




linopy: Linear optimization with n-dimensional labeled variables

Fabian Hofmann ¹

¹ Institute of Energy Technology, Technical University of Berlin

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

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

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

linopy is an open-source package written in Python to facilitate linear and mixed-integer optimization with n-dimensional labeled input data. Using state-of-the-art data analysis packages, linopy enables a high-level algebraic syntax and memory-efficient, performant communication with open and proprietary solvers. While similar packages use object-oriented implementations of single variables and constraints, linopy stores and processes its data in an array-based data model. This allows the user to build large optimization models quickly in parallel and lays the foundation for features such as fast writing to NetCDF file, masking, solving on remote servers, and model scaling.

Statement of need

Mathematical programming and optimization is a broad field in the research community. There exists a list of open-source and proprietary solvers such as GLPK ([GLPK - GNU Project - Free Software Foundation \(FSF\), n.d.](#)), Gurobi (["Gurobi - The Fastest Solver," n.d.](#)), Xpress (["FICO® Xpress Solver," n.d.](#)), each with its own strengths and weaknesses. To ensure comparability and re-usability for a wide range of users, many research projects dealing with optimization aim to make their work compatibles with many solvers as possible. Besides the solver specific interfaces like Gurobipy ([Gurobipy, n.d.](#)) and others, there is therefore the general need for a solver-independent interface allowing the user to formulate optimization problems and solve it with the solver of choice.

Fulfilling this need, Algebraic modeling languages (AML) allow to formulate large scale, complex problems using a high-level syntax similar to the mathematical notation. Well established AMLs such as GAMS([Corporation, n.d.](#)) and AMPL Fourer et al. (2003) support a wide range of solvers, but are license-restricted and rely on closed-source code. In contrast, AMLs as JuMP([Dunning et al., 2017](#)), Pyomo ([Hart et al., 2017](#)), GEKKO ([Beal et al., 2018](#)) and PuLP ([Pulp, 2022](#)) are open-source and have gained increasing attention throughout the recent years. While the Julia package JuMP is characterized by high-performance in-memory communication with the solvers, the Python packages Pyomo, GEKKO and PuLP lack parallelized, low-level operations and communicate slower with the solver via intermediate files written to disk. Further, the Python AMLs do not make use of state-of-the-art data handling packages. In particular, the assignment of coordinates or indexes is often not supported or memory extensive due to use of an object-oriented implementation where every single combination of coordinates is stored separately.

linopy tackles these issues together. By introducing an array-based data model for variables and constraints, it makes mathematical programming compatible with Python's advanced data handling packages Numpy([Harris et al., 2020](#)), Pandas ([Reback et al., 2022](#)) and Xarray ([Hoyer & Hamman, 2017](#)) while significantly increasing speed and flexibility.

Basic Structure

The core data classes `Variable`, `LinearExpression` and `Constraint` are subclasses of `xarray`'s `DataArray` and `Dataset` class. These contain n-dimensional arrays with unique labels referencing the optimization variables and coefficients. Hence, variables and constraints are defined together with a set of dimensions and their corresponding coordinates. For example, creating a variable $x(d_1, d_2)$ defined on $d_1 \in \{1, \dots, N\}$ and $d_2 \in \{1, \dots, M\}$, would only require passing d_1 and d_2 to the variable initialization, with optional lower and upper bounds $l_x(d_1, d_2)$ and $u_x(d_1, d_2)$ being defined on (a subset of) $\{d_1, d_2\}$. The returned object is an $N \times M$ array of integer labels referencing to the optimization variables used by the solver. ...

Benchmark

Benchmarking against two well-established open source AMLs Gurobi and Pyomo, `linopy` outperforms Pyomo in speed and memory-efficiency and outperform JuMP in memory-efficiency.

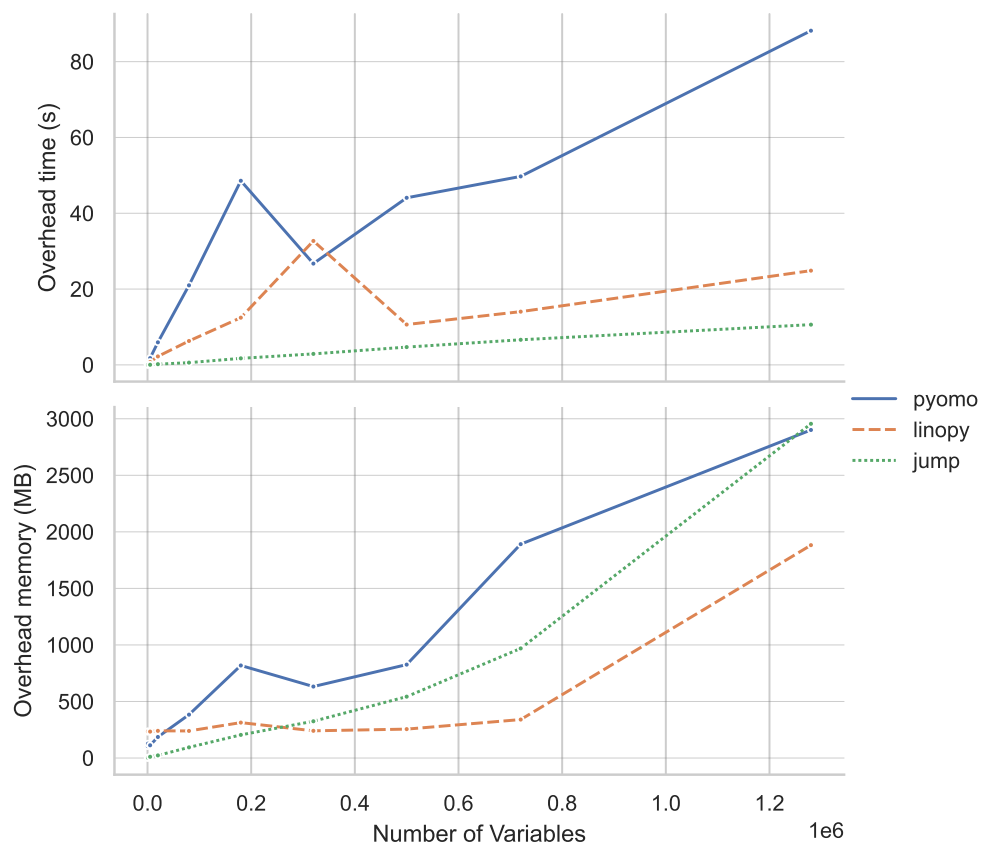


Figure 1: Benchmark of `linopy` against Gurobi and Pyomo. The figure show the net overhead that the AMLs used for the communication to the solver.

Related Research

`linopy` is used by several research projects and groups. The [PyPSA package](#) (Brown et al., 2018) is an open-source software for (large scale) energy system modelling and optimization.

Not yet: Together with the [PyPSA-Eur workflow](#) (Hörsch et al., 2018) and the sector-coupled

57 extension [PyPSA-Eur-sec](#) (Brown et al., 2018), it uses `linopy` to solve large linear problems
58 with up to 10^8 variables.

59 Availability

60 Stable versions of the `linopy` package are available for Linux, MacOS and Windows via `pip` in
61 the [Python Package Index \(PyPI\)](#). Development branches are available in the project's [GitHub](#)
62 [repository](#) together with a documentation on [Read the Docs](#). The `linopy` package is released
63 under [GPLv3](#) and welcomes contributions via the project's [GitHub repository](#).

64 Acknowledgements

65 We thank all [contributors](#) who helped to develop `linopy`. Fabian Hofmann is funded by the
66 BreakthroughEnergy initiative.

67 References

- 68 Beal, L., Hill, D., Martin, R., & Hedengren, J. (2018). GEKKO Optimization Suite. *Processes*,
69 6(8), 106. <https://doi.org/10.3390/pr6080106>
- 70 Brown, T., Hörsch, J., & Schlachtberger, D. (2018). PyPSA: Python for Power System
71 Analysis. *Journal of Open Research Software*, 6, 4. <https://doi.org/10.5334/jors.188>
- 72 Corporation, G. D. (n.d.). *GAMS - Cutting Edge Modeling*. <https://www.gams.com/>.
- 73 Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A Modeling Language for Mathematical
74 Optimization. *SIAM Review*, 59(2), 295–320. <https://doi.org/10.1137/15M1020575>
- 75 FICO® Xpress Solver. (n.d.). In *FICO*. <https://www.fico.com/en/products/fico-xpress-solver>.
- 76 Fourer, R., Gay, D. M., & Kernighan, B. W. (2003). *AMPL: A modeling language for*
77 *mathematical programming* (2nd ed). Thomson/Brooks/Cole. ISBN: 978-0-534-38809-6
- 78 *GLPK - GNU Project - Free Software Foundation (FSF)*. (n.d.). <https://www.gnu.org/software/glpk/>.
- 79 Gurobi - The Fastest Solver. (n.d.). In *Gurobi*. <https://www.gurobi.com/>.
- 81 *Gurobipy: Python interface to Gurobi*. (n.d.).
- 82 Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau,
83 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van
84 Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ...
85 Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362.
86 <https://doi.org/10.1038/s41586-020-2649-2>
- 87 Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L.,
88 & Sirola, J. D. (2017). *Pyomo — Optimization Modeling in Python* (Vol. 67). Springer
89 International Publishing. <https://doi.org/10.1007/978-3-319-58821-6>
- 90 Hörsch, J., Hofmann, F., Schlachtberger, D., & Brown, T. (2018). PyPSA-Eur: An open
91 optimisation model of the European transmission system. *Energy Strategy Reviews*, 22,
92 207–215. <https://doi.org/10.1016/j.esr.2018.08.012>
- 93 Hoyer, S., & Hamman, J. J. (2017). Xarray: N-D labeled Arrays and Datasets in Python.
94 *Journal of Open Research Software*, 5, 10. <https://doi.org/10.5334/jors.148>
- 95 Inc., A. O. (n.d.). *AMPL - Stramlined modeling for real optimization*. In *AMPL*.
96 <https://ampl.com/>.

- 97 *Pulp*. (2022). COIN-OR Foundation.
- 98 Reback, J., Jbrockmendel, McKinney, W., Van Den Bossche, J., Roeschke, M., Augspurger,
99 T., Hawkins, S., Cloud, P., Gfyoung, Sinhrks, Hoefler, P., Klein, A., Petersen, T., Tratner,
100 J., She, C., Ayd, W., Naveh, S., Darbyshire, J., Shadrach, R., ... Li, T. (2022). *Pandas-*
101 *dev/pandas: Pandas 1.4.3*. Zenodo. <https://doi.org/10.5281/ZENODO.3509134>

DRAFT