Jamil and Kor *Green Technology, Resilience, and Sustainability* (2022) 2:1
https://doi.org/10.1007/s44173-021-00001-9

Green Technology, Resilience, and Sustainability

ORIGINAL RESEARCH Open Access

Analyzing energy consumption of nature-inspired optimization algorithms

Mohammad Newaj Jamil and Ah-Lian Kor*

Abstract

Nature-Inspired Optimization (NIO) algorithms have become prevalent to address a variety of optimization problems in real-world applications because of their simplicity, flexibility, and effectiveness. Some application areas of NIO algorithms are telecommunications, image processing, engineering design, vehicle routing, etc. This study presents a critical analysis of energy consumption and their corresponding carbon footprint for four popular NIO algorithms. Microsoft Joulemeter is employed for measuring the energy consumption during the runtime of each algorithm, while the corresponding carbon footprint of each algorithm is calculated based on the UK DEFRA guide. The results of this study evidence that each algorithm demonstrates different energy consumption behaviors to achieve the same goal. In addition, a one-way Analysis of Variance (ANOVA) test is conducted, which shows that the average energy consumption of each algorithm is significantly different from each other. This study will help guide software engineers and practitioners in their selection of an energy-efficient NIO algorithm. As for future work, more NIO algorithms and their variants can be considered for energy consumption analysis to identify the greenest NIO algorithms amongst them all. In addition, future work can also be considered to ascertain possible relationships between NIO algorithms and the energy usage of hardware resources of different CPU architectures.

Keywords: Nature-inspired optimization algorithms, Energy consumption, Carbon footprint, Environmental impact, Green software, Microsoft Joulemeter, UK, DEFRA

1 Introduction

Optimization is a commonly encountered mathematical problem in many disciplines such as engineering, business activities, industrial designs, etc. Many real-world applications in science and engineering encompass various forms of optimization such as: minimizing energy consumption, cost or maximizing performance and efficiency. Traditional optimization algorithms for real-world problems are predominantly highly nonlinear with large numbers of local optima accompanied with a set of complex nonlinear constraints [1]. On the other hand, nature-Inspired Optimization (NIO) algorithms are population-based metaheuristics that mimic diverse phenomena in nature [2]. They circumvent local optima more successfully in contrast to traditional optimization algorithms. Therefore, they are widely used for solving highly nonlinear real-world optimization problems in various fields such as manufacturing, environmental engineering, finance, biomedical, etc.

Due to the proliferation of mobile and IoT devices, software energy consumption should be taken into account during the design of applications for high-performance and mobile computing systems. Software applications could be made more energy-efficient and environment-friendly by means of their optimized algorithms and data structures. In the past decades, an algorithm's performance was evaluated based on runtime, and by default, it was the only performance metric that was considered for the analysis and optimization for an algorithm [3]. However, in recent years, the extensive growth of high-performance computers and embedded systems with

*Correspondence: A.Kor@leedsbeckett.ac.uk
School of Built Environment, Engineering and Computing, Leeds Beckett University, Leeds LS1 3HE, UK

faster processors have resulted in increased energy consumption. Therefore, the energy usage of an algorithm is a crucial element that needs to be factored in for the evaluation of an algorithm (i.e. in terms of performance and sustainability). The effectiveness and efficiency of an algorithm need to be assessed in the context of a target application because its deployment would have an impact on energy consumption and environmental impact.

Carbon footprint is one way to assess the environmental impacts of ICT. Companies can make the software an integral part of their sustainability efforts by assessing its energy efficiency level, associated carbon footprint and deploying it to green business operations or processes [4]. Over the past few years, the deployment of Machine Learning (ML) models has grown exponentially in size [5]. The associated energy consumption and cost involved for training as well as building ML models has become a growing concern [6]. It is found that designing and training translation engines in Natural Language Processing (NLP) can emit between 0.6 and 280 tonnes of CO_2 [7]. Hence, an algorithm's carbon footprint needs to be appropriately addressed during its design, implementation, and deployment.

Currently, there are more than a hundred existing NIO algorithms and their variants in literature [2]. However, this study aims to analyze energy consumption and corresponding carbon footprint for four widely used NIO algorithms, namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC) algorithm. These four algorithms have been considered in this study due to their wide ranging application areas. This research aims to demonstrate how the energy consumption of these algorithms could be empirically assessed. Note that other NIO algorithms could be addressed in future work.

Genetic Algorithm (GA) evolves from the Darwinian evolution of biological systems, where the main characteristics are three genetic operators, namely crossover, mutation, and selection [8]. It has been applied to solve complex problems such as computer-automated design [9], power electronics design [10], gene expression profiling analysis [11], bioinformatics multiple sequence alignment [12], facility layout problem [13, 14], etc. However, a limitation of GA is its premature convergence that can lead to the loss of alleles, which makes it difficult to identify a gene [15].

Particle Swarm Optimization (PSO) is derived from the swarm intelligence of flocking of birds and schooling of fishes in search of food, where each particle contains its own velocity and position [16]. It has been used to address several problems such as software cost estimation [17], human motion tracking [18], resource allocation in the cloud [19], assembly line balancing [20], data clustering [21], etc. However, a limitation of PSO is it easily falls into local optimum in high-dimensional space and has a low convergence rate in the iterative process [22].

Differential Evolution (DE) is formulated on the vector population evolution, where new individuals are generated by differential crossover and mutation operators [23]. It has been used in tackling many different problems such as electromagnetic inverse scattering problems [24], electromagnetic imaging [25], antenna array design [26], robot motion planning and navigation [27], etc. However, a limitation of DE is its unstable convergence, and it easily falls into regional optimum [28].

Artificial Bee Colony (ABC) algorithm is elicited from the food searching behavior of honey bee swarm, where the swarm comprises three components, namely food source, employee foragers, and unemployed foragers [29]. It has been applied to solve a number of problems such as economic dispatch problem [30], traveling salesman problem [31], mobile robot path planning [32], load balancing in the cloud [33], image segmentation [34], etc. However, a limitation of the ABC algorithm is it converges slowly in the process of searching and easily suffers from premature convergence [35].

A set of objectives to support the aim of this study is as follows:

- Conduct a critical literature review on the environmental impact of ICT, green or energy-efficient software, the impact of hardware energy consumption by software, and the analysis of energy consumption in algorithms implementations as well as nature-inspired algorithms and energy efficiency.
- Implement the four NIO algorithms using MATLAB and conduct a set of experiments for measuring the energy consumption of these algorithms by utilizing a commonly used benchmark function, named Sphere Function.
- Estimate the equivalent carbon footprint of each algorithm based on the amount of energy consumed by each algorithm.
- Investigate whether the average energy consumption of these four algorithms is significantly different or not by performing a one-way ANOVA (Analysis of Variance) test.

Several existing work has conducted a comparative analysis of the energy consumption of different programming languages [36–38] and sorting algorithms implementations [39–42]. To date, in existing literature, energy consumption and carbon footprint of NIO algorithms have not been analyzed to the best of our knowledge.

Hence, the novel contribution of this study is to analyze the energy impact of NIO algorithms during execution in MATLAB by conducting experiments with a benchmark function. The limitation of this research is that only four types of NIO algorithms have been investigated. However,

the energy consumption-related experiment procedures could be replicated for all other non-listed NIO algorithms. In addition, the equivalent carbon footprint of each algorithm has been estimated based on their energy consumption, and an ANOVA test has been conducted to test the significant difference in the average energy consumption among these algorithms. This study will help developers choose the greenest NIO algorithm to solve a particular domain problem when energy consumption minimization is the top priority.

The remainder of this article is organized as follows: a literature review on the environmental impact of ICT, green or energy-efficient software, the influence of hardware energy consumption by software, and related works on the analysis of energy consumption in algorithms implementations as well as nature-inspired algorithms and energy efficiency is presented in Section 2. Section 3 provides a brief overview of methodologies, which include both macro and micro methodology as well as experiment setup and design. Findings and discussion are presented in Section 4, which discusses energy consumption and corresponding carbon footprint of each algorithm, the ANOVA test, and ethical issues and challenges of this study. A summary of the discussion and recommendations for future studies is presented in Section 5.

2 Literature review

2.1 Environmental impact of ICT

The ICT sector is accountable for two percent of global carbon emissions, while the rest of the sectors, for example, health, transportation, education, etc., are responsible for the remaining 98% of carbon emissions [43]. The increase of carbon emissions due to GreenHouse Gases (GHGs) and other factors will have a negative impact on the environment as well as the economy [44]. Due to the increasing global demand for ICT products and services, the ICT sector can play a crucial role in reducing global carbon emissions by minimizing the carbon footprints of its products and services. Though a lot of research has been undertaken for making hardware and other embedded systems energy-efficient [45–48], a similar focus needs to be given to the development of energy-efficient software and applications [49, 50].

2.2 Energy-efficient software

A piece of software is labeled as green or energy-efficient when it consumes less energy for its efficient computation that results in minimal adverse effects on environment [51]. Several research work has been conducted on the investigation of energy efficiency of web-based software application [52] and software features [53]. The direct impact of software on a laptop or mobile battery can easily be measured as approximately 25% to 40% of the total energy consumed by a device [54]. However, the indirect impact of software is comparatively harder to assess as they are related to the life cycle of the host device [55]. An energy-efficient or green software can be achieved when its positive and negative impacts are duly considered during the design as well as deployment phase. In view of this, optimization of ICT application services is highly necessary to lessen unpropitious effects on the environment.

2.3 Impact of hardware-Related energy consumption by software

The hardware-related energy consumption behavior of software has a direct influence on the energy consumed by hardware as well as the battery life of a device [56]. A poorly designed software or application can nullify several energy-efficient features embedded into the hardware that may eventually increase the energy consumption of a device [57]. For example, it can inhibit energy-saving features of hardware as well as affect hardware utilization, which may ultimately increase indirect energy consumption [58]. One of the most challenging tasks during the design phase of an embedded system is the development of energy-efficient software that makes a piece of hardware more energy-efficient as well. Therefore, some trade-offs between performance and sustainability will have to be considered so that software and applications can be made more productive yet energy-efficient [59].

2.4 Analysis of energy consumption in algorithms implementations

In existing literature, there is relatively little work that provides insights into algorithm implementation-related energy consumption. Rashid and colleagues have analyzed the energy efficiency of four sorting algorithms, namely Bubble, Merge, Quick, and Counting sort [39]. In this study, an experiment was set up on an ARM-based device, and the energy consumption of four sorting algorithms implemented in three programming languages was measured. This study found that the ARM assembly language implementation of Counting sort was the greenest solution.

Measurement of the energy consumption for five sorting algorithms, namely Bubble, Insertion, Quick, Selection, and Counting sort, has been conducted in [40]. In this study, five different Apps were developed for each sorting algorithm to measure energy consumption. This study found that Quick sort is the most energy-efficient sorting method in average cases, while Bubble sort is the most energy-consuming algorithm.

Deepthi and colleagues have conducted experiments to study how different sorting algorithms have an impact on energy consumption using C language implementation [41]. This study found that both time and energy have an impact on the efficiency of these sorting algorithms. This

study considered six sorting algorithms, namely Quick, Merge, Shell, Insertion, Selection, and Bubble sort. It has been found that the energy consumption of Quick, Merge, and Shell sort is similar while Insertion and Selection sorts are far better than Bubble sort.

Ayodele and colleagues have carried out a comparative experimental analysis of the energy consumption of Quick, Merge, and Insertion sort algorithms using three programming languages (C, Java, and Python) and two algorithm implementation styles (Iterative and Recursive) [42]. This study found that data size, the programming language used, and algorithm implementation styles affect the total energy consumption. Besides, this study provides information for choosing sorting algorithm type and its algorithm implementation style in order to minimize energy consumption.

Though several existing research related to energy consumption analysis of different sorting algorithms implementations have been performed, the energy consumption of NIO algorithms has not been analyzed till now to the best of our knowledge. Therefore, the initial motivation and primary focus of this study is to analyze the energy consumption of four NIO algorithms based on a benchmark function.

2.5 Nature-inspired algorithms and energy efficiency

Existing nature-inspired algorithms research primarily addresses the following areas of research: optimization [1, 2, 60, 61] using metaheuristics [62] or heuristics [63] approaches; greening processes, for example greening the supply chain [64], smart energy management [65], data center energy efficiency [66]; energy efficiency [67] and energy optimization [68] in wireless sensor network clustering. Our critical literature review has shown that to date, there is no research on energy efficient nature-inspired algorithms and thus, our research aims to address this identified gap.

3 Methodology

3.1 Macro methodology

In this study, an LCA (Life Cycle Assessment) methodology for ICT GNS (Goods, Networks, and Services) [69] has been considered as the macro methodology. It is a systematic analytical method to assess the potential environmental effects of ICT GNS in a consistent and transparent manner. It has a cradle-to-grave scope that comprises four life cycle stages, namely Raw Material Acquisition (RMA), Production, Use, and End-of-Life Treatment (EoLT). The ICT LCA application considered in this study is the assessment of energy consumption as well as evaluation of an environmental impact, namely carbon footprint of four NIO algorithms and comparative analysis between them. Figure 1 shows the structure of the LCA methodology specification for ICT GNS.

The first phase of LCA is General Requirements, which consists of high-level requirements of life cycle stages that will be assessed in the context of the problem. In this study, the four NIO algorithms have been considered as an ICT service (application), and their use stage has been considered, where the algorithms impact on energy use. The second phase is Goal and Scope Definition, where the goal phase states the intended application, reasons for conducting the study as well as the intended audience to whom the results of the study will be beneficial, while the scoping phase identifies the building blocks and system boundaries of the ICT GNS. In this study, the goal and scope have been defined in Section 1.

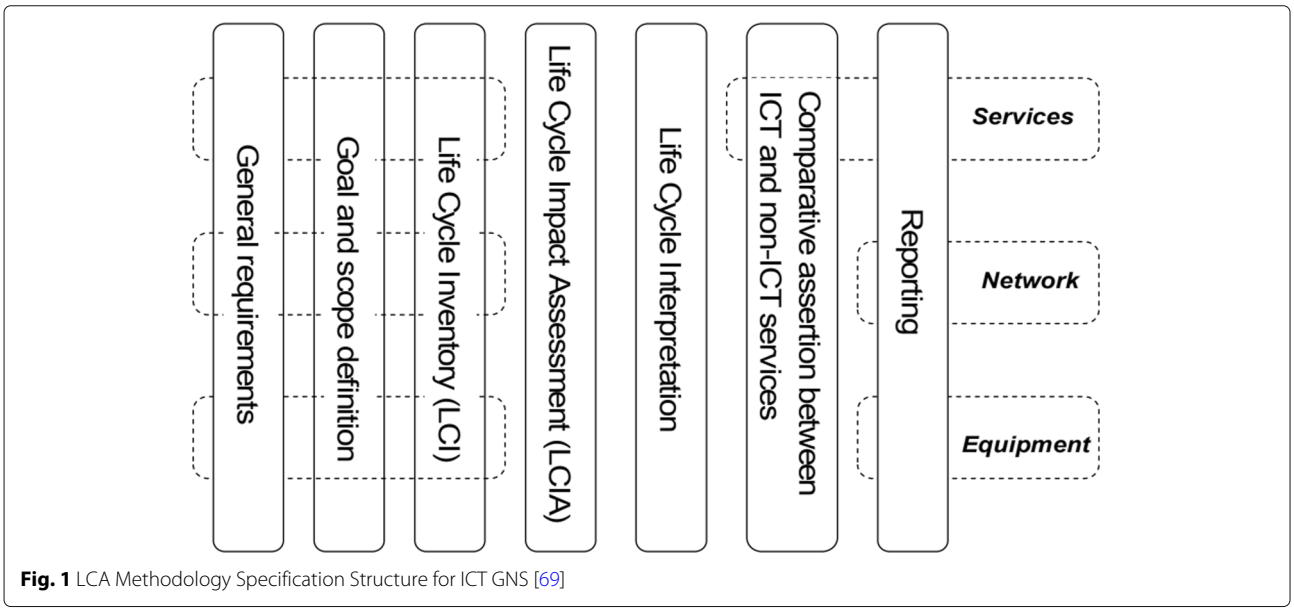

Fig. 1 LCA Methodology Specification Structure for ICT GNS [69]

The third phase is the Life Cycle Inventory (LCI) which includes data collection within the system boundary and analysis of the collected data to quantify the inputs and outputs of a unit process. In this study, a discussion of data collection has been presented in Sections 3.2 and 3.4, while the analysis of the collected data has been presented in Sections 4.1 and 4.3. The fourth phase is the Life Cycle Impact Assessment (LCIA) that estimates the potential environmental impact quantified in the LCI analysis. In this study, the potential environmental impact has been estimated in Section 4.2.

The fifth phase is the Life Cycle Interpretation, where findings from the LCI analysis and LCIA are considered together to answer queries raised in the goal definition as well as draw conclusions, identify limitations and provide recommendations. The sixth and final phase is reporting which summarizes the various reporting requirements and additional reporting considerations in order to make evidence-based decisions. In this study, a summary of the discussion and some scope of improvement for future studies have been presented in Section 5.

3.2 Micro methodology

The micro methodology involves the use of a number of tools or software, as discussed below.

3.2.1 Data collection

The four NIO algorithms considered in this study have been gathered from existing literature [8, 16, 23, 29] and implemented accordingly using MATLAB.

3.2.2 Energy profiling

The energy consumption of each NIO algorithm has been estimated using Microsoft Joulemeter software [70], which has the ability to measure the energy consumed by a running application or software and individual hardware resources, such as CPU, Monitor, Disk, and Idle or Base power.

3.2.3 Carbon footprint

The DEFRA guideline for greenhouse gas conversion factors [71] has been used for calculating carbon footprint corresponding to the energy consumption of each algorithm. After obtaining energy consumption in terms of kilowatt-hours (kWh), the recorded data is converted from kWh into equivalent CO_2 emission based on the emission factor for electricity consumption ($0.25319\ kgCO_2e/kWh$).

3.3 Experiment setup

3.3.1 System specification

Different hardware specifications would bring about different results. Therefore, all experiments were conducted on a laptop with the following specifications shown in Table 1.

Table 1 System Specification

Specification of Laptop Used	
Model	DELL Inspiron 5448
Operating System	Windows 8.1 Pro, 64bit
Processor	Intel core i3-5005U, 2.00GHz, 2000 Mhz
RAM	4GB
Storage	1TB

3.3.2 Calibrating Joulemeter

Since Windows 8.1 Pro operating system lacks the support for automatic calibration (in Joulemeter), the calibration has been done manually based on the Joulemeter user manual, which is shown in Fig. 2.

3.4 Experiment design

All four NIO algorithms have been implemented using MATLAB programming language. A commonly used benchmark function, namely sphere function, has been used for measuring the energy consumption of these four algorithms. The sphere function can be written as:

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{1}$$

where search space, $x_i \in [-5.12, 5.12]$, for $i = 1, ..., d$, and global minimum, $f(x^*) = 0$, at $x^* = (0, ..., 0)$.

For each algorithm, the dimension size (D) has been taken as 50, population size as $10 * D$, maximum iteration as $100 * D$, and optimal value to reach is set as 10^{-8}. All algorithms have been executed ten times separately, and during each run, the estimated energy consumption of each algorithm has been captured by Joulemeter.

For each algorithm, the dimension size, population size, maximum iteration, optimal value to reach, and the number of execution have been considered the same to analyze the measurements' consistency and avoid outliers.

The corresponding result of each algorithm has been stored in a separate CSV file. All results of each algorithm have been aggregated in an Excel file for analyzing the energy consumption of these four algorithms. The design of experiments can be summarized as follows.

- Nature-Inspired Optimization (NIO) Algorithms: {GA, PSO, DE, ABC}
- Programming Language: MATLAB
- Benchmark Function: Sphere Function
- Search Space: [-5.12, 5.12]
- Dimension Size (D): 50
- Population Size: $10 * D$
- Maximum Iteration: $100 * D$
- Optimal Value to reach: 10^{-8}

4 Findings and discussion

4.1 Energy consumption

The details of energy consumed by each optimization algorithm for all experiment runs are shown in Table 2,

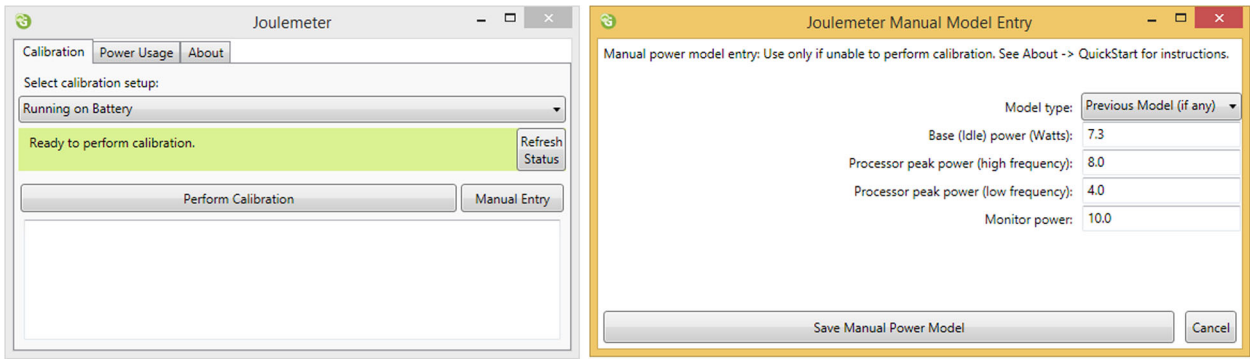

Fig. 2 Calibration of Joulemeter

while Table 3 depicts the aggregated data for all the experiments conducted for each algorithm.

The convergence time of each algorithm is different. Therefore, to perform a fair comparison and maintain the measurements' consistency between four algorithms, the corresponding energy consumption of each algorithm is normalized for $t = 1$s, which is shown in Table 4.

For better illustration, Fig. 3 shows a comparison of normalized hardware energy consumption for each algorithm, while Fig. 4 depicts the energy consumption by the hardware and application for each algorithm.

From Table 4 and Fig. 3, it can be seen that the DE algorithm consumes the least energy, followed by PSO and GA, while ABC algorithm has the highest energy consumption.

As mentioned earlier, different hardware specifications would bring about different results. Therefore, if the experiments are conducted on a laptop with different specifications, the result will vary.

From Fig. 4, it can be observed that the energy consumption of hardware that runs the application is much higher compared to the energy consumed by the application for each algorithm. In order to facilitate the reduction of the overall energy consumption, more research and investigation need to focus on the optimization of interaction and processes between hardware and software [52].

Since DE algorithm is found to have the lowest energy consumption, it is used as a base to investigate the energy consumption ratio of other algorithms which is shown in Table 5. From Table 5, it is obvious that running ABC algorithm consumes almost four times more energy compared to DE algorithm, while the energy consumed by GA algorithm is approximately double in comparison with DE algorithm. The energy consumed by PSO is comparatively lower than ABC and GA algorithms.

4.2 Carbon footprint

The carbon footprint of each algorithm has been calculated by converting the aggregated total energy consumption (kWh) into equivalent CO_2 emission based on the emission factor for electricity consumption ($0.25319/kgCO_2e/kWh$) [71], which is shown in Table 6.

From Table 6, it can be seen that DE algorithm has the lowest carbon footprint, followed by PSO and GA, while ABC algorithm has the highest carbon footprint among all algorithms.

4.3 One-way ANOVA (Analysis of variance) test

A statistical parametric test, named one-way ANOVA, has been conducted to verify whether the average energy consumption per second of these four algorithms is significantly different or not at the significance level ($\alpha = 0.05$) using Excel Data Analysis ToolPak.

The statistical hypotheses for one-way ANOVA are formulated as follows.

- Null Hypothesis: The average energy consumed by each algorithm is equal.
- Alternative Hypothesis: The average energy consumption of each algorithm is significantly different.

The null hypothesis for the test will be rejected if the 481 p-value is less than the significance level ($\alpha = 0.05$) [72] or otherwise. In order to understand the energy impact of each algorithm, the ANOVA test is performed on the average energy consumption per second (J/s) of each algorithm for all experiment runs, as shown in Table 7. Results obtained after the ANOVA test are depicted in Table 8.

From Table 8, it is evident that the p-value ($5.85E-37$) is less than the significance level ($\alpha = 0.05$), so the null hypothesis is rejected. Therefore, it can be concluded from the ANOVA test that the average energy consumed by each algorithm is significantly different.

Table 2 Energy Consumption of Each Algorithm for all Experiments

Algorithm	Experiment No	Hardware Power Consumption (W)					Application (W)	Total Power Consumption (W)	Total Time (s)	Total Energy Consumption (KJ)	Average Energy Consumption per second (J/s)	Aggregated Average Energy Consumption per second (J/s)
		CPU (W)	Monitor (W)	Disk (W)	Base (W)	Total Hardware Power (W)						
Genetic Algorithm (GA)	1	272.600	318.000	0.000	773.800	1364.400	247.500	1611.900	106.518	171.70	1611.90	
	2	263.200	315.000	0.000	766.500	1344.700	242.900	1587.600	105.748	167.89	1587.60	
	3	204.700	243.000	0.000	591.300	1039.000	188.700	1227.700	81.431	99.97	1227.70	
	4	241.800	282.000	0.000	686.200	1210.000	215.100	1425.100	94.789	135.08	1425.10	
	5	228.000	273.000	0.000	664.300	1165.300	209.700	1375.000	91.371	125.64	1375.00	
	6	252.900	306.000	0.000	744.600	1303.500	235.800	1539.300	102.510	157.79	1539.30	
	7	211.200	249.000	0.000	605.900	1066.100	190.700	1256.800	83.556	105.01	1256.80	
	8	243.900	285.000	0.000	693.500	1222.400	219.100	1441.500	95.479	137.63	1441.50	
	9	213.900	261.000	0.000	635.100	1110.000	199.900	1309.900	87.579	114.72	1309.90	
	10	255.600	306.000	0.000	744.600	1306.200	233.900	1540.100	102.566	157.96	1540.10	1443.33
Particle Swarm Optimization (PSO)	1	149.500	180.000	0.000	438.000	767.500	139.500	907.000	60.460	54.84	907.00	
	2	149.500	177.000	0.000	430.700	757.200	138.100	895.300	59.477	53.25	895.30	
	3	150.900	174.000	0.000	423.400	748.300	135.900	884.200	58.164	51.43	884.20	
	4	155.700	171.000	0.000	416.100	742.800	136.200	879.000	57.250	50.32	879.00	
	5	158.100	177.000	0.000	430.700	765.800	142.600	908.400	59.335	53.90	908.40	
	6	144.000	171.000	0.000	416.100	731.100	133.600	864.700	57.365	49.60	864.70	
	7	150.700	174.000	0.000	423.400	748.100	138.100	886.200	58.403	51.76	886.20	
	8	148.900	171.000	0.000	416.100	736.000	134.500	870.500	57.150	49.75	870.50	
	9	153.300	171.000	0.000	416.100	740.400	136.600	877.000	57.443	50.38	877.00	
	10	146.300	168.000	0.000	408.800	723.100	132.100	855.200	56.271	48.12	855.20	883.08
Differential Evolution (DE)	1	105.400	153.000	0.000	372.300	630.700	101.500	732.200	51.198	37.49	732.20	
	2	106.500	150.000	0.000	365.000	621.500	101.800	723.300	50.107	36.24	723.30	
	3	108.200	147.000	0.000	357.700	612.900	102.300	715.200	49.323	35.28	715.20	
	4	105.400	147.000	0.000	357.700	610.100	99.800	709.900	49.079	34.84	709.90	
	5	105.500	150.000	0.000	365.000	620.500	101.800	722.300	50.228	36.28	722.30	
	6	104.400	150.000	0.000	365.000	619.400	98.800	718.200	50.294	36.12	718.20	
	7	105.800	150.000	0.000	365.000	620.800	100.900	721.700	50.342	36.33	721.70	
	8	110.400	147.000	0.000	357.700	615.100	100.900	716.000	49.317	35.31	716.00	
	9	103.900	144.000	0.000	350.400	598.300	98.100	696.400	48.319	33.65	696.40	
	10	106.500	150.000	0.000	365.000	621.500	99.500	721.000	50.250	36.23	721.00	717.75
Artificial Bee Colony (ABC) Algorithm	1	397.600	528.000	0.000	1284.800	2210.400	377.300	2587.700	176.611	457.02	2587.70	
	2	395.000	537.000	0.000	1306.700	2238.700	371.400	2610.100	180.079	470.02	2610.10	
	3	404.000	540.000	0.000	1314.000	2258.000	382.500	2640.500	180.691	477.11	2640.50	
	4	388.400	531.000	0.000	1292.100	2211.500	371.000	2582.500	178.104	459.95	2582.50	
	5	391.500	528.000	0.000	1284.800	2204.300	369.600	2573.900	176.851	455.20	2573.90	
	6	393.600	549.000	0.000	1335.900	2278.500	369.100	2647.600	184.169	487.61	2647.60	
	7	407.200	567.000	0.000	1379.700	2353.900	379.000	2732.900	190.420	520.40	2732.90	
	8	391.000	540.000	0.000	1314.000	2245.000	372.800	2617.800	181.169	474.26	2617.80	
	9	393.500	537.000	0.000	1306.700	2237.200	368.000	2605.200	180.032	469.02	2605.20	
	10	393.100	519.000	0.000	1262.900	2175.000	370.500	2545.500	173.765	442.32	2545.50	2615.54

Table 3 Aggregated Energy Consumption for Each Algorithm

Algorithm	Aggregated Average Hardware Energy Consumption (KJ)					Aggregated Application (KJ)	Aggregated Total Time (s)	Aggregated Total Energy Consumption (KJ)	Aggregated Average Energy Consumption per second (J/s)
	CPU (KJ)	Monitor (KJ)	Disk (KJ)	Base (KJ)	Total Hardware Energy (KJ)				
Genetic Algorithm (GA)	229.10	272.27	0.00	662.53	1163.91	209.49	951.55	1373.40	1443.33
Particle Swarm Optimization (PSO)	87.61	100.84	0.00	245.39	433.84	79.51	581.32	513.35	883.08
Differential Evolution (DE)	52.93	74.19	0.00	180.53	307.65	50.12	498.46	357.77	717.75
Artificial Bee Colony (ABC) Algorithm	712.78	969.25	0.00	2358.50	4040.52	672.39	1801.89	4712.91	2615.54

Table 4 Normalized Energy Consumption for Each Algorithm (for $t = 1$s)

Aggregated Energy Consumption for t = 1s	Aggregated CPU (J)	Aggregated Monitor (J)	Aggregated Disk (J)	Aggregated Base (J)	Aggregated Hardware Energy (J)	Aggregated Application (J)	Aggregated Total Energy Consumption (J)	Time (t=1s)
Genetic Algorithm (GA)	240.77	286.14	0.00	696.27	1223.17	220.15	1443.33	1.00
Particle Swarm Optimization (PSO)	150.71	173.47	0.00	422.12	746.31	136.77	883.08	1.00
Differential Evolution (DE)	106.20	148.84	0.00	362.17	617.21	100.55	717.75	1.00
Artificial Bee Colony (ABC) Algorithm	395.57	537.90	0.00	1308.90	2242.38	373.16	2615.54	1.00

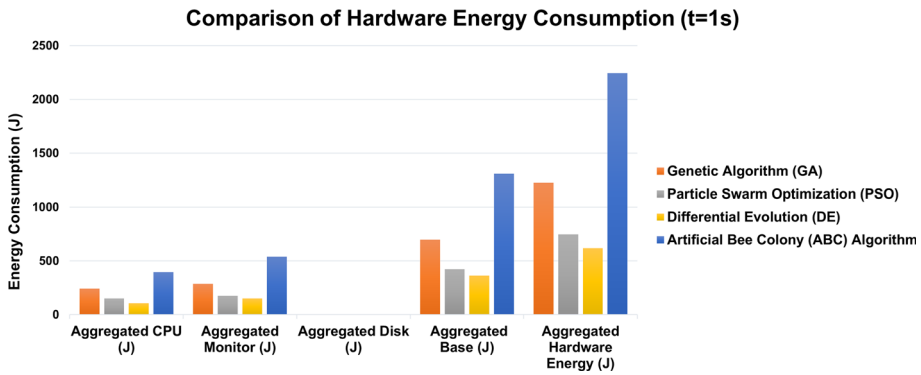

Fig. 3 Comparison of Normalized Hardware Energy Consumption for Each Algorithm

4.4 Ethical issues and challenges

The goal of this study is to measure and understand the energy consumption behavior of four NIO algorithms. This study will provide greater insight into how one NIO algorithm performs compared to other algorithms in terms of energy consumption. Some possible validity-related problems in this study can be divided into four categories [73, 74], namely conclusion validity, internal validity, construct validity, and external validity.

4.4.1 Conclusion validity

This category describes factors that may influence the validity of drawn conclusions [74]. From the experiments conducted in this study, it is clear that different NIO algorithms incur varying amount of processing-related energy consumption. For a better comparison, the energy consumption of the CPU as well as other hardware resources consumption, such as Monitor, Disk, and Idle or Base power, have also been measured. Additionally, to present the analysis in this study, only four NIO algorithms have been chosen due to their applicability to solving a wide range of problems in diverse areas. Nevertheless, the experiments conducted in this research could be replicated for other NIO algorithms.

4.4.2 Internal validity

This category concerns itself with what factors may interfere with the results of a study [74]. When measuring the energy consumption of the four NIO algorithms, other factors alongside the different implementations of algorithms using different programming languages may contribute to variations, for example, a specific version of a laptop. In order to avoid this, every NIO algorithm has been implemented using MATLAB, and the energy consumption of these algorithms is measured by utilizing a commonly used benchmark function named Sphere Function. Besides, on a specific laptop, each NIO algorithm was executed 10 times, and the corresponding energy consumption of each algorithm was measured accordingly. This allowed us to minimize the particular states of the tested machine, including uncontrollable system processes and software. However, the measured results are quite consistent and thus reliable. In addition, the used energy measurement tool has also been proven to be accurate.

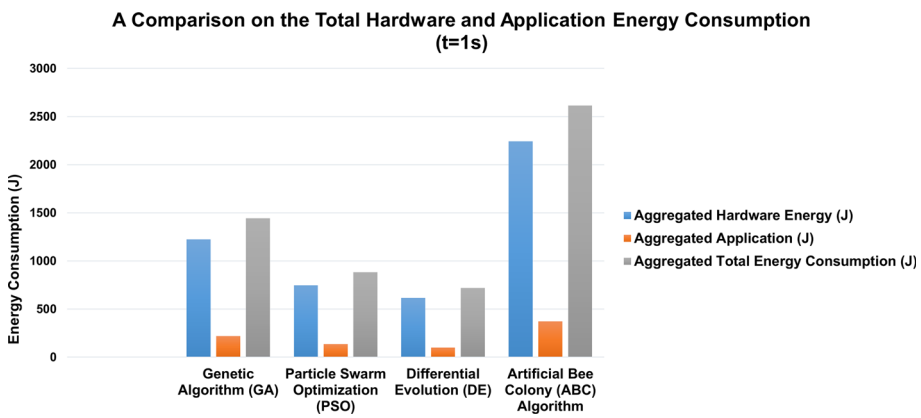

Fig. 4 Comparison of Normalized Hardware and Application Energy Consumption for Each Algorithm

Table 5 Energy Consumption Ratio for Each Algorithm

Aggregated Energy Consumption for t = 1s	Aggregated CPU (J)	Ratio Comparison to DE Algorithm
Genetic Algorithm (GA)	240.77	2.27
Particle Swarm Optimization (PSO)	150.71	1.42
Differential Evolution (DE)	106.20	1.00
Artificial Bee Colony (ABC) Algorithm	395.57	3.72

Aggregated Energy Consumption for t = 1s	Aggregated Application (J)	Ratio Comparison to DE Algorithm
Genetic Algorithm (GA)	220.15	2.19
Particle Swarm Optimization (PSO)	136.77	1.36
Differential Evolution (DE)	100.55	1.00
Artificial Bee Colony (ABC) Algorithm	373.16	3.71

Aggregated Energy Consumption for t = 1s	Aggregated Hardware Energy (J)	Ratio Comparison to DE Algorithm
Genetic Algorithm (GA)	1223.17	1.98
Particle Swarm Optimization (PSO)	746.31	1.21
Differential Evolution (DE)	617.21	1.00
Artificial Bee Colony (ABC) Algorithm	2242.38	3.63

Aggregated Energy Consumption for t = 1s	Aggregated Total Energy Consumption (J)	Ratio Comparison to DE Algorithm
Genetic Algorithm (GA)	1443.33	2.01
Particle Swarm Optimization (PSO)	883.08	1.23
Differential Evolution (DE)	717.75	1.00
Artificial Bee Colony (ABC) Algorithm	2615.54	3.64

4.4.3 Construct validity

This category concerns the generalization of the results to concepts or theories that underlie the experiments [74]. In practice, all the four NIO algorithms have been implemented in different programming languages. However, in this research, each algorithm has been implemented using MATLAB, and the energy consumption of these algorithms is measured by utilizing a commonly used benchmark function named Sphere Function. This allows us to compare the energy consumption of different NIO algorithms for the same task execution. Steps have been taken to ensure that the experimental procedures remain consistent for all the four algorithms.

4.4.4 External validity

This is concerned with the generalization of the results [74]. The greenest NIO algorithm found in this study is based on the outcome of several conducted set of experiments. Since there exist more than a hundred NIO algorithms and their variants in the literature [2], we could not make a generalization about the greenest NIO algorithm because it requires further experiments with all other NIO algorithms. Additionally, experiments run on varying devices would yield different results. In order to ensure consistency and valid results, the algorithms would have to be deployed on the same device. Thus, it is necessary to report the experiment procedures in order to understand its applicability to other contexts [74]. To summarize, the actual approach and methodology used in this study fosters easy replications. Thus, other researchers would be able to easily replicate the applied methodology for future work.

Table 6 Carbon Footprint for Each Algorithm

Algorithm	Aggregated Total Energy Consumption (KJ)	Aggregated Total Energy Consumption (kWh)	Carbon Emission (kgCO2e/kWh)
Genetic Algorithm (GA)	1373.40	0.38	0.10
Particle Swarm Optimization (PSO)	513.35	0.14	0.04
Differential Evolution (DE)	357.77	0.10	0.03
Artificial Bee Colony (ABC) Algorithm	4712.91	1.31	0.33

5 Conclusion and future work

In this study, energy consumption and associated carbon footprint of four widely used Nature-Inspired Optimization (NIO) algorithms have been examined. Each optimization algorithm exhibits significantly different energy consumption, where Differential Evolution (DE) is found to be greenest compared to other algorithms. In addition, a one-way Analysis of Variance (ANOVA) test has been performed, where the test reveals that the average energy consumed by each algorithm is significantly different from each other. The aim of this study will raise awareness of the energy efficiency issue of NIO algorithms and encourage researchers to conduct more research in this field to provide pragmatic yet greener solutions. Research in

Table 7 Average Energy Consumed by Each Algorithm for all Experiments

Exp. No	Average Energy Consumption per second (J/s)			
	GA	PSO	DE	ABC
1	1611.900	907.000	732.200	2587.700
2	1587.600	895.300	723.300	2610.100
3	1227.700	884.200	715.200	2640.500
4	1425.100	879.000	709.900	2582.500
5	1375.000	908.400	722.300	2573.900
6	1539.300	864.700	718.200	2647.600
7	1256.800	886.200	721.700	2732.900
8	1441.500	870.500	716.000	2617.800
9	1309.900	877.000	696.400	2605.200
10	1540.100	855.200	721.000	2545.500

Table 8 ANOVA Test Result

Anova: Single Factor						
SUMMARY						
Groups	*Count*	*Sum*	*Average*	*Variance*		
GA	10	14314.9	1431.49	18944.5		
PSO	10	8827.5	882.75	299.9428		
DE	10	7176.2	717.62	90.39067		
ABC	10	26143.7	2614.37	2673.149		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	22083418	3	7361139	1337.904	5.85E-37	2.866266
Within Groups	198071.8	36	5501.995			
Total	22281490	39				

this area will eventually help developers choose the most appropriate yet green algorithm for a particular domain problem when energy efficiency is the top priority.

However, there exist some challenging concerns of NIO algorithms in spite of their popularity and effectiveness. All NIO algorithms contain algorithm-dependent parameters, and the value of these parameters can significantly influence their performance under consideration [75]. It is still not evident what the best value of these parameters is to achieve an optimal balance of exploration and exploitation for a given algorithm and a given set of problems as parameter settings can be algorithm or problem-dependent [61]. Therefore, an investigation of tuning and controlling of parameters for NIO algorithms can be explored so that their performance can be maximized with reduced energy consumption.

As far as future work is concerned, more NIO algorithms and their variants can be considered for energy consumption analysis so that the top ten greenest NIO algorithms could be identified. Besides, more benchmark functions and real-life optimization problems can also be taken into account while evaluating the energy consumption of NIO algorithms. Moreover, different CPU architectures, such as AMD and ARM, can also be considered for running these algorithms. Energy usage of hardware resources for different CPU architectures can be collected in order to identify possible relationships between NIO algorithms and the energy usage of hardware resources. All these proposed efforts will shed light energy efficiency of NIO algorithms for complex applications.

Authors' contributions

Mohammad Newaj Jamil: Conceptualization, Formal analysis, Software, Investigation, Visualization, Validation, Writing - Original draft preparation; Ah-Lian Kor: Supervision, Project administration, Methodology, Data curation, Writing - Reviewing & Editing. Both authors read and approved the final manuscript.

Funding

This research has been supported by EMJMD GENIAL (610619-EPP-1-2019-1-FR-EPPKA1-JMD-MOB).

Availability of data and material

Data and material of this study is available on request from the corresponding author.

Code availability

Code used in this study is available on request from the corresponding author.

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Competing interests

The authors declare that they have no conflict of interest.

Received: 3 September 2021 Accepted: 8 December 2021
Published online: 27 January 2022

References

1. Li H, Liu X, Huang Z, Zeng C, Zou P, Chu Z, Yi J. Newly emerging nature-inspired optimization-algorithm review, unified framework, evaluation, and behavioural parameter optimization. IEEE Access. 2020;8:72620–49.
2. Yang X-S. Nature-inspired Optimization Algorithms. London: Academic Press; 2020.
3. Bayer H, Nebel M. Evaluating algorithms according to their energy consumption. Math Theory Comput Pract. 2009;48:1–25.
4. Podder S, Burden A, Singh SK, Maruca R. Sustainable Business practices – How Green Is Your Software? 2020. Harvard Business Review. https://hbr.org/2020/09/how-green-is-your-software Accessed 8 Jan 2022.
5. Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT press; 2016.
6. Schwartz R, Dodge J, Smith NA, Etzioni O. Green ai. Commun ACM. 2020;63(12):54–63.
7. Strubell E, Ganesh A, McCallum A. Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics; 2019. p. 3645–50.
8. Taylor CE. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Complex adaptive systems. John H. Holland. Q Rev Biol. 1994;69(1):88–9.
9. Li Y, Ang KH, Chong GC, Feng W, Tan KC, Kashiwagi H. Cautocsd-evolutionary search and optimisation enabled computer automated control system design. Int J Autom Comput. 2004;1(1):76–88.
10. Zhang J, Chung HS, Lo W-L. Pseudocoevolutionary genetic algorithms for power electronic circuits optimization. IEEE Trans Syst Man Cybern C (Appl Rev). 2006;36(4):590–8.
11. To CC, Vohradsky J. A parallel genetic algorithm for single class pattern classification and its application for gene expression profiling in streptomyces coelicolor. BMC Genom. 2007;8(1):1–13.
12. Gondro C, Kinghorn BP. A simple genetic algorithm for multiple sequence alignment. Genet Mol Res. 2007;6(4):964–82.
13. Kia R, Khaksar-Haghani F, Javadian N, Tavakkoli-Moghaddam R. Solving a multi-floor layout design model of a dynamic cellular manufacturing system by an efficient genetic algorithm. J Manuf Syst. 2014;33(1):218–32.
14. Vitayasak S, Pongcharoen P, Hicks C. A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a genetic algorithm or modified backtracking search algorithm. Int J Prod Econ. 2017;190:146–57.
15. Katoch S, Chauhan SS, Kumar V. A review on genetic algorithm: past, present, and future. Multimed Tools Appl. 2021;80(5):8091–126.
16. Shi Y, Eberhart R. A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360). Anchorage: IEEE; 1998. p. 69–73.
17. Sheta AF, Ayesh A, Rine D. Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for nasa projects: a comparative study. Int J Bio-Inspired Comput. 2010;2(6):365–73.

18. Saini S, Bt Awang Rambli DR, Zakaria MNB, Bt Sulaiman S. A review on particle swarm optimization algorithm and its variants to human motion tracking. Math Probl Eng. 2014;2014:1–16.
19. Mohana R. A position balanced parallel particle swarm optimization method for resource allocation in cloud. Indian J Sci Technol. 2015;8(S3): 182–8.
20. Delice Y, Kızılkaya Aydoğan E, Özcan U, undefinedlkay MS. A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. J Intell Manuf. 2017;28(1):23–36.
21. Esmin AA, Coelho RA, Matwin S. A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. Artif Intell Rev. 2015;44(1):23–45.
22. Li M, Du W, Nian F. An adaptive particle swarm optimization algorithm based on directed weighted complex network. Math Probl Eng. 2014;2014:1–7.
23. Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11(4): 341–59.
24. Qing A. Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems. IEEE Trans Geosci Remote Sens. 2005;44(1):116–25.
25. Michalski KA. Electromagnetic imaging of elliptical–cylindrical conductors and tunnels using a differential evolution algorithm. Microw Opt Technol Lett. 2001;28(3):164–9.
26. Pal S, Qu B, Das S, Suganthan P. Optimal synthesis of linear antenna arrays with multi-objective differential evolution. Prog Electromagn Res B. 2010;21:87–111.
27. Chakraborty J, Konar A, Jain LC, Chakraborty UK. Cooperative multi-robot path planning using differential evolution. J Intell Fuzzy Syst. 2009;20(1, 2):13–27.
28. Wu Y-C, Lee W-P, Chien C-W. Modified the performance of differential evolution algorithm with dual evolution strategy. In: 2009 International conference on machine learning and computing IPCSIT. vol. 3. Singapore: IACSIT Press; 2011. p. 57–63.
29. Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report, Citeseer. 2005.
30. Secui DC. A new modified artificial bee colony algorithm for the economic dispatch problem. Energy Convers Manag. 2015;89:43–62.
31. Karaboga D, Gorkemli B. A combinatorial artificial bee colony algorithm for traveling salesman problem. In: 2011 International Symposium on Innovations in Intelligent Systems and Applications. Istanbul: IEEE; 2011. p. 50–3.
32. Contreras-Cruz MA, Ayala-Ramirez V, Hernandez-Belmonte UH. Mobile robot path planning using artificial bee colony and evolutionary programming. Appl Soft Comput. 2015;30:319–28.
33. Pan J-S, Wang H, Zhao H, Tang L. Interaction artificial bee colony based load balance method in cloud computing. In: Genetic and Evolutionary Computing. Yangon: Springer; 2015. p. 49–57.
34. Bose A, Mali K. Fuzzy-based artificial bee colony optimization for gray image segmentation. Signal Image Video Process. 2016;10(6):1089–96.
35. Zhao J, Lv L, Sun H. Artificial bee colony using opposition-based learning. In: Genetic and Evolutionary Computing. Yangon: Springer; 2015. p. 3–10.
36. Pereira R, Couto M, Ribeiro F, Rua R, Cunha J, Fernandes JP, Saraiva J. Energy efficiency across programming languages: how do energy, time, and memory relate? In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering. New York: Association for Computing Machinery; 2017. p. 256–67.
37. Georgiou S, Kechagia M, Spinellis D. Analyzing programming languages' energy consumption: An empirical study. In: Proceedings of the 21st Pan-Hellenic Conference on Informatics. New York: Association for Computing Machinery; 2017. p. 1–6.
38. Pereira R, Couto M, Ribeiro F, Rua R, Cunha J, Fernandes JP, Saraiva J. Ranking programming languages by energy efficiency. Sci Comput Program. 2021;205:102609.
39. Rashid M, Ardito L, Torchiano M. Energy consumption analysis of algorithms implementations. In: 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). Beijing: IEEE; 2015. p. 1–4.
40. Verma M, Chowdhary K. Analysis of energy consumption of sorting algorithms on smartphones. In: Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT). Rochester: ELSEVIER-SSRN; 2018. p. 472–5.
41. Deepthi T, Birunda A. Time and energy efficiency: A comparative study of sorting algorithms implemented in c. In: International Conference on Advancements in Computing Technologies-ICACT 2018. vol. 4. India: IJFRCSCE; 2018. p. 25–7.
42. Ayodele OS, Oluwade B. A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms using Three Programming Languages II: Energy Consumption Analysis. Afr J MIS. 2019;1(2):44–63.
43. Webb M, et al. Smart 2020: Enabling the low carbon economy in the information age. Clim Group Lond. 2008;1(1):1.
44. Murugesan S. Going Green with IT: Your Responsibility Toward Environmental Sustainability. Arlington: Cutter Consortium; 2007.
45. Simunic T, Benini L, De Micheli G. Energy-efficient design of battery-powered embedded systems. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2001;9(1):15–28.
46. Schmitz MT, Al-Hashimi BM, Eles P. System-level Design Techniques for Energy-efficient Embedded Systems. Berlin: Springer Science & Business Media; 2004.
47. Hosangadi A, Kastner R, Fallah F. Energy efficient hardware synthesis of polynomial expressions. In: 18th International Conference on VLSI Design Held Jointly with 4th International Conference on Embedded Systems Design. India: IEEE; 2005. p. 653–8.
48. Shiri A, Mazumder AN, Prakash B, Manjunath NK, Homayoun H, Sasan A, Waytowich NR, Mohsenin T. Energy-efficient hardware for language guided reinforcement learning. In: Proceedings of the 2020 on Great Lakes Symposium on VLSI. New York: Association for Computing Machinery; 2020. p. 131–6.
49. Capra E, Francalanci C, Slaughter SA. Is software "green"? Application development environments and energy efficiency in open source applications. Inf Softw Technol. 2012;54(1):60–71.
50. D'Agostino D, Merelli I, Aldinucci M, Cesini D. Hardware and software solutions for energy-efficient computing in scientific programming. Sci Prog. 2021;2021:1–9.
51. Naumann S, Dick M, Kern E, Johann T. The greensoft model: A reference model for green and sustainable software and its engineering. Sustain Comput Inf Syst. 2011;1(4):294–304.
52. Kor A-L, Pattinson C, Imam I, AlSaleemi I, Omotosho O. Applications, energy consumption, and measurement. In: 2015 International Conference on Information and Digital Technologies. Zilina: IEEE; 2015. p. 161–171.
53. Pattinson C, Olaoluwa PO, Kor A-L. A comparative study on the energy consumption of PHP single and double quotes. In: 2015 IEEE International Conference on Data Science and Data Intensive Systems. Sydney: IEEE; 2015. p. 232–9.
54. Engel M. Sustainable software design. In: Green Information Technology. San Francisco: Elsevier; 2015. p. 111–27.
55. Dastbaz M, Pattinson C, Akhgar B. Green Information Technology: A Sustainable Approach. San Francisco: Morgan Kaufmann; 2015.
56. Ardito L, Procaccianti G, Torchiano M, Vetro A. Understanding green software development: A conceptual framework. IT Prof. 2015;17(1):44–50.
57. Murugesan S. Harnessing green IT: Principles and practices. IT Prof. 2008;10(1):24–33.
58. Ferreira MA, Hoekstra E, Merkus B, Visser B, Visser J. Seflab: A lab for measuring software energy footprints. In: 2013 2nd International Workshop on Green and Sustainable Software (GREENS). San Francisco: IEEE; 2013. p. 30–7.
59. Bener AB, Morisio M, Miranskyy A. Green software. IEEE Softw. 2014;31(3): 36–9.
60. Barontini A, Masciotta M-G, Ramos LF, Amado-Mendes P, Lourenço PB. An overview on nature-inspired optimization algorithms for structural health monitoring of historical buildings. Proc Eng. 2017;199:3320–5. https://doi.org/10.1016/j.proeng.2017.09.439. X International Conference on Structural Dynamics, EURODYN 2017.
61. Yang X-S. Nature-inspired optimization algorithms: Challenges and open problems. J Comput Sci. 2020;46:101104.
62. Abdollahzadeh B, Soleimanian Gharehchopogh F, Mirjalili S. Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. Int J Intell Syst. 2021;36(10):5887–958. https://doi.org/10.1002/int.22535.
63. Mohanty A, Nag KS, Bagal DK, Barua A, Jeet S, Mahapatra SS, Cherkia H. Parametric optimization of parameters affecting dimension precision of fdm printed part using hybrid taguchi-marcos-nature inspired heuristic optimization technique. Mater Today Proc. 2021. https://doi.org/10.1016/j.matpr.2021.06.216.

64. Sadrnia A, Soltani HR, Zulkifli N, Ismail N, Ariffin MKA. A review of nature-based algorithms applications in green supply chain problems. Int J Eng Technol. 2014;6(3):204–11.
65. Nguyen T-H, Nguyen LV, Jung JJ, Agbehadji IE, Frimpong SO, Millham RC. Bio-inspired approaches for smart energy management: State of the art and challenges. Sustainability. 2020;12(20):. https://doi.org/10.3390/su12208495.
66. Usman MJ, Ismail AS, Abdul-Salaam G, Chizari H, Kaiwartya O, Gital AY, Abdullahi M, Aliyu A, Dishing SI. Energy-efficient nature-inspired techniques in cloud computing datacenters. Telecommun Syst. 2020;71:275–302. https://doi.org/10.1007/s11235-019-00549-9.
67. Sharma R, Vashisht V, Singh U. Nature inspired algorithms for energy efficient clustering in wireless sensor networks. In: 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence); 2019. p. 365–70. https://doi.org/10.1109/CONFLUENCE.2019.8776618.
68. Agbehadji IE, Millham RC, Abayomi A, Jung JJ, Fong SJ, Frimpong SO. Clustering algorithm based on nature-inspired approach for energy optimization in heterogeneous wireless sensor network. Appl Soft Comput. 2021;104:107171. https://doi.org/10.1016/j.asoc.2021.107171.
69. ITU-T L.1410. Methodology for Environmental Life Cycle Assessments of Information and Communication Technology Goods, Networks and Services. 2014. Series L: Construction, Installation and Protection of Cables and Other Elements of outside Plant. Geneva. https://www.itu.int/rec/T-REC-L.1410-201412-I.
70. Kansal A, Goraczko M, Liu J, Zhao F. Joulemeter: Computational Energy Measurement and Optimization; 2010. Microsoft Research, Redmond, United States https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization/ Accessed 8 Jan 2022.
71. DEFRA. UK Government Greenhouse Gas Conversion Factors for Company Reporting. Department for Business, Energy & Industrial Strategy. 2020. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/901692/conversion-factors-2020-methodology.pdf.
72. Ross A, Willson VL. One-way anova. In: Basic and Advanced Statistical Tests. Rotterdam: SensePublishers; 2017. p. 21–4.
73. Campbell DT, Cook TD. Quasi-experimentation: Design & Analysis Issues for Field Settings. Boston: Houghton Mifflin; 1979.
74. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. Experimentation in Software Engineering. Berlin: Springer Science & Business Media; 2012.
75. Yang X-S. Nature-inspired Algorithms and Applied Optimization, vol. 744. Gewerbestrasse, Cham: Springer; 2017.

Publisher's Note