ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL164

XOPT: A SIMPLIFIED FRAMEWORK FOR OPTIMIZATION OF ACCELERATOR PROBLEMS USING ADVANCED ALGORITHMS *

R. Roussel†, C. Mayes, A. Edelen, SLAC National Accelerator Laboratory, Menlo Park, CA, USA
A. Bartnik, Cornell University, Ithaca, NY, USA

Abstract

The recent development of advanced black box optimization algorithms has promised order of magnitude improvements in optimization speed when solving accelerator physics problems. However in practice, these algorithms remain inaccessible to the general accelerator community, due to the expertise and infrastructure required to apply them towards solving optimization problems. In this work, we introduce the Python package, Xopt, which implements a simple interface for connecting arbitrarily specified optimization problems with advanced optimization algorithms. Users specify optimization problems and algorithms with a minimal Python script, allowing flexible interfacing with both experimental online control systems and simulated design problems, while minimizing the need for algorithmic expertise or software development. We describe case-studies where cutting-edge Bayesian optimization and genetic algorithms implemented in Xopt are used to solve online control and accelerator design problems. The same algorithms are also used to solve simulated optimization problems in high performance computing clusters using the same interface.

INTRODUCTION

The need to solve complex optimization problems is widely prevalent in the field of accelerator physics. For example, accelerator control parameters must be tuned during accelerator operations to improve performance (so-called "online" tuning) or accelerator design parameters must be optimized in simulation to implement novel operational modes ("offline tuning"). These problems can be solved using a wide variety of conventional optimization algorithms, including Nelder-Mead Simplex [1] and Robust Conjugate Direction Search [2]. In addition, evolutionary algorithms, such as NSGA-II [3], can leverage the power of parallelized computation to solve multi-objective optimization problems. More recently, advanced machine learning based algorithms have been used to solve both online and offline accelerator optimization problems, most notably Bayesian Optimization (BO) [4, 5].

Current attempts of applying optimization methods towards solving accelerator physics problems has been done in an ad hoc way, with individual teams re-implementing the same optimization algorithms for specific use cases. This needlessly "reinvents the wheel", wasting significant amounts of time and expertise, makes it difficult for non-experts to use advanced algorithms to solve their specific optimization problems, and prevents the use of meaningful standard benchmarks to compare new algorithms against. To address these issues, we have developed a Python-based, modular framework called `Xopt` [6] that provides a uniform framework for implementing and applying optimization algorithms to arbitrary problems in experiment or simulation. Furthermore, `Xopt` aims to enable user-friendly, "off-the-shelf" application of advanced algorithms towards solving accelerator physics optimization problems, while allowing easy customization for specific problems by advanced users.

XOPT OVERVIEW

`Xopt` breaks down optimization problems into three modular components contained in the main `Xopt` class, as shown in Fig. 1. The `VOCS` class defines the optimization space, including the variables, objectives, constraints and constants (statics). The `Evaluator` class defines how to evaluate objectives and constraints given points passed to it using serial or parallel (multithreading, MPI etc.) processes. Finally, the `Generator` class implements the optimization algorithm, and is used to generate points in variable space to be evaluated. The main `Xopt` class choreographs the execution and communication between these three modules in order to perform an optimization cycle with the `step()` command.

This modular, object-oriented approach enables easy modification and customization of optimization routines for specific use cases. For example, the same optimization algorithm can be applied to both simulation and experiment, or shared between different accelerator facilities by swapping out the `Evaluator` object. On the other hand, generators can also be swapped out to compare the performance of different algorithms on the same optimization problem. These objects can also be sub-classed to customize evaluation or generation of points to solve specific problems.

Optimization algorithms are defined by `Generator` objects, which are used to generate future points to be evaluated by calling their `generate()` method. While users are free to

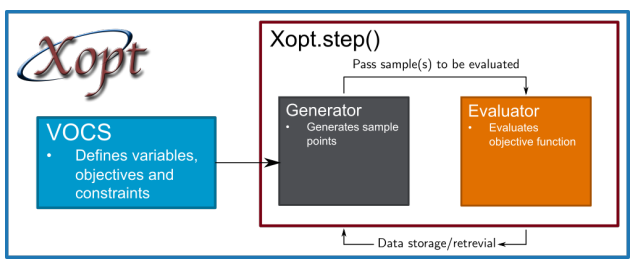

Figure 1: Normal control flow structure of Xopt.

* This work was supported by the U.S. Department of Energy, under DOE Contract No. DE-AC02-76SF00515 and the Office of Science, Office of Basic Energy Sciences.
† rroussel@slac.stanford.edu

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL164

implement their own optimization algorithms, Xopt comes pre-packaged with a number of conventional and advanced optimization algorithms tuned by experts to be applicable to a wide variety of optimization problems "off-the-shelf". Currently these algorithms include

- Autonomous Characterization
  - Bayesian Exploration [7]
- Single Objective Optimization
  - Nelder-Mead Simplex [1]
  - Robust Conjugate Direction Search [2]
  - Extremum Seeking [8]
  - Upper Confidence Bound BO [9]
  - Expected Improvement BO [10]
  - Trust Region BO [11]
  - Multi-fidelity BO [12]
- Multi-Objective Optimization
  - Continuous NSGA-II [13]
  - Expected Hypervolume Improvement BO [14]
  - Multi-Generation Expected Hypervolume Improvement BO [15]
  - Muli-Fidelity Expected Hypervolume Improvement [12]

These generators can be used or modified via Python code inside or outside of Xopt.

Xopt communicates with experimental or computational systems by a single, user-provided evaluate function passed to the Evaluator class. An evaluate function is required to receive a dictionary corresponding to points that should be evaluated and returns a dictionary of evaluated outputs. This greatly simplifies sharing the same algorithms between arbitrary optimization problems, as this Python function handles specific code required to interface with specific accelerator control systems or simulation codes. An example of interfacing with a simple EPICS control system is shown in Fig. 2. One additional benefit of this evaluation method is the ability to track inputs and outputs that do not participate in optimization, such as additional simulation outputs or evaluation time.

Finally, Xopt can be configured and run through a text based interface. This is useful for running large scale computing workflows at high performance computing clusters, or interacting with control system user interfaces. In addition, after every evaluation Xopt can optionally dump the data and configuration of VOCS, the Generator and the Evaluator objects to a human readable text file. The text interface of Xopt allows these dump files to be used to restart a run after every step. Finally, this text interface allows users or software programs access to running Xopt with little to no coding required.

```python
from epics import caget, caput, cainfo
import time

outputs = ["XRMS","YRMS"]
def make_epics_measurement(input_dict):
  # set inputs
  for name, val in input_dict.items():
    caput(name, val)

  # wait for inputs to settle
  time.sleep(1)

  # get output values, current time
  output_dict = caget_many(outputs)
  output_dict["time"] = time.time()

  # compute geometeric avg of beamsizes
  output_dict["RMS"] = (
    output_dict["XRMS"]*\
    output_dict["YRMS"]
  )**0.5

  return output_dict
```

Figure 2: Example evaluate function for setting accelerator parameters and making measurements using an EPICS interface.

EXAMPLE: CHARACTERIZING BEAM EMITTANCE AT THE ARGONNE WAKEFIELD ACCELERATOR

The Argonne Wakefield Accelerator [16] operates an electron photoinjector that contains several elements which can effect the transverse beam emittance, including a set of solenoids and focusing quadrupoles. In order to characterize the effect of these parameters on the emittance, AWA uses a single-shot, multi-slit emittance diagnostic [17]. While this diagnostic provides detailed information about the transverse phase space it has a limited dynamic range. Beamlets on the downstream measurement screen must not overlap or extend beyond the confines of the screen boundary. As a result, the multi-slit diagnostic can only provide accurate emittance measurements for beams with a narrow range of transverse sizes and divergences. Properly characterizing beam emittances in this case requires the use of additional quadrupole focusing elements to satisfy these measurement constraints.

We used Xopt to run a constrained characterization algorithm named Bayesian Exploration [7] to autonomously characterize beam emittances as a function of photoinjector solenoids. A simple Python function was used to allow Xopt to control current settings of two solenoids and two quadrupoles in the AWA beamline using pyEPICS [18]. This was combined with a custom interface with AWA's camera systems to do measurements of the beam distribution (number of beamlets, beamlet size, beamlet centroid, etc.) in order to calculate the beam emittance. We specified minimum and maximum magnet currents and measurement

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL164

constraints (minimum number of beamlets, maximum beamlet size, no beamlet overlaps) using the `VOCS` object. We then used the "off-the-shelf" Bayesian Exploration generator class to define characterization parameters (maximum travel distance in input space, number of emittance measurements to take for each set of input parameters, etc.). Finally, we ran a set number of characterization steps by calling the `step()` method in a simple for loop. This process explored the input parameter space autonomously (as shown in Fig. 3), avoiding parameter sub-spaces that led to invalid emittance measurements while prioritizing scanning over variables that have the most effect on beam emittance. Configuring, executing and analyzing results from the characterization run took place in a single Jupyter notebook using a web browser interface. The same algorithm has been used repeatedly at other facilities (FACET-II) with different interfaces to characterize beam dynamics in experiment and simulation.

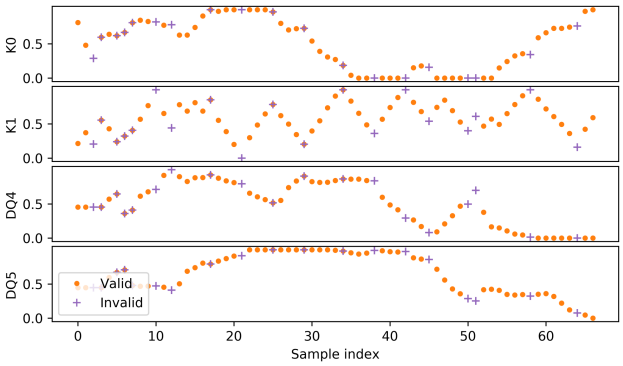

Figure 3: Evaluation trace in input space during Bayesian Exploration characterization of emittance as a function of photoinjector solenoids (K0, K1) and drive linac quadrupoles (DQ4, DQ5). Valid and invalid measurements are due to limited multi-slit emittance diagnostic dynamic range. Reproduced from Ref. [7].

EXAMPLE: MULTI-OBJECTIVE GENETIC OPTIMIZATION OF THE CORNELL PHOTOINJECTOR

This Cornell injector, originally designed for generation of 100 pC-scale bunches for high repetition rate light source applications [19], presents a unique test bed for finding the ultimate bunch length and emittance limits in an MeV ultrafast electron diffraction apparatus [20]. Our goal was to explore both the stroboscopic limit (single electron per bunch) and the single shot limit ($\sim 10^5$ electrons per bunch), and in both cases to find machine settings that simultaneously minimize three things: the width of the beam (allowing for smaller samples), the bunch length and shot-to-shot jitter (for best temporal resolution), and emittance (for best momentum resolution).

As this is a multiobjective optimization problem, we used the NSGA-II algorithm within `Xopt`, which was easily combined with the simulation code GPT (General Particle

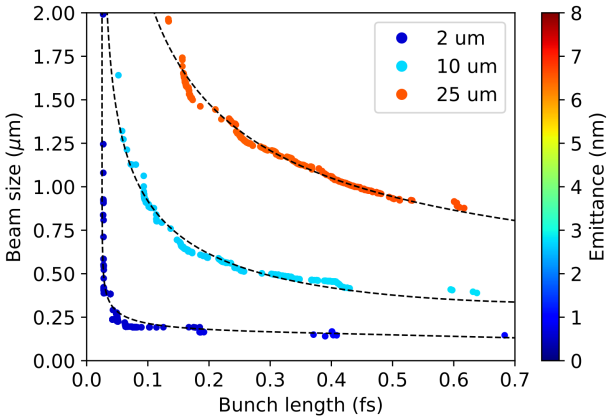

Figure 4: Dependence of achievable bunch length on final beam size in the absence of space charge forces for three different initial laser sizes: 2, 10, and 25 μm (dots), compared to a simple model (lines). Reproduced from Ref. [20].

Tracer) [21], in order to optimize this system. All code was written in Python, and the interface between GPT and Python is publicly available [22]. The code was run on a computing cluster at Cornell University, with each evaluation of GPT occupying a single core, and each member of a population within the genetic algorithm run simultaneously, which scales well with available computing resources. Pareto fronts found by `Xopt` are shown in Fig. 4 and are further discussed in Ref. [20]. For previous research projects, we have used a C-based implementation of NSGA-II. However, using and debugging multi-objective optimization in `Xopt` is much easier than in previous implementations. As a result, `Xopt` has completely supplanted the C-based NSGA-II code at Cornell. Furthermore, `Xopt` allows for future experimentation with advanced multi-objective optimization algorithms which use machine learning techniques.

CONCLUSION

Here we have described `Xopt`, a unified framework for implementing and connecting algorithms to arbitrary optimization problems. This framework has already been used at a wide number of accelerator facilities and institutions including LCLS, LCLS-II, FACET-II, Cornell, LBNL, AWA, DESY, and ESRF. We welcome and encourage community contributions of algorithms to this open-source library in order to facilitate easy community access to state-of-the-art optimization algorithms for accelerator control and design.

REFERENCES

[1] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965, Publisher: Oxford Academic. doi:10.1093/comjnl/7.4.308

[2] X. Huang, "Robust simplex algorithm for online optimization," *Phys. Rev. Accel. Beams*, vol. 21, no. 10, p. 104 601, 2018. doi:10.1103/PhysRevAccelBeams.21.104601

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL164

[3] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994. doi:10.1162/evco.1994.2.3.221

[4] J. Duris *et al.*, "Bayesian Optimization of a Free-Electron Laser," *Phys. Rev. Lett.*, vol. 124, no. 12, 2020. doi:10.1103/physrevlett.124.124801

[5] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proc. of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016. doi:10.1109/JPROC.2015.2494218

[6] C. Mayes *et al.*, *Xopt v1.4.0*, 2023. doi:10.5281/zenodo.7894002

[7] R. Roussel *et al.*, "Turn-key constrained parameter space exploration for particle accelerators using Bayesian active learning," *Nature Communications*, vol. 12, no. 1, p. 5612, 2021. doi:10.1038/s41467-021-25757-3

[8] A. Scheinker and D. Scheinker, "Bounded extremum seeking with discontinuous dithers," *Automatica*, 2016. doi:10.1016/j.automatica.2016.02.023

[9] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009. doi:10.48550/arXiv.0912.3995

[10] D. Zhan and H. Xing, "Expected improvement for expensive optimization: A review," *Journal of Global Optimization*, pp. 1–38, 2020. doi:10.1007/s10898-020-00923-x

[11] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," *Advances in neural information processing systems*, vol. 32, 2019. doi:10.48550/arXiv.1910.01739

[12] F. Irshad, S. Karsch, and A. Döpp, "Expected hypervolume improvement for simultaneous multi-objective and multi-fidelity optimization," *arXiv preprint arXiv:2112.13901*, 2021. doi:10.48550/arXiv.2112.13901

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002. doi:10.1109/4235.996017

[14] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization," *Advances in Neural Information Processing Systems*, vol. 33, 2020. doi:10.48550/arXiv.2006.05078

[15] M. Song, X. Huang, L. Spentzouris, and Z. Zhang, "Storage ring nonlinear dynamics optimization with multi-objective multi-generation Gaussian process optimizer," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 976, p. 164 273, 2020. doi:10.1016/j.nima.2020.164273

[16] M. E. Conde *et al.*, *Research Program and Recent Results at the Argonne Wakefield Accelerator Facility (AWA)*. doi:10.18429/JACoW-IPAC2017-WEPAB132

[17] M. Zhang, "Emittance formula for slits and pepper-pot measurement," Fermi National Accelerator Lab., Batavia, IL (United States), Tech. Rep. FNAL-TM–1988, 1996. doi:10.2172/395453

[18] *Website for pyEPICS*, 2023. https://github.com/pyepics/pyepics

[19] C. Gulliford *et al.*, "Demonstration of low emittance in the cornell energy recovery linac injector prototype," *Phys. Rev. ST Accel. Beams*, vol. 16, p. 073 401, 7 2013. doi:10.1103/PhysRevSTAB.16.073401

[20] A. Bartnik, C. Gulliford, G. H. Hoffstaetter, and J. Maxson, "Ultimate bunch length and emittance performance of an mev ultrafast electron diffraction apparatus with a dc gun and a multicavity superconducting rf linac," *Phys. Rev. Accel. Beams*, vol. 25, p. 093 401, 9 2022. doi:10.1103/PhysRevAccelBeams.25.093401

[21] *Pulsar website for GPT*, 2011. http://www.pulsar.nl/gpt/

[22] *Website for Lume-GPT*, 2023. https://github.com/ColwynGulliford/lume-gpt