




Linopy: Linear optimization with n-dimensional labeled variables

Fabian Hofmann ¹

¹ Department of Digital Transformation in Energy Systems, Technical University of Berlin

DOI: [10.21105/joss.04823](https://doi.org/10.21105/joss.04823)

Software

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

Editor: Fei Tao 

Reviewers:

- [@torressa](#)
- [@g4brielvs](#)

Submitted: 23 September 2022

Published: 22 April 2023

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

Linopy is an open-source package written in Python to build and process 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, fast 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 and lays the foundation for features such as fast writing to array-oriented scientific data formats, masking, automatic solving on remote servers and model scaling.

Statement of need

Decades after its inception ([Dantzig, 1963](#)), mathematical optimization is nowadays of immense importance for business, industry and governmental decision-making. Optimization is used to address various sorts of complex problems, such as challenges related to climate change, energy transitions, and food supply. Typically, an optimization problem, i.e. a mathematical program, consists of one objective function to be numerically minimized and a set of constraints that restrict the underlying variables to external conditions. Algebraic Modeling Languages (AML) aim at facilitating mathematical programming by allowing the user to formulate large scale, complex problems with a high-level syntax similar to the mathematical notation. The formulated problem is then passed to the solver of choice where a solution is calculated. AMLs provide the most user-friendly interface possible to various solvers, each with its own set of features.

Well established AMLs such as GAMS ([Bussieck & Meeraus, 2004](#)) and AMPL ([Fourer et al., 1990](#)) 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](#)), CVXPY ([Diamond & Boyd, 2016](#)), Pyomo ([Hart et al., 2017](#)), GEKKO ([Beal et al., 2018](#)) and PuLP ([Mitchel et al., 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. An exception is CVXPY, which supports fast array-based operations and uses low-level wrappers to the solvers. However, it is common among Python AMLs not to 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 is an open-source Python package representing a new kind of AML that tackles these issues together. By introducing an array-based data model for variables and constraints, Linopy 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](#)).

The approach follows the idea that a variable $x(d_1, d_2, \dots, d_K)$ may be defined on an arbitrary number of $K \geq 0$ dimensions, each dimension spanning over a set of N_i discrete coordinates of arbitrary data type (integer, string, date-time, etc.), i.e. $d_i \in \{c_{i,1}, \dots, c_{i,N}\}$. The variable $x(d_1, d_2, \dots, d_K)$ is then stored as an array of shape $N_1 \times N_2 \times \dots \times N_K$ containing integer labels referencing the optimization variables used by the solver. Coordinates are automatically aligned when variables are used in linear expressions or when applying built-in functions, such as summing over specific dimension or grouping by user-defined labels. Note that if a variable should not be defined on the full set of coordinates given by $\{d_1, d_2, \dots, d_K\}$, a boolean mask of the same shape may be used to select where the variable is defined and where not.

The array-based modelling approach does not only lead to more flexibility but also increases the overall performance. The following figure shows the [benchmark](#) against the AMLs JuMP, Pyomo, PuLP and CVXPY as well as the solver specific interface Gurobipy. The included AMLs packages are all open source and well-established and therefore suitable for comparison. Linopy outperforms all Python AMLs in memory efficiency and is close to CVXPY and Gurobipy in terms of speed while being faster than JuMP.

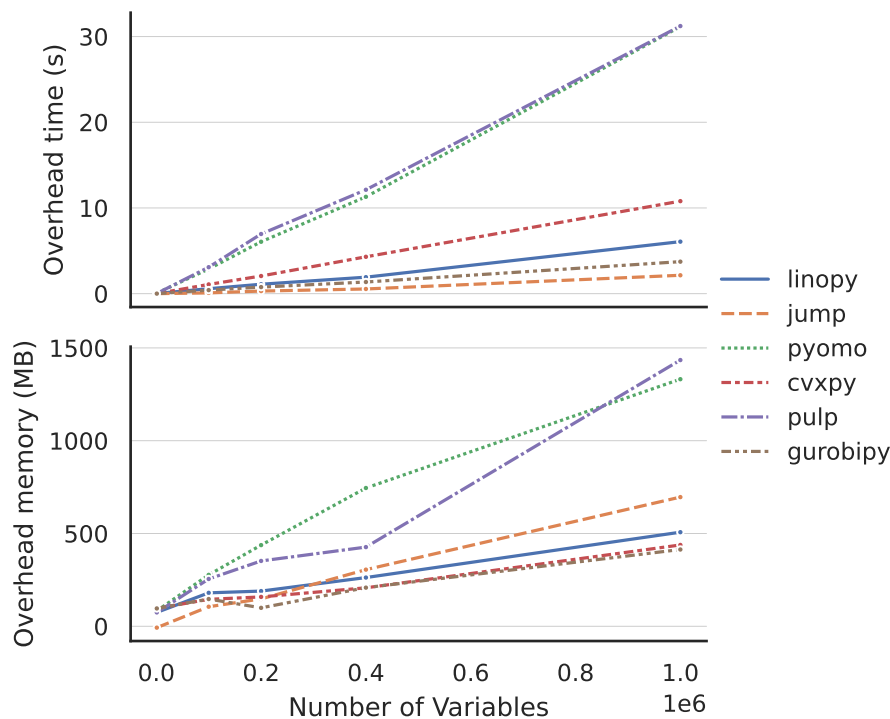


Figure 1: Benchmark of Linopy against comparable packages. The producing Snakemake workflow is available [here](#). The software and hardware specifications are detailed [here](#). The benchmark is based on a 1-dimensional knapsack problem and uses the Gurobi solver. The overhead is calculated from the difference of the whole solving process via the AML and the solving process on the solver side alone. Note that the benchmark is hardly dependent on the complexity of the problem. Thus, adding more terms to the constraints, setting different kind of index labels or changing it to a purely linear problem does hardly have an effect on the overhead.

Due to a strong alignment to the Xarray package, Linopy supports storing the optimization model as a NetCDF file ([Rew & Davis, 1990](#)), which allows users to quickly share optimization problems with others. Using the [Paramiko package](#), Linopy offers the user to send unsolved

problems to a server and retrieve the solution after running the optimization remotely, which is particularly helpful if large computing resources are needed.

Linopy supports a list of well-established solvers, namely

- GLPK ([GLPK - GNU Project - Free Software Foundation \(FSF\)](#), n.d.)
- CBC ([Forrest et al., 2022](#))
- HiGHS ([Huangfu & Hall, 2018](#))
- Gurobi (“[Gurobi - The Fastest Solver](#),” n.d.)
- Xpress (“[FICO® Xpress Solver](#),” n.d.)
- CPLEX ([Cplex, 2009](#))

while other solvers such as PIPS-IPM++ ([Rehfeldt et al., 2022](#)), the SCIP solver ([Bestuzheva et al., 2021](#)) and MOSEK ([ApS, 2019](#)) are planned to be integrated in future versions. Further, upcoming features target model coefficient scaling as for example presented and performed in ([Morshed & Noor-E-Alam, 2020](#)) and ([Göke, 2021](#)) as well as the integration of non-linear expressions.

Related Research

Linopy is used by several research projects and groups, mostly related to energy system modelling. The energy system modelling tool [PyPSA package](#) ([Brown et al., 2018](#)), which is used by [various institutions](#) and builds the core of the [PyPSA-Eur workflow](#) ([Hörsch et al., 2018](#)), ([Brown et al., 2018](#)), uses Linopy as the primary optimization interface. The Fraunhofer Institute for Energy Economics and Energy System Technology is using Linopy in order to create an interface to GPU-based solvers. The German Aerospace Center uses Linopy for calculating stochastic optimization problems. Finally, a TU Berlin and Google Inc. cooperate on a [research project](#) that uses Linopy to analyze system-level impacts of 24/7 carbon-free electricity procurement in Europe.

Availability

Stable versions of the Linopy package are available for Linux, MacOS and Windows via `pip` in the [Python Package Index \(PyPI\)](#). Development branches are available in the project’s [GitHub repository](#) together with a documentation on [Read the Docs](#). For continuous integration, Linopy uses automated tests on Github together with [Pre-Commit hooks](#). The Linopy package is released under [GPLv3](#) and welcomes contributions via the project’s [GitHub repository](#).

Acknowledgements

We thank all [contributors](#) who helped to develop Linopy, in particular Jonas Hörsch who contributed important changes to the architecture of the package. Linopy was inspired by the [Nomopyomo](#) prototype written by Tom Brown and its approach to store variables as integer labels.

Fabian Hofmann is funded by the [BreakthroughEnergy initiative](#).

References

- ApS, M. (2019). *MOSEK Optimizer API for Python 9.3.22*.
- Beal, L., Hill, D., Martin, R., & Hedengren, J. (2018). GEKKO Optimization Suite. *Processes*, 6(8), 106. <https://doi.org/10.3390/pr6080106>

- Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., Doornmalen, J. van, Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., Hulst, R. van der, Koch, T., Lübbecke, M. E., Maher, S. J., ... Witzig, J. (2021). *The SCIP Optimization Suite 8.0 – Optimization Online*.
- Brown, T., Hörsch, J., & Schlachtberger, D. (2018). PyPSA: Python for Power System Analysis. *Journal of Open Research Software*, 6, 4. <https://doi.org/10.5334/jors.188>
- Bussieck, M. R., & Meeraus, A. (2004). General Algebraic Modeling System (GAMS). In P. M. Pardalos, D. W. Hearn, & J. Kallrath (Eds.), *Modeling Languages in Mathematical Optimization* (Vol. 88, pp. 137–157). Springer US. https://doi.org/10.1007/978-1-4613-0215-5_8
- Cplex, I. I. (2009). V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53), 157.
- Dantzig, G. (1963). *Linear Programming and Extensions*. RAND Corporation. <https://doi.org/10.7249/R366>
- Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83), 1–5.
- Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, 59(2), 295–320. <https://doi.org/10.1137/15M1020575>
- FICO® Xpress Solver. (n.d.). In *FICO*. <https://www.fico.com/en/products/fico-xpress-solver>.
- Forrest, J., Ralphs, T., Santos, H. G., Vigerske, S., Forrest, J., Hafer, L., Kristjansson, B., Jpfasano, EdwinStraver, Lubin, M., Rlougee, Jgoncal1, Jan-Willem, H-l-Gassmann, Brito, S., Cristina, Saltzman, M., Tostost, Pitrus, B., ... To-St. (2022). *Coin-or/Cbc: Release releases/2.10.8*. Zenodo. <https://doi.org/10.5281/ZENODO.6522795>
- Fourer, R., Gay, D. M., & Kernighan, B. W. (1990). A Modeling Language for Mathematical Programming. *Management Science*, 36(5), 519–554. <https://doi.org/10.1287/mnsc.36.5.519>
- GLPK - GNU Project - Free Software Foundation (FSF). (n.d.). <https://www.gnu.org/software/glpk/>.
- Göke, L. (2021). A graph-based formulation for modeling macro-energy systems. *Applied Energy*, 301, 117377. <https://doi.org/10.1016/j.apenergy.2021.117377>
- Gurobi - The Fastest Solver. (n.d.). In *Gurobi*. <https://www.gurobi.com/>.
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., & Siirola, J. D. (2017). *Pyomo — Optimization Modeling in Python* (Vol. 67). Springer International Publishing. <https://doi.org/10.1007/978-3-319-58821-6>
- Hörsch, J., Hofmann, F., Schlachtberger, D., & Brown, T. (2018). PyPSA-Eur: An open optimisation model of the European transmission system. *Energy Strategy Reviews*, 22, 207–215. <https://doi.org/10.1016/j.esr.2018.08.012>
- Hoyer, S., & Hamman, J. J. (2017). Xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, 5, 10. <https://doi.org/10.5334/jors.148>
- Huangfu, Q., & Hall, J. A. J. (2018). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1), 119–142. <https://doi.org/10.1007/s12532-017-0130-5>

- Mitchel, S., Kean, A., Mason, A., O'Sullivan, M., Phillips, A., & Peschiera, F. (2022). *Pulp*. COIN-OR Foundation.
- Morshed, M. S., & Noor-E-Alam, Md. (2020). Generalized affine scaling algorithms for linear programming problems. *Computers & Operations Research*, *114*, 104807. <https://doi.org/10.1016/j.cor.2019.104807>
- Reback, J., Jbrockmendel, McKinney, W., Van Den Bossche, J., Roeschke, M., Augspurger, T., Hawkins, S., Cloud, P., Gfyoung, Sinhrks, Hoefler, P., Klein, A., Petersen, T., Tratner, J., She, C., Ayd, W., Naveh, S., Darbyshire, J., Shadrach, R., ... Li, T. (2022). *Pandas-dev/pandas: Pandas 1.4.3*. Zenodo. <https://doi.org/10.5281/ZENODO.3509134>
- Rehfeldt, D., Hobbie, H., Schönheit, D., Koch, T., Möst, D., & Gleixner, A. (2022). A massively parallel interior-point solver for LPs with generalized arrowhead structure, and applications to energy system models. *European Journal of Operational Research*, *296*(1), 60–71. <https://doi.org/10.1016/j.ejor.2021.06.063>
- Rew, R., & Davis, G. (1990). NetCDF: An interface for scientific data access. *IEEE Computer Graphics and Applications*, *10*(4), 76–82. <https://doi.org/10.1109/38.56302>