The Journal of Open Source Software

Canopy: Institutional-Grade Hierarchical Portfolio Optimization in Python

Rakesh Bag [1]

1 Anagatam Technologies, India

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 21 February 2026
Published: unpublished

Summary

Canopy is an open-source Python library for hierarchical portfolio optimization that implements three allocation algorithms: Hierarchical Risk Parity (HRP) (Prado, 2016), Hierarchical Equal Risk Contribution (HERC) (Raffinot, 2017), and Nested Cluster Optimization (NCO) (Prado, 2019). The library provides a unified facade interface with institutional-grade covariance estimation, configurable risk measures, walk-forward backtesting, and JSON-serializable audit trails for regulatory compliance. Canopy is designed for quantitative analysts, portfolio managers, and academic researchers who require stable, reproducible portfolio construction without the instabilities inherent in classical mean-variance optimization (Markowitz, 1952).

Statement of Need

Classical mean-variance portfolio optimization requires inverting the covariance matrix, which becomes numerically unstable as the number of assets grows or when the estimation window is short relative to the asset universe (Prado, 2016). Hierarchical methods address this limitation by leveraging the correlation structure of asset returns through hierarchical clustering, allocating capital through the resulting dendrogram without matrix inversion.

While existing libraries such as PyPortfolioOpt (Martin, 2021) provide comprehensive mean-variance tooling, no Python package offers a unified, production-ready implementation of all three major hierarchical allocation algorithms (HRP, HERC, NCO) with institutional features such as covariance denoising, multiple risk measures, weight constraints, and compliance audit trails.

Canopy fills this gap by providing:

- **Three hierarchical algorithms** in a single `MasterCanopy` facade class that can be configured and executed in one line of code.
- **Four covariance estimators** — Sample, Ledoit-Wolf shrinkage (Ledoit & Wolf, 2004), Marchenko-Pastur denoising (Marchenko & Pastur, 1967), and Exponentially Weighted Moving Average (EWMA) — with optional detoning (Prado, 2020) to remove the market mode before clustering.
- **Four risk measures** for HERC inter-cluster allocation — Variance, Conditional Value-at-Risk (CVaR) (Rockafellar & Uryasev, 2000), Conditional Drawdown-at-Risk (CDaR), and Mean Absolute Deviation (MAD).
- **Walk-forward backtesting** with configurable rebalance frequency and lookback windows.
- **Full audit trails** with JSON-serializable computation logs for MiFID II, SEC Rule 15c3-5, and Basel III/IV regulatory compliance.

Canopy is actively used in quantitative research and portfolio management workflows, and is available on PyPI (`pip install canopy-optimizer`).

Algorithms

Hierarchical Risk Parity (HRP)

The HRP algorithm (Prado, 2016) applies agglomerative hierarchical clustering to the distance matrix derived from the correlation matrix of asset returns. After constructing the dendrogram, it applies optimal leaf ordering (Bar-Joseph et al., 2001) and then uses recursive bisection to allocate weights based on inverse-variance risk parity. Unlike Markowitz optimization, HRP does not require inverting the covariance matrix, making it robust to estimation error and applicable to singular or near-singular covariance matrices.

Hierarchical Equal Risk Contribution (HERC)

HERC (Raffinot, 2017) extends HRP with a two-stage allocation process. First, it determines the optimal number of clusters using the gap statistic or a user-specified maximum. Then, it allocates capital between clusters using inter-cluster risk parity (with a user-selected risk measure: Variance, CVaR, CDaR, or MAD), and within each cluster using inverse-variance weighting. This produces cluster-aware diversification that respects the hierarchical structure of the asset universe.

Nested Cluster Optimization (NCO)

NCO (Prado, 2019) addresses the instability of mean-variance optimization by applying Tikhonov-regularized optimization within each cluster: $\mathbf{w}_k = (\Sigma_k + \lambda I)^{-1} \cdot \mathbf{1}$, where λ is a regularization parameter. The per-cluster optimal weights are then combined using inter-cluster inverse-variance allocation. This nested approach reduces the effective dimensionality of each optimization subproblem, yielding portfolios with lower tail risk compared to full-universe optimization.

Architecture

Canopy follows a facade design pattern through its `MasterCanopy` class, which orchestrates five internal engines:

- `CovarianceEngine` — computes and conditions the covariance matrix using the selected estimator, with eigenvalue analysis and condition number diagnostics.
- `ClusterEngine` — performs agglomerative hierarchical clustering with seven linkage methods (Ward, single, complete, average, weighted, centroid, median) and optimal leaf ordering.
- `HRP`, `HERC`, `NCO` — the three optimizer implementations, each producing normalized weight vectors with optional min/max weight constraints.
- `ChartEngine` — generates nine dark-theme Plotly visualizations including dendrograms, correlation heatmaps, allocation charts, and risk decomposition plots.
- `DataLoader` — ingests data from Yahoo Finance, CSV, Parquet, or in-memory DataFrames, with automatic log-return computation and benchmark alignment.

Every computation step is recorded in a timestamped audit trail (`AuditEntry` objects) that can be exported to JSON for compliance archival.

Availability and Dependencies

Canopy is available on PyPI and GitHub under the Apache License 2.0. It requires Python 3.10+ and depends on NumPy (Harris et al., 2020), pandas (McKinney, 2010), SciPy (Virtanen & others, 2020), and scikit-learn (Pedregosa & others, 2011). Documentation is hosted on ReadTheDocs.

References

Bar-Joseph, Z., Gifford, D. K., & Jaakkola, T. S. (2001). Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, *17*(suppl_1), S22–S29. https://doi.org/10.1093/bioinformatics/17.suppl_1.S22

Harris, C. R., Millman, K. J., Walt, S. J. van der, & others. (2020). Array programming with NumPy. In *Nature* (Vol. 585, pp. 357–362). https://doi.org/10.1038/s41586-020-2649-2

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411. https://doi.org/10.1016/S0047-259X(03)00096-4

Marchenko, V. A., & Pastur, L. A. (1967). Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, *1*(4), 457–483. https://doi.org/10.1070/SM1967v001n04ABEH001994

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77–91. https://doi.org/10.2307/2975974

Martin, R. A. (2021). PyPortfolioOpt: Portfolio optimization in Python. *Journal of Open Source Software*, *6*(61), 3066. https://doi.org/10.21105/joss.03066

McKinney, W. (2010). *Data structures for statistical computing in Python* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Pedregosa, F., & others. (2011). Scikit-learn: Machine learning in Python. In *Journal of Machine Learning Research* (Vol. 12, pp. 2825–2830).

Prado, M. L. de. (2016). Building diversified portfolios that outperform out-of-sample. *The Journal of Portfolio Management*, *42*(4), 59–69. https://doi.org/10.3905/jpm.2016.42.4.059

Prado, M. L. de. (2019). A robust estimator of the efficient frontier. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3469961

Prado, M. L. de. (2020). *Machine learning for asset managers*. https://doi.org/10.1017/9781108883658

Raffinot, T. (2017). Hierarchical clustering-based asset allocation. *The Journal of Portfolio Management*, *44*(2), 89–99. https://doi.org/10.3905/jpm.2018.44.2.089

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, *2*(3), 21–41. https://doi.org/10.21314/JOR.2000.038

Virtanen, P., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. In *Nature Methods* (Vol. 17, pp. 261–272). https://doi.org/10.1038/s41592-019-0686-2