




IWOPY: Fraunhofer IWES optimization tools in Python

Jonas Schulte ¹✉

¹ Fraunhofer IWES, Küpkersweg 70, 26129 Oldenburg, Germany ✉ Corresponding author

DOI: [10.21105/joss.06014](https://doi.org/10.21105/joss.06014)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: Frauke Wiese  

Reviewers:

- [@BenWinchester](#)
- [@StewMH](#)

Submitted: 27 March 2023

Published: 09 October 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

Optimization problems are described by optimization variables, which are scalars that are modified during the process of optimization; objective functions, which depend on the variables and whose values define the metric for rating the choices of the latter; and constraints, which are also functions of the optimization variables and define validity of the solution. The variables can be discrete or continuous, bounded or unbounded; the number of objectives can be one or many; and constraints can require equality or inequality. Many Python packages formulate a framework for the description of such problems, accompanied by a library of optimizers. Hence, switching from one to another optimization package can often be tedious and a meta-solution is required that can serve as a single interface to multiple optimization packages. The package `iwopy` solves this problem by providing a convenient and flexible interface which is general enough to be applicable to a wide range of optimization problems.

Statement of need

The Python package `iwopy` provides a general object-oriented formulation for optimization problems, objective functions, and optimization constraints. The optimization problem class defines the optimization variables, their bounds and their types. Objectives and constraints can then be added in an independent step to the problem, such that they can easily be exchanged and modified by the user. The framework is general enough for supporting complex science and engineering problems, and it supports single, multi and many objective optimization problems.

The core functionality of `iwopy` is to provide interfaces to other existing Python optimization packages, like `pymoo` (Blank & Deb, 2020), `pygmo` (Biscani & Izzo, 2020), or `scipy` (Virtanen et al., 2020). Once the problem is formulated within the framework sketched above, all individual optimizers from the supported linked packages can be selected and switched easily.

Note that more optimization packages are available which are not yet supported, like `pyomo` (Bynum et al., 2021), `Platypus` (Yermanos, 2023), `DEAP` (Fortin et al., 2012) and others. Each package is well suited for solving a wide range of optimization problems, and they all come with extensive user interfaces. However, `iwopy` addresses a unification of those interfaces, enabling the user to benefit from all supported optimizers without the need of extensive changes of the code base.

The design of `iwopy` has a focus on vectorized evaluation approaches, as for example often provided by heuristic algorithms that rely on the concept of populations. If the vectorized evaluation of a complete population of individual choices of optimization variables is implemented by the user, this enables a vast speed-up of optimizations compared to the one-by-one evaluation through a loop.

Acknowledgements

The development of iwopy and has been supported through multiple publicly funded research projects. We acknowledge in particular the funding by the Federal Ministry of Economic Affairs and Climate Action (BMWK) through the projects Smart Wind Farms (grant no. 0325851B) and GW-Wakes (0325397B) as well as the funding by the Federal Ministry of Education and Research (BMBF) in the framework of the project H2Digital (03SF0635).

References

- Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53), 2338. <https://doi.org/10.21105/joss.02338>
- Blank, J., & Deb, K. (2020). Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8, 89497–89509. <https://doi.org/10.1109/access.2020.2990567>
- Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Sirola, J. D., Watson, J.-P., & Woodruff, D. L. (2021). *Pyomo—optimization modeling in Python* (Third, Vol. 67). Springer Science & Business Media. <https://doi.org/10.1007/978-3-030-68928-5>
- Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13, 2171–2175. <https://dl.acm.org/doi/10.5555/2503308.2503311>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Yermanos, A. (2023). *Platypus: Single-cell immune repertoire and gene expression analysis*. <https://doi.org/10.1093/nargab/lqab023>