The Journal of Open Source Software

qPOTS: A Python Package for Sample-efficient Batch Constrained Multiobjective Bayesian Optimization

Ashwin Renganathan [1,2], **Peter E. Bachman**[1], **and Kade E. Carlson**[1]

1 Department of Aerospace Engineering, The Pennsylvania State University, United States **2** Institute for Computational and Data Sciences, The Pennsylvania State University, United States

DOI: 10.21105/joss.10867

Software

- Review
- Repository
- Archive

Editor: Fabian Scheipl

Reviewers:

- @jbussemaker
- @gialmisi

Submitted: 14 June 2026
Published: 22 September 2026

Summary

Designing an engineered system involves optimizing the system for some performance objective under several design constraints. In practice, though, there is typically more than one objective to meet and the objectives may conflict with each other. For instance, an aircraft designer may need to reduce the aerodynamic drag on the aircraft while ensuring passenger safety. Aerodynamic efficiency typically leads to slender aircraft cross-sections which can, however, compromise the safety of the aircraft under gusty conditions. Similar tradeoffs occur in chemical process optimization, materials discovery, and machine learning hyperparameter optimization. Optimizing simultaneously for more than one objective is called multiobjective optimization, and the solution to such problems is no longer a single point but a set of points that offer a pragmatic compromise between the objectives.

`qPOTS` is a Python package that solves constrained multiobjective optimization when there is no derivative information and every query of the objectives and constraints is expensive, thus necessitating "sample efficiency" (i.e., solving the problem with as few evaluations of objectives and constraints as possible). The package implements batch Pareto optimal Thompson sampling (qPOTS), a Bayesian optimization (BO) method (Frazier, 2018; Garnett, 2023; Renganathan & Carlson, 2025). The method uses Thompson sampling (Thompson, 1933), and the prefix q denotes the number of design points selected in each batch. The package also includes `qPOTS-Decoupled`, which supports decoupled objective and constraint evaluations, that is, when objectives and constraints can be evaluated independently by computer models using separate resources. In this setting, the algorithm improves sample efficiency by evaluating only a subset of objectives and constraints during each design iteration (Renganathan et al., 2026). The cited manuscript calls this method qPOTS-DOE, where DOE stands for decoupled oracle evaluations.

`qPOTS` is released under the GNU General Public License v3.0 and is built on widely used scientific Python libraries including `PyTorch` (Paszke et al., 2019), `GPyTorch` (Gardner et al., 2018), `BoTorch` (Balandat et al., 2020), and `Pymoo` (Blank & Deb, 2020).

Statement of need

The solution to a multiobjective optimization problem is called the "Pareto set", and its image in the objective space is called the "Pareto frontier". Computing the Pareto set/frontier is nontrivial. When derivative information is available and the objectives and constraints are smooth and not expensive to evaluate, one may use gradient-based optimization techniques over several scalarized versions of the objectives (Fliege et al., 2009; Fliege & Vaz, 2016). However, in real-world problems, high-quality derivatives are rarely available, objective/constraint smoothness is not guaranteed, and the Pareto frontier may have a complex topology (disconnected, nonconvex, etc.) In such situations, multiobjective Bayesian optimization (MOBO) serves as a

suitable alternative. MOBO learns global surrogate models for the objectives and constraints, and leverages Bayesian decision theory to generate a sequence of design points (iterates) that balance exploration and exploitation of the design space. This involves constructing an "acquisition" function that quantifies the utility of a candidate iterate, and solving an "inner" optimization problem to determine the iterates. This inner optimization depends only on the surrogate model and is independent of the expensive objectives and constraints.

MOBO is an active area of research; there are several existing works that innovate on the Bayesian decision-theoretic acquisition function (Belakaria et al., 2019; Daulton et al., 2020; Tu et al., 2022), or use scalarization to adapt multiobjective problems to single-objective BO (Knowles, 2006), or modify existing approaches to promote diversity in the iterates (Konakovic Lukovic et al., 2020), among others. However, existing methods have one or more of the following limitations: (1) they require nontrivial inner acquisition function optimization due to nonconvexity and/or stochasticity, which ultimately affects the accuracy of the predicted Pareto set/frontier; (2) they do not support batch sampling (generating a batch of iterates); or (3) they cannot handle nonlinear constraints. See Renganathan & Carlson (2025) for further details. `qPOTS` was introduced to address all of these limitations, with improved sample efficiency and solution accuracy being our overarching goal.

Evolutionary algorithms (Deb et al., 2002) are quite popular for derivative-free multiobjective optimization. However, they rely on generating large populations of candidate solutions that undergo repeated mutation and crossover. Although this leads to accurate solutions, it can penalize sample efficiency. `qPOTS` combines the sample efficiency of BO with the accuracy of evolutionary algorithms (Renganathan & Carlson, 2025). By fitting Gaussian process (GP) surrogates (Rasmussen & Williams, 2006) for the objectives and constraints, `qPOTS` replaces the conventional acquisition function optimization in MOBO with a cheap inner multiobjective optimization over the GP posterior sample paths, leveraging evolutionary algorithms. Nonlinear constraints are handled as part of the Gaussian process Thompson sampling (Renganathan & Carlson, 2025).

Thompson sampling approaches similar to `qPOTS` already exist for MOBO, including `TS-EMO` (Bradford et al., 2018) and `TS-TCH` (Paria et al., 2020). The main difference between `qPOTS` and `TS-EMO` is that `TS-EMO` generates GP posterior Thompson samples, but still has to compute the maximum hypervolume improvement over them to choose the candidate — this inner optimization problem can be difficult to solve (Renganathan & Carlson, 2025). Additionally, `TS-EMO` is unable to handle constraints. `TS-TCH` scalarizes the Thompson samples to leverage single-objective acquisition functions. This inherits the limitations of scalarized approaches in computing disconnected and nonconvex Pareto frontiers.

A further opportunity arises when the objectives and constraints can be evaluated asynchronously using independent compute resources. For example, in aerodynamic design, objectives may be aircraft performance at different operating conditions that may be evaluated separately. In such situations, if the objectives and constraints are correlated with each other, MOBO can proceed with only a subset of objective and constraint evaluations at each iterate. The `qPOTS` package includes `qPOTS-Decoupled` (Renganathan et al., 2026) to address this, which combines the `qPOTS` posterior sampling workflow with a multitask Gaussian process (MTGP) (Bonilla et al., 2007) that jointly learns all objectives and constraints. MTGPs learn the correlations across the objectives and constraints, which can be exploited to determine which iterates warrant a decoupled evaluation. Decoupled MOBO methods such as PESMOC and HVKG can already progress with partial information (Daulton et al., 2023; Garrido-Merchán & Hernández-Lobato, 2019). However, existing methods depend on differences in evaluation cost across the objectives and constraints to determine decoupling. In contrast, `qPOTS-Decoupled` can decouple evaluations under uniform evaluation costs; differences in evaluation cost will further improve the sample efficiency of `qPOTS-Decoupled`.

`qPOTS` builds on `GPyTorch` (Gardner et al., 2018) and existing GP and MTGP wrappers in `BoTorch` (Balandat et al., 2020) for surrogate modeling, `Pymoo` (Blank & Deb, 2020) for

evolutionary algorithms, and fully supports GPU acceleration. Even though qPOTS builds on these existing libraries, there is a compelling case for a standalone package for the following reasons. First, qPOTS is anchored on Pareto optimal Thompson sampling — this makes it compatible with probabilistic models beyond GPs such as Bayesian neural networks and variational autoencoders. Future versions of the package could therefore support these alternative surrogate models. Second, generative models may be integrated into the framework to learn the distribution of the Pareto set in Thompson sampling. Third, `qPOTS-Decoupled` could be applied to other problems such as optimal experiment design and sensor placement, where optimal selection of a subset from a correlated set of tasks is of interest. Therefore, a standalone package will promote wider usage, contributions, and extensions that might not be possible when integrated into another package.

The target audience is researchers and engineers who need a sample-efficient derivative-free multiobjective optimization software package in Python. qPOTS is designed for method developers benchmarking against standard acquisition strategies, and for application researchers optimizing expensive custom black-box functions under limited evaluation budgets.

State of the field

While multiobjective optimization problems can be solved via a variety of approaches including gradient-based approaches and evolutionary algorithms, we only review libraries relevant to MOBO here. `BoTorch` provides a PyTorch-based platform for Bayesian optimization (including MOBO) and already implements MOBO acquisition functions such as qNEHVI, qNParEGO, and their logarithmic variants (Ament et al., 2023; Balandat et al., 2020; Daulton et al., 2020), entropy-search acquisitions (PESMOC, MESMO, and JESMO) (Belakaria et al., 2019; Garrido-Merchán & Hernández-Lobato, 2019; Tu et al., 2022), and hypervolume knowledge gradient (Daulton et al., 2023). `Trieste` provides a modular TensorFlow-based Bayesian-optimization loop with multiobjective and constrained workflows (Picheny et al., 2023). `TS-EMO` (Bradford et al., 2018) is another MOBO algorithm based on Thompson sampling (similar to qPOTS) and implemented in MATLAB. qPOTS differs methodologically from these libraries, as explained in the previous section and in more detail in Renganathan & Carlson (2025), and has demonstrated better empirical performance in the reported comparisons (Renganathan & Carlson, 2025).

Software design

qPOTS coordinates the modeling, search, and evaluation steps of its optimization algorithms. Each iteration must keep design points in physical coordinates consistent with the normalized tensors used by the surrogate models and the candidate populations used by `Pymoo`. In decoupled runs, it must also track which objectives and constraints are evaluated at each point. The `QPOTSRunner` class manages this sequence: fit the surrogate model, search its posterior sample paths, select a batch of points and outputs to evaluate, evaluate them, and update the observations.

The runner exposes model and acquisition objects and accepts callbacks that are invoked during optimization. These interfaces allow users to compare surrogate models, change candidate selection policies, and connect external simulators without copying the optimization loop. This flexibility introduces additional layers of software and more interfaces for users to understand. Default settings, configuration validation, results with explicit types, and a fixed sequence of optimization steps help manage that complexity. Users can call `run()` to execute an optimization run or `step()` to advance one iteration under the control of an external system.

Coupled and decoupled modes can both use MTGPs and share the same representation of the optimization state. Task identifiers record which outputs are requested, and unevaluated outputs are represented as missing observations. This avoids maintaining separate implementations,

but requires stricter validation and a surrogate model that can handle missing data. The package therefore coordinates more than a `BoTorch` acquisition function: it manages posterior path construction, Pareto set search, batch and output selection, and updates from partial observations across its dependencies.

`BoTorch` and `GPyTorch` provide models, posterior sampling, and baseline acquisition functions that operate on tensors (Balandat et al., 2020; Gardner et al., 2018). Reimplementing these components would add numerical code to maintain without contributing functionality specific to `qPOTS`. `Pymoo` supports bounded optimization problems with vectorized evaluations, seeded execution, nondominated solution sets, and callbacks, making it suitable for the inner multiobjective optimization (Blank & Deb, 2020). Alternatives include jMetalPy, pagmo/pygmo, and DEAP (Benítez-Hidalgo et al., 2019; Biscani & Izzo, 2020; Fortin et al., 2012). Using `Pymoo` adds a dependency and requires conversion between its CPU-based NumPy arrays and the PyTorch tensors used by the surrogate models. The package keeps these conversions in an adapter so that replacing the optimizer would require fewer changes elsewhere. We did not use a file-based workflow system such as Snakemake (Köster & Rahmann, 2012) because each new design, and sometimes the choice of outputs to evaluate, depends on a newly fitted posterior model held in memory. `QPOTSRunner` does not provide file-based checkpointing, scheduling, or provenance tracking; external systems can provide these capabilities by calling `step()` and using callbacks.

Research impact statement

`qPOTS` is the reference implementation for the AISTATS 2025 paper introducing Pareto optimal Thompson sampling for batch multiobjective Bayesian optimization (Renganathan & Carlson, 2025). That paper evaluates the method on synthetic and real-world benchmarks against analytical, entropy-search, Thompson-sampling, and random baselines. The software has also supported an aerodynamic design study of the NASA Common Research Model with open reproducibility materials (Carlson & Renganathan, 2026). These studies demonstrate research applications of the method and the code on which this package is based.

Independent evaluation is beginning to emerge: SPREAD includes `qPOTS` in a numerical comparison of MOBO methods (Hotegni & Peitz, 2025). Other publications cite qPOTS while discussing generative molecular design, sustainable process systems, class-imbalance learning, and many-objective optimization (Jiang et al., 2026; Kudva et al., 2026, Ch. 3; Muthyala et al., 2026; Wang et al., 2025). We treat those citations as evidence both of awareness of `qPOTS` and of its use as a state-of-the-art benchmark in the MOBO community.

AI usage disclosure

Anthropic Claude assisted with code refactoring and README structure before submission. OpenAI Codex assisted during review revisions with drafting tests, examples, and documentation, and with manuscript language and citation corrections. Generative AI was not used to design the `qPOTS` or `qPOTS-Decoupled` algorithms or to generate benchmark results. The authors reviewed every change, reran the software tests and documented examples, checked citations against publication records, and remain responsible for the manuscript and software.

Acknowledgements

The authors acknowledge the Penn State Institute for Computational and Data Sciences for access to computational research infrastructure through the Roar Core Facility.

References

Ament, S., Daulton, S., Eriksson, D., Balandat, M., & Bakshy, E. (2023). Unexpected improvements to expected improvement for Bayesian optimization. *Advances in Neural Information Processing Systems*, *36*, 20577–20612. https://doi.org/10.52202/075280-0904

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, *33*, 21524–21538. https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html

Belakaria, S., Deshwal, A., & Doppa, J. R. (2019). Max-value entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, *32*. https://proceedings.neurips.cc/paper_files/paper/2019/hash/82edc5c9e21035674d481640448049f3-Abstract.html

Benítez-Hidalgo, A., Nebro, A. J., García-Nieto, J., Oregi, I., & Del Ser, J. (2019). jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, *51*, 100598. https://doi.org/10.1016/j.swevo.2019.100598

Biscani, F., & Izzo, D. (2020). A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, *5*(53), 2338. https://doi.org/10.21105/joss.02338

Blank, J., & Deb, K. (2020). pymoo: Multi-objective optimization in Python. *IEEE Access*, *8*, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567

Bonilla, E. V., Chai, K. M. A., & Williams, C. K. I. (2007). Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems*, *20*. https://proceedings.neurips.cc/paper/2007/hash/66368270ffd51418ec58bd793f2d9b1b-Abstract.html

Bradford, E., Schweidtmann, A. M., & Lapkin, A. (2018). Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm. *Journal of Global Optimization*, *71*(2), 407–438. https://doi.org/10.1007/s10898-018-0609-2

Carlson, K., & Renganathan, A. (2026). Multiobjective aerodynamic design optimization of the NASA Common Research Model. *Aerospace Science and Technology*, *168*, 111120. https://doi.org/10.1016/j.ast.2025.111120

Daulton, S., Balandat, M., & Bakshy, E. (2020). Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, *33*, 9851–9864. https://proceedings.neurips.cc/paper/2020/hash/6fec24eac8f18ed793f5eaad3dd7977c-Abstract.html

Daulton, S., Balandat, M., & Bakshy, E. (2023). Hypervolume knowledge gradient: A lookahead approach for multi-objective Bayesian optimization with partial information. *Proceedings of the 40th International Conference on Machine Learning*, *202*, 7167–7204. https://proceedings.mlr.press/v202/daulton23a.html

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Fliege, J., Graña Drummond, L. M., & Svaiter, B. F. (2009). Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, *20*(2), 602–626. https://doi.org/10.1137/08071692X

Fliege, J., & Vaz, A. I. F. (2016). A method for constrained multiobjective optimization based on SQP techniques. *SIAM Journal on Optimization*, *26*(4), 2091–2119. https://doi.org/10.1137/15M1016424

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175. https://jmlr.org/papers/v13/fortin12a.html

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv Preprint arXiv:1807.02811*. https://doi.org/10.48550/arXiv.1807.02811

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., & Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, *31*, 7576–7586. https://proceedings.neurips.cc/paper_files/paper/2018/hash/27e8e17134dd7083b050476733207ea1-Abstract.html

Garnett, R. (2023). *Bayesian optimization*. Cambridge University Press. https://doi.org/10.1017/9781108348973

Garrido-Merchán, E. C., & Hernández-Lobato, D. (2019). Predictive entropy search for multi-objective Bayesian optimization with constraints. *Neurocomputing*, *361*, 50–68. https://doi.org/10.1016/j.neucom.2019.06.025

Hotegni, S. S., & Peitz, S. (2025). SPREAD: Sampling-based Pareto front refinement via efficient adaptive diffusion. *arXiv Preprint arXiv:2509.21058*. https://doi.org/10.48550/arXiv.2509.21058

Jiang, C., Huang, J., & Li, M. (2026). Do we really need to approach the entire Pareto front in many-objective Bayesian optimisation? *arXiv Preprint arXiv:2604.09417*. https://doi.org/10.48550/arXiv.2604.09417

Knowles, J. (2006). ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, *10*(1), 50–66. https://doi.org/10.1109/TEVC.2005.851274

Konakovic Lukovic, M., Tian, Y., & Matusik, W. (2020). Diversity-guided multi-objective Bayesian optimization with batch evaluations. *Advances in Neural Information Processing Systems*, *33*, 17708–17720. https://proceedings.neurips.cc/paper/2020/hash/cd3109c63bf4323e6b987a5923becb96-Abstract.html

Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. https://doi.org/10.1093/bioinformatics/bts480

Kudva, A., Tang, W.-T., & Paulson, J. A. (2026). Multi-objective Bayesian optimization for networked black-box systems: A path to greener profits and smarter designs. In C. Li (Ed.), *Optimization of sustainable process systems: Multiscale models and uncertainties* (pp. 63–89). John Wiley & Sons. https://doi.org/10.1002/9781394205646.ch3

Muthyala, M. R., Sorourifar, F., Tan, T., Peng, Y., & Paulson, J. A. (2026). Generative multiobjective Bayesian optimization with scalable batch evaluations for sample-efficient de novo molecular design. *Industrial & Engineering Chemistry Research*, *65*(1), 628–642. https://doi.org/10.1021/acs.iecr.5c03166

Paria, B., Kandasamy, K., & Póczos, B. (2020). A flexible framework for multi-objective Bayesian optimization using random scalarizations. *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, *115*, 766–776. https://proceedings.mlr.press/v115/paria20a.html

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024–8035. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Picheny, V., Berkeley, J., Moss, H. B., Stojic, H., Granta, U., Ober, S. W., Artemev, A., Ghani,

K., Goodall, A., Paleyes, A., Vakili, S., Pascual-Diaz, S., Markou, S., Qing, J., Loka, N. R. B. S., & Couckuyt, I. (2023). Trieste: Efficiently exploring the depths of black-box functions with TensorFlow. *arXiv Preprint arXiv:2302.08436*. https://doi.org/10.48550/arXiv.2302.08436

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press. ISBN: 978-0-262-18253-9

Renganathan, A., & Carlson, K. (2025). qPOTS: Efficient batch multiobjective Bayesian optimization via Pareto optimal Thompson sampling. In Y. Li, S. Mandt, S. Agrawal, & E. Khan (Eds.), *Proceedings of the 28th international conference on artificial intelligence and statistics* (Vol. 258, pp. 4051–4059). PMLR. https://proceedings.mlr.press/v258/renganathan25a.html

Renganathan, A., Carlson, K., & Bachman, P. E. (2026). *qPOTS-DOE: Batch Pareto optimal Thompson sampling with decoupled oracle evaluations for constrained multiobjective Bayesian optimization*. Under review.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3-4), 285–294. https://doi.org/10.1093/biomet/25.3-4.285

Tu, B., Gandy, A., Kantas, N., & Shafei, B. (2022). Joint entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, *35*, 9922–9938. https://doi.org/10.52202/068431-0721

Wang, Z., Wang, S., Ernst, D., & Xiao, C. (2025). Automated class imbalance learning via few-shot multi-objective Bayesian optimization with deep kernel Gaussian processes. *IEEE Access*, *13*, 131839–131855. https://doi.org/10.1109/access.2025.3591034