Teeport: BREAK THE WALL BETWEEN THE OPTIMIZATION ALGORITHMS AND PROBLEMS*

Z. Zhang†, X. Huang[1], M. Song, SLAC National Accelerator Laboratory, Menlo Park, CA, USA
[1]now at Argonne National Laboratory, Lemont, IL, USA

Abstract

Optimization algorithms/techniques such as genetic algorithm (GA), particle swarm optimization (PSO) and Gaussian process (GP) have been widely used in the accelerator field to tackle complex design/online optimization problems. However, connecting the algorithm with the optimization problem can be difficult, sometimes even unrealistic, since the algorithms and problems could be implemented in different languages, might require specific resources, or have physical constraints. We introduce an optimization platform named Teeport that is developed to address the above issues. This real-time communication (RTC) based platform is particularly designed to minimize the effort of integrating the algorithms and problems. Once integrated, the users are granted a rich feature set, such as monitoring, controlling, and benchmarking. Some real-life applications of the platform are also discussed.

INTRODUCTION

As the accelerator properties such as beam emittance and brightness are being pushed to their limit, the nonlinear effect is more and more important and must be taken into consideration and handled properly to realize the designed goals in operation. That's why applying modern optimization algorithms to tune the big machines online to optimize the machine performance has became a trend over the last few years in the accelerator field [1–7]. In a regular online optimization scenario, the evaluation script that controls the machine parameters and reads or calculates the objective to be optimized usually lives in the accelerator control room (ACR), while the codes of optimization algorithms are copied to the same computer in the ACR and adapted to the evaluation script and perform the optimization task there. There are a few problems posed in this simple and straightforward method. If the optimization algorithm was tested in a simulation setup and then copied to the ACR, some reconfiguration may be needed, such as adapting the API to the experimental evaluation script and setting up the algorithm run-time environment. These seemingly trivial tasks could be complicated, time-consuming, and error-prone. The work may need to be done each time a new algorithm is used or a new experimental problem is optimized. Furthermore, it could be a daunting task to connect the algorithm and the evaluation scripts if they are written in different languages. Sometimes for security considerations an externally developed algorithm run-time environment is not allowed to be deployed in the ACR.

In this stduy, we developed an online optimization platform, Teeport, to addresses the aforementioned communication difficulties between the optimization algorithms and application problems. It is task-based, extensible, embeddable, and can be used for optimization and real-time testing. With Teeport, the algorithms and problems can be effortlessly integrated into a real-time messaging service, which gives the ability for the two sides to talk to each other freely. Teeport has been applied to solve real-life remote optimization tasks in several national laboratories, including SLAC and ANL.

PHILOSOPHY

We first analyze the following real-life online optimization case as a way to introduce the philosophy behind Teeport.

Evaluator

Assume that we have a Matlab script that reads and writes the PVs through EPICS. When the optimization algorithm evaluates a solution (a point in the parameter space), the script writes the PVs with the values given by the algorithm, then reads and returns the PV value of the objective. There could be some configurable parameters during the evaluation, such as the waiting time between the PV writing and reading. Therefore, the whole evaluation process can be abstracted as a function:

$$\mathbf{Y} = \mathrm{evaluate}(\mathbf{X}, \mathrm{configs}). \tag{1}$$

Here $\mathbf{X}$ and $\mathbf{Y}$ are 2D arrays, have shape of (n, v) and (n, o) respectively, where n denotes the number of the points to be evaluated, v the number of the variables, and o the number of the objectives. Any evaluation process, including simulation, experiment, and parallel evaluation tasks that run on a cluster, could be abstracted as the `evaluate` function as shown in Eq. (1). In Teeport, we call the evaluation process that has been implemented in the form shown in Eq. (1) an evaluator.

Optimizer

On the other hand, assume the optimization algorithm is a Python script that imports several optimization-related packages, that accepts an `evaluate` function and tries to optimize it. The algorithm usually takes in parameters such as the dimension of the problem to be optimized, the number of the objectives, and parameters related to the termination conditions. The optimization algorithm can be abstracted

* Work supported by DOE, Office of Science, Office of Basic Energy Sciences, DE-AC02-76SF00515 and FWP 2018-SLAC-100469 Computing Science, Office of Advanced Scientific Computing Research, FWP 2018-SLAC-100469ASCR.
† zhezhang@slac.stanford.edu

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB305

like this:

$$[\mathbf{X}_{\text{opt}}, \mathbf{Y}_{\text{opt}}, \cdots] = \text{optimize}(\text{evaluate}, \text{configs}), \qquad (2)$$

where $[\mathbf{X}_{\text{opt}}, \mathbf{Y}_{\text{opt}}, \cdots]$ are optional return arguments. Any optimization process, including MOGA, GP optimizer, and even a human being who decides which data points to evaluate in the next step, could be abstracted as such a `optimize` function as shown in Eq. (2). In Teeport, we call the optimization process implemented in the form shown in Eq. (2) an optimizer.

Adapter

Since the evaluator and the optimizer are implemented in different languages, to enable them to talk to each other, Teeport provides an adapter, or client, for each language, and a messaging engine as a middleware between the evaluator and the optimizer. With the corresponding adapter, the data flowing in and out of the optimizer and the evaluator will be normalized to a standard format and subsequently forwarded by the messaging engine to complete the optimization loop. The process is as illustrated in Fig. 1.

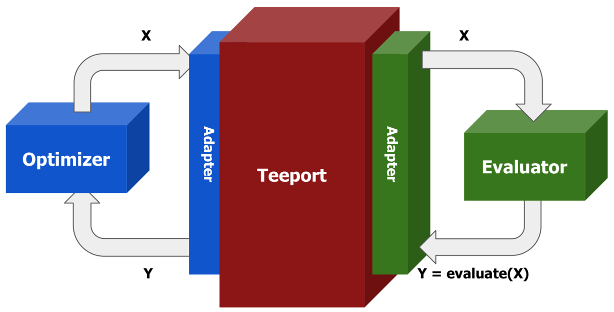

Figure 1: The architecture of Teeport. With the help of the adapters provided by Teeport, the optimizer and the evaluator can exchange data in a normalized format.

Monitor

Since the optimization data flow through the Teeport messaging middleware, one can add the control and monitor layers to the middleware, to make the online optimization more controllable and visible. A visualization of the optimization process based on the data flow is called a monitor, and is provided by the Teeport GUI through a browser. Examples of monitors provided by Teeport are shown in Fig. 2.

FEATURES

Remote Online Optimization

Teeport completely decouples the evaluator and the optimizer, which makes performing an online experimental optimization as simple as doing a local optimization. The workflow to do an online optimization experiment is as follows:

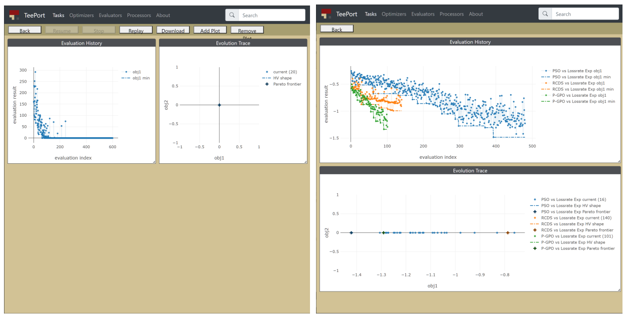

Figure 2: Left: the history data of an online optimization experiment that was performed through Teeport; Right: comparison among the performance of three optimization algorithms against the SPEAR3 beam loss rate online optimization problem.

1. Code the experimental evaluator, and integrate it to Teeport with the `run_evaluator` API. Teeport will generate an id and assign it to the evaluator.
2. On the local computer, use the Teeport adapter for the language of the optimizer, and get a local `evaluate` function through the `use_evaluator` API, with the id of the last step.
3. Call the local `optimize` function on the local `evaluate` function to perform the optimization.

After going through the above steps, the user will be automatically granted a set of features through the Teeport GUI, such as monitoring the optimization progress, pausing or resuming the optimization, terminating the optimization, and so on. Figure 2 left plot shows a monitored single objective optimization task.

Fast Switch Between Different Optimization Settings

Since we can run multiple evaluators and optimizers on Teeport, we can switch the optimization settings by: 1) Select the target evaluator and optimizer pair through the Teeport GUI, or 2) Use the `use_evaluator` API and/or the `use_optimizer` API to get local evaluator and/or optimizer, then do the optimization normally.

The only actions that are needed to switch between the different optimization settings with regard to the code are switching the evaluator/optimizer id and/or update the configurations of the evaluator/optimizer accordingly. The process is visualized in Fig. 3.

Optimization Performance Comparison

As data flow between the optimizer and the evaluator through the Teeport platform, Teeport can archive the data for future reference. With this data monitoring and archiving capability, we can easily compare the performance of an optimizer on a series of testing evaluators, or compare the efficiency of different optimizers against the same to-be-optimized evaluator, as shown in the right plot in Fig. 2.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB305

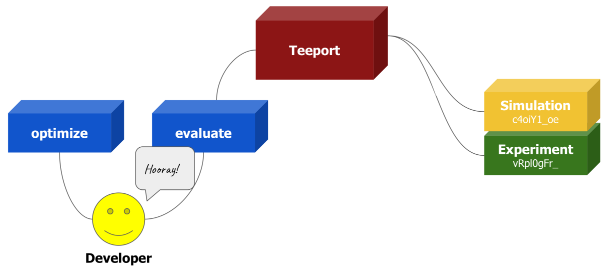

Figure 3: Fast switching between the simulation evaluator and the experimental evaluator. The user can select the evaluator to be optimized upon by the corresponding id, get the specific local evaluator, and then perform the optimization.

Through the comparison feature of Teeport, we could determine how efficient each optimizer could be with respect to the specific evaluator on the fly during the experiment, and adjust our optimization strategy accordingly.

Optimization Algorithm Benchmark

Teeport has the ability to turn a local `optimize` function into an online optimizer, and provides the user full control: one can start, pause, resume, and terminate it. With this capability, we can benchmark any optimization algorithm effortlessly. One can pick an optimization configuration for the optimizer-evaluator pair, and tells Teeport to repeat the run for multiple times. Teeport will automatically perform statistics on the results of the multiple runs and generate some useful plots (objective mean and variation, Pareto front distribution for multi-objective optimizations, etc) to demonstrate the algorithm performance.

APPLICATIONS

SPEAR3 Beam Loss Rate Remote Optimization

The most important application of Teeport (which was also the problem that lead to the creation of Teeport) is remote online optimization. Usually to perform a beam-based online optimization, such as the SPEAR3 beam loss rate optimization, one needs to clone the algorithm from the local computer to the ACR computer. With Teeport, to perform a remote online optimization is effortless. For example, below is what was done to optimize the SPEAR3 beam loss rate [8]:

1. Run the beam loss rate evaluation script as an evaluator through the Teeport adapter for Matlab in the ACR.

2. Get the corresponding local evaluator through the Teeport adapter for Matlab on the local laptop.

3. Call the `optimize` function with the local `evaluate` function.

The remote optimization results and more details can be found in the corresponding paper [8].

Enhance MG-GPO With GPy

In addition to be able to convert an `evaluate` function to an online evaluator, Teeport is also capable to convert an arbitrary function to an online processor, as long as the function is pure [9] and the arguments and returns of the function are serializable. That means we can use the API provided by packages that are written in different languages through Teeport.

When we were developing MG-GPO [10, 11], initially we were not able to find a good Matlab Guassian process package, which is important as GP modeling part is at the core of the algorithm. In Python, there does exist an excellent GP package called GPy [12]. The problem was, how to use GPy to handle the GP modeling part, while keeping all other logic in Matlab? This is solved with Teeport by running GPy's GP modeling function as a processor on Teeport, applying the `use_processor` API to get a Matlab version of the GP modeling function, and using it in our algorithm evolution loop.

Unified Interface for the Optimization Platforms

Optimization algorithms have been widely used in the accelerator field, and people are introducing more and more algorithms to tackle the complex optimization problems. While it is usually good to have more options to address a hard problem, it could become problematic when the number of the optimization platforms are overwhelming. There are currently more than 500 optimization platforms, each of them is equipped with a collection of optimization algorithms/test problems. Since the platforms were developed by different people in different fields, and usually targeted different challenges, the usages are quite diverse. This diversity could bring confusion and frustration to the users, especially when the user tries to use the algorithms and test problems from multiple platforms at the same time. Teeport addresses this problem by providing a minimal set of integration APIs to effortlessly integrate the algorithms and test problems. It enables the users to use any of them through a unified API (`use_optimizer` and `use_evaluator`), therefore alleviates the problem.

We have already integrated a large number of optimization algorithms and test problem from various platforms, such as PyGMO [13], pymoo [14], PlatEMO [15] and Ocelot Optimizer [16], to Teeport.

CONCLUSION

We developed a RTC based online optimization platform, Teeport, to break the communication wall between the optimization algorithms and the application problems that live in different environments. We applied Teeport to perform and control remote online optimizations, monitor and benchmark the performance of the optimization algorithms, and help developing and enhancing algorithms. Teeport has been tested and deployed at SLAC and ANL. More information about Teeport can be found at [17].

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB305

REFERENCES

[1] X. Huang, J. Corbett, J. Safranek, and J. Wu, "An algorithm for online optimization of accelerators", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 726, pp. 77–83, Oct. 2013. doi:10.1016/j.nima.2013.05.046

[2] X. Pang and L. Rybarcyk, "Multi-objective particle swarm and genetic algorithm for the optimization of the lansce linac operation", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 741, pp. 124–129, Mar. 2014. doi:10.1016/j.nima.2013.12.042

[3] X. Huang, "Development and Application of Online Optimization Algorithms", in *Proc. North American Particle Accelerator Conf. (NAPAC'16)*, Chicago, IL, USA, Oct. 2016, pp. 1287-1291. doi:10.18429/JACoW-NAPAC2016-FRA2IO01

[4] W. F. Bergan, I. V. Bazarov, C. J. Duncan, D. B. Liarte, D. L. Rubin, and J. P. Sethna, "Online storage ring optimization using dimension-reduction and genetic algorithms", *Physical Review Accelerators and Beams*, vol. 22, no. 5, p. 054601, May 2019. doi:10.1103/physrevaccelbeams.22.054601

[5] K. Tian, J. Safranek, and Y. Yan, "Machine based optimization using genetic algorithms in a storage ring", *Physical Review Special Topics - Accelerators and Beams*, vol. 17, no. 2, p. 020703, Feb. 2014. doi:10.1103/physrevstab.17.020703

[6] J. Duris *et al.*, "Bayesian optimization of a free-electron laser", *Physical Review Letters*, vol. 124, no. 12, p. 124801, Mar. 2020. doi:10.1103/PhysRevLett.124.124801

[7] D. K. Olsson, "Online Optimisation of the MAX IV 3 GeV Ring Dynamic Aperture", in *Proc. 9th Int. Particle Accelerator Conf. (IPAC'18)*, Vancouver, Canada, Apr.-May 2018, pp. 2281-2283. doi:10.18429/JACoW-IPAC2018-WEPAL047

[8] Z. Zhang, M. Song, and X. Huang, "Online accelerator optimization with a machine learning-based stochastic algorithm", *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 015014, Dec. 2020. doi:10.1088/2632-2153/abc81e

[9] S. P. Jones, *Haskell 98 language and libraries: the revised report*, Cambrigde, UK: Cambridge University Press, 2003.

[10] X. Huang, M. Song, and Z. Zhang, "Multi-objective multigeneration gaussian process optimizer for design optimization", 2020. arXiv:1907.00250

[11] X. Huang, M. Song, and Z. Zhang, "Multi-Objective Multi-Generation Gaussian Process Optimizer", presented at the 12th Int. Particle Accelerator Conf. (IPAC'21), Campinas, Brazil, May 2021, paper WEPAB304, this conference.

[12] GPy: A gaussian process framework in python, http://github.com/SheffieldML/GPy

[13] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: Pagmo", *Journal of Open Source Software*, vol. 5, no. 53, p. 2338, 2020. doi:10.21105/joss.02338

[14] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python", *IEEE Access*, vol. 8, pp. 89497–89509, Apr. 2020. doi:10.1109/access.2020.2990567

[15] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]", *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, Nov. 2017. doi:10.1109/mci.2017.2742868

[16] S. I. Tomin *et al.*, "Progress in Automatic Software-based Optimization of Accelerator Performance", in *Proc. 7th Int. Particle Accelerator Conf. (IPAC'16)*, Busan, Korea, May 2016, pp. 3064-3066. doi:10.18429/JACoW-IPAC2016-WEPOY036

[17] Teeport, https://teeport.ml/intro