Scikit-Topt: A Python Library for Algorithm Development in Topology Optimization

Kohei Watanabe [1]

1 JTEKT Corporation, Kariya, Japan

Summary

Topology optimization is a computational method for optimizing the shape and material layout of structures and components, typically using the finite element method (FEM). It automatically generates optimal structural configurations by adjusting the distribution of material to achieve target performance—for example, minimizing compliance under a volume constraint. Specifically, the method optimizes an objective function by adjusting design variables that represent the presence or absence of material in each region.

This technique can be applied to a wide range of physical simulations, including structural analysis, heat conduction, fluid dynamics, and electromagnetic field analysis, making it a powerful tool for optimal design.

Scikit-Topt is a Python library that implements topology optimization algorithms. It provides tools for visualizing parameter transitions and optimization results, thereby facilitating algorithm tuning and comparative analysis of optimization strategies. The library is tested via GitHub Actions and includes documentation generated with Sphinx. Installation and usage instructions are available in the GitHub repository.

Statement of Need

There is a growing demand for accessible and flexible tools for topology optimization that support unstructured meshes and can be easily integrated into Python-based workflows. While many existing libraries focus on specific problems with rigid architectures or heavy dependencies, Scikit-Topt offers a lightweight, modular, and fully Pythonic framework suitable for both academic research and industrial applications.

In topology optimization, the evolution of material density fields and the scheduling of algorithmic parameters (e.g., via continuation methods) play a crucial role in achieving high-quality results. However, most existing libraries provide limited or no support for managing and visualizing such transitions, making it difficult to interpret the optimization process.

Scikit-Topt addresses this gap by enabling users to track and visualize the progression of density distributions and parameter schedules throughout the optimization process. This feature is particularly valuable for algorithm development, comparative studies, and educational purposes.

Purpose and Prior Art

There are numerous open-source projects for topology optimization. Examples include FEniTop (Jia et al., 2024), TopOpt.jl (Tarek, 2019), Topology optimization using PETSc (Smit et al., 2021), DL4TO(Erzmann et al., 2023), and pytopo3d (Kim & Kang, 2025). However, many

---

projects are specialized implementations targeted at specific problems and are not well-suited for solving general topology optimization tasks. Common limitations include:

- Parameters and problem settings are hard-coded, limiting flexibility.
- They rely on grid- or voxel-based finite element analysis, making it difficult to handle arbitrary geometries.
- They are not packaged properly or depend on outdated software, making installation and usage difficult.
- They are not designed as standard experimental platforms, making it hard to compare algorithms across different parameters and case studies.

Scikit-Topt avoids these issues by not including compiled code and depending only on SciPy (Virtanen et al., 2020), a widely used and stable Python library, making packaging and installation straightforward. For finite element matrix assembly, it uses scikit-fem (Gustafsson & McBain, 2020), which also contains no compiled components.

Furthermore, Scikit-Topt visualizes parameter transitions and material density evolution through Matplotlib (Hunter, 2007) and PyVista (Sullivan & Kaszynski, 2019).

Scikit-Topt implements topology optimization algorithms based on the density method, and currently supports the following features:

Physics & Analysis

- Structural analysis and heat conduction on unstructured 2D/3D meshes

- Finite element solvers powered by:

    - SciPy (Virtanen et al., 2020)
    - PyAMG (Bell et al., 2023)
    - scikit-fem (Gustafsson & McBain, 2020)

Density-Based Topology Optimization

- SIMP and RAMP material interpolation
- Heaviside projection for binarization acceleration
- Helmholtz filtering for density smoothing
- Sensitivity analysis via adjoint-based backpropagation through SIMP, the Helmholtz filter, and the Heaviside projection
- Optimization under multiple load cases
- Continuation strategies for scheduling parameters (e.g., β in Heaviside, penalization (p))

Optimization Algorithms

- Optimality Criteria Method (OC)
- Modified Optimality Criteria (MOC)

Usage Example

The following example demonstrates how to use Scikit-Topt to perform topology optimization on an unstructured mesh using the Optimality Criteria (OC) method. First, a mesh is loaded using `scikit-fem`, and a `LinearElasticity` object is defined with material properties and boundary conditions. Next, a configuration object for the optimizer is created, and the optimization process is launched. In this example, a `LinearElasticity` is created from a mesh file using `scikit-fem`, which defines the FEM model, material properties, and boundary conditions. Then, `OC_Config` sets the optimization parameters such as volume fraction and iteration limits. Finally, the optimizer is executed.

```
import skfem, sktopt

mesh_path = "./data/model.msh"
basis = sktopt.mesh.loader.basis_from_file(mesh_path, intorder=1)

task = sktopt.mesh.task.LinearElasticity.from_facets(
    basis, dirichlet_facets...
)
cfg = sktopt.core.OC_Config(
    dst_path="./result",
    vol_frac=vol_frac=sktopt.tools.SchedulerConfig(
      target_value=0.6, scheduler_type="Step"
    ),
    max_iter=100...
)

optimizer = sktopt.core.OC_Optimizer(cfg, task)

optimizer.parameterize()
optimizer.optimize()
```

During the optimization process, intermediate density fields are automatically visualized and saved using PyVista and Matplotlib. This enables users to monitor the convergence behavior and spatial evolution of the material distribution. An example is shown below. The left figure shows the boundary conditions, while the right shows the optimized density distribution:
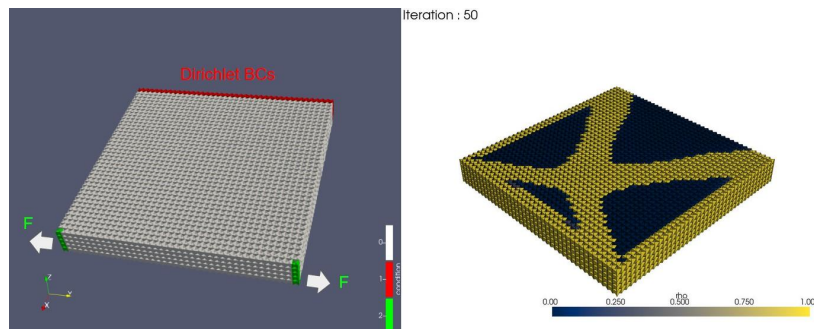

Figure 1: Topology optimization setup under multiple load cases (left) and resulting optimized density distribution (right)

Watanabe. (2025). Scikit-Topt: A Python Library for Algorithm Development in Topology Optimization. *Journal of Open Source Software*, *10*(116), 9092. https://doi.org/10.21105/joss.09092.
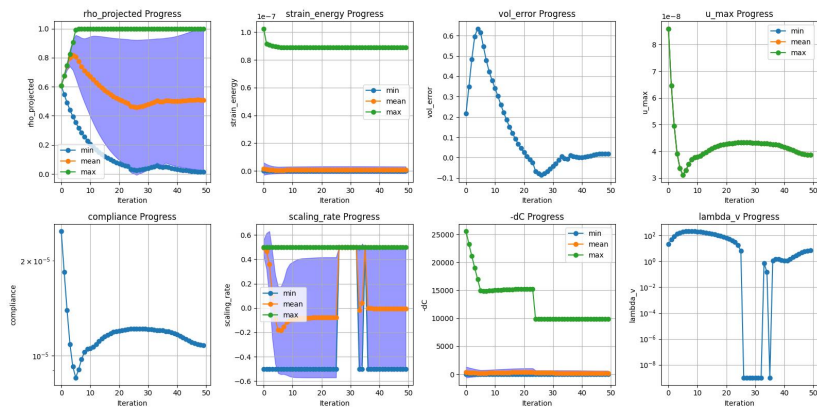
Figure 2: Parameter progression during optimization

As an extensible and open framework, Scikit-Topt aims to facilitate reproducible experimentation and accelerate the development of new topology optimization strategies.

Acknowledgements

References

Bell, N., Olson, L. N., Schroder, J., & Southworth, B. (2023). PyAMG: Algebraic Multigrid Solvers in Python. *Journal of Open Source Software*, *8*(87), 5495. https://doi.org/10.21105/joss.05495

Erzmann, D., Dittmer, S., Harms, H., & Maaß, P. (2023). DL4TO : A Deep Learning Library for Sample-Efficient Topology Optimization. In *Geometric science of information* (pp. 543–551). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-38271-0_54

Gustafsson, T., & McBain, G. D. (2020). *kinnala/scikit-fem: Simple finite element assemblers for Python* (Version 1.1.0). Zenodo. https://doi.org/10.5281/zenodo.3862391

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science &Amp; Engineering*, *9*(3), 90–95. https://doi.org/10.1109/mcse.2007.55

Jia, Y., Wang, C., & Zhang, X. S. (2024). FEniTop: A simple FEniCSx implementation for 2D and 3D topology optimization supporting parallel computing. *Structural and Multidisciplinary Optimization*, *67*(8). https://doi.org/10.1007/s00158-024-03818-7

Kim, J., & Kang, N. (2025). PyTopo3D: A Python Framework for 3D SIMP-based Topology Optimization. *arXiv Preprint arXiv:2504.05604*.

Smit, T., Aage, N., Ferguson, S. J., & Helgason, B. (2021). Topology optimization using PETSc: A Python wrapper and extended functionality. *Structural and Multidisciplinary Optimization*, *64*(6), 4343–4353. https://doi.org/10.1007/s00158-021-03018-7

Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. https://doi.org/10.21105/joss.01450

Tarek, M. (2019). TopOpt.jl: An efficient and high-performance package for topology optimization of continuum structures in the Julia programming language. *Proceedings of the 13th World Congress of Structural and Multidisciplinary Optimization*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2