The Journal of Open Source Software

OPTIMEO: Bayesian Optimization Web App for Process Tuning, Modeling, and Orchestration

Colin Bousige [1]

1 Universite Claude Bernard Lyon 1, CNRS, LMI UMR 5615, Villeurbanne, F-69100, France

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @sgbaird
- @VasanthRajendran

Submitted: 04 June 2025
Published: unpublished

Summary

We present `OPTIMEO`, a Bayesian optimization web app for process tuning, modeling, and orchestration. `OPTIMEO` is a web application that helps the user optimize their experimental process by generating a design of experiment (DoE), generating new experiments using Bayesian optimization, and analyzing/predicting the results of their experiments using machine learning (ML) models. It can also be used as a Python package for more advanced usage, but it is not its main purpose. The web application is designed to be user-friendly and accessible to researchers and students alike, providing a powerful tool for optimizing experimental processes in various research fields. The Bayesian optimization part is based on the `ax-platform` package (Bakshy et al., 2018), which in turn is based on `BoTorch` (Balandat et al., 2020). It therefore allows for numerical and categorical variables, as well as multi-objective optimization. To use the web application, no knowledge of python is required: the user can run it on the streamlit.io web page, https://optimeo.streamlit.app/, or run it locally on their machine for better performance. The user can simply (1) select the desired options to generate a DoE; or (2) upload their data and perform Bayesian optimization to look for optimal parameters to minimize/maximize their experimental response; or (3) upload their data and plot it, assess possible correlations between variables, as well as run various regression models using `scikit-learn` (Pedregosa et al., 2011) to make predictions and analyze the importance of each feature on their experimental process. Extensive yet accessible descriptions of the different options are provided in the web application to help the user understand what they are doing. Using the package in python is also possible and it offers more versatility, like the possibility to make an optimization loop (in case experiments and their characterizations are done by a robot, for example) or to provide more parameters to the ML models – the heavy lifting is done under the hood by the package (e.g. categorical variables encoding and decoding, data formatting, normalizations, workflows, etc.).

Statement of need

Experimental processes are often complex and time-consuming, requiring careful planning and execution to achieve reliable results. In many cases, researchers must conduct multiple experiments to optimize their processes, which can be both costly and time-consuming. The traditional approach to experimental design often relies on trial and error, resulting in inefficient use of resources and time. In recent years, there has been a growing interest in using advanced computational techniques to optimize experimental processes using active learning techniques such as genetic algorithms or Bayesian optimization (Guo et al., 2023; Shields et al., 2021). For example, `Sycofinder` (Moosavi et al., 2019; Talirz & SeyedMohamadMoosavi, 2019), an application with web-based User Interface (UI) was developed to optimize the synthesis of metal-organic frameworks (MOFs) using a genetic algorithm, and Bayesian optimization was used to find optimal synthesis parameters – the optimization target can be either the yield and quality of the synthesized material as well as its physical properties or performances (Lambard

et al., 2022; Matsuda et al., 2022; Osada et al., 2020). While these techniques have now proven their effectiveness in various fields, they often require a deep understanding of the underlying algorithms and programming skills to be implemented effectively. This can be a barrier for many researchers, especially those without a strong background in computer science or data analysis – which is often the case for experimentalists. To address these challenges, we present `OPTIMEO`, a powerful tool that streamlines the experimental process by providing a comprehensive platform for design of experiment, Bayesian optimization, and machine learning analysis.

State of the Field

Bayesian optimization is known for its data efficiency, meaning it can find optimal solutions with fewer evaluations compared to other methods. However, it can become computationally expensive as the number of features and evaluations increases, leading to longer computation times (Lan et al., 2022). Genetic algorithms, on the other hand, are generally faster in terms of computation time per evaluation but may require more evaluations to converge to an optimal solution (Lan et al., 2022).

However, the trade-off between data efficiency, computation time, and experimentation time is a key consideration when choosing between these two optimization methods. The `OPTIMEO` package is aimed at helping scientists of any field to reach the optimum parameters of their process using the minimum amount of ressources and effort. As a result, `OPTIMEO` relies on Bayesian optimization for its superior data efficiency. When each experiment can take a day or more to complete and analyze (or costs a lot of money), minimizing the total number of experiments is crucial. Bayesian optimization is preferred over genetic algorithms in this context, as it typically requires fewer experiments to reach optimal parameters – even if the algorithm itself takes a few extra minutes to suggest the next experiments.

There are several free and open source software that provide similar functionality to `OPTIMEO`, such as `AutoOED` (Tian et al., 2021), `BOXVIA` (Ishii et al., 2022), and `MADGUI` (Bajan & Lambard, 2025). All three rely on Bayesian optimization to minimize the number of experimental evaluations, and are available via executables or source code. However, none of them provide the ability to build an experimental design, which, from interacting with our experimentalist colleagues, is a feature they often need and highly appreciate – it gives a good idea of where to start an optimization process from scratch. Also, while we agree that it's probably very suggestive, we feel that they lack the usability and accessibility that we aim to provide with our UI. For example, the upper and lower bounds of the features need to be entered manually in these packages – which is a tedious task. These values are automatically set (while editable) in `OPTIMEO` based on the data provided by the user. Finally, these programs are not available as standalone Python packages for more advanced use. Such advanced usage might be required if the user wants to run the optimization loop in a robotic high-throughput lab, for example – which is why this package was developed in the first place.

All in all, while other software is available to perform some of the tasks of `OPTIMEO`, this package offers a unique concatenation of features that makes it a powerful and versatile tool for optimizing experimental processes.

Acknowledgements

This work was supported by the French National Research Agency (N° ANR-24-CE08-7639). It would also not have been possible without the development of the open source packages `ax-platform` (Bakshy et al., 2018), `BoTorch` (Balandat et al., 2020), `scikit-learn` (Pedregosa et al., 2011), `pyDOE3` (tisimst et al., 2024), `dexpy` (*Statease/Dexpy*, 2025), `doepy` (dirkaudaz, 2018), and `definitive-screening-design` (Ongari, 2024).

References

Bajan, C., & Lambard, G. (2025). MADGUI: Multi-Application Design Graphical User Interface for active learning assisted by Bayesian optimization. *Chemometrics and Intelligent Laboratory Systems*, *258*, 105323. https://doi.org/10.1016/j.chemolab.2025.105323

Bakshy, E., Dworkin, L., Karrer, B., Kashin, K., Letham, B., Murthy, A., & Singh, S. (2018). AE: A domain-agnostic platform for adaptive experimentation. *Workshop Syst. ML Open Source Softw. NeurIPS*.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). *BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization* (No. arXiv:1910.06403). arXiv. https://doi.org/10.48550/arXiv.1910.06403

dirkaudaz. (2018). *Dirkaudaz/DOEPy*.

Guo, J., Ranković, B., & Schwaller, P. (2023). Bayesian Optimization for Chemical Reactions. *CHIMIA*, *77*(1-2), 31–38. https://doi.org/10.2533/chimia.2023.31

Ishii, A., Kamijyo, R., Yamanaka, A., & Yamamoto, A. (2022). BOXVIA: Bayesian optimization executable and visualizable application. *SoftwareX*, *18*, 101019. https://doi.org/10.1016/j.softx.2022.101019

Lambard, G., Sasaki, T. T., Sodeyama, K., Ohkubo, T., & Hono, K. (2022). Optimization of direct extrusion process for Nd-Fe-B magnets using active learning assisted by machine learning and Bayesian optimization. *Scripta Materialia*, *209*, 114341. https://doi.org/10.1016/j.scriptamat.2021.114341

Lan, G., Tomczak, J. M., Roijers, D. M., & Eiben, A. E. (2022). Time efficiency in optimization with a bayesian-Evolutionary algorithm. *Swarm and Evolutionary Computation*, *69*, 100970. https://doi.org/10.1016/j.swevo.2021.100970

Matsuda, S., Lambard, G., & Sodeyama, K. (2022). Data-driven automated robotic experiments accelerate discovery of multi-component electrolyte for rechargeable Li–O2 batteries. *CR-PHYS-SC*, *3*(4). https://doi.org/10.1016/j.xcrp.2022.100832

Moosavi, S. M., Chidambaram, A., Talirz, L., Haranczyk, M., Stylianou, K. C., & Smit, B. (2019). Capturing chemical intuition in synthesis of metal-organic frameworks. *Nat Commun*, *10*(1), 539. https://doi.org/10.1038/s41467-019-08483-9

Ongari, D. (2024). *Danieleongari/definitive_screening_design*.

Osada, K., Kutsukake, K., Yamamoto, J., Yamashita, S., Kodera, T., Nagai, Y., Horikawa, T., Matsui, K., Takeuchi, I., & Ujihara, T. (2020). Adaptive Bayesian optimization for epitaxial growth of Si thin films under various constraints. *Materials Today Communications*, *25*, 101538. https://doi.org/10.1016/j.mtcomm.2020.101538

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, *12*(85), 2825–2830.

Shields, B. J., Stevens, J., Li, J., Parasram, M., Damani, F., Alvarado, J. I. M., Janey, J. M., Adams, R. P., & Doyle, A. G. (2021). Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, *590*(7844), 89–96. https://doi.org/10.1038/s41586-021-03213-y

Statease/dexpy. (2025). Stat-Ease, Inc.

Talirz, L., & SeyedMohamadMoosavi. (2019). *Ltalirz/sycofinder: Release v0.2.0*. Zenodo. https://doi.org/10.5281/zenodo.2554380

Tian, Y., Luković, M. K., Erps, T., Foshey, M., & Matusik, W. (2021). *AutoOED: Automated*

Optimal Experiment Design Platform (No. arXiv:2104.05959). arXiv. https://doi.org/10.48550/arXiv.2104.05959

tisimst, Sjögren, R., Lafage, R., mwang, Vidner, O., Antoine-Averland, Valle, E., Andrei, Imoto, H., Schueller, J., Dietrich, J., Becker, M. R., Todd, & Fehlner, A. (2024). *Relf/pyDOE3: 1.0.4*. Zenodo. https://doi.org/10.5281/zenodo.13340611