Journal of
open research software

Opfunu: An Open-source Python Library for Optimization Benchmark Functions

SOFTWARE METAPAPER

NGUYEN VAN THIEU

]u[ubiquity press

ABSTRACT

Opfunu is a Python library designed to address the need for a comprehensive suite of benchmark functions for numerical optimization algorithms. It offers a rich collection of functions, including all those used in the Congress on Evolutionary Computation (CEC) competition between 2005 and 2022, alongside over 200 traditional functions with varying complexities and dimensions. Opfunu is built on top of Numpy for ease of use, fast computation, and adheres to a modular structure. The library is freely available on GitHub, promoting open-source development and encouraging the reuse of these benchmark functions by researchers and students working in the field of optimization algorithms.

CORRESPONDING AUTHOR:
Nguyen Van Thieu
Faculty of Computer Science, PHENIKAA University, Yen Nghia, Ha Dong, Hanoi, 12116, Vietnam
thieu.nguyenvan@phenikaa-uni.edu.vn

KEYWORDS:
Benchmark Functions; CEC Functions; Mathematical Optimization Functions

TO CITE THIS ARTICLE:
Van Thieu N 2024 Opfunu: An Open-source Python Library for Optimization Benchmark Functions. *Journal of Open Research Software,* 12: 8. DOI: https://doi.org/10.5334/jors.508

(1) OVERVIEW

INTRODUCTION

In the realm of numerical optimization, benchmark functions play a pivotal role in evaluating and comparing the performance of optimization algorithms [1]. These functions serve as standardized test cases, allowing researchers and practitioners to gauge the efficacy and robustness of different optimization techniques across a variety of problem domains. By subjecting algorithms to a diverse set of benchmark functions, researchers can assess their ability to handle various challenges such as multimodality, nonlinearity, and high dimensionality, thereby gaining insights into strengths and limitations of their algorithms. One notable initiative in the realm of benchmark functions is the Congress on Evolutionary Computation (CEC) competition [2]. The CEC competition, held annually since 2005, has been instrumental in driving advancements in optimization research by providing a platform for the evaluation and comparison of optimization algorithms on standardized benchmark functions. These benchmark functions, meticulously curated by the competition organizers, represent real-world optimization problems and pose significant challenges to algorithm developers. However, despite the importance of benchmark functions in driving innovation and progress in optimization research, there has been a notable gap in readily accessible and diverse benchmark function libraries. This gap underscores the critical need for a comprehensive suite of benchmark functions readily available to researchers and practitioners alike.

To address this gap, we introduce opfunu, an open-source Python library meticulously designed to provide a vast array of benchmark functions for numerical optimization tasks. Opfunu offers a rich collection of benchmark functions, including those featured in the CEC competition and over 200 traditional functions characterized by varying complexities and dimensions. Opfunu is built on top of Numpy [3] library, a powerful numerical computing library for Python, ensuring ease of use, computational efficiency, and seamless integration with existing Python-based optimization frameworks. Its modular structure enhances flexibility, allowing users to easily incorporate custom benchmark functions or extend the library's functionality as needed.

Opfunu has been utilized in various research studies across the optimization community, facilitating rigorous experimentation, algorithmic development, and comparative analysis. For instance, [4] utilized CEC2014 benchmark functions to evaluate and compare the Hybridization of Galactic Swarm and Evolution Whale Optimization model. In study [5] proposed the nQSV-net model for workload modeling and demonstrated the effective optimization capabilities of the Improved Queuing search algorithm on the CEC2014 benchmark function set. Additionally [6], selected 20 benchmark functions from CEC2014 and CEC2015 to assess the Improved Sea Lion Optimization algorithm against other techniques. Moreover [7], proposed a specialized library for metaheuristic algorithms and utilized functions from CEC2017 to evaluate the performance of these algorithms. Notably, recent metaheuristic algorithms, such as the Q-learning based Vegetation Evolution algorithm [8], have incorporated Opfunu library functions for testing on the CEC2020 function set, as well as engineering problems and WSN coverage optimization problems. Similarly [9], proposed the strengthened RIME (rime-ice) algorithm and utilized functions from the Opfunu library's CEC2020 set. Furthermore [10], proposed the evolutionary multi-mode slime mold optimization algorithm for addressing continuous optimization problems and validated its performance using the CEC2013 function set from Opfunu. These studies collectively highlight the utility and versatility of Opfunu in advancing optimization research and fostering algorithmic innovation.

In comparison with existing software packages, modules, and libraries offering similar functionalities, Opfunu stands out due to its comprehensive collection of benchmark functions. Additionally, Opfunu provides extensive documentation, user guides, examples, test cases, and notably, it is easy to install and use. For example [11], is a package that only includes the CEC2017 benchmark functions, is not packaged as a library, and lacks documentation, making it challenging for users to understand and utilize effectively. Similarly, [12], [13], and [14] are three modules containing sets of benchmark functions for multi-dimensional problems, but they are not packaged as libraries and lack user guides and examples. [15] includes unimodal and multimodal benchmark functions but lacks a documentation website and is not packaged as a library. [16] provides basic benchmark functions in the Julia programming language but lacks examples and a documentation website. Moreover [17], offers a library providing standard functions for single and multi-objective optimization problems. However, despite having documentation and examples, it is written in the R programming language and only offers standard functions without the inclusion of CEC function sets. Opfunu, on the other hand, not only provides a comprehensive range of benchmark functions but also offers clear documentation, ease of installation, and usage, making it a standout choice for researchers and practitioners in the field of optimization.

IMPLEMENTATION AND ARCHITECTURE

Opfunu is implemented in Python, leveraging the powerful numerical computing capabilities of the NumPy library. The software follows a modular architecture, designed to provide flexibility, extensibility, and ease of maintenance. Figure 1 provides a visual representation of Opfunu's modular architecture and the relationships

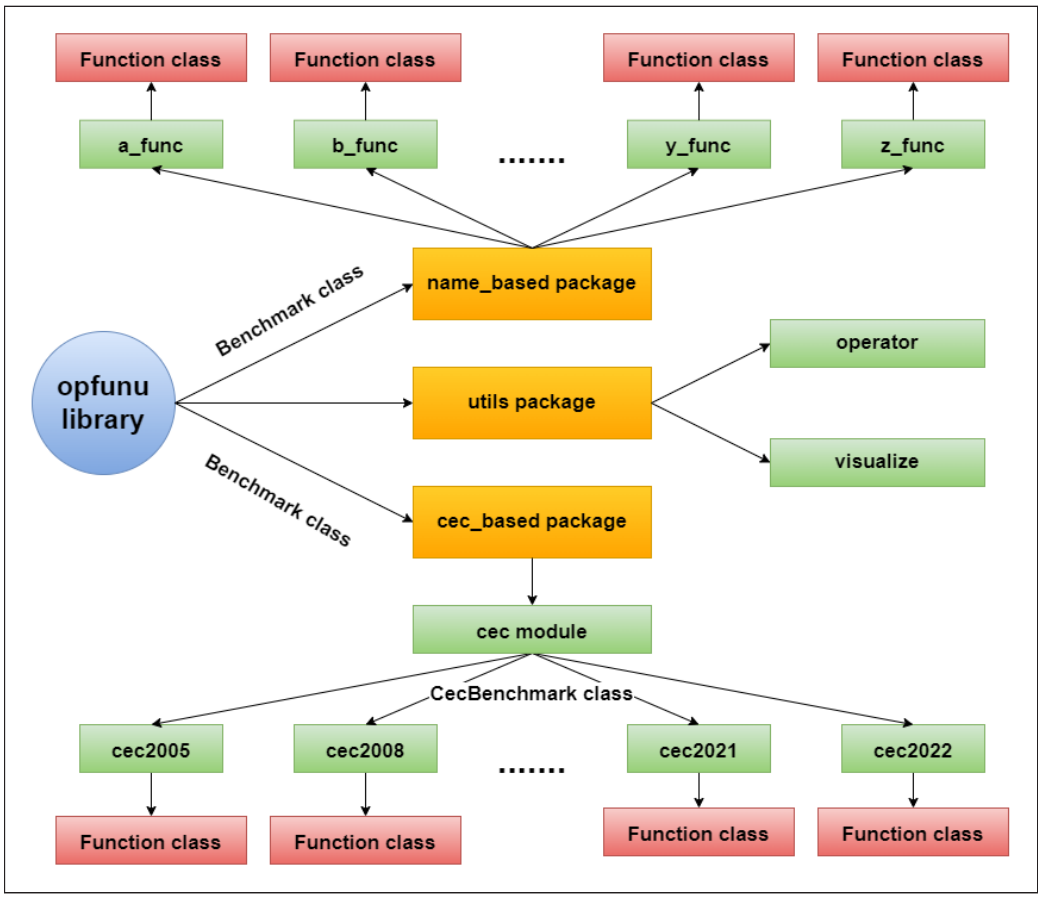

Figure 1 The overview structure of Opfunu library.

between its components. Within this architecture, several key components are outlined, including the `benchmark` module and three primary packages: “name_based,” “cec_based,” and “utils.”

- benchmark module: Contains a class named `Benchmark` that all functions in the library must inherit from. The aim is to utilize object-oriented programming inheritance to reuse code and optimize the library structure.
- name_based package: Contains all traditional benchmark functions (excluding CEC functions). It is divided into 26 modules corresponding to the alphabet. For instance, functions with names starting with the letter “a” are placed in the “a_func” module, such as “Ackley01”, “Adjiman”, “Alpine01”, etc. Similarly, all traditional functions are distinguished based on their names.
- cec_based package: Contains all CEC functions spanning from 2005 to 2022. It includes a `cec` module that contains the `CecBenchmark` class. This class inherits from the `Benchmark` class and serves as the parent class for all CEC benchmark functions. The rationale behind this class is to accommodate the unique characteristics of CEC benchmark functions, such as external data loading requirements and the need for bias value settings. The package is further divided into submodules based on the year CEC was organized, such as cec2014, cec2015, etc. The classes defining CEC functions are separated by the year of the CEC competition.
- utils package: Contains helper functions used throughout the library components to promote code reuse. It has two main modules: operator and visualize. The operator module contains functions for performing common mathematical operations and frequently reused functions, while the visualize module provides functions for visualizing benchmark functions in 2D or 3D.

CODE TEMPLATE

Users can access more indepth examples either in the Github repository or in the documentation. In this section, we provide code templates for integrating benchmark functions into some popular libraries such as ScikitOpt [18], Mealpy [19], and Opytimizer [20].

In the example below, we use the Differential Evolution (DE) algorithm in the ScikitOpt library to solve the F10 function in the CEC 2015 competition. As can be seen, the first 2 lines define the F10 function from the `opfunu` library, while the remaining lines define the DE algorithm from the ScikitOpt library. Metaheuristic algorithms typically require a fitness (objective) function and need to know the lower bounds and upper bounds for the decision variables of the problem. Therefore, the function classes in Opfunu provide these attributes, and users only need to call them for use. For instance: f10.evaluate is the fitness function, f10.lb and f10.ub are the lower bounds and upper bounds for the decision variables.

```
from opfunu.cec_based import cec2015
f10 = cec2015.F102015(ndim=30)

from sko.DE import DE

de = DE(func=f10.evaluate, lb=f10.lb, ub=f10.ub,
                          size_pop=50, max_iter=800)
best_x, best_y = de.run()
print(f"best_x: {best_x}, best_y: {best_y}")
```

Similarly to the example above, the following example utilizes the Genetic Algorithm (GA) to solve the F3 problem in the CEC 2017 competition. As can be seen, the first 2 lines still define the F3 problem from the opfunu library, while the remaining lines define the GA algorithm from the Mealpy library. Here, f3.evaluate represents the fitness function, and f3.lb and f3.ub denote the lower bounds and upper bounds of the decision variables.

```
from opfunu.cec_based import cec2017
f3 = cec2017.F32017(ndim=30)

from mealpy import GA, FloatVar

problem = {
    "obj_func": f3.evaluate,
    "bounds": FloatVar(lb=f3.lb, ub=f3.ub),
    "minmax": "min",
}
model = GA.BaseGA(epoch=100, pop_size=50)
gbest = model.solve(problem_dict1)
print(f"Solution: {gbest.solution},
             Fit: {gbest.target.fitness}")
```

Similarly to the examples above, in the following example, we utilize the Artificial Bee Colony (ABC) algorithm from the Opytimizer library to solve the F5 problem in the CEC 2022 competition. As can be observed, the first 2 lines define the F5 function from the opfunu library. The remaining lines define the ABC algorithm from the Opytimizer library. Here, f5.ndim indicates the number of decision variables, f5.evaluate represents the fitness value, and f5.lb and f5.ub denote the lower and upper bounds of the problem.

```
from opfunu.cec_based import cec2022
f5 = cec2022.F52022(ndim=30)

from opytimizer import Opytimizer
from opytimizer.core import Function
from opytimizer.optimizers.swarm import PSO
from opytimizer.spaces import SearchSpace

space = SearchSpace(n_agents=20, n_variables=f5.ndim,
               lower_bound=f5.lb, upper_bound=f5.ub)
optimizer = PSO()
function = Function(f5.evaluate)

opt = Opytimizer(space, optimizer, function)
opt.start(n_iterations=1000)
```

Through the examples above, it is evident that integrating the opfunu library into other optimization algorithm libraries is straightforward.

QUALITY CONTROL

Opfunu undergoes rigorous testing to ensure its reliability, functionality, and performance across various scenarios. We utilize pytest library for both functional and component testing to verify the integrity of the software. These tests involve running the library with various benchmark functions and comparing the obtained results against known optimal solutions or established benchmarks.

Additionally, Opfunu is tested in multiple environments to ensure compatibility and reliability across different platforms. These environments include:

- Local development environments, such as Windows, macOS, and Linux, using popular Python distributions like pip and Pypi.
- Continuous integration (CI) pipelines, where automated tests are run on cloud-based platforms like GitHub Actions to validate code changes and ensure consistency across different code branches. Opfunu's testing primarily occurs in different Python environments like Python 3.7, 3.8, 3.9, 3.10, and 3.11.

To assist users in quickly assessing the software's correctness, Opfunu provides the following resources:

- Documentation: Opfunu's documentation includes detailed instructions on installing the software, using its features, and interpreting the output.
- Example Usage: Opfunu includes example scripts demonstrating how to use its functionalities with sample input data. These examples showcase how to import the library, select functions, and execute them with sample data.
- Test Script: A well-structured test script is included within the codebase. This script demonstrates how to run functional tests, allowing users to verify the library's integrity on their machines.
- Sample Code Snippets: The GitHub repository may also include sample code snippets that users can directly employ to test Opfunu's functionalities with various benchmark functions.

These testing and support mechanisms ensure Opfunu's reliability, functionality, and ease of use for researchers and practitioners in the optimization community.

(2) AVAILABILITY

OPERATING SYSTEM

This package can be run on any operating system where python can be run (GNU/Linux, Mac OSX, Windows).

PROGRAMMING LANGUAGE

python 3.7+

ADDITIONAL SYSTEM REQUIREMENTS

None

DEPENDENCIES

numpy >= 1.16.5, matplotlib >= 3.3.0, Pillow >= 9.1.0, requests >= 2.27.0

LIST OF CONTRIBUTORS

Nguyen Van Thieu

SOFTWARE LOCATION

Archive

Name: Zenodo
Persistent identifier: https://zenodo.org/doi/10.5281/zenodo.3620960
Licence: Creative Commons Attribution 4.0 International
Publisher: Nguyen Van Thieu
Version published: 1.0.2
Date published: 22/03/24

Code repository

Name: Github
Identifier: https://github.com/thieu1995/opfunu
Licence: GNU General Public License (GPL) V3 license
Date published: 06/12/2019

LANGUAGE

English

(3) REUSE POTENTIAL

Opfunu, with its comprehensive collection of benchmark functions and user-friendly design, holds significant potential for reuse by researchers both within and outside the field of optimization. Below, we outline several potential use cases for the software:

- Benchmarking Optimization Algorithms: Opfunu provides a diverse set of benchmark functions suitable for evaluating and comparing optimization algorithms. Researchers across various domains, including machine learning, operations research, and engineering, can leverage Opfunu to assess the performance of their optimization techniques on standardized test cases.
- Educational Purposes: Opfunu serves as a valuable tool for teaching and learning about optimization problems. Students and instructors can leverage the library to explore various function types, understand optimization challenges, and test different algorithms.
- The empirical evidence indicates that numerous studies have utilized the Opfunu library, as exemplified by [21–25]. The quantity of such implementations continues to increase daily, demonstrating the utility of the proposed library. Moreover, as of the time of manuscript preparation, Opfunu has surpassed 100,000 downloads on PyPI.

Opfunu is designed to be flexible and extensible, allowing users to modify or extend its functionality to suit their specific research needs. Researchers interested in contributing to Opfunu or extending its capabilities can do so by:

- Adding New Benchmark Functions: Researchers can contribute novel benchmark functions by following the existing code structure and submitting pull requests on the GitHub repository.
- Extending Functionality: The library's modular design allows for the addition of new functionalities, such as visualization tools for analyzing optimization results. Contributions can be made through pull requests or by creating separate packages that interact with Opfunu.
- Reporting Issues and Bugs: Researchers encountering issues or bugs can report them on the GitHub repository's issue tracker. This helps maintain the library's quality and functionality.
- Enhancing documentation: Contributors can improve the documentation to provide clearer guidance on software usage, implementation details, and best practices.
- Integrating additional features: Opfunu can be extended to incorporate new features, such as visualization tools for analyzing optimization results or compatibility with other optimization frameworks.

There are several ways users can get the assistance include:

- Documentation: Opfunu's documentation provides comprehensive guidance on software installation, usage, and customization.
- GitHub Repository: The project's GitHub repository serves as a central hub for discussions, bug reports, feature requests, and contributions. Users can open issues or pull requests to report problems or propose changes.
- Community Forums: By actively engaging with the developer community and leveraging online resources, researchers can effectively utilize Opfunu and contribute to its ongoing development.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for valuable feedbacks. We also thank the community of Opfunu users for bug reports, improvement suggestions and their general friendly support and encouragement for the further development of Opfunu.

COMPETING INTERESTS

The author has no competing interests to declare.

AUTHOR AFFILIATIONS

Nguyen Van Thieu orcid.org/0000-0001-9994-8747
Faculty of Computer Science, PHENIKAA University, Yen Nghia, Ha Dong, Hanoi, 12116, Vietnam

REFERENCES

1. **Digalakis JG, Margaritis KG.** An experimental study of benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics*. 2002; 79(4): 403–416. DOI: https://doi.org/10.1080/00207160210939
2. **Liang JJ, Qu BY, Suganthan PN.** Problem definitions and evaluation criteria for the CEC *2014* special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*. 2013; 635(2): 2014.
3. **Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R.** Array programming with NumPy. *Nature*. 2020; 585(7825): 357–362. DOI: https://doi.org/10.1038/s41586-020-2649-2
4. **Nguyen BM, Tran T, Nguyen T, Nguyen G.** Hybridization of galactic swarm and evolution whale optimization for global search problem. *IEEE Access*. 2020; 8: 74991–75010. DOI: https://doi.org/10.1109/ACCESS.2020.2988717
5. **Nguyen BM, Hoang B, Nguyen T, Nguyen G.** nQSV-Net: a novel queuing search variant for global space search and workload modeling. *Journal of Ambient Intelligence and Humanized Computing*. 2021; 12: 27–46. DOI: https://doi.org/10.1007/s12652-020-02849-4
6. **Nguyen BM, Tran T, Nguyen T, Nguyen G.** An improved sea lion optimization for workload elasticity prediction with neural networks. *International Journal of Computational Intelligence Systems*. 2022; 15(1): 90. DOI: https://doi.org/10.1007/s44196-022-00156-8
7. **Van Thieu N, Mirjalili S.** MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*. 2023; 139: 102871. DOI: https://doi.org/10.1016/j.sysarc.2023.102871
8. **Zhong R, Peng F, Yu J, Munetomo M.** Q-learning based vegetation evolution for numerical optimization and wireless sensor network coverage optimization. *Alexandria Engineering Journal*. 2024; 87: 148–163. DOI: https://doi.org/10.1016/j.aej.2023.12.028
9. **Zhong R, Yu J, Zhang C, Munetomo M.** SRIME: a strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems. *Neural Computing and Applications*. 2024; 1–20. DOI: https://doi.org/10.1007/s00521-024-09424-4
10. **Zhong R, Zhang E, Munetomo M.** Evolutionary multi-mode slime mold optimization: a hyper-heuristic algorithm inspired by slime mold foraging behaviors. *The Journal of Supercomputing*. 2024; 1–32. DOI: https://doi.org/10.1007/s11227-024-05909-0
11. **Tilley D.** "CEC2017-py". 2020. https://github.com/tilleyd/cec2017-py.
12. **Plevris V.** "Collection30Functions". 2021. https://github.com/vplevris/Collection30Functions.
13. **Ardeh MA.** "BenchmarkFcns". 2016. https://github.com/mazhar-ansari-ardeh/BenchmarkFcns.
14. **Diessner M.** "benchfuncs". 2022. https://github.com/mikediessner/benchfuncs.
15. **Tomochika K.** "optimization-evaluation". 2017. https://github.com/tomochi222/optimization-evaluation.
16. **Alexander R.** "BenchmarkFunctions.jl". 2020. https://github.com/rbalexan/BenchmarkFunctions.jl.
17. **Bossek J.** "smoof". 2015. https://github.com/jakobbossek/smoof.
18. **Fei G.** "scikit-opt". 2019. https://github.com/guofei9987/scikit-opt.
19. **Van Thieu N.** "mealpy". 2020. https://github.com/thieu1995/mealpy.
20. **de Rosa G.** "opytimizer". 2019. https://github.com/gugarosa/opytimizer.
21. **Nguyen T, Hoang B, Nguyen G, Nguyen BM.** A new workload prediction model using extreme learning machine and enhanced tug of war optimization. *Procedia Computer Science*. 2020; 170: 362–369. DOI: https://doi.org/10.1016/j.procs.2020.03.063
22. **Alfadhli J, Jaragh A, Alfailakawi MG, Ahmad I.** FP-SMA: an adaptive, fluctuant population strategy for slime mould algorithm. *Neural Computing and Applications*. 2022; 34(13): 11163–11175. DOI: https://doi.org/10.1007/s00521-022-07034-6
23. **Van Thieu N, Oliva D, Pérez-Cisneros M.** MetaCluster: An open-source Python library for metaheuristic-based clustering problems. *SoftwareX*. 2023; 24: 101597. DOI: https://doi.org/10.1016/j.softx.2023.101597
24. **Abed AM, AlArjani A, ElAttar S.** Reduce the delivery time and relevant costs in a chaotic requests system via lean-Heijunka model to enhance the logistic Hamiltonian route. *Results in Engineering*. 2024; 21: 101745. DOI: https://doi.org/10.1016/j.rineng.2023.101745
25. **Van Thieu N, Barma SD, Van Lam T, Kisi O, Mahesha A.** Groundwater level modeling using augmented artificial ecosystem optimization. *Journal of Hydrology*. 2023; 617: 129034. DOI: https://doi.org/10.1016/j.jhydrol.2022.129034

TO CITE THIS ARTICLE:
Van Thieu N 2024 Opfunu: An Open-source Python Library for Optimization Benchmark Functions. *Journal of Open Research Software,* 12: 8. DOI: https://doi.org/10.5334/jors.508

Submitted: 24 March 2024 **Accepted:** 22 April 2024 **Published:** 22 May 2024