Report for Beam Based Optimization and Machine Learning for Synchrotrons at SSRL

Xiaobiao Huang, Minghao Song, and Zhe Zhang

SLAC National Accelerator Laboratory

January 26, 2021

Abstract

Here we report the research activities and achievements during the two-year R&D project of 'beam-based optimization and machine learning for synchrotrons' at SSRL. These include development of machine learning based optimization algorithms, application of such algorithms to accelerator design and online tuning, and development of neural network based method to analyze accelerator operation data and study the impact of underlying environmental factors to machine performance.

One particular achievement is the development of the multi-generation Gaussian process optimizer (MG-GPO) and the demonstration of the method to nonlinear beam dynamics optimization in storage ring lattice design study and to online optimization of dynamic aperture. The method was found to substantially outperform traditional stochastic algorithms in both simulations and experiments. An online optimization platform was developed to facilitate the development, testing, and application of online optimization methods on various systems. A deep-learning method was also developed and successfully applied to storage nonlinear lattice optimization.

Contents

1 Overview

The R&D project 'beam-based optimization and machine learning for synchrotrons' is jointly funded by BES and ASCR for a period of two years, from August 2018 to August 2020. The goals of the project are to develop machine learning-based, advanced methods for accelerator design, commissioning, and operation, with applications focused on synchrotrons. This project is in collaboration with a team at Advanced Light Source. This report only covers the activities and achievements for the SSRL effort.

Currently the synchrotron light source community is experiencing a phase of fast changes from the traditional 3rd generation light sources to the multi-bend achroamt (MBA)-based 4th generation storage ring light sources. In the US, the Advanced Photon Source (APS) and the Advanced Light Source (ALS) are both being upgraded to MBA-based new rings. The upgrades will substantially increase the photon beam brightness from what is afforded by the current rings. However, the improvement is accompanied by new challenges. As the new rings push for extremely lower emittances, the beam dynamics in the rings become significantly nonlinear, causing smaller dynamic aperture and potentially smaller momentum aperture. It is very important to optimize the lattice in the design phase to achieve good nonlinear beam dynamics performances. Highly efficient global optimizers are needed for such design optimizations. It is also very important to develop methods to implement the lattice on the real machine in order to achieve the design performance. Advanced online optimization provides an effective way of finding the ideal machine configuration.

In this study, we developed two machine learning-based global optimization algorithms and applied them to storage ring nonlinear beam dynamics optimization. The first is the multi-generation Gaussian process optimizer (MG-GPO) [1, 2]. The second one is a neural network based method [3]. Both methods and their applications are discussed in Section 2.

We explored various methods of online optimization for storage ring applications. The physics informed Gaussian process optimizer, developed for the FEL power optimization on LCLS [4], was adopted and applied to the SPEAR3 storage ring in experiments [5]. The MG-GPO algorithm was also applied to multiple real-life experiments on SPEAR3 and was found to outperform other stochastic optimization algorithms [6]. To facilitate the application of optimization algorithms developed in one programming environment to accelerators equipped with many different control systems and programming environments, we developed a universal online optimization platform. These studies are discussed in Section 3.

The extremely small emittance in a new ring requires much higher beam stability, which in turn requires a good understanding of the impact of environmental factors on the accelerator and the beams. Accelerator operation history data contain vast amount of information of machine status, which could provide insight on how the environment affects the machine. However, advanced methods are needed to analyze the data and extract useful information. We developed a neural network-based method to analyze accelerator operation data and applied it to the analysis of the dependence of SPEAR3 injection efficiency performance on environmental factors [7]. This study is discussed in Section 4.

A summary is given in Section 5.

2 Accelerator design optimization

2.1 The multi-generation Gaussian process optimizer

We proposed the multi-generation Gaussian process optimizer (MG-GPO) to solve multiple objective optimization problems [1]. MG-GPO builds up Gaussian process models with the aid of Gaussian process regression and use the models to pre-select the fixed population size of candidates with high potential yielding good performance. It will improve the algorithm

efficiency because of Gaussian process models are based on the previous evaluated solutions without losing the measured information.

2.1.1 Gaussian process regression

Gaussian process regression (GPR) [8, 9, 10] implements the Gaussian process for continuous input and output. As a non-parametric method, it will calculate the probability distribution over all acceptable functions that best fit the samples rather than calculating parameters for a specific function, therefore, a Gaussian process is a multivariate Gaussian distribution with infinite dimensions and joint distribution of any variables is also a normal distribution. In addition, as a type of Bayesian approach, it would also calculate the predictive posterior based on the prior assumption and observations. Generally, the Gaussian process can be specified by the mean function $m(\mathbf{x})$ and the co-variance function or called kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ [10]. The mean function can be zero or non-zero values. There are also many options for the kernel function, for example, linear, Matern kernel or a combination of different kernels. The selection of proper kernel function plays an important role in Gaussian process regression. Here the popular kernel squared exponential was used and described in the following [10]:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\ell}^{-2}(\mathbf{x}_i - \mathbf{x}_j)) + \sigma_n^2 \delta_{ij}, \tag{1}$$

There are three types of hyper-parameters: σ_f^2, σ_f^2 and $\boldsymbol{\ell}$. Where σ_f^2 and σ_n^2 represent the signal variance and Gaussian noise, respectively; $\boldsymbol{\ell} = \text{diag}(\ell_1, \ell_2, \cdots, \ell_P)$ is a diagonal matrix, P is the dimension of input space and the length-scales in the diagonal reflects the correlation of two points located at different positions. The bigger length-scale will cause less correlation. It is expected that best setup of hyper-parameters can be adopted to improve the model accuracy and consequent optimized results. A commonly used approach to tune the hyper-parameters is to maximize the log marginal likelihood of training samples.

In Gaussian process regression, all or part of observed data can be used as training data to construct models to make predictions on the data of interest. From the Gaussian process prior, the joint distribution of training samples $\mathbf{x}_i$, $i = 1, 2, \cdots, N$ and testing samples $\mathbf{x}^*$ is a multivariate Gaussian distribution and in the form of [10]:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \boldsymbol{N} \left(0, \quad \begin{bmatrix} \mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \\ \mathbf{K}(\mathbf{x}^*, \mathbf{x}) & \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix} \right) \tag{2}$$

The mean function of Gaussian process prior is often assumed to be zero without learning any knowledge in advance. Where $\mathbf{K}(\mathbf{x}, \mathbf{x})$ is the kernel matrix with its entries correspond to kernel function obtained from training points. $\mathbf{I}$ is the N dimensional matrix. $\mathbf{K}(\mathbf{x}, \mathbf{x}^*) = \mathbf{K}(\mathbf{x}^*, \mathbf{x})^T$ are the kernel vectors given by observed and test points. Given the joint distribution function and sample data, it is tractable to calculate predictive posterior distribution, which is a normal distribution that could be completely described by mean and covariance functions [10].

$$\mu_{y^*} = \mathbf{K}(\mathbf{x}^*, \mathbf{x})(\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{3}$$

$$\sigma_{y^*}^2 = \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{K}(\mathbf{x}^*, \mathbf{x})(\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}(\mathbf{x}^*, \mathbf{x})^\top, \tag{4}$$

Where mean function μ_{y^*} gives the estimated prediction and covariance function $\sigma_{y^*}^2$ shows the standard deviation.

With the aid of Gaussian process posterior, acquisition function will guide the search of new solutions in the complex parameter space. There are many options for the acquisition function which includes probability of improvement [11], expected improvement [12, 9]. For a minimization type optimization problem in our study, the lower confidence bound (LCB) was used and given by [13]

$$\text{GP-LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}), \tag{5}$$

Where μ and σ are predicted mean and variance, respectively. There is a trade-off between exploration and exploitation, which was determined by the positive paremeter κ. A large κ tends to explore the region with high uncertainty, while a small κ favors to exploit the region with less uncertainty that might yield bigger gain. It is important to choose a suitable κ to balance two searching strategies. κ could be either a proper constant or gradually decreased with a constant decay rate.

2.1.2 Description of MG-GPO

Similar to traditional evolutionary algorithms such as NSGA-II [14] and MOPSO [15], MG-GPO also manipulates a fixed population size over the optimization. Differently, MG-GPO employs the Gaussian process regression model as surrogate model to replace the actual physics model, which is cheaper to evaluate objective functions. Meanwhile, the GP models are used to pre-select the candidates that have the potential to produce good results from a large quantities of trial solutions.

In the initialization stage, a population of solutions $\mathcal{G}_0$ was generated randomly in the whole parameter space or within a small region. The population size is N. Then the prior Gaussian process model $\mathcal{GP}_0$ was built up based on the evaluated solutions of $\mathcal{G}_0$.

In the optimization loop, at nth generation. For each solution in $\mathcal{G}_{n-1}$, it will generate a large number of trial solutions through simple random operations. Among them, m_1 new solutions will be produced by polynomial mutation (PLM) [16] with m_2 new solutions by simulated binary crossover (SBX) [17], which is the same as done in NSGA-II. Besides, m_3 new individuals could be generated by using the method in PSO. It is worth noting that part or all techniques can be adopted in generating trial solutions, and the multiplication factor $(m_1 + m_2 + m_3)$ can be varied. The $(m_1 + m_2 + m_3)N$ trial solutions will then be evaluated by GP model $\mathcal{GP}_{n-1}$. The predicted mean and variance will be used in acquisition function to direct the sampling in the area with bigger gain. A non-dominated sorting will be performed to select N best solutions $\mathcal{F}_n$ and evaluate them in the actual system, which is then combined with $\mathcal{G}_{n-1}$ and update the best solutions with another non-dominated sorting. Either evaluated solutions in the last few generations or current generation can be combined with $\mathcal{G}_n$ to construct $\mathcal{GP}_n$.

The MG-GPO algorithm can be summarized as:

$n \leftarrow 0$, Initialize the population, $\mathcal{G}_0$.

Evaluate all solutions in $\mathcal{G}_0$
Construct Gaussian process models, $\mathcal{GP}_0$, with $\mathcal{G}_0$
while $n < G_{max}$ **do**
  $n \leftarrow n + 1$
  For each solution in $\mathcal{G}_{n-1}$, generate m_1 solutions with mutation, m_2 solutions with crossover and m_3 solutions with swarming
  Evaluate the $(m_1 + m_2 + m_3)N$ solutions with $\mathcal{GP}_{n-1}$
  Use non-dominated sorting to select N best solutions, which forms the set $\mathcal{F}_n$.
  Evaluate the solutions in $\mathcal{F}_n$ in the actual system.
  Use non-dominated sorting to select N best solutions from the combined set of $\mathcal{G}_{n-1}$ and $\mathcal{F}_n$, the results of which form $\mathcal{G}_n$.
  Construct Gaussian process models, $\mathcal{GP}_n$, with evaluated solutions and $\mathcal{G}_n$.
end while

2.2 Testing of MG-GPO with analytic problems

Extensive testing has been conducted to characterize and benchmark the performance of the MG-GPO algorithm [1], using common testing problems such as the ZDT series [18]. In tests with 30 decision variables, MG-GPO substantially outperforms other advanced multi-objective optimization algorithms such as NSGA-II [14], MOPSO [19], MMOPSO [20], and WOF-SMPSO [21] in terms of both the convergence speed and the completeness of the Pareto front (see Figure 1). Similar performance was found for cases with 100 decision variables.

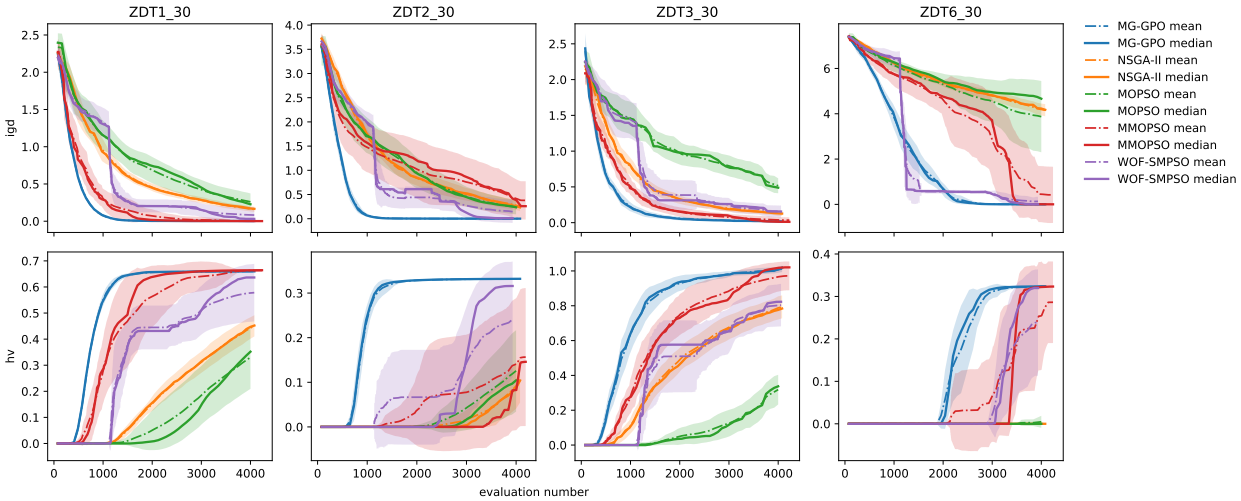

Figure 1: Comparison of the hypervolume (HV) and inverted generational distance (IGD) measures of MG-GPO with other advanced multi-objective optimization algorithms. Figure reproduced from Ref. [1].

2.3 Application of MG-GPO on storage ring lattice design

MG-GPO was applied to the SPEAR3 upgrade lattice design optimization, and compared with traditional stochastic algorithms, NSGA-II and MOPSO [2]. In this optimization problem, the optimization variables are 8 combined sextupole knobs made out of 10 sextupole families and the two objectives are dynamic aperture and local momentum aperture. The evaluations of two objectives are done by particle tracking simulation.

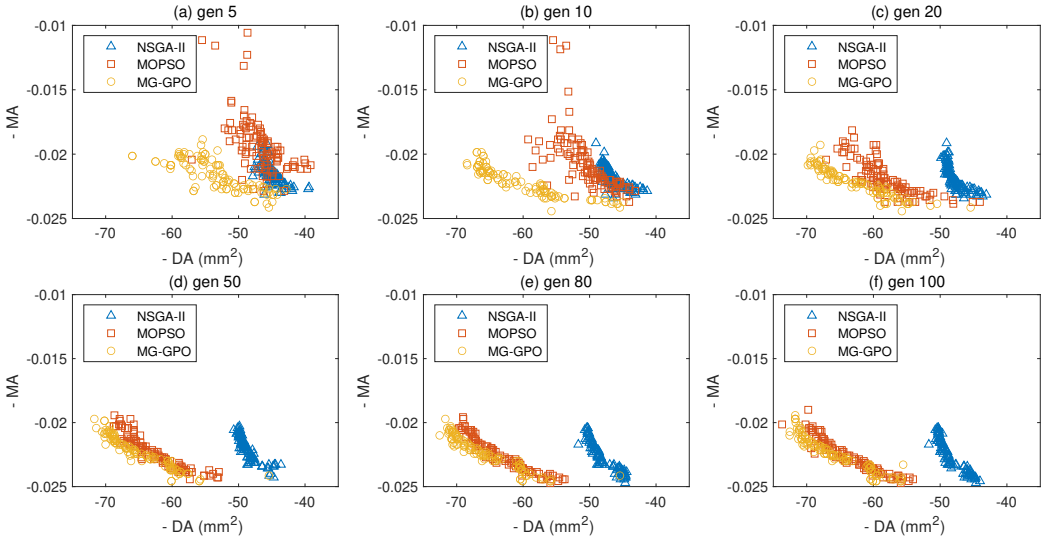

Figure 2: Comparison of the best solutions in the objective space at generation 5, 10, 20, 50, 80 and 100 for the three algorithms, NSGA-II, MOPSO, and MG-GPO. Figure reproduced from Ref. [2].

Figure 2 shows the comparison of the best solutions of dynamic aperture (DA) and local momentum aperture (LMA) in the objective space at generation 5, 10, 20, 50, 80 and 100 for the three algorithms. It is clearly shown that MG-GPO converges faster than the other two algorithms. It almost only takes 20 generations to reach the final distribution found so far. However, MOPSO needs 50 generations to reach the same level, and NSGA-II does not converge to the same front even after 100 generations. NSGA-II may prematurely converge to local optimum if solutions lack of diversity. It was obvious that MG-GPO converges significantly faster than MOPSO and NSGA-II, which allows us to speed up lattice design studies as fewer evaluations are now needed to achieve the optimal solutions. A popular metric, the hypervolume (HV) was used to measure the performances of algorithms [2, 22, 23]. It is particularly useful in real-life optimization problems as the true Pareto front may not be precisely known. A larger HV indicates better performance in terms of convergence rate and diversity. Figure 3 illustrates the comparison of HV for the three algorithms over 100 generations, which again demonstrates MG-GPO outperforms MOPSO and NSGA-II.

The final best solutions optimized by the algorithms are significantly improved, which allow us to select appropriate solutions found by MG-GPO as a trade-off between two competing objectives. Two solutions are selected as examples from the MG-GPO Pareto front to compare with initial solution with flat sextupole configuration (i.e. all SDs and SFs have

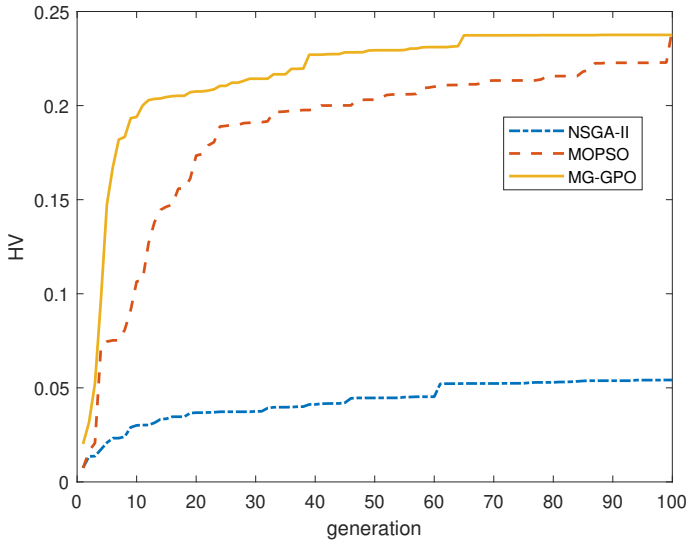

Figure 3: Comparison of hypervolume (HV) for the three algorithms over 100 generations. Figure reproduced from Ref. [2].

the same strength). One solution has larger DA but at the cost of the LMA (option 1), while the other emphasizes the latter more (option 2). Figure 4 shows the comparison of DA and LMA for the initial solution and and the two selected options. It is clear that both solutions substantially improve the DA in the negative horizontal plane, where the beam was injected.

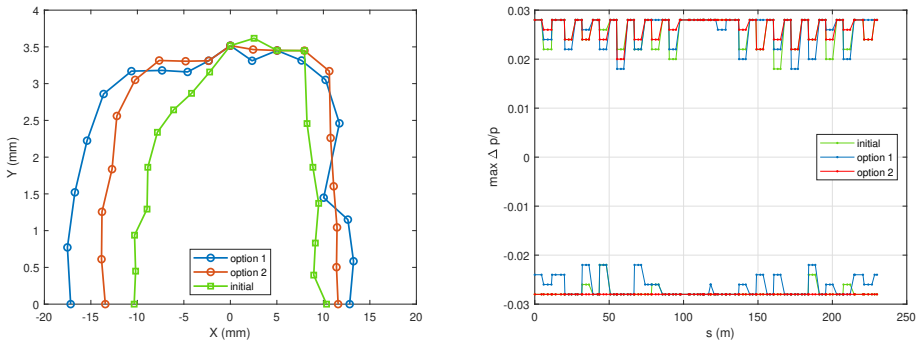

Figure 4: Comparison of DA (left) and LMA (right) for the initial lattice and two selected optimized solutions for SPEAR3 nonlinear lattice optimization with MG-GPO. Figure reproduced from Ref. [2].

2.4 Neural network based optimizer and application

Stochastic algorithms manage to outperform pure random search because they use the past solutions to guide the search in the parameter space. Modeling all past solutions with machine learning approaches provide an effective way to extract the information contained

in the past solutions. Compared to GP regression, neural networks can model a large number of solutions without introducing computation difficulty in the training process.

We developed an NN-based method for multi-objective design optimization [3]. In this method, an NN is trained for each objective function using the decision variables as input parameters. All past solutions are used as the training data set. Unlike MG-GPO, where the ML model is used to filter out the trial solutions, here the NN models are used to optimize for new trial solutions. To cover the full Pareto front, the objective functions are combined to form a single objective with various combination of weights. For example, 20 combinations can be made for the case of two objectives. For each combination, the genetic algorithm NSGA-II is used to find the best solution for the trained model. A number of random solutions are generated within a small hypercube around the best solution are used a trial solutions to be evaluated on the real system. The random solutions are introduced to improve exploration of the parameter space and to fill the gaps on the Pareto front between the best solutions of the combined objectives.

This approach has been demonstrated with storage ring nonlinear lattice optimization, using the SPEAR3 upgrade lattice as an example. The 10 sextupole families in the ring are used as decision variables to optimize DA and LMA. The NN consists of 5 fully connected layers. A leaky rectified linear unit activation function (LeakyReLU) [24] is applied to all intermediate layers. The output is the average of an ensemble of 20 NN models, each of which has a random LeakyReLU slope coefficient drawn from within $(0, 1)$.

In one test case, the algorithm evaluates $N = 1000$ solutions (i.e., 50 random solutions are generated for each of the 20 weight combinations). Starting from an initial distribution of N solutions, the algorithm reached the Pareto front within two iterations (as shown in Figure 5).

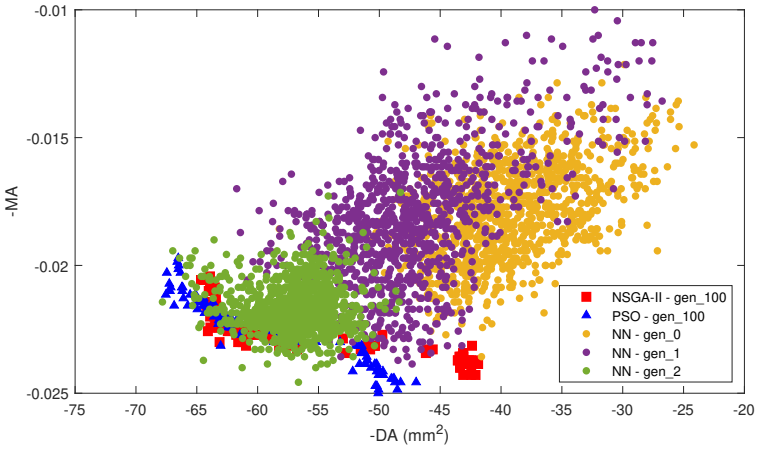

Figure 5: DA/MA evolution with NN-based optimizer for $N = 1000$ in comparison with NSGA-II and PSO best solutions after 10000 evaluations. Figure reprodeced from Ref. [3].

Similar to MG-GPO, this method could be generalized to other design optimization problems.

3 Beam-based optimization

3.1 A brief review of online optimization algorithms

In recent years, online optimization has drawn attention in the accelerator community. It is used to improve accelerator performance when beam-based correction techniques cannot be applied, which could happen if there are inadequate diagnostics, if the correction targets are not well defined, or if there a deterministic solution cannot be established based on the diagnostics and the target. The tuning of storage ring nonlinear beam dynamics performance is a good example where beam-based optimization is essential.

Traditional manual tuning is a primitive form of online optimization. Typically it can only handle a very small number of tuning knobs and the time of evaluating one configuration is long. Automatic tuning talks directly to the control systems in modifying the machine configuration and measuring the machine performance and relies on advanced optimization algorithms to process the measurement data. Therefore, it is faster in both data taking and decision making; it can also handle substantially more complex tuning problems that have more tuning knobs and coupled, nonlinear parameter spaces.

Successful online optimization algorithms need to overcome a number of challenges. First of all, they need to be very efficient: the algorithms should find the best solution in with as few function evaluations as possible. Secondly, the algorithms should be robust against the inevitable measurement noise that comes into the function evaluation. Thirdly, they need to behave properly in case of data outliers or machine anomaly.

An optimization problem can have a single optimization objective or multiple objectives. In the later case, often times, the multiple objectives can be combined into one with weighted sum. However, in some cases, it is desirable to obtain the Pareto front, i.e., the best solutions in the sense of all non-dominated solutions in the parameter space. We will discuss the single objective optimization algorithms first, followed by the multi-objective algorithms that are aimed at uncovering the Pareto front. Optimization algorithms can also be classified according to their nature of being deterministic or stochastic. Deterministic algorithms follow the same convergence path every time when launched from the same starting point, while the paths of stochastic algorithms vary from one run to another as they employ random operations or random decisions.

The deterministic single objective optimization algorithms could be grouped into two categories: gradient-based algorithms or gradient-free algorithms. Gradient-based algorithms use the derivatives (first or second order) of the objective function with respect to the decision variables in the algorithms. The derivatives may be given in analytic forms (if available) or computed with numeric differences. Gradient-free algorithms do not use the derivatives.

The following gradient-based algorithms are common in mathematical optimization:

- Gradient descent, also known as steepest descent algorithm, for which only the first derivatives (i.e., the gradient) are used.
- Newton method, which uses a quadratic approximation of the objective function to solve for the increment toward the optimum. The first and the second order derivative (Hessian matrix) are both used.

- Quasi-Newton methods, which builds up approximations to the inverse Hessian matrix through past evaluations of function values and gradients. The BFGS algorithm is widely used in practice.

- Conjugate-gradient method, which does not explicitly use the Hessian or its inverse but achieves a similar performance by iteratively updating the gradient and the conjugate direction.

The gradient descent method may not be efficient for some problems with coupling between the decision variables. The other methods that use the second order derivatives or equivalents can be very efficient for noise-free objective functions. However, in online optimization applications, typically the gradient and the Hessian matrix are not given in analytic forms. As the measurement noise will introduce errors to the derivatives calculated with numeric differences, especially for the second order derivatives, the application of these methods in online optimization would be very challenging.

Gradient-free algorithms include direct search methods, such as pattern search and Nelder-Mead simplex search [25]. These methods only use function value comparison to decide search directions. They also include methods that use the function values in determining the search, for example, Powell's method [26].

The simplex method follows a routine to update a set of points that form a non-degenerate geometric body, the simplex, in the parameter space. It uses function values on the simplex vertices and new sample points to find a new point to replace an existing vertex. Powell's method does iterative 1-dimensional search and use the past solutions to build up a conjugate direction set. Both the simplex method and Powell's method are very efficient for smooth functions. However, they are very sensitive to noise in the function values.

The robust conjugate direction search (RCDS) method was based on Powell's method, but with substantial modifications, which enable it to cope with the measurement noise [27]. The RCDS method has found applications in over 30 accelerator facilities [28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Recently the simplex method is also modified to gain robustness against noise, resulting the robust simplex method [38].

Stochastic optimization algorithms include random search, simulated annealing, genetic algorithms, particle swarm optimization (PSO), etc. These algorithms involve random operations in generating new trial solutions or random decisions in choosing the convergence path. While stochastic algorithms are typically not very efficient, they often have better ability in locating the global optima. The genetic algorithm NSGA-II [14] has been applied to online optimization [39, 34]. The PSO has also been used in online applications [28, 40]. It is desirable to develop efficient stochastic optimization algorithms which keep the ability to find global optima.

Machine learning-based optimization algorithms have recently been adopted for online optimization. Physics-informed Gaussian process (GP) optimizer, a form of Bayesian optimization, has been used in FEL tuning [4]. The GP optimizer was found to be an efficient method in experiments when used to improve FEL pulse intensity with optics matching quadrupole knobs.

In the R&D for the present project, we explored the application of the GP optimizer on storage rings and evaluated its performance. Neural networks trained with simulation data

were used to construct a physics-informed GP kernel, which helped to gain much higher efficiency than other methods. We also explored applying the MG-GPO algorithm to online optimization problems on storage rings. It was found that it converges faster than traditional stochastic algorithms. Simulation and experiments were both used to demonstrate the efficiency of the algorithms.

3.2 Application of MG-GPO on online optimization of storage rings

Stochastic optimization algorithms have better ability to locate the global optima in a complex parameter space, given the random operations they employ for exploration. Some online tuning problems are better addressed with stochastic optimization algorithms, especially the ones without a known "good" starting point. As an efficient stochastic optimization algorithm, MG-GPO can be an effective tool for online tuning. To demonstrate the online optimization capability of the MG-GPO algorithm, we applied it experimentally to the SPEAR3 storage ring. We'll show the results for two applications: the vertical emittance minimization problem, and the dynamic aperture maximization problem. The MG-GPO algorithm was configured to use population size 30 in all the cases.

3.2.1 Vertical emittance minimization in SPEAR3

SPEAR3 is a third generation storage ring, in which the beam vertical emittance is determined by linear coupling and vertical dispersion, both of which can be corrected by skew quadrupole magnets. With a high bunch current, the beam loss rate is dominated by Touschek scattering, which is affected by the vertical emittance. Therefore, the beam loss rate can be used as an indirect measure of the vertical emittance. To minimize the vertical emittance in SPEAR3, one can maximize the beam loss rate for a high current beam by tuning the 13 skew quadrupoles magnets [27].

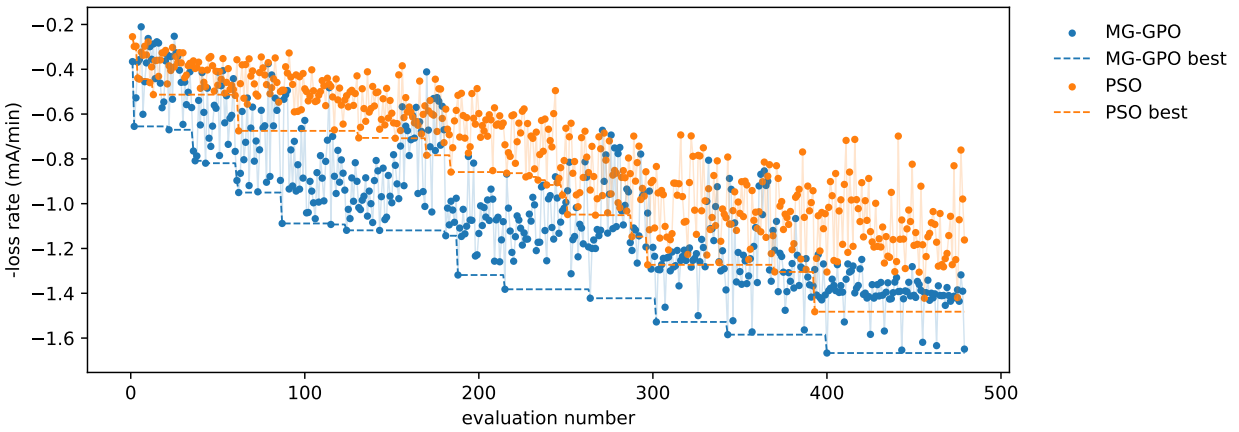

Figure 6: Comparison of the history of all evaluated solutions and the best-to-date solution during the beam loss rate optimization experiment with the MG-GPO and PSO algorithms.

The PSO algorithm was used to benchmark MG-GPO during the experiment. As shown in Figure 6, within around 480 evaluations, the loss rate reached -1.66 mA/min for MG-

GPO, which is quite close to the maximum loss rate achieved in the SPEAR3 ring in recent studies. While for PSO, not only the convergence speed is slower, but also the converged loss rate only reached -1.48 mA/min, obviously outperformed by MG-GPO.

3.2.2 Dynamic aperture maximization in SPEAR3

One key metric of a storage ring is the injection efficiency, which measures the ratio of the particles that can be injected into the ring. To achieve high injection efficiency, a large dynamic aperture is needed. However, in a low emittance storage ring like SPEAR3, strong sextupole magnets are used to correct chromaticities, while the magnets introducing the non-linearity at the same time, which reduces the dynamic aperture. Nevertheless, dynamic aperture can be effectively optimized by tuning the sextupole magnets in some manner that the desired chromaticities are kept [41, 36, 37].

The optimization of dynamic aperture is performed by optimizing the injection efficiency with sextupole knobs, the power supply set points of 10 sextupole families are varied through 8 combined knobs to keep the needed chromaticities [41]. We started the optimization from the flat sextupole solution, in which all SFs/SDs are set to the same value, accordingly. Each sextupole current is limited to vary within ± 20 A from the initial setting for protection and safety consideration.

The optimization approach was like this: at first we set the kicker bump size to the standard value, perform the optimization, hopefully the optimizer would converge soon and push the injection efficiency close to 100%. Then we reduce the kicker bump size by 20%, which will cause the degrading of the injection efficiency, and do the optimization from the converged solution of the first round until converging again. The result is shown in Figure 7 and Figure 8.

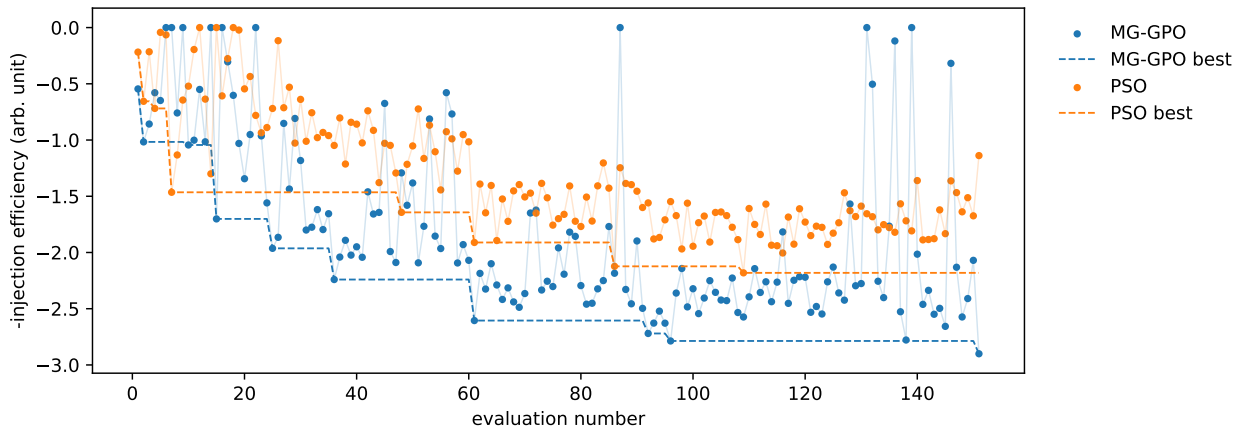

Figure 7: Comparison of the history of all evaluated solutions and the best-to-date solution during the injection efficiency optimization experiments with the MG-GPO and PSO algorithms, starting from the flat sextupole solution.

Since we started from the flat solution, the injection efficiency was quit low (here we normalized the injection efficiency to a certain injector beam intensity when the injector beam is delivered at a 2 Hz repetition rate, so the maximum value was not 1). As shown

in Figure 7, after 150 evalutions (the 1st run, 5 generations), the injection efficiency was improved substantially. Starting from the converged solution, the 2nd run converged within another 200 evaluations and achieved the optimal solution, as shown in Figure 8.

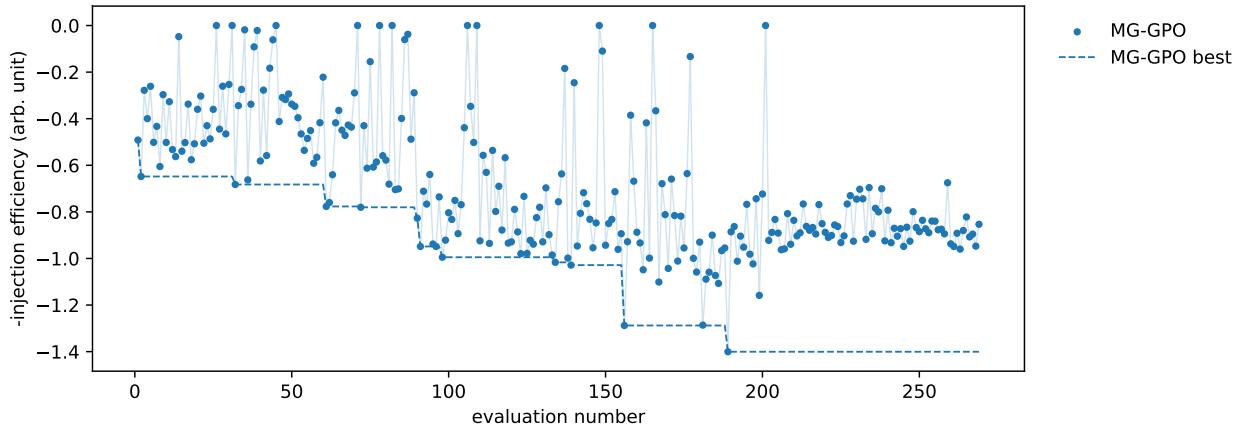

Figure 8: History of all evaluated solutions and the best-to-date solution during the injection efficiency second optimization experiment with the MG-GPO algorithm, starting from the best solution of the first run.

Here we employed PSO to benchmark MG-GPO. It can be clearly seen that the injection efficiency reached approximately -2.9 mA/min in the 1st run for MG-GPO, while PSO only reached -2.2 mA/min. Due to the limited AP time, we only used the better algorithm MG-GPO to continue optimizing in the 2nd run, which improved the injection efficiency from -0.5 mA/min to about -1.4 mA/min.

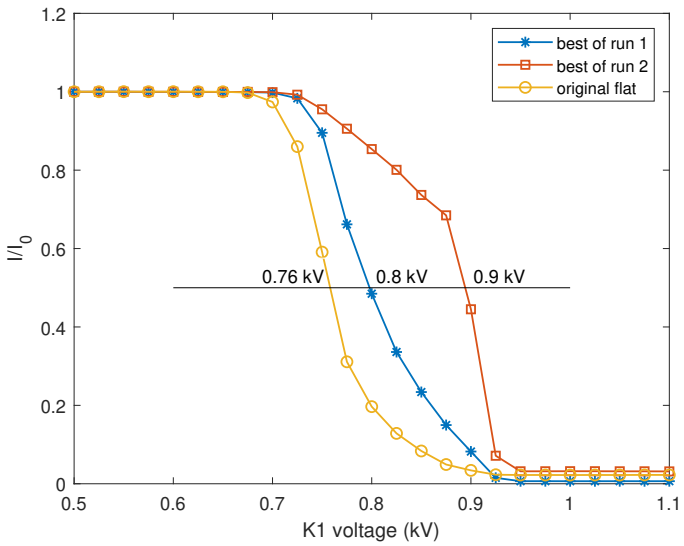

Figure 9: Comparison of measured dynamic aperture for the three cases

To verify the optimized solutions, we directly measured the dynamic aperture by kicking out a small stored beam current with an increasingly larger kick strength. The normalized

(by the initial value) beam currents vs. kicker voltage are shown in Figure 9 for 3 cases: the flat solution, the best solutions from the first run and the best solution from the second run. The corresponding kick voltage is increased from 0.76 kV (flat solution) to 0.9 kV (best solution).

3.3 An online optimization platform

An online optimization platform was developed during the beam-based optimization study to address the communication issue between the optimizer and the evaluator (the instance of the function to be optimized). With the platform, the optimizer and evaluator could communicate freely, which also makes the switch between the simulation objective and the experimental objective smooth and effortless. The fore-mentioned online simulation/experimental optimizations were mostly performed on the platform.

3.3.1 The communication problem in the online optimizations

A common situation in an online optimization is like the following. The evaluator is a Matlab script that reads and writes the PVs through Epics, when the optimizer is trying to evaluate a point, the evaluator writes the PVs of the knobs to the values given by the optimizer, then reads out the PV value of the objective and return it. There could be some 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:

$$\mathbf{Y} = \text{evaluate}(\mathbf{X}, \text{configs})$$

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. On the other hand, the optimizer is a Python script that imports several high quality optimization-related packages, that accepts an evaluate function and tries to optimize it. The optimizer usually has parameters like dimension of the problem to be optimized, the maximum iteration number, and the converging conditions. An abstract form of the optimizer can be described like this:

$$\text{optimize}(\text{evaluate}, \text{configs})$$

Since the evaluator and the optimizer are implemented in the different languages, to let them talk to each other, one has to make sure that the codes are located in the same computer, and figures out a way to call one from the other. Usually the evaluator is only accessible in the accelerator control room (ACR), so basically we have two choices: 1) clone the optimizer to the ACR computer and run it there. Unfortunately the administration permission is often needed to setup the environment for the optimizer on the ACR computer, and the setup also has a chance to break the configuration of the control system. 2) rewrite the optimizer in the same language as the evaluator. This solution is inefficient and error-prone, and not always feasible, due to the fact that the optimization packages support is different for each language, and it's unimaginable to rewrite the high-quality packages in the target language within any reasonable time. That's where the problem kicks in: how

to perform the online optimization task efficiently and effortlessly, given the conditions that we neither want to risk to break the control system just to get the optimizer setting up, nor rewrite a lot of stuff. The online optimization platform, Teeport, was our answer to that problem.

3.3.2 The architecture of the platform

The main idea behind the Task-based extensible embeddable platform for optimization and real-time testing (Teeport) platform is simple: inserting a middleware between the evaluator and the optimizer, which acts as a data normalizer and signal forwarder. Since the data flows through the middleware, one can add the control and monitor layers to the middleware, to make the online optimization process more controllable and visible.

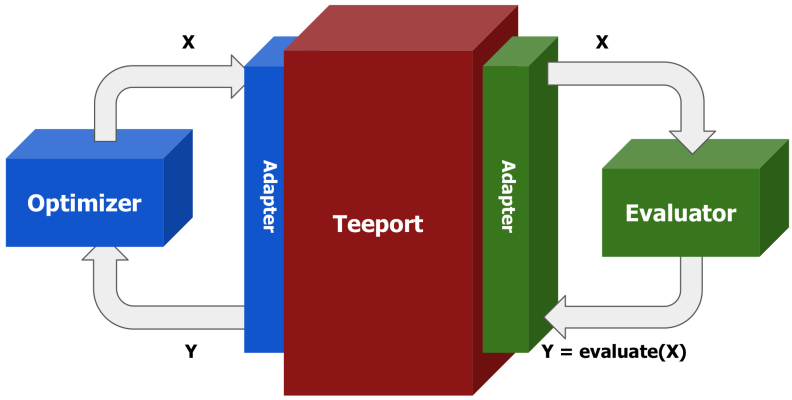

Figure 10: The architecture of Teeport

3.3.3 The features of the platform

Online optimization experiment With Teeport, we can completely decouple the evaluator and the optimizer, so performing an online experimental optimization is just as simple as doing a local optimization test run. The workflow to do an online optimization experiment is as following:

1. Code the evaluation script, and integrate it to Teeport with the `run_evaluator` API, Teeport will generate an id and assign to the evaluator
2. On the local computer, use the Teeport client for the language of the optimization algorithm, 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 rich set of features, such as monitoring the optimization progress, pausing/resuming the optimization,

terminating the optimization, and so on. Figure 11 shows a monitored single objective optimization task.

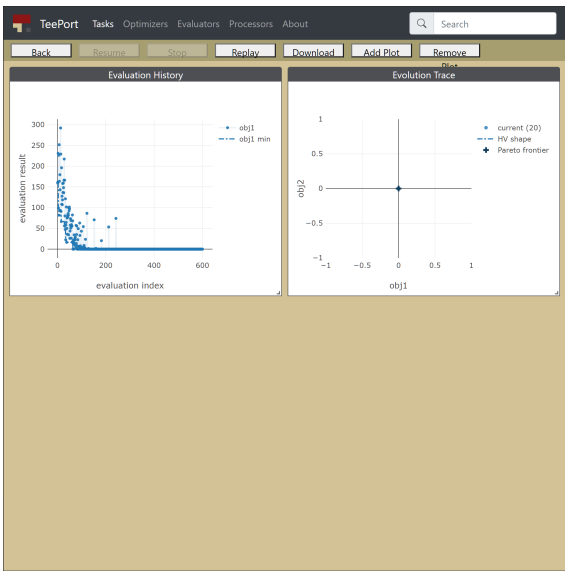

Figure 11: The history data of an online optimization experiment that was performed through Teeport

Fast switch between different optimization settings Since we can run multiple evaluators and optimizers on Teeport, as shown in Figure 12, 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 evaluate and/or optimize function, then do the optimization normally.

The only actions that 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 Figure 13.

Optimization performance comparison The data between the optimizer and the evaluator will flow through the Teeport platform, and Teeport can make use of them by monitoring and archiving the data for future reference. With this data monitor/archive capability, we can easily compare the performance of the to-be-tested algorithm on a series of testing problems, or compare the efficency of the different optimization algorithms against the same to-be-optimized problem.

The left plot in Figure 14 shows several online SPEAR3 beam loss rate experimental optimizations that we performed with Teeport. We used multiple optimization algorithms: RCDS, P-GPO and PSO to tackle the non-linear optimization problem. When the optimizations finished, simply select the interested runs and click the Compare button in the Teeport GUI, we would have the comparison result immediately, as shown in the right plot in Figure 14.

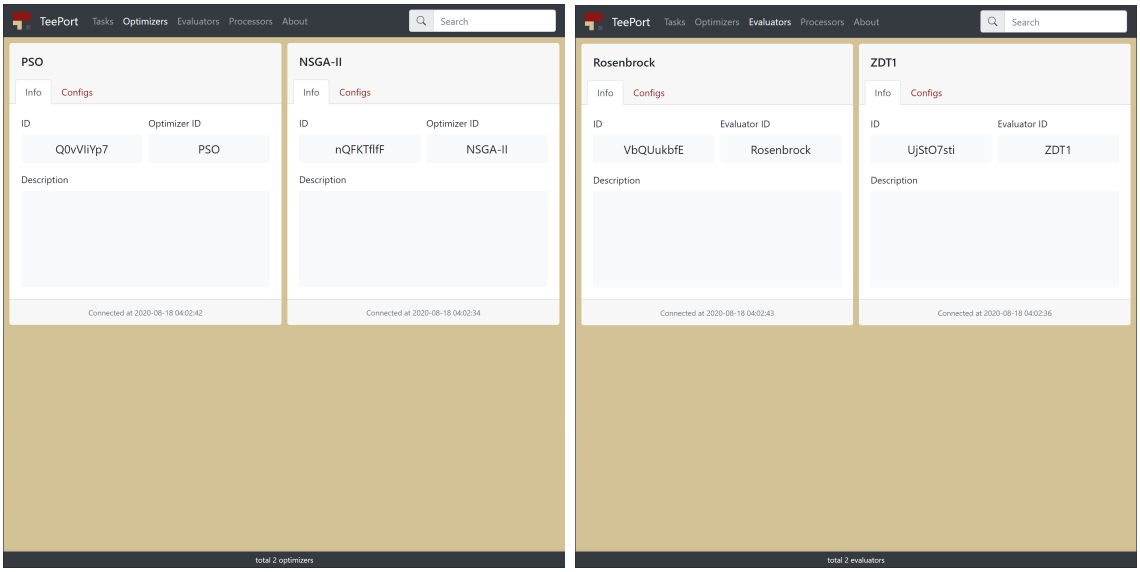

Figure 12: Multiple optimizers and evaluators running on the Teeport platform

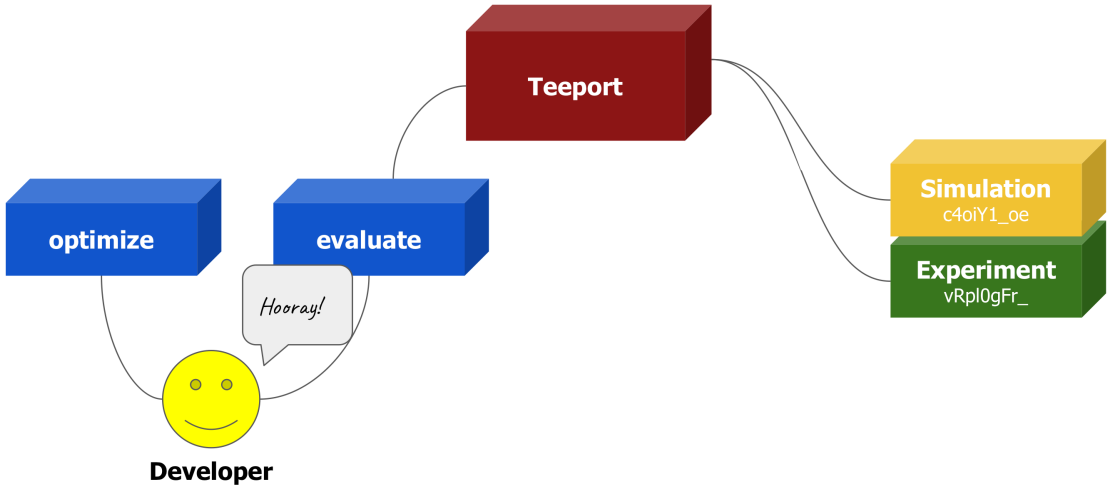

Figure 13: Fast switching between the simulation evaluator and the experimental evaluator.

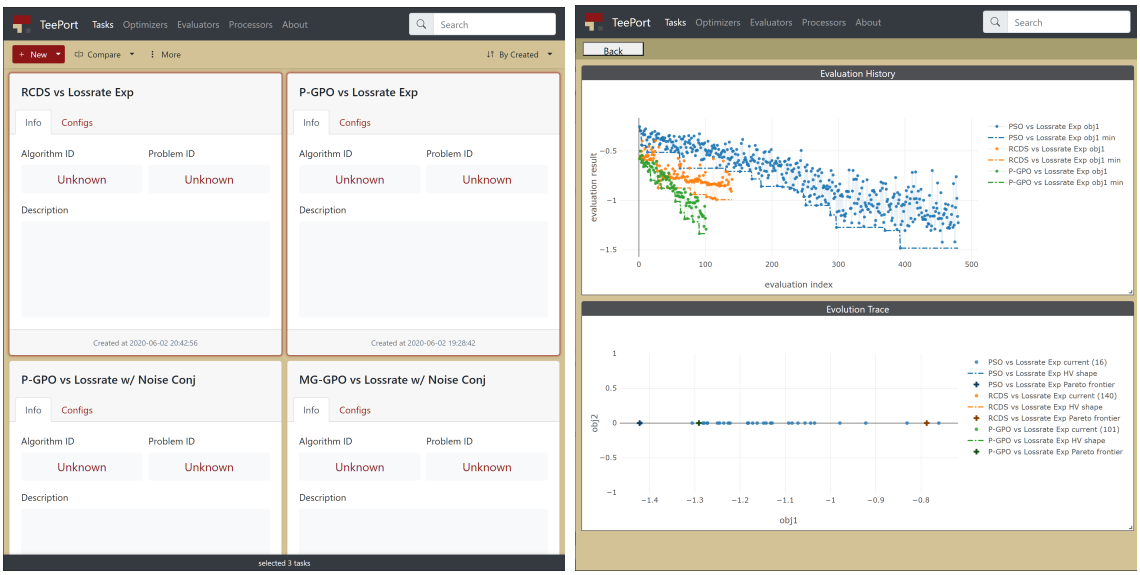

Figure 14: Comparison among the performance of the RCDS, P-GPO and PSO optimization algorithms against the SPEAR3 beam loss rate online optimization problem.

Through the comparison feature of Teeport, we could determine how efficient each optimization algorithm could be with respect to the specific problem 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 has full control over it: one can control when to run the optimizer, when to pause/resume, and when to terminate it. With this capability, we can benchmark any optimization algorithm effortlessly. One just pick an optimization configuration for the optimizer-evaluator pair, and tells Teeport to repeat the run for n times, then Teeport will automatically perform statistics on the results of the multiple runs and generate some meaningful plots (objective mean and variation, Pareto front distribution for multi-objective optimizations, etc) to demonstrate the algorithm performance.

3.3.4 The applications of the platform

SPEAR3 beam loss rate remote optimization The most important use of Teeport (also was the problem that lead to the creation of Teeport) is remote online optimization. Usually to perform an optimization against the beam-based online optimization such as the SPEAR3 beam loss rate optimization, one needs to clone the algorithm from his local computer to the computers in the ACR, and run the algorithm there. With Teeport, a remote optimization is totally possible and effortless. Here is what we did to use Teeport to optimize the SPEAR3 beam loss rate:

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

2. Get the corresponding local evaluate function through the Teeport client for matlab on the local laptop

3. Call the optimize function with the local evaluate function

The remote optimization results are shown in section 3.2.1.

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 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.

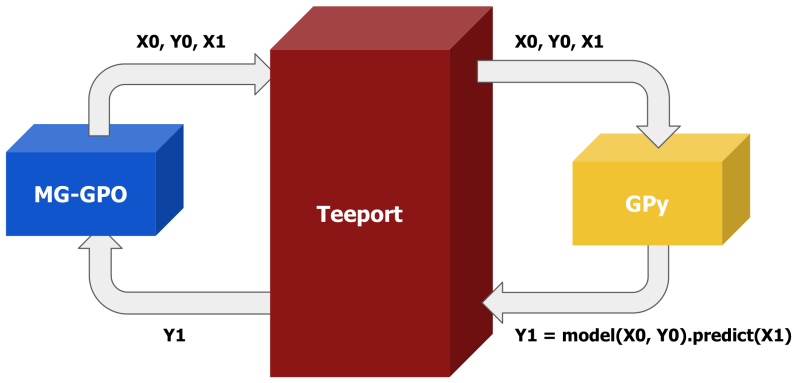

Figure 15: Teeport connects the GPy python package and the MG-GPO matlab optimization algorithm.

When we were developing MG-GPO, we couldn't find any decent matlab Guassian process packages, however the GP modeling part is the core of the whole algorithm. In python, there does exist a excellent GP package called GPy, so the problem was, how can we use GPy to handle the GP modeling part, while keeping all other logic in matlab? We solved this problem with Teeport by running GPy's GP modeling function as a processor on Teeport, then we applied the `use_processor` API to get a matlab version of the GP modeling function, and used it in our algorithm evolution loop. This idea is demonstrated in Figure 15.

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 their complex optimization problems. It's usually a good thing to have more options when dealing with one hard problem but it becomes not so good when the number of the optimization platforms are overwhelming, which is the case we're facing now. We have more than 500 optimization platforms, each of them equips a collection of optimization algorithms/test problems. However since the platforms were developed by different people in different fields, and usually targeted different pain points, so the usages could be quite diverse. This diversity could bring confusion and frustration to the users, especially when the user tries to use the algorithms/test problems from multiple platforms at the same

time. Teeport takes this diversity problem by providing a mininal set of integration APIs to effortlessly integrate the algorithms/test problems on Teeport first, then enable the users to use any of them through a unified API (`use_optimizer` and `use_evaluator`), therefore ease the pain.

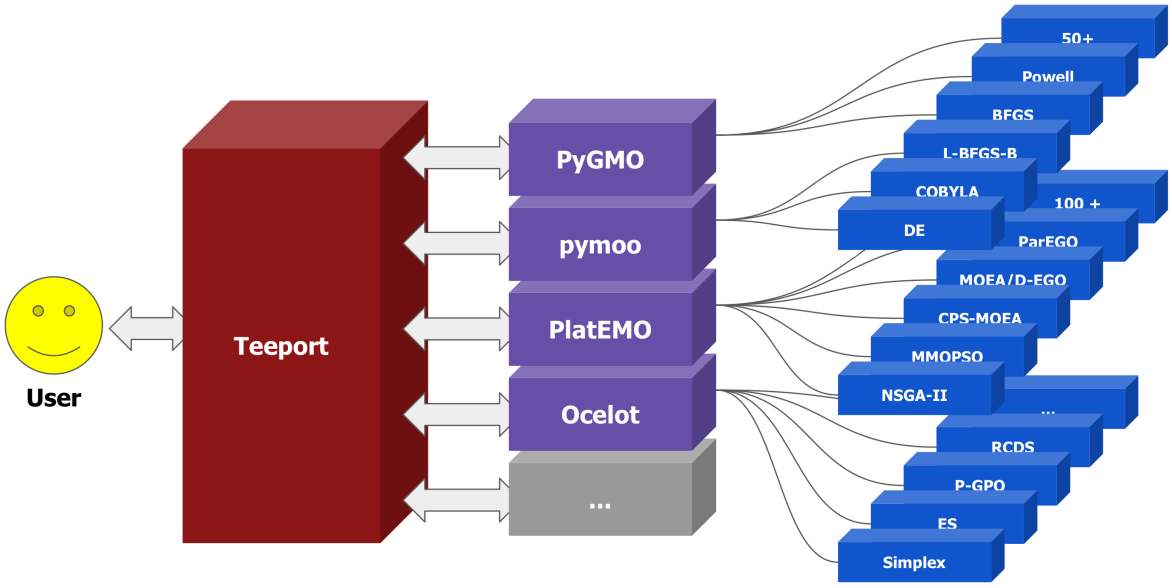

Figure 16: Teeport as a unified interface for the optimization algorithms.

We took this approach and already integrated lots of the optimization algorithms/test problem from various platforms, such as PyGMO, pymoo, PlatEMO and Ocelot, to Teeport. Figure 16 and Figure 17 demonstrate this approach.

4 Analysis of accelerator operation data with neural networks

4.1 The motivation

The performance of an accelerator often drifts with time due to variations in the environment. In some cases, the causes of the performance drifts can be identified and compensated. However, in many other cases, the root causes are not clearly identified and the drifts cannot be eliminated. Ideally, if we have access to the underlying physical model that could predict the performance of the ring with the given input parameters, then we can tune the free knobs to compensate the performance drift. However, due to the complexities of a storage ring, the thousands of parameters could affect each other in a very non-linear way, so a usable physical model is usually not possible to be derived/fit.

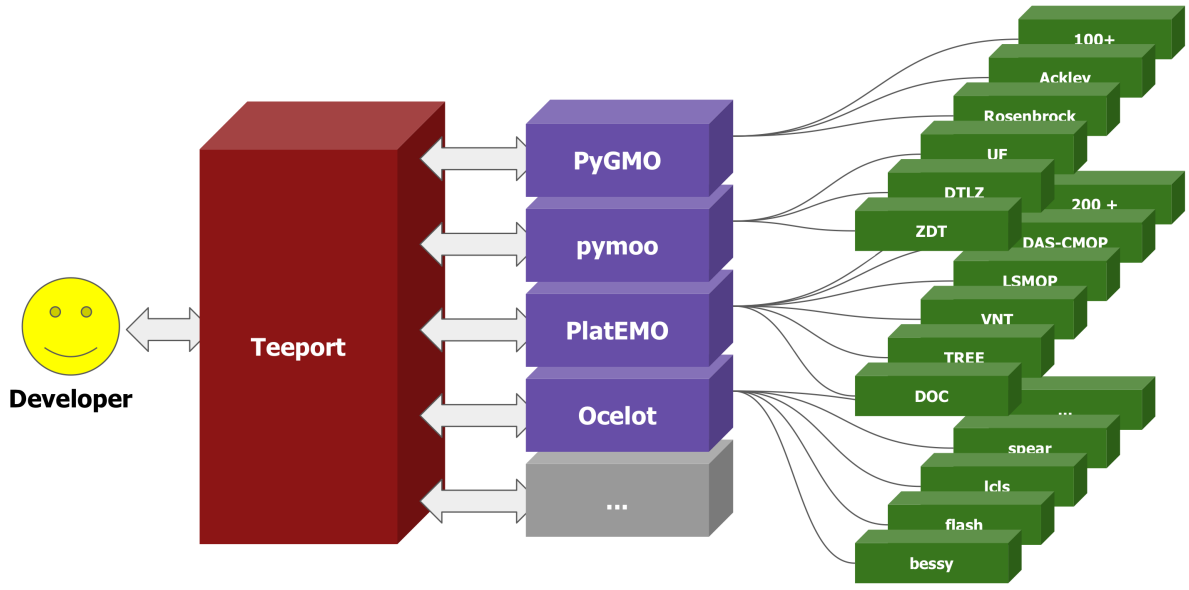

Figure 17: Teeport as a unified interface for the optimization test problems.

Nevertheless, if what we care about are just several performance indicators, such as the injection efficiency, then we could make use of the large amount of data that was produced and archived during the operation of the big machines to train a neural network model. Since a deep enough neural network model could emulate any function, with enough data, we can get an empirical model to analyze the operation data.

4.2 A brief introduction to the artificial neural network

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

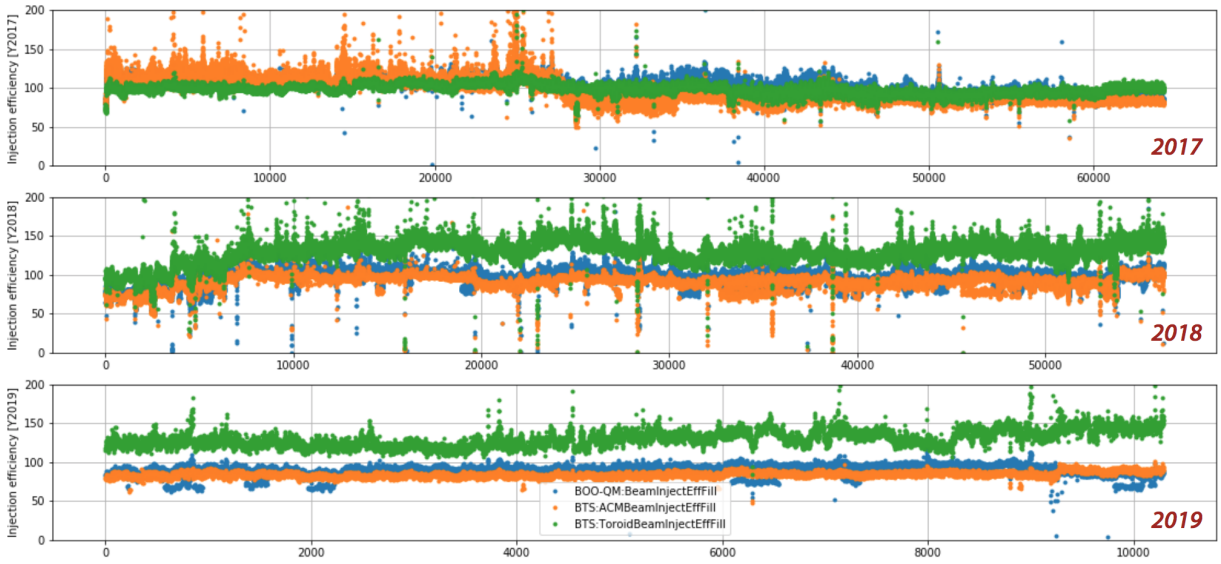

Figure 18: The injection efficiency operation data from three recent runs, including the 2017, 2018 runs and a fraction of the 2019 run. There are about 60,000 data points in each full run and about 10,000 data points from the 2019 run.

4.3 Training the neural network with the operation data

4.3.1 The problem

In SPEAR3, the BTS trajectory is controlled by a feedback that corrects the trajectory every 5 minutes during operation. While the trajectory is kept stable, the injection efficiency still varies over time. Injection efficiency is sensitive to many parameters that affect the injected beam and the storage ring, we picked the beam trajectory parameters (reading from the upstream BPMs), the downstream steering magnets, the air and ground temperatures and two of the insertions devices (ID).

4.3.2 Data preparation

We investigated operation history data of SPEAR3, which include injection efficiency, BPM readings, and steering magnet currents in BTS, insertion device gaps, and ambient air and ground temperatures. These parameters are archived at the different time intervals and there were occasionally missing data points. Therefore, some efforts were necessary to clean up the data and align the data points. Data from three recent runs were used in the analysis, including the 2017, 2018 runs and a fraction of the 2019 run. There are about 60,000 data points in each full run and about 10,000 data points from the 2019 run.

There are three injection efficiency measurements, differing in the monitor used to measure the average intensity of the injected beam (see Figure 1). Among them, the Booster Q-meter based data are the least noisy and were thus used as the target of NN model. There are still some unrealistic data points due to diagnostic issues. To ensure only valid data enter the analysis, we filtered out data points with injection efficiency above 200%, below 50%, or periodic large fluctuation (20%) in 5 minutes interval. About 3% of all data sets

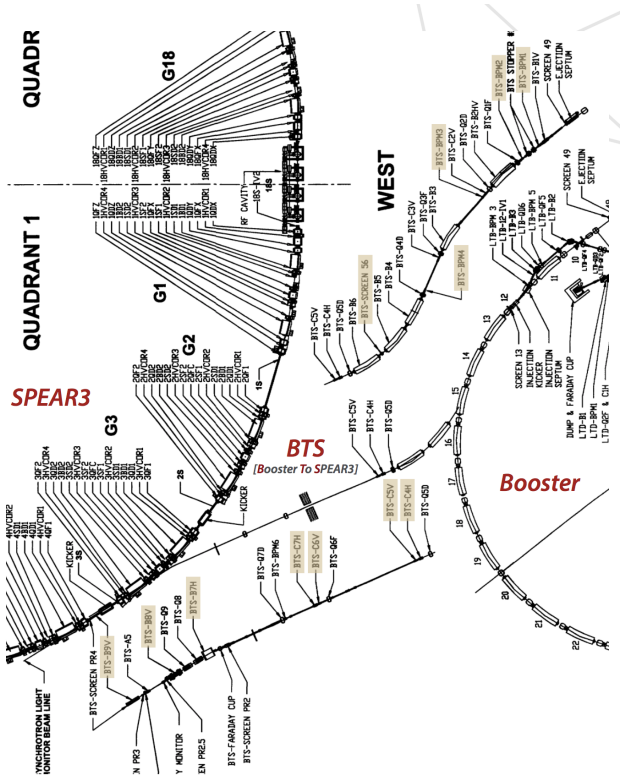

Figure 19: Booster to SPEAR3 layout and the parameters of interest.

were removed from the study.

The beam trajectory has very large shifts between different runs, with some BPM readings change by more than 10 mm, as shown in Fig. 2. Accordingly, the downstream steering magnets had to be tuned to compensate. The vertical orbit at two BPMs and the currents on two vertical steering magnets are shown in Figure 2 as examples.

Two of the SPEAR3 insertions devices (ID) can have particularly large effect on the injection efficiency, including the BL5.elliptical polarized unduator (EPU) and BL15 ID, an in-vacuum undulator (IVU). The EPU is a major source of perturbation to the dynamic aperture. The IVU gap changes the physical aperture and could affect the injection beam loss. The gap changes of two devices for the three runs are shown in Figure 3. The EPU phase is also included in the analysis.

4.3.3 The architecture of the neural network

The fully connected forward NN has a single output, which is the injection efficiency, and 22 input variables, which include the horizontal and vertical readings of 5 upstream BPMs in the BTS (10 variables), horizontal and vertical steering magnets in the downstream end of the BTS (7 variables), the temperatures (2), and the undulator gaps and EPU phase (3). The upstream BPMs determine the initial conditions of the trajectory, which, combined with the downstream steering magnets give the launching orbit of the injected beam into the storage ring.

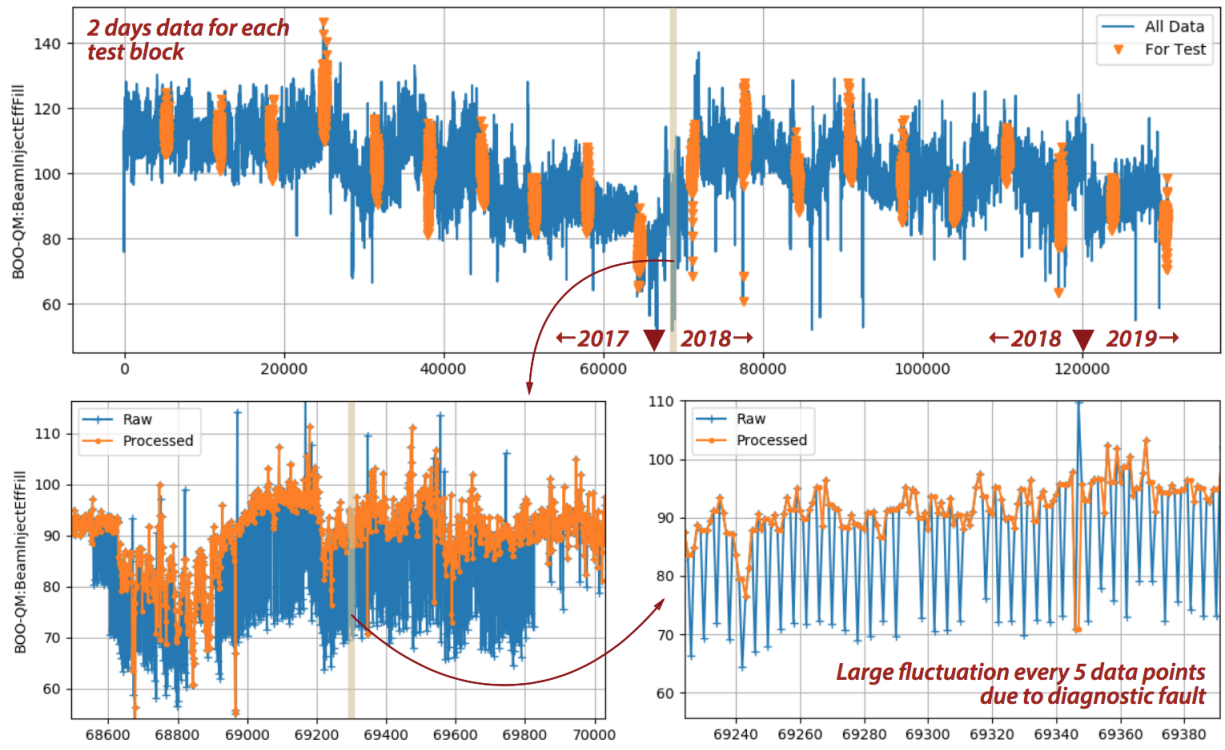

Figure 20: Data preparation. Note the large fluctuation every 5 data points due to diagnostic fault.

The NN model consists of 5 layers of networks. The first layer being a Recurrent NN (RNN). The 2nd through the 4th layers are Convolutional NN (CNN). The 5th layer is the output. There are a total of 5611 training parameters. A finite drop rate is adopted to improve the model reliability in some layers.

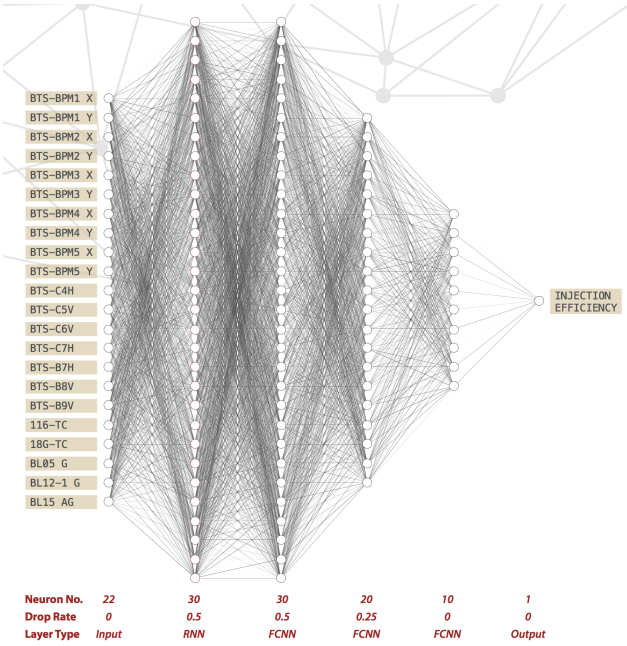

Figure 21: Architecture of the 5-layer neural network.

4.3.4 Training results

The trained NN model fits the data very well. The standard deviation of prediction errors for the validation data is 3.4%. For the test data the standard deviation is 4.4%. Figure 5 shows the comparison of the model predicted injection efficiency and the history data for the test data set.

4.4 Applications of the trained neural network

4.4.1 Analysis on the main factor of the injection efficiency drift

The goal is to find out how the injection efficiency depends on the environment variables, such as ID gaps and temperatures.

The approach we took is essentially to calculate the partial derivative of the output parameter of the NN with respect to the environment parameters. Figure 6 shows the change of the injection efficiency predicted by the model when an environment parameter is changed by 10% and all other parameters are fixed. The three curves represent the partial derivative for the air temperature, ground temperature, and BL5 EPU gap, respectively. The ground temperature causes the biggest variation to the output parameter, up to 30% in the injection efficiency.

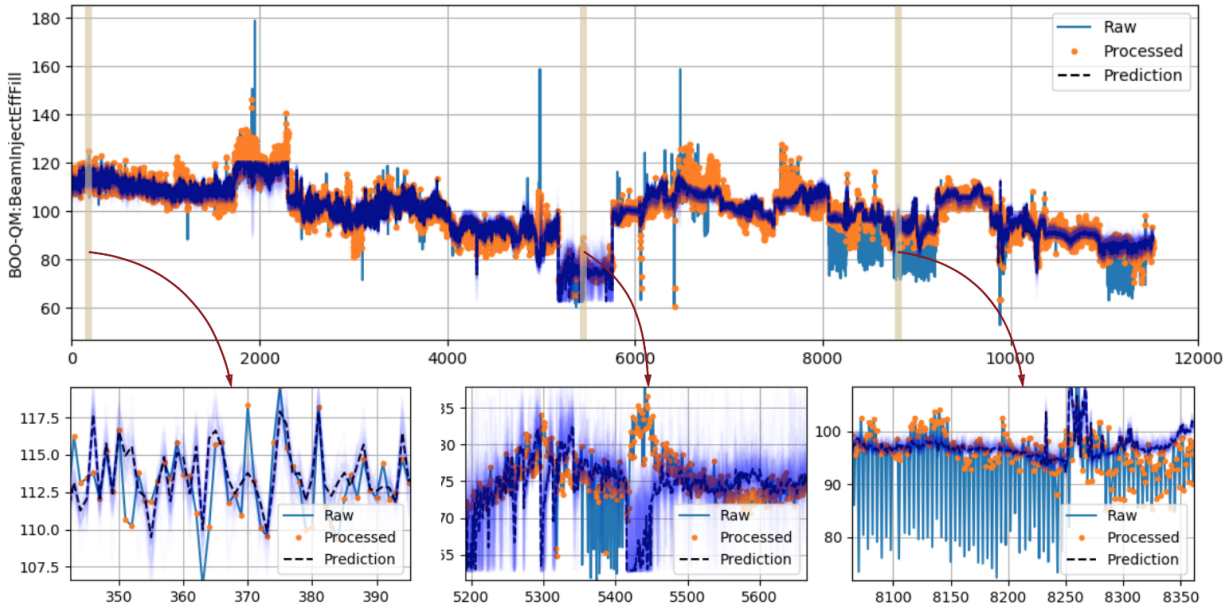

Figure 22: Training results.

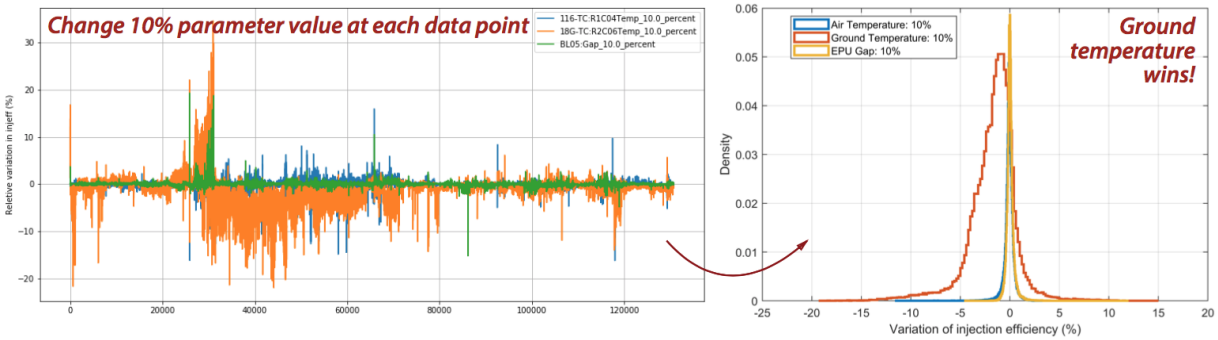

Figure 23: Main factor of the injection efficiency drift.

The environment variables typically vary slowly and their impact to the performance measure could be small compared to that of the other parameters. In addition, the performance measure parameter, such as the injection efficiency data, can be noisy. Therefore, it would be difficult to detect the small dependence on the environment parameters directly from the data.

4.4.2 Deduction of the ideal orbit under different circumstances

After the ground temperature is identified to be an environment parameter with a large impact, we studied the dependence of the ideal trajectory on the parameter. The ground temperature is first divided into 1°C zones. Within each zone, we used to the NN model to find the data points with the top 10% injection efficiency. The distribution of the corresponding trajectory readings on each BPM can then be used to determine the ideal trajectory.

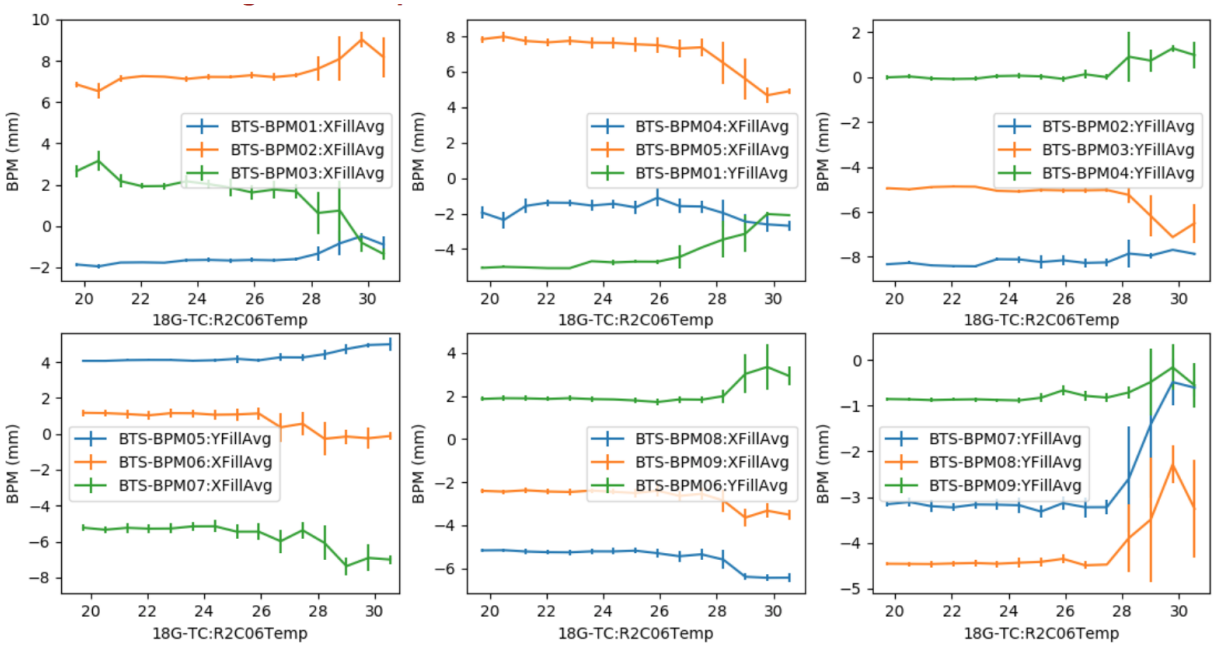

Figure 24: Ideal orbits for the ground temperatures between 20°C and 30°C.

5 Summary

The BES funded project, 'Beam-based optimization and machine learning for Synchrotrons', was successfully carried out during the 2-year period between August 2018 and August 2020. As planned, R&D work was conducted in the development of machine learning methods for synchrotron applications in three areas: accelerator design optimization, beam-based optimization, and analysis of accelerator operation data.

We developed a machine learning based stochastic optimization algorithm, the multi-generation Gaussian process optimizer (MG-GPO) [1]. The method combines the power of population based stochastic algorithms, such as NSGA-II [14] and MOPSO, with the

prediction capability of machine learning models afforded by Gaussian process regression. By using the ML models to pre-select trial solutions before sending them for evaluations, the MG-GPO algorithm substantially speeds up the convergence toward the global optima (the Pareto front), which is critical for design optimization problems that involve time consuming simulations. We demonstrated the fast convergence of MG-GPO with many commonly used test problems by comparing its performance against a number of advanced stochastic optimization algorithms [1].

We applied the MG-GPO method to a real-life lattice design optimization problem – optimization of nonlinear beam dynamics of a SPEAR3 emittance upgrade lattice [2]. It was shown to converge substantially faster than traditional methods NSGA-II and MOPSO.

We also proposed and implemented a neural network (NN)-based method for fast design optimization [3]. In this method deep learning techniques are used to build models that approximate the physical systems, which are subsequently used to generate new trial solutions. The algorithm update and refine the ML models with new trial solutions in an iterative manner. This method was also applied to the SPEAR3 nonlinear beam dynamics optimization problem and was demonstrated to have much faster convergence speed.

Beam-based optimization is critical for accelerator operation and commissioning. Development of efficient beam-based optimization methods would benefit existing accelerators as well as future machines such as the diffraction limited storage rings. A brief review of the online optimization algorithms is given in this report. The physics informed GP optimizer (P-GPO) has been previously demonstrated on tuning a free electron laser [4]. We adopted the method and applied it to storage ring applications. In the minimization of vertical emittance for the SPEAR3 storage ring, 13 skew quadrupoles are used as tuning knobs, while the Touschek loss rate is used as the surrogate objective for the vertical emittance. In both simulations and experiments, the P-GPO method was found to converge to the same level of minimum emittance, but with faster convergence speed than other methods [5].

As a stochastic algorithm, the MG-GPO method also has advantages in online optimization in its ability to locate the global optimum in a complex parameter space. We successfully applied the method to a number of storage ring tuning applications in experiments, including kicker bump matching, vertical emittance minimization, and dynamic aperture maximization. When compared to particle swarm optimization (PSO), another stochastic optimization algorithm suitable for online application, the MG-GPO method is found to have significantly faster convergence [6].

Because the control systems and programming environments on different machines may be different, online optimization algorithms implemented for one system may have difficulty applying to other systems. To solve this problem, we developed a general online optimization platform, Teeport, which decouples the algorithm implementation and the experimental systems by providing a universal middle layer that communicate between the optimizer and the evaluator. The Teeport platform can potentially become as a centralized service for advanced optimization applications. It has been extensively tested, both in simulation and in experiments.

Accelerator operation typically accumulates large amount of data about the status of the machines, which contain vast amount of information about the machines. However, usually such data are not exploited as it is difficult to analyze large data sets to extract hidden information. We developed a deep learning-based method to model the long-term time

evolution of accelerator performance [7]. The method was applied to analyze the injection efficiency variation over more than two years. The neural network model trained by the data can predict the performance with a good accuracy ($\pm 4.4\%$). By examining the dependence of the model predicted performance on the various environment variables, we are able to identify the latent variables that are correlated with injection efficiency.

Acknowledgment

This work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515 and FWP 2018-SLAC-100469 and by Computing Science, Office of Advanced Scientific Computing Research under FWP 2018-SLAC-100469ASCR.

References

[1] Xiaobiao Huang, Minghao Song, and Zhe Zhang. Multi-objective multi-generation gaussian process optimizer for design optimization, 2019.

[2] Minghao Song, Xiaobiao Huang, Linda Spentzouris, and Zhe Zhang. Storage ring nonlinear dynamics optimization with multi-objective multi-generation gaussian process optimizer. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 976:164273, 2020.

[3] Faya Wang, Minghao Song, Auralee Edelen, and Xiaobiao Huang. Machine learning for design optimization of storage ring nonlinear dynamics, 2019.

[4] J. Duris, D. Kennedy, A. Hanuka, J. Shtalenkova, A. Edelen, P. Baxevanis, A. Egger, T. Cope, M. McIntire, S. Ermon, and D. Ratner. Bayesian optimization of a free-electron laser. *Phys. Rev. Lett.*, 124:124801, Mar 2020.

[5] Adi Hanuka, J Duris, J Shtalenkova, D Kennedy, A Edelen, D Ratner, and X Huang. Online tuning and light source control using a physics-informed gaussian process adi. *arXiv preprint arXiv:1911.01538*, 2019.

[6] Zhe Zhang, Minghao Song, and Xiaobiao Huang. Online accelerator optimization with a machine learning-based stochastic algorithm. *Machine Learning: Science and Technology*, 2(1):015014, dec 2020.

[7] Z. Zhang F. Wang, X. Huang. Analyzing accelerator operation data with neural networks. In *Proceedings of NA-PAC2019*, Lansing, MI, USA, 2019.

[8] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964.

[9] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[10] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2(3). MIT press Cambridge, MA, 2006.

[11] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 1964.

[12] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[13] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Trans. Evol. Comp*, 6(2):182–197, April 2002.

[15] CA Coello Coello and Maximino Salazar Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE, 2002.

[16] Konstantinos Liagkouras and Konstantinos Metaxiotis. An elitist polynomial mutation operator for improved performance of moeas in computer networks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–5. IEEE, 2013.

[17] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[18] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, June 2000.

[19] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[20] Qiuzhen Lin, Jianqiang Li, Zhihua Du, Jianyong Chen, and Zhong Ming. A novel multiobjective particle swarm optimization with multiple search strategies. *European Journal of Operational Research*, 247(3):732–744, 2015.

[21] Heiner Zille, Hisao Ishibuchi, Sanaz Mostaghim, and Yusuke Nojima. Weighted optimization framework for large-scale multi-objective optimization. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, page 83–84, New York, NY, USA, 2016. Association for Computing Machinery.

[22] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[23] Zhun Fan, Yi Fang, Wenji Li, Xinye Cai, Caimin Wei, and Erik Goodman. MOEA/D with angle-based constrained dominance principle for constrained multi-objective optimization problems. *Applied Soft Computing*, 74:621–633, 2019.

[24] A. Hannun A. Maas and A. Ng. Rectifier nonlinearities improve neural network acoustic models. *Proc. ICML. 30 (1)*, June 2013.

[25] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[26] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[27] 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*, 726:77 – 83, 2013.

[28] Xiaobiao Huang and James Safranek. Online optimization of storage ring nonlinear beam dynamics. *Phys. Rev. ST Accel. Beams*, 18:084001, Aug 2015.

[29] H.-F. Ji, Y. Jiao, S. Wang, D.-H. Ji, C.-H. Yu, Y. Zhang, and X. Huang. A simplex method for function minimization. *Chinese Physics C*, 39(12):127006, 2015.

[30] S.M. Liuzzo, N. Carmignani, L. Farvacque, B. Nash, T. Perron, P. Raimondi, R. Versteegen, and S. M. White. RCDS optimizations for the ESRF storage ring. In *Proceedings of IPAC2016*, pages 3420–3422, Busan, Korea, 2016.

[31] I.P.S. Martin, M. Apollonio, and R. Bartolini. Online suppression of the sextupole resonance driving terms in the DIAMOND storage ring. In *Proceedings of IPAC2016*, pages 3381–3383, Busan, Korea, 2016.

[32] G.M. Wang, W.X. Cheng, X. Yang, J. Choi, and T. Shaftan. Storage ring injection kickers alignment optimization in nsls-ii. In *Proceedings of IPAC2017*, pages 4683–4685, Copenhagen, Denmark, 2017.

[33] T. Pulampong, P. Klysubun, S. Kongtawong, S Krainara, and S. Sudmuang. Online optimization applications at sps. In *Proceedings of IPAC2017*, pages 4086–4088, Copenhagen, Denmark, 2017.

[34] W. F. Bergan, A. C. Bartnik, I. V. Bazarov, H. He, D. L. Rubin, and J. P. Sethna. Using sloppy models for constrained emittance minimization at the cornell electron storage ring (cesr). In *Proceedings of IPAC2017*, pages 2418–2420, Copenhagen, Denmark, 2017.

[35] J. Wu et al. RECENT ON-LINE TAPER OPTIMIZATION ON LCLS. In *Proceedings of FEL2017*, pages 229–234, Santa Fe, NM, USA, 2017.

[36] D. K. Olsson. Online optimisation of the MAX-IV 3 GeV ring dynamic aperture. In *Proceedings of IPAC2018*, pages 2281–2283, Vancouver, BC, Canada, 2018.

[37] X. Yang, G. Ganetis, Y. Hidaka, T.V. Shaftan, V.V. Smaluk, G.M. Wang, L.-H. Yu, and P. Zuhoski. Online optimization of NSLS-II dynamic aperture and injection transient. In *Proceedings of IPAC2019*, Melbourne, Australia, 2019.

[38] Xiaobiao Huang. Robust simplex algorithm for online optimization. *Phys. Rev. Accel. Beams*, 21:104601, Oct 2018.

[39] K. Tian, J. Safranek, and Y. Yan. Machine based optimization using genetic algorithms in a storage ring. *Phys. Rev. ST Accel. Beams*, 17:020703, Feb 2014.

[40] Xiaobiao Huang. Development and application of online optimization algorithms. In *Proceedings of NAPAC2016*, pages 1287–1291, Chicago, IL, 2016.

[41] Xiaobiao Huang and James Safranek. Online optimization of storage ring nonlinear beam dynamics. *Phys. Rev. ST Accel. Beams*, 18:084001, Aug 2015.