meters: focus on CPU and GPU," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2023, pp. 106–118.

[20] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, 2010, pp. 1689–1696.

[21] J. J. Merelo-Guervós, C. Merelo-Molina, P. García-Sánchez, and M. García-Valdez, "Is there a (carbon-) free lunch? energy/performance tradeoffs in population-based metaheuristics," January 2026, accepted, LION 20.

[22] J. J. Merelo, G. Romero López, and M. García-Valdez, "Analyzing per-component energy consumption of evolutionary algorithms implemented in low-level languages: Comparison of C++ and zig in key evolutionary algorithm functions," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '25 Companion. New York, NY, USA: Association for Computing Machinery, 2025, p. 139–142. [Online]. Available: https://doi.org/10.1145/3712255.3726694

[23] L. Pérez Cáceres, F. Pagnozzi, A. Franzin, and T. Stützle, "Automatic configuration of GCC using Irace," in *Artificial Evolution*, E. Lutton, P. Legrand, P. Parrend, N. Monmarché, and M. Schoenauer, Eds. Cham: Springer International Publishing, 2018, pp. 202–216.

[24] J. Merelo, Z. Chelly, A. Mora, A. Fernández-Ares, A. I. Esparcia-Alcázar, C. Cotta, P. de las Cuevas, and N. Rico, "A statistical approach to dealing with noisy fitness in evolutionary algorithms," *Studies in Computational Intelligence*, vol. 620, p. 79 – 95, 2016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84949818091&doi=10.1007%2f978-3-319-26393-9_6&partnerID=40&md5=5c243b4b43eb1159ab92938624c16aa7