Analyzing and Rating Greenness of Nature-Inspired Algorithms

Chander Jindal [1†], Kanaishk Garg [1*†], Shobhit Kumar [1†]

[1] Department of Computer Science and Engineering, Maharaja Agrasen Institute of Technology, Delhi, India

*** Correspondence:**

Kanaishk Garg

kanaishkgarg@gmail.com

†These authors contributed equally to this work and share first authorship

Abstract: Nature-inspired optimization algorithms (NIOAs) are modeled after natural phenomena such as the behavior of birds, bees, and other animals. These algorithms are designed to mimic the way that nature optimizes solutions to problems, and they can be used to find the optimal solution to a wide range of problems in fields such as machine learning, computer science, and engineering. This study aims to objectively evaluate energy use and the associated carbon footprint for a few well-known NIOAs. To measure the energy utilized by each algorithm, we used Microsoft Joulemeter. The associated carbon footprint is determined using the recommendations of the Central Electricity Authority of India. According to the study's findings, each algorithm uses varying amounts of energy to accomplish the same task. This study aims to enhance our knowledge regarding the energy consumption behavior of different NIOAs and aid software designers in selecting greener NIOAs for their task implementation from the available options. The aim is to bring more focus toward greener software solutions to combat growing energy demands and climate change. Future studies might consider more NIOAs and their modifications for energy consumption analysis to identify the most environmentally friendly NIOAs. Additionally, further investigation into the potential effects of different CPU architectures on the efficiency and power usage of the NIOAs may be taken into account.

Keywords: nature-inspired optimization algorithms; energy consumption; carbon footprint; green software; environmental effect; microsoft joulemeter; central electricity authority of india;

1. Introduction

Traditionally, algorithm engineering has focused on the runtime of algorithms, and performance has been the primary metric for evaluating and optimizing algorithms. Today, analysing and minimizing the power consumption of algorithms is just as crucial as optimizing their runtime [1]. Recent years have seen a significant increase in the use of the Internet of Things (IoT). Alongside this, the proliferation of smartphones, tablets, and other mobile devices through mobile computing has added to the strain on network and energy resources. Green Computing aims to address this by promoting energy-efficient and environmentally friendly practices. This is achieved through the implementation of energy-efficient techniques and procedures at both the hardware and software levels, aimed at reducing energy consumption, carbon dioxide emissions, and the overall impact on the environment. By doing so, existing devices, applications, and services can become more sustainable [2]. An algorithm's effectiveness and efficiency must be assessed in the context of a particular application since how it is implemented will influence energy consumption and the environment. Software programs may become more eco- and energy-friendly as a result of enhanced algorithms and data structures.

One approach for assessing the ecological consequences of computers and other computing devices is through carbon footprint [3]. By evaluating a program's degree of power efficiency related to carbon footprint and implementing it into ecologically friendly company operations or procedures, organizations may make the application a crucial component of their corporate social responsibility activities.

Machine learning is a powerful tool that enables the use of data to gain insights, make predictions, and automate decision-making. Machine learning model deployment has grown massively in recent years [4]. With this much utilization, it becomes crucial that algorithms used to solve these problems use as minimal resources as possible be it in compute time, power, or energy consumption to save ongoing expenses and deliver effective results faster. Thus, considerable issues have emerged regarding energy usage and expense related to developing ML models and training them [5]. Therefore, it is crucial to consider an application's carbon footprint while planning, constructing, and deploying it.

Optimization is the process of modifying an existing system to improve the chances of achieving desired outcomes and reducing the risk of undesirable ones. This concept is widely applied in various fields, including engineering, business operations, industrial design, and many more. Optimization problems may have different objectives such as reducing energy consumption and increasing performance or efficiency. In machine learning, an optimizer is an algorithm or technique to adjust the various parameters of a model in order to minimize the loss function (in simpler terms, the error). The Nature Inspired Optimization Algorithms (NIOAs) are one such family of optimizers.

The majority of traditional optimization algorithms used to address real-world problems are highly non-linear, have several local optima, and involve complex nonlinear constraints. Nature Inspired Optimization Algorithms (NIOAs) are better at avoiding local optima in comparison to conventional optimization algorithms [6]. NIOAs are population-based metaheuristics that replicate a wide range of natural phenomena [7]. Therefore, they are utilized extensively across a wide range of sectors, including manufacturing, environmental engineering, finance, biology, data mining, and more, to tackle complex and nonlinear optimization problems.

More than a hundred NIO algorithms and their variations are now known and available in the literature [6]. The Bat Algorithm (BAT), Camel Algorithm (CAM), Cuckoo Search (CS), Firefly Algorithm (FIR), and Particle Swarm Optimization (PAR) are some of the regularly utilized NIO algorithms that will be the subject of this study's examination of energy consumption and associated carbon footprint. These algorithms were taken into account for this study due to the wide range of applications for them. This work aims to demonstrate how one may experimentally assess the energy consumption of different algorithms. Future research may concentrate on different NIO algorithms or methodologies to eliminate architectural biases.

The objectives of this study are as follows:

1. Conducting an analytic review of existing literature on
   - Global Energy usage by Information and Communications Technology (ICT)
   - Effects of Information and Communications Technology (ICT) on the environment,
   - Green or energy-efficient programs,

- Effects of software's power usage on hardware,
- Analysis of software's consumption of electricity in algorithm implementations,
- Current state of Energy-efficient algorithms
- Nature-inspired optimization algorithms
- Bayesian Optimization

2. Implement the NIO Algorithms in Python programming language using NatureInspiredSearchCV provided by the sklearn_nature_inspired_algorithms library and NiaPy for nature-inspired algorithms.
3. Performing Hyper-parameter tuning on NIOAs using Bayesian Optimization.
4. Determine the Energy each algorithm consumes using Microsoft Joulemeter.
5. Based on how much energy each algorithm consumes, calculate its equivalent carbon footprint.

Previous works have compared the energy usage of various programming languages [8–10] and implementations of sorting algorithms [11–14]. Only one other team has evaluated the NIO algorithms' energy usage and greenness. They assessed the effectiveness of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC) [15]. According to their results, while Differential Evolution is the most efficient, the result cannot be directly compared to this study due to the use of different methodologies to test the algorithm suite.

The energy consumption behavior of NIOAs has not been well studied when applied to real-world problems. The novel contribution of this study leans more towards finding out how the NIOAs may compare to each other when applied to real-world optimization problems. Python is a more widely utilized language for machine learning tasks. So, analyzing the algorithms' performance, and subsequent consumptions will provide more information regarding how they may be applied in industries for optimization tasks. While this approach has its limitations, it is hoped that the results provide an idea about how the NIOAs perform if they were to be applied today to optimize a model. This study aims to compare the energy consumption of different NIOAs when solving a commonly encountered optimization problem. The issue of conducting hyper-parameter tuning is addressed by using Bayesian optimization to select the parameter values and then applying the resultant model to optimize a classification model to see the effects of NIOAs. By understanding the energy consumption of these algorithms, more informed decisions can be made on which algorithm to use in energy-constrained systems, which can contribute to the development of energy-efficient algorithms for technological devices, appliances, and systems. This may have a significant impact on their design and implementation and help to reduce their environmental impact.

The results of this study (as well as other studies in this field) can help in many ways:

1. Choosing Energy efficient options: By understanding the energy consumption of these algorithms, one can make more informed decisions on which algorithm to use in energy-constrained systems, such as mobile and embedded devices, which will lead to the development of more energy-efficient algorithms.
2. Cost reduction: More efficient algorithms will consume less energy which will also help in reducing the operating costs of mobile and embedded devices, benefiting both individuals and Organizations that utilize these devices.
3. Improved performance: By selecting the most appropriate algorithm for a given task, the performance of mobile and embedded systems can be improved, which will have a positive impact on various fields, such as machine learning, computer vision, and control systems.

4. Environmental Impact: By reducing the energy consumption and thus in turn carbon footprint of mobile and embedded devices, this study can help lower environmental impact due to the growth of the given technologies.

The rest of this paper is divided into the following sections:

- Section 2 presents a review of the literature on global energy usage by ICT, the effects of ICT on the environment, green or energy-efficient software, the influence of hardware energy consumption on software, current state of energy-efficient algorithms, and related works on the analysis of energy consumption in algorithm implementations as well as on nature-inspired optimization algorithms and Bayesian Optimization.
- Section 3 covers both macro and micro methodology as well as the planning and design of experiments.
- Section 4 presents the findings and commentary, which goes through the ethical concerns and difficulties of this study as well as the energy usage and associated carbon footprint of each algorithm.
- Section 5 provides a summary of the topic and suggestions for additional research.

2. Literature Review

Global Energy Usage by Information and Communication Technology (ICT):

Due to the rapidly increasing demand for Communication Technology devices and the establishment of the Internet of Things (IoT), most devices are interconnected nowadays [16]. Be it cloud computing [17], thin clients like smartphones [18], or highspeed network access [19], they all have had a disruptive impact on the ICT sector. While efficiency improvements have both been made in computations [20] and mass-storage operations [21] but still the growth of electricity usage has outweighed these improvements [22]. As this trend continues [23] and more and more applications utilizing computation power develop over time, the necessity for reliable electricity increases. It is predicted that energy usage increase might become completely unsustainable by 2040 if nothing is done to tackle the issue [24, 25]. The growing electricity usage will increase carbon emissions, may it be in production or expanding the energy delivery network. This is not good for the environment for obvious reasons thereby steps need to be taken to make the process and devices more efficient on every scale and frontier possible.

Effect of ICT on the Environment:

Between 2008 and 2021, the ICT sector has grown responsible for almost 1.9 increased percent of the world's carbon emissions, with the remaining 97.9 to 96.1% coming from other industries including the transportation and agricultural industries [26, 27]. The environment and the economy suffer as a result of the rise in carbon emissions brought on by Greenhouse Gases and other causes [28]. The ICT sector can significantly contribute to lowering global carbon emissions by reducing the carbon footprints of its products and services because there is growing global demand for ICT goods and services. Energy-efficient hardware and other embedded systems have been the subject of extensive study [29–32], but software and application development should also receive significant attention [33, 34].

Green Software:

Green or energy-efficient software is defined as using less energy for effective computing while causing little environmental harm [35]. The energy efficiency of web-based software applications and software features have been the subject of numerous research [36, 37]. The software can easily be estimated to use between 25% and 40% of the total energy used by a device, depending on the laptop or mobile battery [38]. However, because it is correlated with the host device's life cycle, the indirect impact of software is more challenging to measure [39]. Only when both the positive and negative impact is taken into consideration throughout the design and deployment phases can the energy efficiency of software be truly accomplished. In light of this, optimizing ICT application services is essential to lowering harmful environmental effects.

Software's Impact on Hardware-Related energy consumption:

The software's energy usage patterns strongly impact how much energy hardware uses and how long a device's battery lasts [40]. A device's energy usage may eventually increase if software or application that is poorly built disables various hardware-based energy-saving capabilities [41]. For instance, it can prevent hardware from using energy-saving features and impact how the hardware is used, which could ultimately increase indirect energy usage [42]. One of the trickiest tasks during the design stage of an embedded system is the development of energy-efficient software that enhances the energy efficiency of a piece of hardware. Various trade-offs between productivity and sustainability will need to be considered to increase software and application productivity while maintaining energy efficiency [43].

Analysis of Energy Consumption in algorithms implementations

Overall, there has been little focus on algorithm-related energy consumption in the currently existing literature. In 2015, Rashid and the team evaluated the energy efficiency of four sorting algorithms implemented in three different programming languages. They found that ARM assembly was the most energy-efficient language and Java was the most power-hungry among the three. Among the algorithms, the one with the best performance was the Counting sort. They also concluded that a large part of energy consumption was determined by the computational complexity (or time performance) [11].

Murlidhar and K.R. Chowdhary conducted a study regarding the analysis of energy consumption by sorting algorithm implementations on smartphones in 2018. They found Quick sort to be the most energy-efficient and Bubble Sort to be the least efficient. They also proposed the concept of energy complexity which could be extended in the future to be a similar parameter to time complexity to keep track of the energy consumption of both algorithms and software systems [12]. Another study in 2018 tracked the time and energy efficiency of sorting algorithms when implemented in C. They similarly found Quicksort to be the most energy efficient one among the tested algorithms [13].

A 2019 study by Ayodele and Oluwade found that energy consumption scales up with increasing data size. Among iterative and recursive approaches, the iterative approach is more energy efficient for Merge sort but the Recursive approach is more efficient for Quick sort and C is the most energy efficient among the languages they tested [14].

In research conducted by Jamil and Kor, the energy consumption of a few nature-inspired algorithms has been analyzed on a dataset [15]. Particle swarm optimization, differential evolution, the artificial bee colony algorithm, and the genetic algorithm were all employed. They found Differential Evolution (DE) to be the most environmentally friendly among other optimization algorithms, using much less energy than each of the other methods.

Current state of Energy-efficient algorithms:

As seen from the previous subsection, the current state of energy-efficient algorithms mainly focuses on comparing sorting algorithms in different implementations, platforms, and languages. While this type of research provides valuable insights for specific use cases, it does not provide a comprehensive understanding of the general energy efficiency of algorithms across different fields of software development. Additionally, most of the existing literature deals with creating specific energy-efficient algorithmic solutions for use cases such as flow time minimization [44], RFID estimation problem [45], or cloud computing algorithms [46]. However, the field is still in its early stage and more research is needed to better understand the energy efficiency of different types of algorithms in different application fields. This can help to identify the most energy-efficient algorithms for various use cases and to reduce energy consumption in software development.

Nature-inspired optimization algorithms:

Current research on nature-inspired algorithms primarily focuses on the following areas: optimization [6, 7, 47, 48] utilizing metaheuristics [49] or heuristic methods [50]; improving environmental sustainability, such as optimizing supply chain processes [51], managing energy resources intelligently [52], and increasing energy efficiency in data centers [53]; and energy efficiency [54] and optimization [55] in wireless sensor network clustering. A little detail about the algorithms this study will cover is below, to help with a better understanding of the algorithms undergoing examination.

Bat Algorithm (BAT/BA): There are around 1000 species of bats. The Bat Algorithm (BA) is based on the Echolocation behavior of microbats [56]. Microbats are medium-sized bats that eat insects. They used a SONAR technique called echolocation to detect prey. Artificial bats' that imitate actual bats' natural pulse loudness and emission rate serve as search agents in the search process carried out by the Bat Algorithm. Furthermore, it aids in undertaking global optimization since it uses a meta-heuristic approach [57]. In a variety of fields, including data mining, big data, and machine learning, BA has been used to address challenging issues.

Camel Algorithm (CAM/CA): It is a cutting-edge optimization method motivated by camel migration patterns in the desert and other challenging situations. A camel will often travel toward an area with food and water. In light of that, several variables and operators are taken into account to outline the CA algorithm procedure, including the temperature effect, the supply (water and food), the camel endurance, the camel visibility (and/or hearing) range, random walk, the group effect (multi-solution), the termination condition (dying or moving back), the land conditions (oasis, quicksand, storms, etc.), and limitations (max speed, age, and carrying weight). The camel algorithm's simple structure, along with its efficient search ability allows it to deal effectively with unimodal and multimodal test functions, to find an optimal solution even in the case of difficult ones [58].

Cuckoo Search (CUC/CS): Xin-She Yang and Suash Deb created the Cuckoo Search optimization algorithm in 2009 [59]. The obligate brood parasitism of some cuckoo species, which involves the cuckoos' laying their eggs in the nests of host birds of other species, served as its model. Direct combat between some host birds and the trespassing cuckoos is possible. For instance, if the host bird learns the eggs are not its own, it may either discard the alien eggs or quit the nest and make a new one elsewhere. Some cuckoo species, like the New World brood-parasitic Tapera, have developed in such a way that female parasitic cuckoos are frequently extremely skilled at mimicking the colors and patterns of the eggs of a select few host species. Such breeding behavior was idealized by cuckoo search, which can be used to solve numerous optimization issues [60].

Firefly Algorithm (FIR/FA): The glowing pattern that firefly swarms exhibit serves as the inspiration for Firefly Algorithm [6]. FA is incredibly flexible and easy to use. It is based on the idea that attractiveness and brightness are inversely correlated and that Fireflies are attracted to one another if two fireflies have the same level of brightness. The software creates creative approaches and continues to search the solution space. The Random Walk unpredictability factor refers to this. There are several applications for FA, including image compression, antenna design optimization, classification, feature selection, etc.

Particle Swarm Optimization (PAR/PSO): The concept of Particle Swarm Optimization (PSO) is inspired by the swarm intelligence of fish schooling and bird flocks that are in search of food [61]. Each particle in these groups has its velocity and position. It has been applied to solve a variety of issues, including data clustering [62], human motion tracking [63], cloud resource allocation [64], assembly line balance [65], and cost prediction for software [66]. However, a drawback of PSO is that it has a poor rate of convergence during the iterative process and is prone to falling into local optimum in high-dimensional space [67].

Since there are currently very few studies on energy-efficient, nature-inspired algorithms, as this analytical literature evaluation has indicated, this study intends to address this issue to further encourage research in this field.

Bayesian Optimization:

Bayesian optimization (BO) is a powerful technique used to find the optimal set of parameters for a given model. BO is widely adopted in various fields, such as machine learning, computer vision, and control systems, due to its ability to effectively handle high-dimensional parameter spaces. It is useful in a situation where traditional optimization methods may be computationally expensive or impractical [68]. In the context of energy efficiency, Bayesian optimization is used to optimize the energy consumption of different algorithms by searching for the optimal set of parameter values that minimize energy consumption. A study conducted in 2016 demonstrated the suitability of Bayesian optimization as a tool for tuning parameters in Evolutionary Algorithms [69]. This study will also utilize Bayesian optimization to perform Hyper-Parameter tuning for the algorithm implementations.

3. Research / Methodology

The next section will go through the various tools and software that are used in the study.

- Macro Methodology

The initial step includes data collection and its subsequent preprocessing. Afterward, the accuracy, energy consumption, and carbon footprint of the five algorithms, that are BAT, CUC, FIR, CAM, and PAR are estimated and analyzed. The findings are used to draw conclusions that specify the algorithm having the best ratio of Accuracy to Energy Consumption. Lastly, all the results are summarized and discussed.

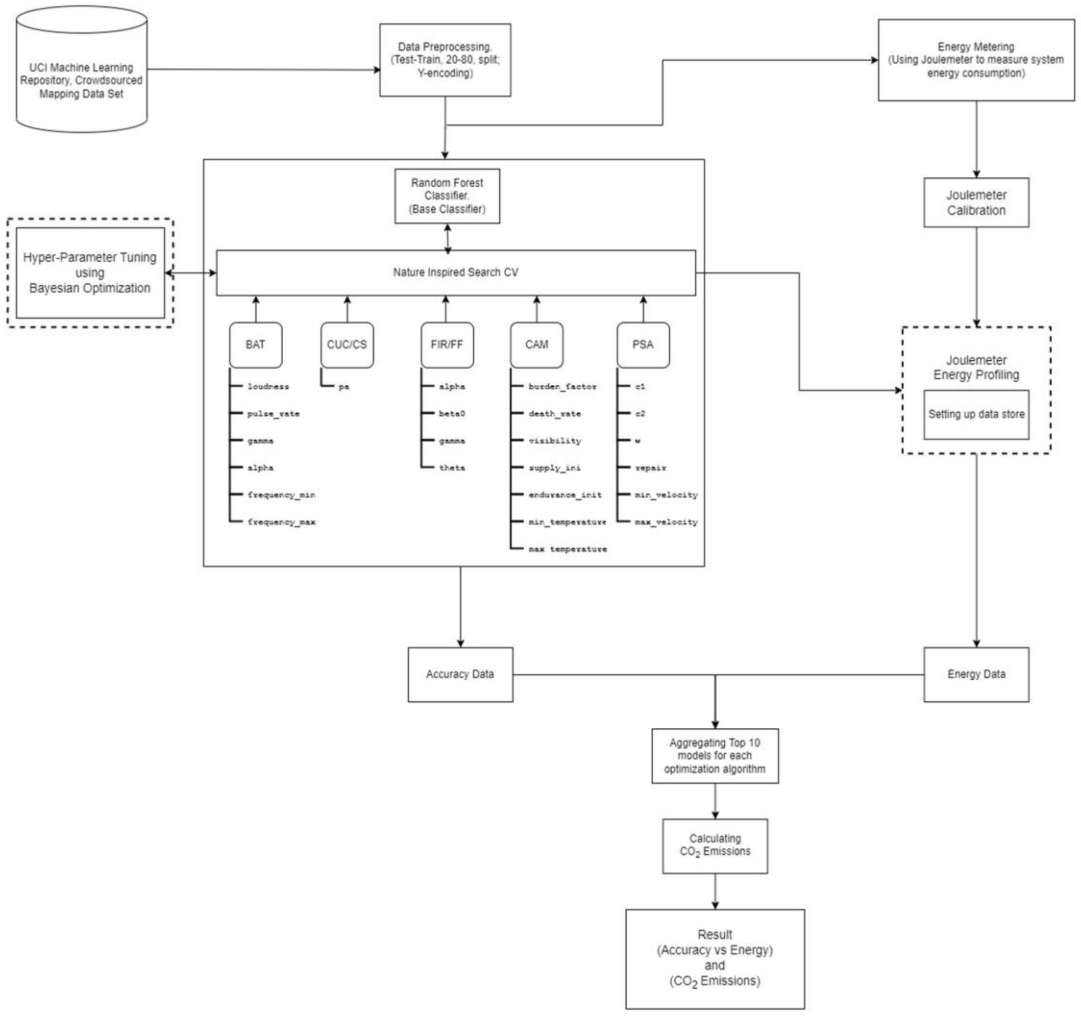

UCI Machine Learning Repository, Crowdsourced Mapping Data Set
Data Preprocessing. (Test-Train, 20-80, split; Y-encoding)
Energy Metering (Using Joulemeter to measure system energy consumption)
Random Forest Classifier. (Base Classifier)
Hyper-Parameter Tuning using Bayesian Optimization
Nature Inspired Search CV
Joulemeter Calibration
BAT
CUC/CS
FIR/FF
CAM
PSA
loudness
pulse_rate
gamma
alpha
frequency_min
frequency_max
pa
alpha
beta0
gamma
theta
burden_factor
death_rate
visibility
supply_ini
endurance_init
min_temperature
max temperature
c1
c2
w
repair
min_velocity
max_velocity
Joulemeter Energy Profiling
Setting up data store
Accuracy Data
Energy Data
Aggregating Top 10 models for each optimization algorithm
Calculating CO2 Emissions
Result (Accuracy vs Energy) and (CO2 Emissions)

Figure 1. Methodology

- **Micro Methodology**

a. Data Availability:

The dataset used in this study has been obtained from the UCI machine learning repository. It's called the Crowdsourced Mapping Data Set. Automated classification of satellite pictures into several land cover classifications (impervious, farm, forest, grass, orchard, and water) is done using crowdsourced data from OpenStreetMap [70].

The NIO methods taken into consideration in this study were implemented accordingly using Python programming language with the sklearn-nature-inspired-algorithms, a machine learning library [71], and NiaPy, a library dedicated to Nature Inspired Optimization Algorithm in Python [72].

b. Energy Profiling:

To measure and calculate the expected energy consumption of each NIO technique, we used the Microsoft Joulemeter software [73]. This software is designed to monitor the energy usage of running applications or software, as well as specific hardware resources such as CPU, monitor, disk, and idle or base power. It provides a detailed breakdown of energy usage by different components, making it an ideal tool for evaluating the energy efficiency of different NIO techniques.

c. Carbon Footprint:

To calculate carbon emissions, the Central Electricity Authority of India's recommendations have been followed [74]. After collecting the amount of energy used for an experiment (in kWh), the data is converted to the equal amount of carbon released using the formula below.

CO2 Emissions = 0.85* E(kW-hr/year) where E is the energy consumed.

1kWhr of Energy Consumed = 0.85Kg of CO_2 emission

72 Joules = 17 mg of CO_2 emissions

- Experiment Setup

a. System Specification:

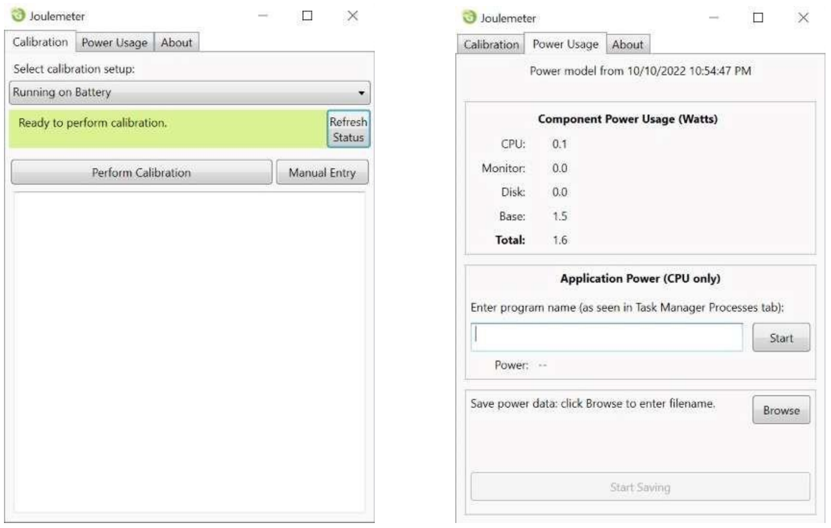
Different hardware requirements would produce various outcomes. Consequently, a laptop with the following characteristics was used for all experiments:

Specification of Laptop Used	
Model	Lenovo Ideapad 530S
Operating System	Windows 10 (19043.2006)
Processor	Intel® Core™ i5-8250U CPU @ 1.60Hz
RAM	8 GB
Storage	256 GB

Table 1. System Specification

b. Calibrating Joulemeter

In case the auto-calibration (in Joulemeter) does not work, one needs to manually calibrate Joulemeter to get the required power readings. For that define the model using the Manual entry option and add system parameters.



Select calibration setup:
Running on Battery
Ready to perform calibration.

Power model from 10/10/2022 10:54:47 PM

Component Power Usage (Watts)

CPU:	0.1
Monitor:	0.0
Disk:	0.0
Base:	1.5
Total:	1.6

Application Power (CPU only)

Enter program name (as seen in Task Manager Processes tab):

Power: --

Save power data: click Browse to enter filename.

Figure 2. Calibration of Joulemeter

c. Experiment design:

Initially, Microsoft Joulemeter is calibrated on the specified system. Random forest classifier was implemented as the base classifier and the NIO algorithms were utilized to optimize the results. The CPU energy consumption for each instance for all the different algorithms was calculated by hyper-parameter tuning using Bayesian Optimization. Microsoft Joulemeter was used to evaluate power usage with a precision of 0.1 watts. The obtained results were then plotted in form of graphs using the Pandas and PyPlot libraries. The results for each algorithm were recorded in individual CSV files. To compare the energy usage of these five methods, all the results from each algorithm were combined in an Excel file. The experiment design can be summarized as follows

- Nature Inspired Optimization Algorithm (NIOA): PAR, CUC, CAM, BAT, FIR
- Programming Language: Python
- Benchmark Function: Sphere Function
- Space for Search: [-5.12, 5.12]

4. Results:

The details of energy consumed by each optimization algorithm for top 10 parameter sets are shown in Table 2, while Table 3 depicts the average power, accuracy and CO_2 emission of given models.

Name of Algorithm	Keys	Time Taken(s)	Total Energy(J)	CPU Energy(J)	Disk Energy(J)	Base Energy(J)	Power Consumption(W)
Bat Algorithm	BAT-0042	100	323.20	177.20	0.0	150.0	3.232
	BAT-0034	793	2728.30	1561.6	0.1	1189.5	3.440
	BAT-0008	76	247.1	136.40	0.0	114.0	3.251
	BAT-0029	230	761.90	427.20	0.0	345.0	3.313
	BAT-0011	78	251.80	138.5	0.0	117.0	3.228
	BAT-0017	533	1741.1	964.30	0.0	799.5	3.267

	BAT-0003	306	980.20	535.40	0.0	459.0	3.203
	BAT-0040	150	490.00	270.80	0.0	225.0	3.267
	BAT-0016	115	371.60	204	0.0	172.5	3.231
	BAT-0020	9	29.4	16.4	0.0	13.5	3.267
Camel Algorithm	CAM-0000	164	527.00	287.30	0.3	264.0	3.213
	CAM-0049	704	2264.00	1241.10	0.0	1056.0	3.215
	CAM-0024	222	708.40	384.10	0.0	333.0	3.191
	CAM-0003	164	529.00	289.40	0.0	246.0	3.226
	CAM-0026	97	313.80	172.60	0.0	145.5	3.235
	CAM-0008	19	61.3	33.5	0.0	28.5	3.226
	CAM-0041	38	123.7	69.10	0.0	57.0	3.255
	CAM-0033	840	2695.80	1475	0.6	1260.0	3.209
	CAM-0025	124	398.60	218.50	0.0	186.0	3.214
	CAM-0006	53	172.80	95.70	0.0	79.5	3.260
Cuckoo Search	CUC-0009	16	51.10	27.70	0.0	24.0	3.194
	CUC-0031	264	852.80	468.30	0.0	396.0	3.230
	CUC-0010	84	269.3	146.50	0.0	126.0	3.206
	CUC-0042	678	2185.10	1198.30	0.0	1017.0	3.223
	CUC-0035	40	130.0	71.90	0.0	60.0	3.25
	CUC-0040	612	1978.50	1086.70	0.1	918.0	3.233
	CUC-0041	258	832.00	455.3	0.0	387.0	3.225
	CUC-0039	63	206.30	114.50	0.0	94.5	3.275
	CUC-0046	84	273.6	151.30	0.0	126.0	3.257
	CUC-0048	69	222.40	122.40	0.0	103.5	3.223
Firefly Algorithm	FIR-0042	340	1045.90	551.20	0.0	510.0	3.076
	FIR-0037	218	709.10	388.80	0.0	327.0	3.253
	FIR-0003	933	3013.40	1654.10	0.0	1399.5	3.230
	FIR-0049	93	288.2	152.8	0.0	139.5	3.099
	FIR-0043	527	1623.00	855.80	0.0	790.5	3.080
	FIR-0041	129	397.70	209.50	0.0	193.5	3.083
	FIR-0014	74	238.5	131.50	0.0	111.0	3.223
	FIR-0009	119	386.5	212.30	0.0	178.5	3.248
	FIR-0012	86	280.20	154.70	0.0	129.0	3.258
	FIR-0000	92	295.70	162.2	0.0	138.0	3.214
Particle Swarm Optimization	PAR-0016	186	609.60	338.9	0.0	279.0	3.277
	PAR-0043	21	67.90	37.8	0.0	31.5	3.233
	PAR-0030	75	269.00	157.5	0.3	112.5	3.587
	PAR-0003	168	551.70	307.70	0.0	252.0	3.284
	PAR-0022	116	380.20	211.10	0.1	174.0	3.278
	PAR-0004	87	285.70	159.30	0.0	130.5	3.284

	PAR-0045	64	207.2	114.4	0.0	96.0	3.238
	PAR-0044	61	200.80	111.80	0.0	91.5	3.292
	PAR-0032	153	493.50	270.90	0.0	229.5	3.224
	PAR-0015	33	107.9	59.50	0.0	49.5	3.270

Table 2. Energy Consumption of Each Algorithm for Top 10 models

Name Of Algorithm	Keys	Avg. Accuracy	Avg. Time Taken(s)	Avg. Energy Used(J)	Avg. Equivalent CO$_2$ Emission(mg)
Bat Algorithm	['BAT-0042', 'BAT-0034', 'BAT-0008', 'BAT-0029', 'BAT-0011', 'BAT-0017', 'BAT-0003', 'BAT-0040', 'BAT-0016', 'BAT-0020']	93.13047	239.0	792.46	187.10861
Camel Algorithm	['CAM-0000', 'CAM-0049', 'CAM-0024', 'CAM-0003', 'CAM-0026', 'CAM-0008', 'CAM-0041', 'CAM-0033', 'CAM-0025', 'CAM-0006']	93.12125	242.5	779.44	184.03444
Cuckoo Search	['CUC-0009', 'CUC-0031', 'CUC-0010', 'CUC-0042', 'CUC-0035', 'CUC-0040', 'CUC-0041', 'CUC-0039', 'CUC-0046', 'CUC-0048']	93.07515	216.8	700.11	165.30375
Firefly Algorithm	['FIR-0042', 'FIR-0037', 'FIR-0003', 'FIR-0049', 'FIR-0043', 'FIR-0041', 'FIR-0014', 'FIR-0009', 'FIR-0012', 'FIR-0000']	93.14431	261.1	827.82	195.45750
Particle Swarm Optimization	['PAR-0016', 'PAR-0043', 'PAR-0030', 'PAR-0003', 'PAR-0022', 'PAR-0004', 'PAR-0045', 'PAR-0044', 'PAR-0032', 'PAR-0015']	93.12125	96.4	317.35	74.92986

Table 3. Average Accuracy and Energy Consumption for Each Algorithm

Fig. 3 compares the energy usage of each method for clarity, and Fig. 4 displays the average accuracy of the five algorithms for the top 10 epochs.

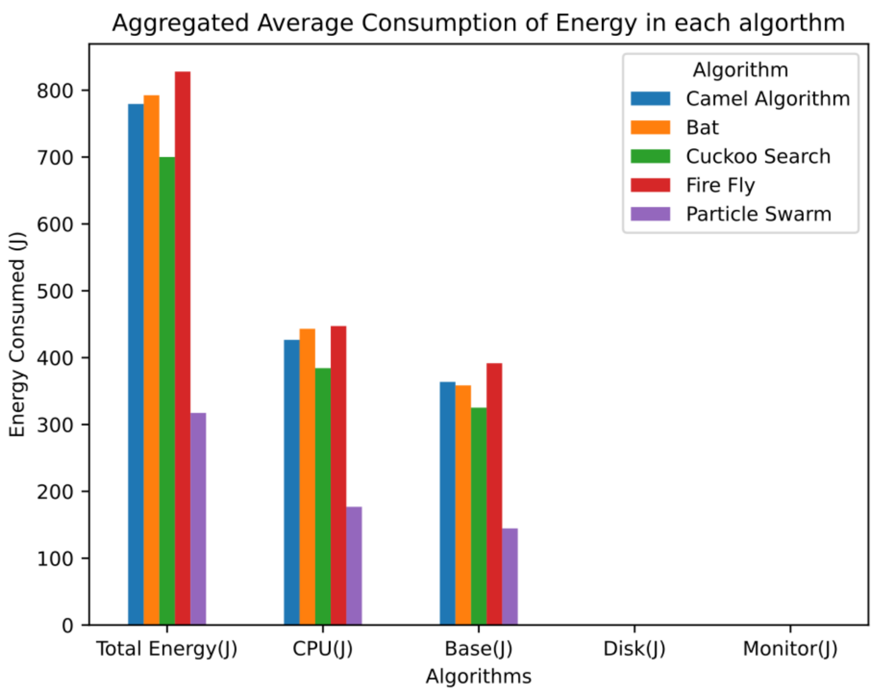


Figure 3. Aggregated Average consumption of Energy in each Algorithm

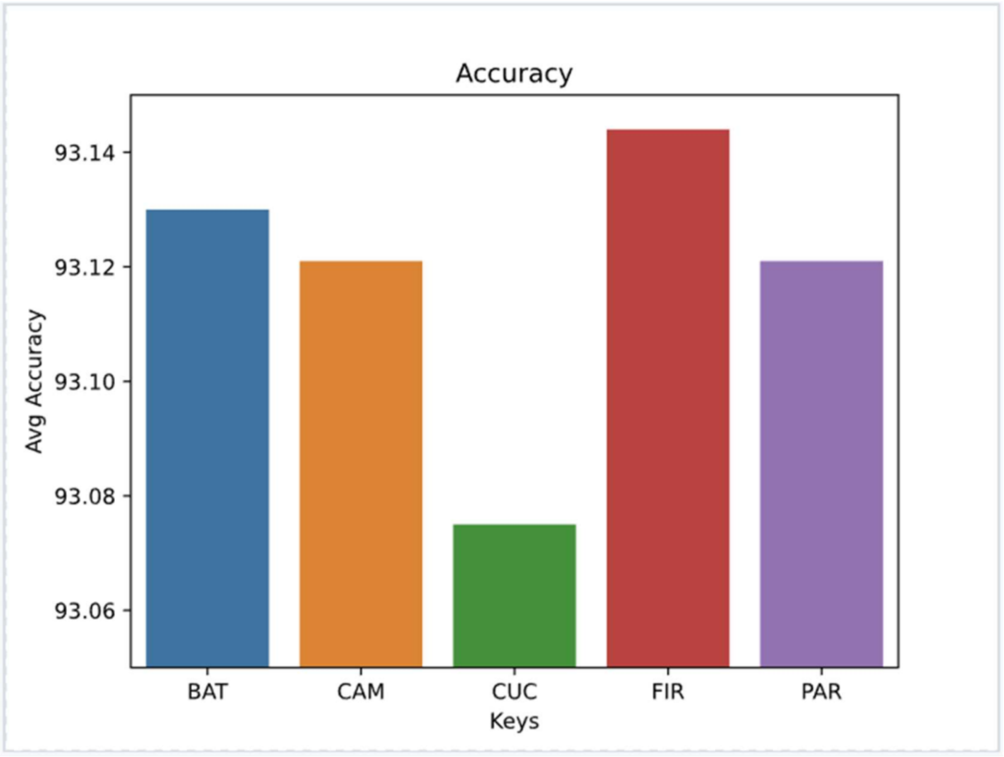


Figure 4. Average Accuracy of top 10 epochs achieved for each of the five nature-inspired algorithms

Fig. 5 shows the average energy consumed by the five algorithms for top 10 epochs, while Fig. 6 shows the average CO_2 emitted by the five algorithms for top 10 epochs.

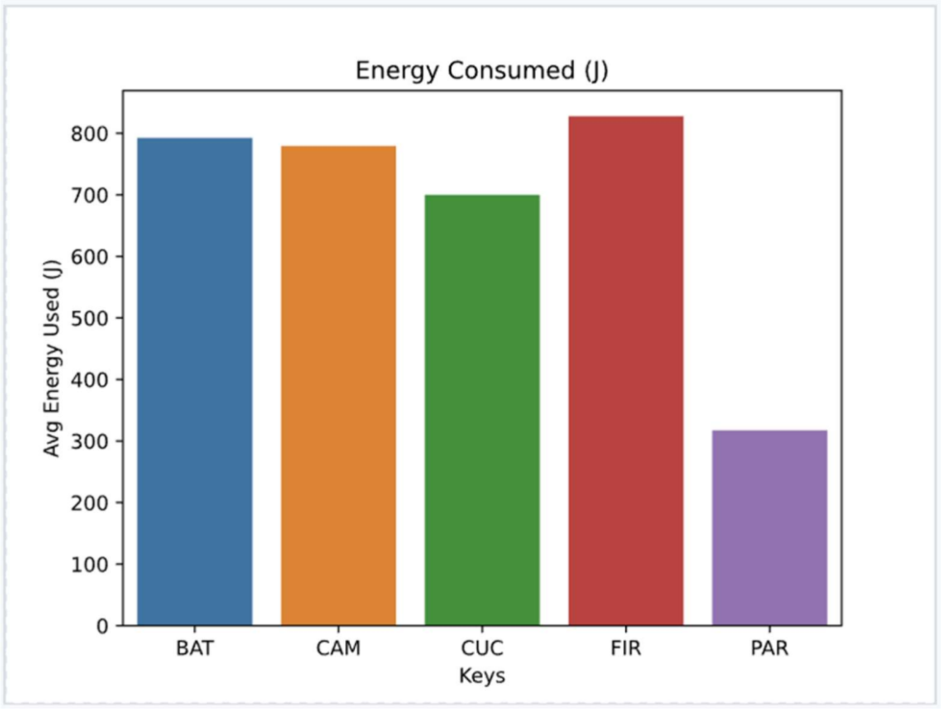


Figure 5. Average Energy Consumed of top 10 epochs achieved for each of the five nature-inspired algorithms

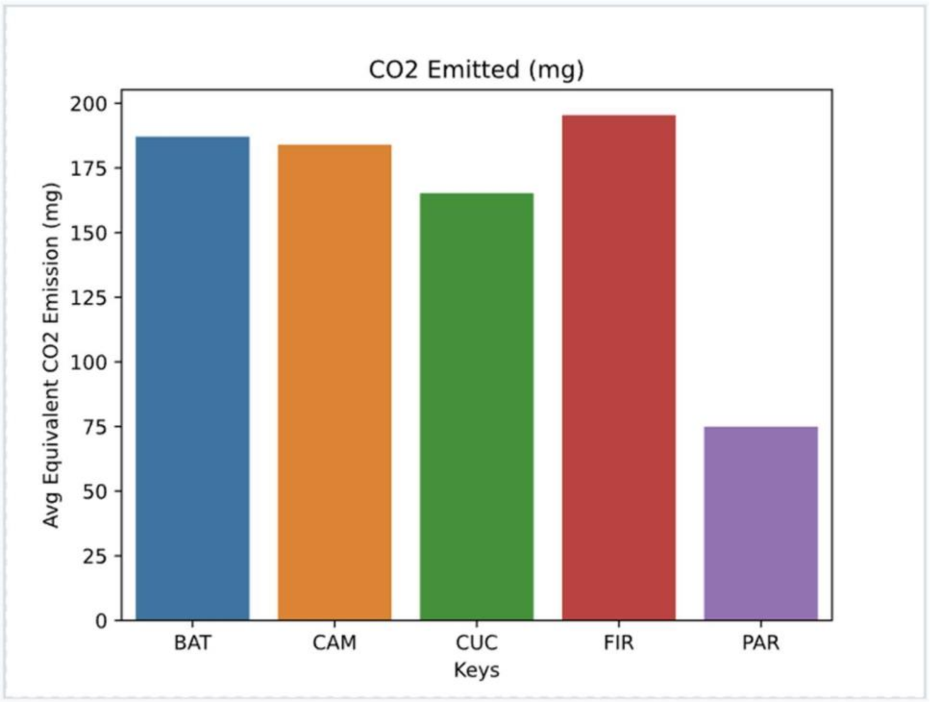


Figure 6. Average Equivalent CO_2 emissions of top 10 epochs achieved for each of the five nature-inspired algorithms

As was previously said, different hardware specs would produce various outcomes. As a result, the outcomes will alter if the trials are carried out on a laptop with different specifications. PAR is used as the foundation to explore the energy consumption ratio of other algorithms, which is presented in Table 4, because it is discovered to have the lowest energy consumption. .

Algorithm	CPU Energy Consumption(J)	Ratio Comparison to PAR
Bat Algorithm	443.18	2.505
Camel Algorithm	426.63	2.412
Cuckoo Search	384.29	2.172
Firefly Algorithm	447.29	2.529
Particle Swarm	176.89	1.0
Algorithm	Total Energy Consumed(J)	Ratio Comparison to PAR
Bat Algorithm	792.46	2.497
Camel Algorithm	779.44	2.456
Cuckoo Search	700.11	2.206
Firefly Algorithm	827.82	2.609
Particle Swarm	317.35	1.0

Table 4. Energy Consumption Ratio for Each Algorithm

Energy usage for each optimization algorithm varies greatly. But as the number of decision trees increases, it is observed that Particle Swarm Optimization algorithm has the highest accuracy to energy consumption ratio of 0.29343. Firefly Algorithm performs the worst with the accuracy to energy consumption ratio of 0.11252 (Fig 7).

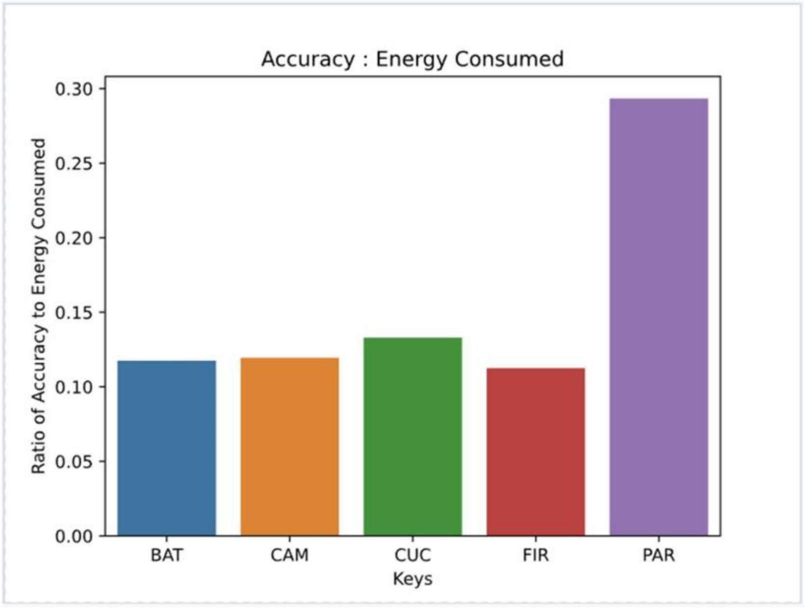


Figure 7. Ratio of Accuracy to Energy Consumption of top 10 epochs achieved for each of the five nature-inspired algorithm

5. Limitations

It is important to note that the present study has certain limitations that should be considered when interpreting the results. Firstly, the results are specific to the problem type and algorithm implementations that were examined and may not be generalizable to other scenarios. Secondly, the study may be influenced by the characteristics of the input data and the hardware utilized. Thirdly, energy consumption was measured using the Microsoft Joulemeter software, which has a resolution of 0.1J per reading. Lastly, it is acknowledged that energy efficiency varies among different hardware, operating systems, and CPUs, thus energy consumption may vary depending on the device and configuration used.

6. Conclusion

In the present experiment, it was found that the Particle Swarm Optimization algorithm exhibited the least energy consumption while maintaining relatively high accuracy. As such, it can be concluded that it is the most energy-efficient algorithm among the ones evaluated in this study. The purpose of this study was to emphasize the significance of energy efficiency in technology and the capabilities of nature-inspired optimization algorithms to decrease energy consumption. This will aid in enhancing the energy efficiency of various systems, thereby contributing to a more sustainable future.

7. Future Scope

The potential for further research in this field includes the possibility of incorporating other optimization algorithms and applying them to various real-world problems. Additionally, the study can be extended to optimize the algorithms on different hardware platforms to minimize the effects of architectural differences, thus allowing for a comprehensive examination of the energy characteristics of algorithms. Furthermore, there is scope for further research on other methods of hyper-parameter tuning to provide even more optimal and energy-efficient results.

8. Additional Requirements

Nature-inspired optimization algorithms implementation provided by the NiaPy machine learning library.

9. References

[1] Bayer H, Nebel M. Evaluating algorithms according to their energy consumption. Math Theory Comput Pract. 2009;48:1–25. http://wwwagak.cs.uni-kl.de/downloads/papers/Evaluating_Algorithms_according_to_their_Energy_Consumption.pdf

[2] Ahmad, Rafeeq and Asim, Mohd and Khan, Safwan Zubair and Singh, Bharat, Green IoT — Issues and Challenges (March 11, 2019). Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019, Available at SSRN: https://ssrn.com/abstract=3350317 or http://dx.doi.org/10.2139/ssrn.3350317

[3] 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.

[4] Heaton, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach 19, 305–307 (2018). https://doi.org/10.1007/s10710-017-9314-z.

[5] Schwartz, R., Dodge, J., Smith, N., & Etzioni, O. (2019). Green AI. Communications of the ACM, 63, 54 - 63.

[6] Li, H., Liu, X., Huang, Z., Zeng, C., Zou, P., Chu, Z. and Yi, J., 2020. Newly emerging nature-inspired optimization-algorithm review, unified framework, evaluation, and behavioural parameter optimization. IEEE Access, 8, pp.72620-72649. https://ieeexplore.ieee.org/abstract/document/9064786/

[7] Yang X-S. Nature-inspired Optimization Algorithms. London: Academic Press; 2020. https://www.sciencedirect.com/book/9780128219867/nature-inspired-optimization-algorithms

[8] 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.

[9] 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.

[10] 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.

[11] 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

[12] 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.

[13] 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.

[14] 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.

[15] Jamil, M., Kor, AL. Analyzing energy consumption of nature-inspired optimization algorithms. GRN TECH RES SUSTAIN 2, 1 (2022). https://doi.org/10.1007/s44173-021-00001-9

[16] K. Albrecht and K. Michael, "Connected: To Everyone and Everything [Guest Editorial: Special Section on Sensors]," in IEEE Technology and Society Magazine, vol. 32, no. 4, pp. 31-34, winter 2013, doi: 10.1109/MTS.2013.2291170.

[17] A. Berl et al., "Energy-Efficient Cloud Computing," in The Computer Journal, vol. 53, no. 7, pp. 1045-1051, Sept. 2010, doi: 10.1093/comjnl/bxp080.

[18] Maga, D., Hiebel, M. & Knermann, C. Comparison of two ICT solutions: desktop PC versus thin client computing. Int J Life Cycle Assess 18, 861–871 (2013). https://doi.org/10.1007/s11367-012-0499-3

[19] Marco Ajmone Marsan, Michela Meo, Energy efficient wireless Internet access with cooperative cellular networks, Computer Networks, Volume 55, Issue 2, 2011, Pages 386-398, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2010.10.017 .

[20] Kudtarkar P, DeLuca TF, Fusaro VA, Tonellato PJ, Wall DP. Cost-Effective Cloud Computing: A Case Study Using the Comparative Genomics Tool, Roundup. Evolutionary Bioinformatics. 2010;6. doi:10.4137/EBO.S6259

[21] J. Baliga, R. W. A. Ayre, K. Hinton and R. S. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," in Proceedings of the IEEE, vol. 99, no. 1, pp. 149-167, Jan. 2011, doi: 10.1109/JPROC.2010.2060451.

[22] Global e-Sustainability Initiative. 2012. Available online: https://www.telenor.com/wp-content/uploads/2014/04/SMARTer-2020-The-Role-of-ICT-in-Driving-a-Sustainable-Future-December-2012._2.pdf

[23] Andrae ASG, Edler T. On global electricity usage of communication technology: trends to 2030. Challenges 2015; 6: 117-57. https://doi.org/10.3390/challe6010117

[24] Barlage D. Impact of the Digital Revolution on World Wide Energy Consumption. [cited 2019 Sept 25]: Available from: https://aroundtheworld.ualberta.ca/portfolio/douglas-barlage/

[25] Andrae, Anders. (2019). Comparison of Several Simplistic High-Level Approaches for Estimating the Global Energy and Electricity Use of ICT Networks and Data Centers. International Journal of Green Technology. 5. 50-63. 10.30634/2414-2077.2019.05.06.

[26] Webb M, et al.Smart 2020: Enabling the low carbon economy in the information age. Clim Group Lond. 2008; 1(1):1.

[27] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S. Blair, Adrian Friday, The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations, Patterns, Volume 2, Issue 9, 2021, 100340, ISSN 2666-3899, https://doi.org/10.1016/j.patter.2021.100340

[28] Murugesan S. Going Green with IT: Your Responsibility Toward Environmental Sustainability. Arlington: Cutter Consortium; 2007

[29] 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.

[30] Schmitz MT, Al-Hashimi BM, Eles P. System-level Design Techniques for Energy-efficient Embedded Systems. Berlin: Springer Science & Business Media; 2004.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] Engel M. Sustainable software design. In: Green Information Technology. San Francisco: Elsevier; 2015. p. 111–27.

[39] Dastbaz M, Pattinson C, Akhgar B. Green Information Technology: A Sustainable Approach. San Francisco: Morgan Kaufmann; 2015

[40] Ardito L, Procaccianti G, Torchiano M, Vetro A. Understanding green software development: A conceptual framework. IT Prof. 2015;17(1):44–50.

[41] Murugesan S. Harnessing green IT: Principles and practices. IT Prof. 2008;10(1):24–33.

[42] 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.

[43] Bener AB, Morisio M, Miranskyy A. Green software. IEEE Softw. 2014;31(3): 36–9

[44] Susanne Albers and Hiroshi Fujiwara. 2007. Energy-efficient algorithms for flow time minimization. ACM Trans. Algorithms 3, 4 (November 2007), 49–es. https://doi.org/10.1145/1290672.1290686

[45] T. Li, S. Wu, S. Chen and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 2010, pp. 1-9, doi: 10.1109/INFCOM.2010.5461947.

[46] Q. Zhou et al., "Energy Efficient Algorithms based on VM Consolidation for Cloud Computing: Comparisons and Evaluations," 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, VIC, Australia, 2020, pp. 489-498, doi: 10.1109/CCGrid49817.2020.00-44.

[47] 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.

[48] Yang X-S. Nature-inspired optimization algorithms: Challenges and open problems. J Comput Sci. 2020;46:101104.

[49] 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.

[50] 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.

[51] 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.

[52] 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 .

[53] 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 .

[54] 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 .

[55] 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 .

[56] X.-S. Yang, A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (Eds. J. R. Gonzalez et al.), SCI 284, 65-74 (2010) https://doi.org/10.48550/arXiv.1004.4170

[57] Gandomi, A.H., Yang, XS., Alavi, A.H. et al. Bat algorithm for constrained optimization tasks. Neural Comput & Applic 22, 1239–1255 (2013). https://doi.org/10.1007/s00521-012-1028-9

[58] Ibrahim, M.K., & Ali, R.S. (2016). Novel Optimization Algorithm Inspired by Camel Traveling Behavior. Journal of Electrical and Electronic Engineering, 12, 167-177.

[59] X.-S. Yang; S. Deb (December 2009). Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214. https://doi.org/10.48550/arXiv.1003.1594

[60] Gandomi, A.H., Yang, XS. & Alavi, A.H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers 29, 17–35 (2013). https://doi.org/10.1007/s00366-011-0241-y

[61] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 1998, pp. 69-73, doi: 10.1109/ICEC.1998.699146.

[62] 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.

[63] 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.

[64] Mohana R. A position balanced parallel particle swarm optimization method for resource allocation in cloud. Indian J Sci Technol. 2015;8(S3): 182–8

[65] Delice Y, Kızılkaya Aydogan 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.

[66] 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.

[67] 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

[68] Peter I. Frazier (2018) Bayesian Optimization. INFORMS TutORials in Operations Research null(null):255-278. https://doi.org/10.1287/educ.2018.0188

[69] I. Roman, J. Ceberio, A. Mendiburu and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 2016, pp. 4839-4845, doi: 10.1109/CEC.2016.7744410.

[70] Johnson, B. A., & Iizuka, K. (2016). Integrating OpenStreetMap crowdsourced data and Landsat timeseries imagery for rapid land use/land cover (LULC) mapping: Case study of the Laguna de Bay area of the Philippines. Applied Geography, 67, 140-149.

[71] Sklearn-nature-inspired-algorithm https://pypi.org/project/sklearn-nature-inspired-algorithms/

[72] NiaPy https://niapy.org/en/stable/

[73] 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.

[74] Central Electricity Authority of India Guidelines for energy to CO2 emissions: https://cea.nic.in/wp-content/uploads/baseline/2020/07/user_guide_ver14.pdf

10. Supplementary Materials:

The supplementary figures and tables are available in a separate file.

11. Funding:

No external funding has been utilized for conducting the study.

12. Data Availability Statements:

The dataset used is Crowdsourced data from OpenStreetMap is used to automate the classification of satellite images into different land cover classes (impervious, farm, forest, grass, orchard, water) [70]. Data and material of this study are available on request from the corresponding author.

13. Acknowledgments:

The authors and contributors acknowledge active support from their college Maharaja Agrasen Institute of Technology, Rohini, New Delhi.

14. Conflicts of Interest:

The authors declare no conflict of interest.