The Journal of Open Source Software

NiaPy: Python microframework for building nature-inspired algorithms

Grega Vrbančič[1], Lucija Brezočnik[1], Uroš Mlakar[1], Dušan Fister[2], and Iztok Fister Jr.[1]

1 University of Maribor, Faculty of Electrical Engineering and Computer Science **2** University of Maribor, Faculty of Economics and Business

DOI: 10.21105/joss.00613

Software

- Review
- Repository
- Archive

Submitted: 28 February 2018
Published: 22 March 2018

Summary

Nature-inspired algorithms are a very popular tool for solving optimization problems (Yang 2014), (Hassanien and Emary 2016). Numerous variants of nature-inspired algorithms have been developed (Iztok Fister Jr. and Fister 2013) since the beginning of their era. To prove their versatility, those were tested in various domains on various applications, especially when they are hybridized, modified or adapted. However, implementation of nature-inspired algorithms is sometimes a difficult, complex and tedious task. In order to break this wall, `NiaPy` is intended for simple and quick use, without spending time for implementing algorithms from scratch.

Currently, the framework consists of algorithms, benchmark functions and supporting features. There are 8 basic implemented algorithms: Artificial Bee Colony algorithm, Bat algorithm, Differential Evolution algorithm, Firefly algorithm, Flower Pollination algorithm, Genetic algorithm, Grey Wolf Optimizer, Particle Swarm Optimization and 2 hybrid variants: Hybrid Bat algorithm, self-adaptive Differential Evolution algorithm. The following benchmark functions (Jamil and Yang 2013) are also included in framework: Ackley, Alpine, Alpine1, Alpine2, Chung Reynolds, Csendes, Griewank, Happy cat, Pintér, Qing, Quintic, Rastrigin, Ridge, Rosenbrock, Salomon, Schumer Steiglitz, Schwefel, Schwefel 2.21, Schwefel 2.22, Sphere, Step, Step2, Step3, Stepint, Styblinski-Tang, Sum Squares, Whitley. A supporting feature consists of a runner utility that allows users to run selected algorithms easily, along with predefined or customized benchmark functions. By the same token, it also allows export to various formats, such as, for example, LaTeX, JSON, and Excel.

There are also some existing similar Python implementations/frameworks of nature-inspired algorithms, as for example DEAP (Félix-Antoine Fortin and Gagné 2012), EvoloPy (Hossam Faris and Merelo 2016), PySwarms (Miranda 2018), inspyred (Garrett 2014), and Pygmo (Izzo 2012). However, NiaPy tends to be very minimalistic and easy for use, especially for practitioners and students.

Key features of NiaPy:

- Since it is a very extensible framework, new algorithms can be integrated into it easily.
- Good documentation.
- Practitioner friendly. Users do not need to implement algorithms from scratch.
- Due to the many implemented algorithms, researchers can study how similar algorithms are (the problem of metaphor-based algorithms (Sörensen 2015)).
- The framework allows a fair comparison. Only a number of function evaluations are taken as the stopping criterion.
- Reviewers can check for referential results quickly.

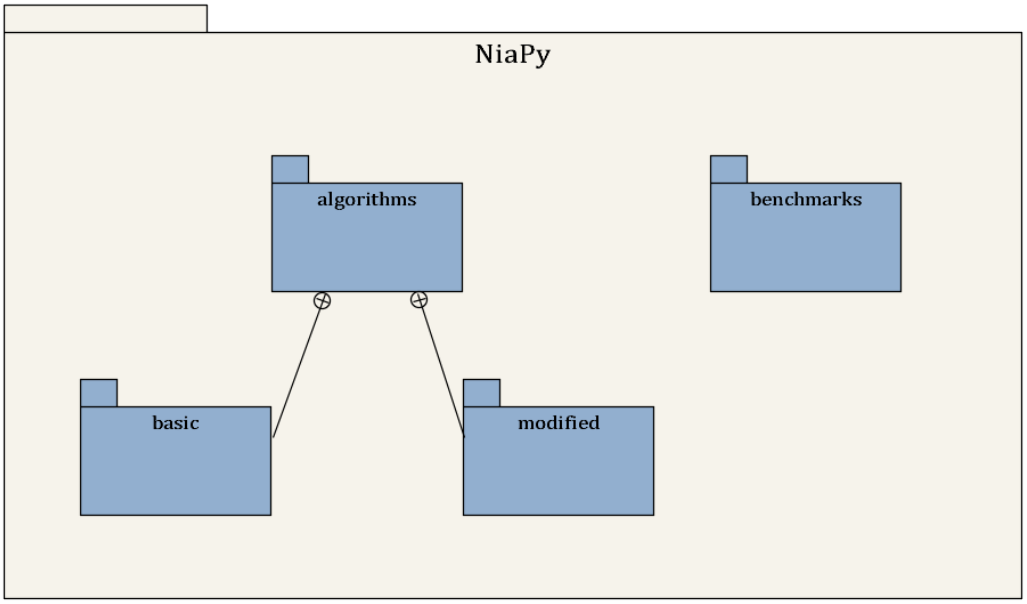

Figure 1: NiaPy architecture

- Rapid prototyping of new approaches, especially modified algorithms.
- Supportive community.

In conclusion, NiaPy is a new microframework for building and using nature-inspired algorithms in Python. Stepping stones for the design and implementation of NiaPy were the limitations of existing software, that were mostly limited in documentation, number of algorithms, inconsistent stopping criteria, non-modular software architecture. In the future, more algorithms will be added to the current collection, while support will also be developed for constraint optimization problems.

Acknowledgement

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).

References

Félix-Antoine Fortin, Marc-André Gardner, François-Michel De Rainville, and Christian Gagné. 2012. "DEAP: Evolutionary algorithms made easy." *Journal of Machine Learning Research* 13 (Jul):2171–5.

Garrett, Aaron. 2014. "inspyred: Bio-inspired Algorithms in Python." 2014. https://pypi.python.org/pypi/inspyred.

Hassanien, Aboul Ella, and Eid Emary. 2016. *Swarm intelligence: principles, advances, and applications*. CRC Press.

Hossam Faris, Seyedali Mirjalili, Ibrahim Aljarah, and Juan J Merelo. 2016. "EvoloPy: An Open-source Nature-inspired Optimization Framework in Python." In *Proceedings of the 8th International Joint Conference on Computational Intelligence*, 171–77.

Iztok Fister Jr., Iztok Fister, Xin-She Yang, and Dušan Fister. 2013. "A Brief Review of Nature-Inspired Algorithms for Optimization." *Elektrotehniški Vestnik* 80 (3):116–22.

Izzo, Dario. 2012. "Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)." In *Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT*.

Jamil, Momin, and Xin-She Yang. 2013. "A Literature Survey of Benchmark Functions For Global Optimization Problems." *International Journal of Mathematical Modelling and Numerical Optimisation* 4 (2):150–94. https://doi.org/10.1504/IJMMNO.2013.055204.

Miranda, Lester James V. 2018. "PySwarms: a research toolkit for Particle Swarm Optimization in Python." *Journal of Open Source Software* 3 (21). https://doi.org/10.21105/joss.00433.

Sörensen, Kenneth. 2015. "Metaheuristics—the metaphor exposed." *International Transactions in Operational Research* 22 (1). Wiley Online Library:3–18. https://doi.org/10.1111/itor.12001.

Yang, Xin-She. 2014. *Nature-inspired optimization algorithms*. Elsevier.