Analyzing and Rating Greenness of Nature-Inspired Algorithms

Shobhit Kumar1†, Chander Jindal1†, Kanaishk Garg1\*†, Shallu Junja1

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

**\* Correspondence:**Kanaishk Garg  
[kanaishkgarg@gmail.com](mailto:kanaishkgarg@gmail.com)

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

Keywords: Nature Inspired Optimization Algorithms, Green software, Environmental Impact , Central Electricity Authority of India, Microsoft Joulemeter, Carbon footprint

Technology is a must for individuals to go about their everyday lives. From streaming to supercomputers, all are designed to make our lives simpler. All these machines work on the basic principle of machine learning. Artificial learning has allowed making prompt and efficient decision-making. Machine learning can be said to be the key to analyzing data for decision-making. With this much usage, it becomes important that these algorithms run under minimum resources so we can reduce recurring costs and provide efficient results in less time. An optimizer is a method or algorithm to update the various parameters that can reduce the loss with much less effort. One such optimizer is the Nature-Inspired Optimization (NIO) algorithm. They are highly efficient in finding optimized solutions to multi-dimensional and multi-modal problems.

This study presents a critical analysis of energy consumption and the corresponding carbon footprint for some popular NIO algorithms. Microsoft Joulemeter is employed for measuring the energy consumption during the runtime of each algorithm. In contrast, the corresponding carbon footprint of each algorithm is calculated based on India's Central Electricity Authority guide. The results of this study evidence that each algorithm demonstrates different energy consumption behaviors to achieve the same goal. This study will help software developers to choose better (greener) options among the tested NIO algorithms. In future work, more NIO algorithms and their variants can be considered for energy consumption analysis to identify the greenest NIO algorithms. In addition, future work can also be considered to ascertain the possible impact of different CPU architectures on the performance and energy usage of the NIO algorithms.

# Introduction

In various disciplines, such as engineering, business operations, industrial designs, etc., optimization is a commonly recurring mathematical problem. Different types of optimizations such as lowering energy costs or raising performance and efficiency are required. 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. Contrarily, NIO algorithms are population-based metaheuristics that replicate a wide range of natural phenomena. They successfully avoid local optima in comparison to traditional optimization methods. As a result, they are frequently used to resolve highly nonlinear optimization issues in various fields, like those of manufacturing, environmental engineering, finance, biology, etc.

Computing energy usage should be considered while designing programs targeting elevated performance and mobile software applications due to the rise of mobile and IoT devices. Software programs' improved algorithms & data structures can make them more environmentally and energy-friendly. In previous decades, the sole performance indicator taken into account for study as well as optimization of an algorithm was runtime, which served as the basis for evaluating an algorithm's performance. But the rapid development of high-performance computers and embedded systems with faster processors in recent years has led to a rise in energy usage. As a result, it is essential to take into account an algorithm's energy consumption while evaluating it (i.e. in terms of performance and sustainability). Since an algorithm's implementation will affect energy usage and environmental impact, its efficacy and efficiency must be evaluated in the context of a specific application.

One approach for assessing the ecological consequences of computers and other computing devices is through carbon footprint by evaluating a program's degree of power efficiency related to carbon footprint and implementing it into ecologically friendly company operations or procedures organisations may make the application a crucial component of their corporate social responsibility activities. Machine learning model deployment has grown massively in recent years. Considerable issues have emerged about the energy usage and expense related to developing ML models and training them. Therefore, it's indeed crucial to consider an application's carbon footprint while planning, constructing, as well as deploying it.

More than a hundred NIO algorithms and their variations are now known and available in the literature. However, this work intends to examine the energy consumption and accompanying carbon footprint for some commonly used NIO algorithms, including the Bat Algorithm, Hybrid Bat Algorithm, Firefly Algorithm, Hybrid Bat Self Adaptive Algorithm, and the Grey Wolf Algorithm. Due to the vast range of applications for these algorithms, they were taken into consideration for this study. This study intends to show how one may experimentally evaluate the energy usage of various algorithms. Keep in mind that future work could focus on alternative NIO algorithms.

The Bat Algorithm (BA) : There are around 1000 species of bats. The Bat Algorithm (BA) is based on the Echolocation behavior of microbats. 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. In a variety of fields, including data mining, big data, and machine learning, BA has been used to address challenging issues.

Firefly Algorithm (FA) : The glowing pattern that firefly swarms exhibit served as inspiration for FA. FA is incredibly flexible and easy to use. It is based on the ideas that the attractiveness and the brightness are inversely correlated and that Fireflies are attracted to one another, if two fireflies have the same brightness. The software creates creative approaches and continues to search 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.

Grey Wolf Algorithm (GWO) : The leadership structure and preying methods of grey wolves served as inspiration for the Grey Wolf Algorithm. Large canines called grey wolves operate in well-organized packs. The typical pack size is between 5 and 12. In a pack, there are four different wolf levels. GWO imitates the grey wolves' leadership model and foraging strategy. Scouting, Stalking, Surrounding, and Attacking are the four major stages of grey wolf hunting. The method has been used to address a variety of issues, including the segmentation of satellite images, the estimate of transmission line characteristics, the assessment of biochemical data, etc.

Hybrid Bat Algorithm (HBA) : Due to Bat Algorithm possibly having certain drawbacks, such as a very quick convergence rate that slows down with time, the accuracy of the outcomes is impacted by this limitation. To overcome this limitation, a modification was proposed. By employing the Differential Evolution Algorithmic techniques to hybridise the classical Bat Algorithm, the HBA was created.

Hybrid Self Adaptive Bat Algorithm (HSABA) : The pulse rate and the loudness are the two strategy parameters used by the original bat algorithm. The former controls how the best solution is improved, while the latter affects how the best solution is accepted. During the execution of the original bat algorithm, both of the aforementioned parameters are fixed. These parameters in the self-adaptive bat algorithm (SABA) are self-adaptive.

The findings of the SABA algorithm have now been further improved by hybridization with local search heuristics, leading to the creation of the hybrid self-adaptive bat algorithm (HSABA).

The objectives of our study are as follows.

1. Implement the above-stated NIO Algorithms using python as the programming language and the class NatureInspiredSearchCV provided by the scikit-learn machine learning library, then run a series of experiments to determine how much energy each method uses.

2. Based on how much energy each technique uses, calculate its equivalent carbon footprint.

3. Conduct a critical review of the literature on the effects of ICT on the environment, green or energy-efficient programs, the effects of software's power usage on hardware, the analysis of software's consumption of electricity in algorithm implementations, and energy-efficient and nature-inspired algorithms.

This study will assist programmers in selecting the greenest NIO technique to address a certain domain problem where minimizing energy usage is of the greatest priority.

# Literature Review

The Information and Communications Technology (ICT) sector is responsible for between 2.1 and 3.9% of the world's carbon emissions, with the remaining 97.9 to 96.1% coming from other industries including transportation and agriculture []. The environment and the economy would suffer as a result of the rise in carbon emissions brought on by Greenhouse Gases and other causes []. Because global demand for ICT products and services is expanding, the ICT sector can play a critical role in lowering global carbon emissions by reducing the carbon footprints of its products and services. Though much research has been conducted to make hardware and other embedded systems more energy-efficient [], a similar emphasis should be placed on the creation of energy-efficient software and applications [].

## Energy-efficient software

When a piece of software uses less energy for its effective computing and does little environmental damage, it is referred to as green or energy-efficient. Several studies have been undertaken on the energy efficiency of web-based software applications and software features. The direct impact of software on a laptop or mobile battery is easily quantified as 25% to 40% of overall energy utilized by a device. However, the indirect influence of software is more difficult to quantify because it is linked to the life cycle of the host device. Energy-efficiency of a software can only be truly achieved only when both the positive and negative impacts are properly taken into account throughout the design and deployment phases. In light of this, ICT application service optimization is crucial to reducing negative environmental effects.

## Software’s Impact on Hardware-Related energy consumption

Software's hardware-related energy usage habits directly affect how much energy hardware uses and how long a device's battery lasts []. A device's energy usage may eventually increase if a software or application that is poorly built disables various hardware-based energy-saving capabilities . For instance, it can prevent hardware from using energy-saving features and impact how the hardware is used, which could ultimately result in an increase in indirect energy usage . The creation of energy-efficient software that improves a piece of hardware's energy efficiency is one of the trickiest challenges during the design phase of an embedded system. In order to make software and applications more productive while still being energy-efficient, various trade-offs between performance and sustainability will need to be taken into account .

## Analysis of Energy Consumption in algorithms implementations

There isn't a lot of research in the literature currently available that offers insights into algorithm implementation-related energy consumption. Four sorting algorithms, namely Bubble, Merge, Quick, and Counting sort, have been examined for their energy efficiency by Rashid and colleagues . In this study, an experiment was set up on an ARM-based device, and it was determined how much energy was used by four sorting algorithms written in three different programming languages. According to this investigation, the Counting sort implementation in ARM assembly language was the most environmentally friendly choice.

Five sorting algorithms—Bubble, Insertion, Quick, Selection, and Counting sort—have had their energy consumption measured in . To measure energy consumption in this study, five separate Apps were created, one for each sorting method. According to this study, Bubble sort is the most energy-intensive algorithm, whereas Quick sort is the most energy-efficient sorting technique in typical situations.

Using C language implementation, Deepthi and colleagues ran tests to investigate the effects of various sorting algorithms on energy consumption. This study discovered that the effectiveness of these sorting algorithms is influenced by both time and energy. Six sorting algorithms were taken into consideration in this study: Quick, Merge, Shell, Insertion, Selection, and Bubble sort. Quick, Merge, and Shell sort have been found to use about the same amount of energy, although Insertion and Selection sorts perform far better than Bubble sort.

Utilizing three programming languages (C, Java, and Python), two algorithm implementation styles (Iterative and Recursive), and three algorithm types (Quick, Merge, and Insertion), Ayodele and colleagues conducted a comparative experimental analysis of the energy consumption of these three algorithms []. According to this study, the amount of energy consumed depends on the size of the data, the programming language used, and the way the algorithms are implemented. Additionally, in order to reduce energy consumption, this study offers guidance for selecting the sorting algorithm type and its algorithm implementation style.

To the best of our knowledge, NIO algorithms' energy consumption has not yet been examined, despite the fact that numerous studies have already been conducted in relation to the examination of the energy consumption of various sorting algorithm implementations. As a result, the fundamental reason for this study and its main objective are to evaluate the energy consumption of four NIO algorithms using a benchmark function.

## Nature-inspired algorithms and energy efficiency

Existing nature-inspired algorithms research primarily addresses the following areas of research: optimization using metaheuristics [] or heuristics approaches; greening processes, for example greening the supply chain , smart energy management, data center energy efficiency; energy efficiency and energy optimization 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.

# Methods

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

## Data Collection:

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. The dataset used for our research has been taken from Kaggle.

## Energy Profiling:

The estimated energy consumption of each NIO method was calculated using Microsoft Joulemeter software, which can track the energy used by a running program or software as well as by particular hardware resources, including CPU, Monitor, Disk, and Idle or Base power.

## Carbon Footprint:

The guidelines of the Central Electricity Authority of India have been used as method for calculating carbon emissions. After obtaining the Energy consumed for an experiment (in terms of kWh), the data is converted to equivalent carbon emitted based on the following formula,

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

## System Specification:

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

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

## Experiment design:

First calibrated Microsoft Joulemeter on the specified system. Next, we implemented a random forest classifier as our base classifier and used the NIO algorithms to optimize our results. We then calculated CPU energy consumption for each of the instances for all the different algorithms by changing the number of decision trees. Joulemeter can evaluate the power usage up to 0.1 watts. As such, any error by the software is 0.1 watts. The results obtained were plotted in the form of graphs using matplotlib and seaborn library.

# Result:

This study examined the energy use and carbon impact of some popular Nature-Inspired Optimization (NIO) methods. Energy usage for each optimization algorithm varies greatly. But as the number of decision trees increases, it is observed that hybrid bat consumes the least energy and gives a decent accuracy.

# Discussion:

Despite their widespread use and efficiency, NIO algorithms have a few difficult issues. Every NIO method has algorithm-dependent parameters, and these parameters' values can greatly impact how well the algorithm performs. As parameter choices might vary depending on the algorithm or issues, it is currently unclear what the appropriate value of these parameters is to achieve an optimal balance between exploration and exploitation for a specific algorithm and a given collection of problems. As a result, it is possible to investigate the tweaking and regulating of NIO algorithms' parameter values to improve their performance while minimizing their energy usage.

It is feasible to gather data on the energy consumption of hardware resources across a range of CPU architectures to look for potential connections between NIO techniques and the energy consumption of hardware resources. All of these initiatives will provide insight into the energy efficiency of NIO algorithms for sophisticated applications.

# 6 Additional Requirements

# Nature inspired algorithms implementation provided by the scikit-learn machine learning library

# Conflict of Interest

*The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest*.

# References

## Harvard Referencing style(author-date)

[1] 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>

[2] Wang, Z.; Qin, C.; Wan, B.; Song, W.W. A Comparative Study of Common Nature-Inspired Algorithms for Continuous Function Optimization. Entropy 2021, 23, 874. <https://doi.org/10.3390/e23070874>

[3] A.M. Hemeida, Salem Alkhalaf, A. Mady, E.A. Mahmoud, M.E. Hussein, Ayman M. Baha Eldin, Implementation of nature-inspired optimization algorithms in some data mining tasks, Ain Shams Engineering Journal, Volume 11, Issue 2, 2020, Pages 309-318, ISSN 2090-4479, <https://doi.org/10.1016/j.asej.2019.10.003>.

[4] Timoteh Zaftko, A faster Hyper Parameter Tuning using Nature-Inspired Algorithms in Python, TowardsDataScience.com 2020 <https://towardsdatascience.com/a-faster-hyper-parameter-tuning-using-nature-inspired-algorithms-in-python-33a32eb34f54>

[5] Timoteh Zaftko, A faster Hyper Parameter Tuning using Nature-Inspired Algorithms in Python, TowardsDataScience.com 2020 <https://towardsdatascience.com/a-faster-hyper-parameter-tuning-using-nature-inspired-algorithms-in-python-33a32eb34f54>

[6] 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>

[7] 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](https://hbr.org/2020/09/how-green-is-your-software%20Accessed%208%20Jan%202022).

[8] 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.

[9] 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.

[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] 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.

# Data Availability Statement

The datasets ANALYZED for this study can be found in the <https://github.com/ChanderJindal/Green-Computing>