Multiobjective Bayesian optimization for online accelerator tuning

Ryan Roussel[*]
Department of Physics, University of Chicago, Chicago, Illinois 60637, USA

Adi Hanuka and Auralee Edelen
SLAC National Laboratory, Menlo Park, California 94025, USA

(Received 19 October 2020; accepted 30 April 2021; published 2 June 2021)

Particle accelerators require constant tuning during operation to meet beam quality, total charge and particle energy requirements for use in a wide variety of physics, chemistry and biology experiments. Maximizing the performance of an accelerator facility often necessitates multiobjective optimization, where operators must balance trade-offs between multiple competing objectives simultaneously, often using limited, temporally expensive beam observations. Usually, accelerator optimization problems are solved off-line, prior to actual operation, with advanced beam line simulations and parallelized optimization methods (NSGA-II, swarm optimization). Unfortunately, it is not feasible to use these methods for online multiobjective optimization, since beam measurements can only be done in a serial fashion, and these optimization methods require a large number of measurements to converge to a useful solution. Here, we introduce a multiobjective Bayesian optimization scheme, which finds the full Pareto front of an accelerator optimization problem efficiently in a serialized manner and is thus a critical step towards practical online multiobjective optimization in accelerators. This method uses a set of Gaussian process surrogate models, along with a multiobjective acquisition function, to reduce the number of observations needed to converge by at least an order of magnitude over current methods. We demonstrate how this method can be modified to specifically solve optimization challenges posed by the tuning of accelerators. This includes the addition of optimization constraints, objective preferences and costs related to changing accelerator parameters.

DOI: 10.1103/PhysRevAccelBeams.24.062801

I. INTRODUCTION

Accelerator optimization during operation (i.e., "online tuning") is a tedious but often necessary part of any experimental facility's operation. This severely limits beam time that is available to experimenters (i.e. "users") as hours of tuning time is required to diagnose issues and make corrections. It would be beneficial to have an automated or semiautomated algorithm take care of normal beam line tuning, reducing downtime while also allowing human experts to tackle more challenging operational problems.

As a response to this, a number of algorithms have been used to optimize current accelerator facilities. Gradient-based algorithms, such as robust conjugate direction search (RCDS) [1], have been used successfully in the past to optimize beam parameters. Heuristic methods such as the Nelder-Mead simplex [2] algorithm can also be used to optimize black box problems, when functional derivative information is not easily accessible. More recently, BOBYQA has also been used to optimize accelerators [3–5] by fitting data to a second order model in a local trust region, which accounts for noisy observations. However, these methods can struggle to handle problems with many local extrema and must be restarted several times to ensure a global extrema is found.

Bayesian optimization [6,7] provides a framework for global optimization, while significantly reducing the number of physical observations needed to find solutions, while also taking into account observational noise. In this method, physical observations are combined with a kernel, which describes the overall functional behavior, to create what is commonly referred to as a Gaussian process (GP) model. The GP model is able to predict the value and uncertainty of a target function [8]. Using this prediction, an optimizer can then choose input points that are likely to be ideal, before a physical measurement is made. Recently, this method was successfully used to efficiently optimize single objective problems at LCLS and SPEAR3, with a lower number of observations needed than Nelder-Mead simplex and RCDS algorithms [9–12].

These algorithms have been used to optimize a single beam characteristic, while in reality, accelerator tuning generally seeks to simultaneously optimize multiple facets of the beam ("objectives") at a time. This presents an issue,

[*]rroussel@uchicago.edu

as individual beam characteristics can often be optimized only at the expense of others. For example, it is difficult to simultaneously minimize both the bunch length and the transverse emittance of a low energy electron beam in a photoinjector due to space charge forces [13]. Solving multiobjective problems during simulated beam line design has become a relatively simple task, given the development of evolutionary algorithms and the availability of computing clusters, which can run a large number of particle physics simulations in parallel [13,14].

By contrast, solving multiobjective optimization problems during accelerator operations presents an extremely difficult challenge due to several factors. Most notably, accelerator operators can only evaluate or observe the objectives for a single set of input parameters at any time (referred to as "serialized observations"). This makes the use of evolutionary algorithms practically infeasible, due to the number of observations needed to converge to a solution if used in a serialized manner. Furthermore, an online optimization algorithm must be able to keep track of constraining functions, as well as account for relative objective preferences specified by the operators. Finally, the optimization algorithm should take into account the costs of changing accelerator parameters during optimization.

In this paper, we use the recent development of multibjective Bayesian optimization (MOBO) [15] to extend online Bayesian optimization of accelerators to solving multiple objective problems using serialized observations. We also demonstrate how to extend this algorithm to solve specific practical challenges associated with online accelerator optimization.

II. ONLINE MULTIOBJECTIVE OPTIMIZATION OF ACCELERATORS

We start with a brief explanation of techniques currently used to solve multiobjective problems. This serves to motivate use of the MOBO algorithm for online accelerator optimization.

A simplistic way of solving a multiobjective optimization problem is to explicitly weight each objective relative to one another *a priori*, and add up the weighted objective values, in a process known as scalarization [16,17]. This optimization method results in a solution found only for a single set of weights (trade-offs), and must be repeated from scratch to explore different trade-offs between objectives. Mapping out the full set of optimal trade-offs in an accelerator is highly desirable, particularly at facilities which must accommodate a variety of working points, or when operators wish to benchmark beam simulation results to experimental realities. In this case, repeating the optimization for a discrete set of weights is relatively inefficient, even when Bayesian optimization methods are used [18].

By contrast, multiobjective optimization algorithms attempt to find a set of points, known as the Pareto front $\mathcal{P}$, that optimally balances the trade-offs between multiple competing objectives simultaneously (see Fig. 1). The Pareto front is defined as the set of "nondominated" points in objective space with respect to a reference point $\mathbf{r}$ (which itself must be dominated by every other observed point). Points are nondominated if they are as good as any other observed point for every objective and are better than any other point for at least one objective. The hypervolume metric $\mathcal{H}$ [19], shown in Fig. 1, is often used to characterize the quality of the Pareto front, where a larger volume corresponds to a better solution set. Adding observations to the current dataset, which dominate over points in the current Pareto front, leads to an expansion of the hypervolume, characterized by the hypervolume improvement $\mathcal{H}_I$ (see Fig. 1). Algorithms generally stop once this metric converges to a maximum as new observations are added, signifying that a correct approximation to the true Pareto front has been reached.

A popular set of techniques, known as evolutionary algorithms, is generally used to solve multiobjective problems. These algorithms, such as nondominated sorting genetic algorithm II (NSGA-II) [20] or multiobjective particle swarm optimization [21–23], are based on the generation of a large collection of candidate solutions, which are then observed via simulation or experiment, usually in a parallelized manner. The results from each observation are then sorted into nondominated and dominated subsets. The nondominated subset of candidate solutions is used to produce the next "generation" of candidate solutions using a stochastic

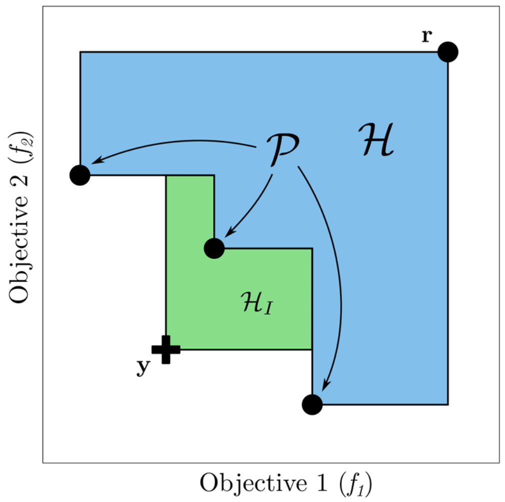

FIG. 1. Cartoon of multiobjective optimization where each objective is to be minimized. Multiobjective optimization attempts to find a set of points known as the Pareto front $\mathcal{P}$ that dominate over a reference point $\mathbf{r}$ and any other observed points in objective space. The Pareto front hypervolume $\mathcal{H}$ (shown in blue) is the axis-aligned volume enclosed by the Pareto front and a reference point $\mathbf{r}$. Making a new observation $\mathbf{y}$, which dominates over points in the current Pareto front, leads to an increase in hypervolume (shown in green), referred to as the hypervolume improvement $\mathcal{H}_I$.

heuristic, which are then reevaluated. The process is repeated over a number of generations until the nondominated set of observations converges to a stationary Pareto front or the hypervolume has converged to a maximum value. It has been shown that these methods are well suited for solving accelerator design optimization problems [13,14].

Recently, surrogate assisted evolutionary algorithms have been developed which combine evolutionary algorithms with surrogate models of the objective functions. A fast executing surrogate model (possibly a Gaussian process [24] or a neural network [25]) is used to predict if a candidate solution generated by the evolutionary algorithm will be Pareto optimal before an observation is made. This significantly speeds up convergence over basic evolutionary algorithms by eliminating candidate observations that are not predicted to improve the Pareto frontier. The surrogate model is retrained after observations are made, thus improving the model's accuracy as the optimization progresses. This further improves the convergence speed of the algorithm, as the surrogate model gains knowledge about the objective functions and can more adequately identify which candidates will be nondominated.

However, these algorithms still are not ideal for online accelerator optimization. Evolutionary algorithms use the binary classification metric of Pareto dominance to identify which candidate points to observe. As a result, this metric does not guarantee optimal expansion of the Pareto front, as it does not consider the relative hypervolume improvement (see Fig. 1) of individuals in the nondominated subset of candidates. This reduces the observation efficiency of these algorithms, making them impractical for use in serialized settings.

The multiobjective Bayesian optimization (MOBO) algorithm [15] achieves maximum efficiency by using an explicit calculation of the hypervolume improvement to determine the next candidate for observation. Each objective is modeled using a Gaussian process (GP) surrogate model which can be used to predict the hypervolume improvement as a function of input parameters. As a result, MOBO can determine a single point that maximally increases the Pareto front hypervolume at every step in a serialized manner, making it ideal for online accelerator optimization.

III. MULTIOBJECTIVE BAYESIAN OPTIMIZATION

We begin the explanation of MOBO by first describing single-objective Bayesian optimization. To maintain consistency with reference texts, we assume that single objective optimization aims to maximize the objective, while in the multiobjective case we wish to minimize each objective. Simply multiplying any objective and its corresponding observed values by -1 allows us to switch the optimization goal from maximization to minimization or vice versa.

A. Single objective Bayesian optimization

The goal of our optimization strategy is to maximize the function $f(\mathbf{x})$ using as few observations of f as possible inside the input domain $\mathbf{x} \in \mathcal{X}$. Bayesian optimization uses two components to achieve this.

The first component is the GP surrogate model. A "surrogate model" in this case refers to a computationally cheap-to-evaluate predictive model, which acts as a stand-in for any computationally expensive or difficult to measure system, and can be either a local or a global model. The GP surrogate produces both the predicted mean $\mu(\mathbf{x})$ and the corresponding uncertainty $\sigma(\mathbf{x})$ of a random function value at the point $\mathbf{x}$: $f(\mathbf{x}) \sim \mathcal{GP}[\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')]$, where $k(\mathbf{x}, \mathbf{x}')$ is the covariance (kernel) function. The kernel function represents the covariance between function values f at two points in input space. For example, we can specify how rapidly $f(\mathbf{x})$ changes as a function of the distance between two points $\mathbf{x}$ and $\mathbf{x}'$ by defining a length scale λ hyperparameter. Given a set of observations $\mathcal{D}_N = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ where y_i is a noise corrupted measurement of $f(\mathbf{x}_i)$, $y_i = f(\mathbf{x}_i) + \varepsilon$, we can then calculate the predicted mean and variance anywhere in input space. Details on creating and using a GP surrogate model can be found in the Appendix A and in Ref. [8].

Algorithm 1. Bayesian optimization (BO).

```
input : input domain $\mathcal{X}$, dataset $\mathcal{D}$, GP prior $\mathcal{M}_0 = \mathcal{GP}(\mu_0, k_0)$,
        acquisition function $\alpha$, noise $\varepsilon$
for $i = 1, 2, 3, \ldots$ do
    $x_i \leftarrow \mathrm{argmin}_{x \in \mathcal{X}} \alpha(x|\mathcal{M}_{i-1})$;    // optimize $\alpha$
    $y_i \leftarrow f(x_i) + \varepsilon$;                                  // do observation
    $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1}|(x_i, y_i)$;                       // update model
end
```

The second component of Bayesian optimization is an acquisition function $\alpha(\mathbf{x})$ which codifies the value gained from potential observation points, based on mean and uncertainty predictions from the GP model. To find the global optimum efficiently we wish to search regions of input space that either take advantage of previously observed extremum points (exploitation) or have a large amount of uncertainty (exploration). We choose our acquisition function such that it is maximized at the point of most interest, one that properly balances the trade-off between exploration and exploitation. We can then use a standard single-objective optimization algorithm to optimize the cheap-to-evaluate acquisition function in order to propose the next observation location, instead of directly optimizing the expensive-to-evaluate physical experiment. Two popular acquisition functions for Bayesian optimization are expected improvement (EI) and upper confidence bound (UCB) [26,27].

Expected improvement calculates the average improvement of a point over the best observed function value f_{best}

$$\begin{aligned}\alpha_{\mathrm{EI}}(\mathbf{x}) &= \mathbb{E}\{\max[f_{\mathrm{best}} - f(\mathbf{x}), 0]\} \\ &= [f_{\mathrm{best}} - \mu(\mathbf{x})]\Phi\left(\frac{f_{\mathrm{best}} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \\ &\quad + \sigma(\mathbf{x})\phi\left(\frac{(f_{\mathrm{best}} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right), \qquad (1)\end{aligned}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the probability distribution function and cumulative distribution function of a Gaussian distribution respectively.

Upper confidence bound calculates an optimistic function value, weighted by an optimization parameter β, which explicitly specifies the trade-off between exploration and exploitation:

$$\alpha_{\mathrm{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\beta}\sigma(\mathbf{x}). \qquad (2)$$

For $\beta \ll 1$ UCB prioritizes exploitation; for $\beta \gg 1$ UCB prioritizes exploration. This parameter can be increased as the optimization progresses, to prevent the optimizer from getting stuck in a local optimum [27]. Combining both the GP surrogate model and the acquisition function we can now perform Bayesian optimization using the method presented in Algorithm 1.

B. Incorporating multiple objectives

We now extend this methodology, following [15], to incorporate P objectives $\mathbf{f} = \{f_1, f_2, \ldots, f_P\}$. We assume that the objectives share the same input domain $\mathbf{x} \in \mathcal{X}$ and are all observed for each input point such that $\mathcal{D}$ now contains the set of N observations of each objective, $\{(\mathbf{x}_1, \mathbf{f}_1), (\mathbf{x}_2, \mathbf{f}_2), \ldots, (\mathbf{x}_N, \mathbf{f}_N)\}$. Each objective is then modeled as an independent GP such that $f_p(\mathbf{x}) \sim \mathcal{GP}_p[\mu_p(\mathbf{x}), k_p(\mathbf{x}, \mathbf{x}')]$, where $p = 1, 2, \ldots, P$, as seen in Fig. 2(a). Each GP has its own independent kernel which is trained on corresponding observations by maximizing the marginal log likelihood [8].

In order to proceed with optimization we must construct a scalar acquisition function $\alpha: \mathcal{X} \to \mathbb{R}$ that finds points which are likely to maximally increase the Pareto frontier hypervolume. We consider two acquisition functions that have been developed for this purpose.

The first multiobjective acquisition function, expected hypervolume improvement (EHVI) [15] seen in Fig. 2(b), is analogous to single-objective expected improvement. This acquisition function calculates the average increase in hypervolume using the posterior probability distribution of each objective function from the surrogate model. The EHVI acquisition function is formally defined as

$$\alpha_{\mathrm{EHVI}}(\mu, \sigma, \mathcal{P}, \mathbf{r}) \coloneqq \int_{\mathbb{R}^P} \mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{r}) \cdot \xi_{\mu,\sigma}(\mathbf{y}) d\mathbf{y}, \qquad (3)$$

where $\mathcal{P}$ is the current set of Pareto optimal points, $\mathbf{r}$ is the reference point, $\mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{r})$ is the hypervolume improvement from an observed point $\mathbf{y}$ in objective space, and $\xi_{\mu,\sigma}$ is the multivariate Gaussian probability distribution function defined by the GP predicted mean μ and standard deviation σ for each objective. The hypervolume improvement is defined by the exclusive hypervolume contribution to the current Pareto front by adding $\mathbf{y}$ to the Pareto set, as seen in the green region in Fig. 2(b).

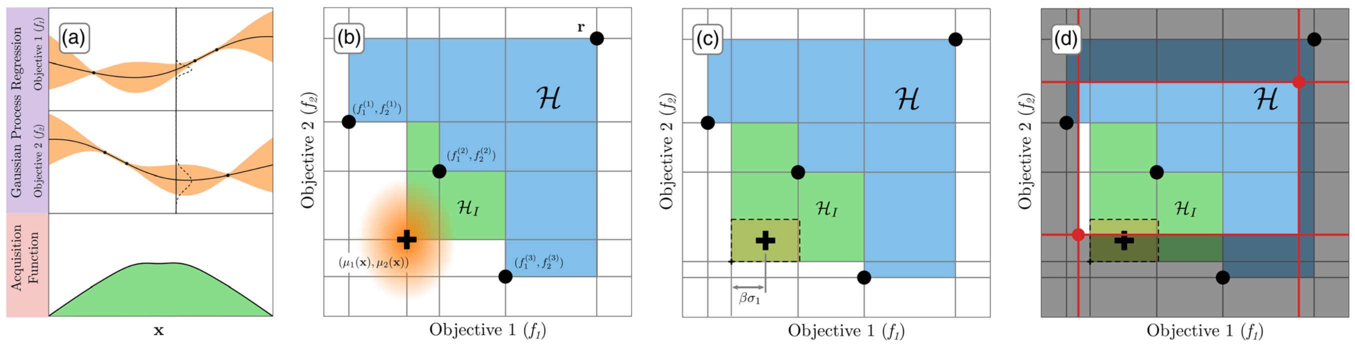

FIG. 2. Cartoons showing hypervolume improvement metrics used for MOBO. Blue regions denote the current Pareto front hypervolume defined by a reference point (upper right) and three observed points. Green regions denote the hypervolume improvement after adding an observation. (a) Each objective is modeled by an independent GP that predicts the function value and an uncertainty, which is used by the multiobjective acquisition function to determine the next observation. (b) Expected hypervolume improvement (EHVI) where the distribution of predicted objective values is given by a probability distribution (orange shading) centered at the black cross. (c) Upper confidence bound hypervolume improvement (UCB-HVI) where the hypervolume improvement is determined by an optimistic view of the predicted objective values. (d) Truncated UCB-HVI where we only consider hypervolume contributions within the unshaded subspace.

The reference point $\mathbf{r}$ is chosen such that all expected observations $\mathbf{f}$ dominate the reference point. Any predicted points in objective space that do not satisfy this condition will not contribute to the hypervolume improvement, and are thus never chosen as observation candidates. It is also important to note for optimization purposes that the prior mean for each objective GP is set to the corresponding component of the reference point. This ensures that in regions of input space where observations have not been made, the mean of the GP model returns to the reference point. As a result, the acquisition function will never predict that unobserved regions of input space contribute to the hypervolume.

The integral in Eq. (3) can become computationally expensive to calculate, as the objective space must be decomposed into cells for which the integral has an analytical form. This is computationally expensive for high dimensional objective spaces, as a naive decomposition algorithm scales as $\mathcal{O}(N^P)$, where N is the number of points on the Pareto front. Work in the multiobjective Bayesian optimization field has produced efficient methods for objective space decomposition, which improves scaling to $\mathcal{O}(N\log N)$ for 2–3 dimensions and $\mathcal{O}(2^{P-1}N^{P/2})$ scaling when $P \geq 4$ [28]. Regardless, this computational complexity results in a significant increase in computation time when used to maximize the acquisition function.

The second multiobjective acquisition function, upper confidence bound hypervolume improvement (UCB-HVI) [15], is similar to the UCB acquisition function for single objective optimization. This acquisition function describes an optimistic view of the hypervolume improvement given the surrogate model prediction,

$$\alpha_{\text{UCB-HVI}}(\mu, \sigma, \mathcal{P}, \mathbf{r}, \beta) \coloneqq \mathcal{H}_I(\mathcal{P}, \mu - \sqrt{\beta}\sigma, \mathbf{r}). \qquad (4)$$

The simplicity of this acquisition function reduces computation time relative to EHVI, especially in high-dimensional objective spaces. This is due to the development of advanced hypervolume computation strategies, such as the walking fish group (WFG) algorithm [29] or approximate hypervolume computation algorithms [30]. These algorithms can be used to calculate the hypervolume improvement by projecting points from the Pareto front onto the subdomain that is dominated by the test point. This allows calculation of UCB-HVI to be much faster than EHVI when the number of objectives is large ($P > 3$), while still achieving similar optimization performance. As a result we exclusively use the UCB-HVI acquisition function as a starting point to perform multiobjective optimization for the rest of the paper.

We now show how the MOBO algorithm tackles a simple two-objective optimization problem in 2D input space, $\mathbf{x} = (x_1, x_2)$. The problem is stated as

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), f_2(\mathbf{x})\} \\ &f_1(\mathbf{x}) = ||\mathbf{x} - \mathbf{1}|| \\ &f_2(\mathbf{x}) = ||\mathbf{x} + \mathbf{1}|| \\ &x_n = [-2, 2], \qquad n = 1, 2. \end{aligned} \qquad (5)$$

The analytical Pareto front for this problem in objective space lies on the line segment from $(f_1, f_2) = (0, 2\sqrt{2})$ to $(2\sqrt{2}, 0)$. In input space, Pareto optimal points lie on the line segment from $(x_1, x_2) = (-1, -1)$ to (1,1). We start with a set of five random input points, drawn from a uniform distribution, which are used to initialize the GP surrogate model. We choose an isotropic radial basis function (RBF) for our kernel [see Eq. (A6)] with a length-scale of $\lambda = 1.0$ and a variance of $\sigma_f^2 = 0.5$. The UCB-HVI acquisition function with $\beta = 0.01$ is then used to determine the next point to sample. This value of β is chosen to heavily weight exploitation since our functions are unimodal.

Figure 3 shows results after 15 optimization iterations. We see that the GP prediction for each of the objectives [Figs. 3(c) and 3(d)] near the observed points closely resembles the true value of each objective [Figs. 3(a) and 3(b)]. After only 15 observations the Pareto front found by the UCB-HVI algorithm closely matches the analytical one seen in Fig. 3(f). We also observe that extrema of the acquisition function Fig. 3(e) (i.e., the most likely points to expand the Pareto front hypervolume) are located near the analytical Pareto optimal region in input space. Each successive extremum is located in between each previously observed point due to the increase in uncertainty in between observations. This means that after the algorithm observes points along the entire Pareto front at a low resolution, the algorithm will continually attempt to increase the hypervolume by sampling intermediate points in between previous observations.

C. Adding optimization preferences and constraints

One major advantage of the MOBO approach, is the ability to specify how the optimizer searches the input space when considering preferential treatment of objectives and adding constraints. In the case of preferential treatment, we wish to specify that the optimizer searches in a given objective subspace [as seen in Fig. 2(d)], thus optimizing one objective or a set of objectives over another. To achieve this, we simply set the acquisition function to zero outside of the selected subdomain [31]. On the other hand, if we wish to specify an objective constraint, we require that an observed objective quantity $g(\mathbf{x})$ satisfies $g(\mathbf{x}) \leq h$ where h is a constant. In this case, we create a surrogate model for $g(\mathbf{x})$ and use it to predict the probability that the constraint will be satisfied [32]. We then multiply the acquisition function by this probability to bias the optimizer against

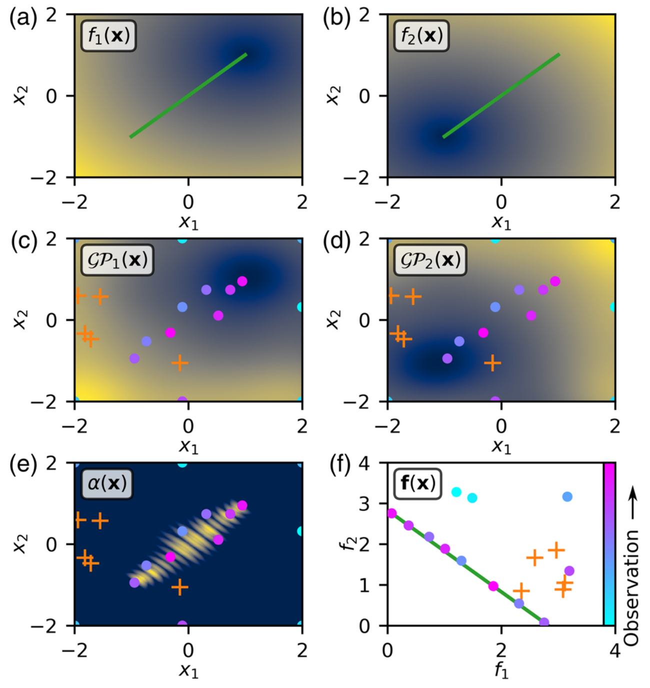

FIG. 3. Optimization results after 15 function observations using the MOBO algorithm on the problem defined in Eq. (10). Functions plotted in (a)–(e) are normalized to the range 0 (blue) to 1 (yellow). (a),(b) Ground truth of f_1, f_2. The green line denotes the analytical location of Pareto optimal points. (c),(d) GP mean prediction for f_1, f_2 respectively. (e) UCB-HVI acquisition function with $\beta = 0.01$. (f) Plot of observation values in objective space and the analytical Pareto front (green line). (c)–(f) Colored dots denote observation locations in input and output space, colored by observation number (blue to pink). Orange crosses denote the locations of five random, initial observation points.

choosing points in a region that will likely violate the constraint.

While at first glance it seems that these two methods would result in the same behavior, i.e., limiting the region where the acquisition function is nonzero, they in fact produce different results. The addition of a preferential objective subdomain results in a Pareto front that is only found within that subdomain, which implies that all objectives are still minimized at the expense of other objectives. On the other hand, a constraint loosens this requirement, allowing any objective value that satisfies the constraint, which in turn can lead to better solutions for the other objectives. The subtle difference between these two methods allows more flexibility during optimization, suiting the different operation requirements for each accelerator. We now look at how to implement each of these methods in the MOBO framework.

The preferential algorithm specifies both a maximum and minimum reference point in objective space, as opposed to a single reference point in normal MOBO. It then calculates what has been coined as the truncated hypervolume improvement [31]. If we specify the truncated domain $\mathcal{T} \in [\mathbf{A}, \mathbf{B}]$ defined by the minimum objective point $\mathbf{A}$ and the maximum objective point $\mathbf{B}$, the truncated expected hypervolume improvement (TEHVI) is given by

$$\alpha_{\text{TEHVI}}(\mu, \sigma, \mathcal{P}, \mathbf{A}, \mathbf{B}) := \int_{\mathbb{R}^p \in \mathcal{T}} \mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{r}) \cdot \xi_{\mu,\sigma}(\mathbf{y}) d\mathbf{y}, \quad (6)$$

where the Pareto set $\mathcal{P}$ is projected onto the truncated domain. In a similar fashion, the truncated version of the UCB-HVI is given by

$$\alpha_{\text{TUCB-HVI}}(\mu, \sigma, \mathcal{P}, \beta, \mathbf{A}, \mathbf{B}) := \begin{cases} \mathcal{H}_I(\mathcal{P}, \mathbf{y}, \mathbf{B}) & \mathbf{y} \in \mathcal{T} \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

where $\mathbf{y} = \mu - \sqrt{\beta}\sigma$.

If we wish to specify an inequality constraint that needs to be satisfied, we create a GP surrogate model that can be used to predict the probability of that constraint being satisfied and modify our acquisition function accordingly. We assume that we have another observed quantity $g(\mathbf{x})$ that must satisfy $g(\mathbf{x}) \leq h$ whose value is stored in a dataset $\mathcal{D}_g$ and used in a GP to predict the probability distribution of $g(\mathbf{x})$. The probability of a point $\mathbf{x}$ satisfying the constraint condition is then

$$P_g(\mathbf{x}) := Pr[g(\mathbf{x}) \leq h] = \int_{-\infty}^{h} p[g(\mathbf{x})|\mathcal{D}_g] dg(\mathbf{x}) \quad (8)$$

which is simply a univariate Gaussian cumulative distribution function. This probability can be adapted to suit a number of different types of constraints by modifying the limits of this integral. Now we can define a new constrained version of the acquisition function $\hat{\alpha}(\mathbf{x})$ as

$$\hat{\alpha}(\mathbf{x}) = \alpha(\mathbf{x}) P_g(\mathbf{x}). \quad (9)$$

Our acquisition function will be reduced anywhere we predict the constraint has a high probability of being violated and remain unchanged where there is a high probability that the constraint is satisfied. Extra constraints can be easily added by multiplying the acquisition function by the respective probabilities of satisfying each additional constraint.

An example of adding a constraint to the problem specified in Eq. (5) is shown in Fig. 4. In this case we add the constraint inequality $g(\mathbf{x}) \leq 0.5$ where $g(\mathbf{x}) = x_1$, to stand in opposition of minimizing the first objective. We see the predicted probability that a point in parameter space will satisfy the constraint in Fig 4(a). Even though only a few observed points violate the constraint, we can clearly see a predicted threshold boundary, consistent with the stated constraint inequality. Furthermore, the boundary is best defined in the region of most interest, namely in the

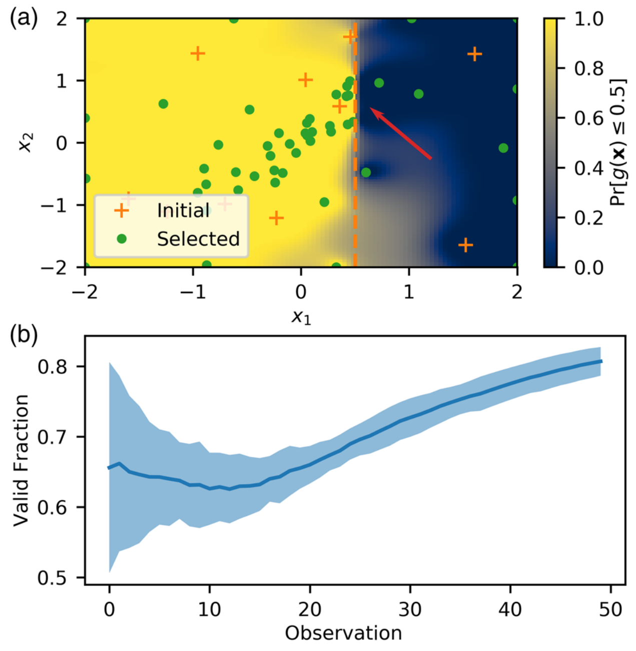

FIG. 4. (a) Probability map of satisfying the constraint $x_1 \leq 0.5$ after 50 observations, selected by constrained UCB-HVI, while optimizing Eq. (5). The red arrow points to where the constraint probability is most accurate, which is where the greatest number of observations are located. (b) Average fraction of points satisfying the constraint over 25 randomly initialized constrained MOBO optimization runs. Shading denotes one sigma spread.

region denoted by the red arrow in Fig. 4(a) where Pareto optimal points would lie. From Fig. 4(b) we see that as the optimization progresses the constraint function surrogate model accuracy is improved, and a smaller fraction of points which do not satisfy the constraint are sampled. This leads to a reduction in the number of iterations needed to converge to a solution, as the constraint reduces the effective input parameter domain of the optimization problem.

D. Proximal input space exploration

One aspect of accelerator optimization that is often overlooked when constructing optimization algorithms is the cost associated with changing input parameters. Changes to input parameters (magnetic field strengths, cavity phases, etc.) often take time that scales proportionally to the magnitude of the change. As a result, it is undesirable or infeasible to make large changes in machine input parameters frequently. We modify the acquisition function so that each optimization step travels a small distance in parameter space during optimization (i.e., “proximal exploration”).

We multiply the acquisition function by a multivariate Gaussian distribution, centered at the most recently observed point in input space $\mathbf{x}_0$ and a precision matrix $\mathbf{\Lambda}$, to produce a proximal UCB-HVI (P-UCB-HVI) acquisition function given by

$$\tilde{\alpha}(\mathbf{x}, \mathbf{x}_0) = \alpha(\mathbf{x}) \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{\Lambda} (\mathbf{x} - \mathbf{x}_0)\right]. \quad (10)$$

The precision matrix in this case specifies the cost associated with changing each input parameter, where larger elements of the matrix correspond to a higher cost associated with changing a given parameter. The matrix can be specified prior to optimization or trained from multiple optimization runs (Bayesian optimization has been used to optimize similar hyperparameters for neural network regression [33]). The addition of this extra term decreases the acquisition function far away from the most recently observed point.

With this modification we expect the MOBO algorithm to sample points along the Pareto front in input space, provided that the objectives are smoothly varying. This is almost always the case in accelerator optimization problems with continuously variable input parameters. However, since the weighting function is nonzero throughout the input domain, large jumps to explore regions of parameter space with high uncertainty are still allowed, only if the acquisition function $\alpha(\mathbf{x})$ is large enough to overcome the travel distance penalty. This is in contrast to simply reducing the UCB-HVI parameter β or restricting the maximum travel distance in input space, which prevents meaningful exploration when necessary. The proximal modification maintains the optimization algorithm’s ability to escape local extrema to explore regions of unobserved input space, while significantly reducing the frequency of large jumps.

We demonstrate how this new method effects optimization by tracking how the UCB-HVI and P-UCB-HVI acquisition functions explore the input space, while optimizing our test problem Eq. (5). We start with a random sample of five points and then use UCB-HVI as our unmodified acquisition function to perform MOBO with 25 iterations, the result of which is shown in Fig. 5(a). We then repeat the optimization with our modified acquisition function P-UCB-HVI. We specify $\mathbf{\Lambda} = 4\mathbf{I}$ where $\mathbf{I}$ is the identity matrix, the result of which is shown in Fig. 5(b). The modified acquisition function reduces the average distance traveled in input space during each optimization step (L), when compared to UCB-HVI (see Fig. 5 insets). While normal UCB-HVI seems to quasirandomly explore the input space, P-UCB-HVI explores the Pareto optimal region in a disciplined manner. It first travels along the Pareto optimal space until it reaches the end, then it explores in the vicinity of the end point to verify it is indeed the end of the Pareto optimal region. Finally, it reverses direction and continues exploring the Pareto optimal space, jumping over regions that have already been explored.

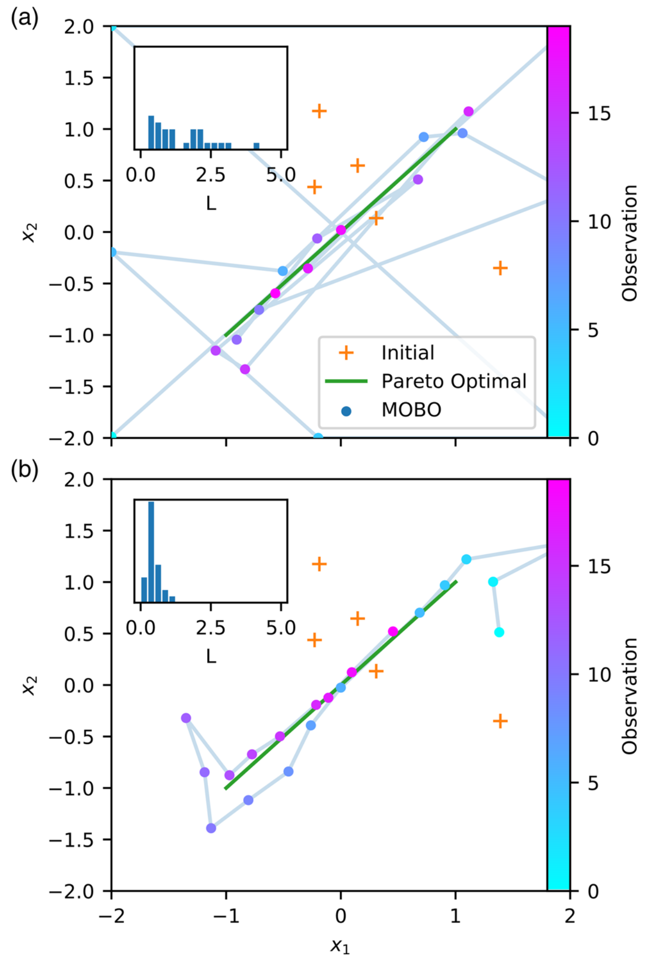

FIG. 5. Optimization trajectories in input space after 25 observations of objective functions in Eq. (5) using (a) the unmodified UCB-HVI acquisition function and (b) the P-UCB-HVI acquisition function. Insets: Distribution of distances (L) traveled per step in input space.

IV. APPLICATION TO ACCELERATOR OPTIMIZATION

We now demonstrate the MOBO framework on a multiobjective accelerator optimization problem, namely, optimizing the parameters of the Argonne wakefield accelerator (AWA) photoinjector [34]. The AWA photoinjector uses a cesium-telluride cathode in a normal conducting radio-frequency cavity, to produce electron beams with a wide variety of bunch charges for the use in wakefield accelerator physics experiments. We consider a case where we can vary a number of parameters inside the injector and the first linac section, seen in Table I and in Fig. 6. Our goal is to simultaneously minimize a collection of beam parameters at the exit of the linac section, also seen in Fig. 6.

This problem was chosen based on previous work done towards creating a surrogate model of the AWA photoinjector [25]. In this previous work, the authors used the full 3D space charge, particle in cell (PIC) code OPAL [35] to simulate a large set of randomly generated input parameters and measure the corresponding beam parameters at the injector exit. They then created a neural network based surrogate model, trained on the simulation dataset. The model can be rapidly queried to retrieve output beam parameters for a given input parameter set. The authors then showed that this surrogate model accurately reproduces results from the 3D PIC simulation. We use this surrogate model for testing our optimization algorithm as it reduces simulation time by several orders of magnitude.

TABLE I. AWA input parameters.

Parameter	Abbreviation	Minimum value	Maximum value	Unit
Solenoid 1 strength	K_1	400	550	m^{-1}
Solenoid 2 strength	K_2	180	280	m^{-1}
Injector phase	ϕ_1	−10	0	deg
Cavity phase	ϕ_2	−10	0	deg
Injector accelerating gradient	G_1	60	75	MV/m
Cavity accelerating gradient	G_2	15	25	MV/m

A. Convergence comparison

In our first experiment, we use MOBO to minimize all seven exit beam parameters as a function of all six input parameters. We wish to compare the convergence rate of MOBO with the convergence rates of standard and recently developed algorithms for solving multiobjective optimization problems. All of the input and output values are normalized to the range $[-1, 1]$ in order to account for the widely varying scaling of each parameter. We assume that the functional form of each objective is smooth, and thus we choose the standard radial basis function kernel [Eq. (A6)] with an anisotropic precision matrix $\Lambda = \mathrm{diag}(\mathbf{l})^{-2}$ where $\mathbf{l}$ is a vector that stores an independent length scale for each

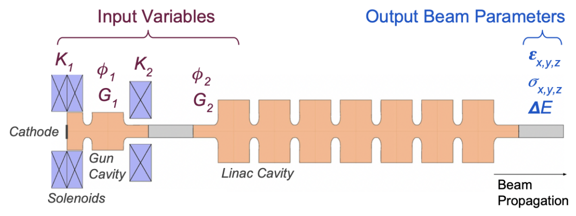

FIG. 6. Cartoon of the AWA photoinjector and first linac cavity. Input and output parameters used in optimization are labeled. Reprinted with permission from A. Edelen *et al.*, Phys. Rev. Accel. Beams **23**, 044601 (2020). Copyright 2020, American Institute of Physics.

input parameter. Initially, a randomly generated Latin-hypercube distribution of 20 input parameter sets with corresponding objective observations is used to train each objective GP. Hyperparameter training is done by maximizing the log marginal likelihood of the GP model with the gradient based Adam optimization algorithm [36], with 5000 iterations and a learning rate of 0.001.

Once trained, we use the UCB-HVI acquisition function to perform multiobjective Bayesian optimization with 500 sequential observations. In this case, empirical testing found that $\beta = 0.01$ gave the fastest convergence. This is likely due to the unimodal nature of each objective function, which allows us to aggressively exploit the GP model for the global extremum without worrying about getting caught in local extrema.

Maximizing the acquisition function is done via particle swarm optimization, implemented using the PyGMO package [37] with 64 individuals over ten generations. In order to account for new information gained from observations during optimization, we retrain the GP kernel hyperparameters with the accumulated dataset every ten observations, again using the Adam algorithm with a learning rate of 0.001 but with 1000 steps.

We rerun this optimization procedure 10 times, each with a different set of 20 randomly generated initial points. After each observation, we calculate the exact hypervolume of the Pareto set in normalized objective space, referenced to the maximum possible value for each objective (in this case 1). The average and variance of the hypervolume as a function of observation number after ten optimization runs is shown in Fig. 7(a).

For comparison, we run the same optimization test, but with previously used methods for multiobjective optimization, evaluated in serial, as would be the case during online optimization. The first, nondominated sorting genetic algorithm II (NSGA-II) [20], is a popular genetic optimization algorithm, which has been previously used to solve multiobjective accelerator design problems [13]. We conducted ten optimization runs using the NSGA-II algorithm, with a population of 20 individuals, which matched the input parameter sets used in the MOBO optimization runs. We then evolved the population for 200 generations, with a crossover probability of 0.8 and mutation probability of 0.05. The hypervolume after the first 25 generations (500 function observations) is shown in Fig. 7(a).

The second algorithm, iterated neural network (I-NN) optimization [25], is a recently developed algorithm using surrogate neural network (NN) models to choose future observation locations. In this method, observations are used to train a NN surrogate model, which in turn is optimized by the NSGA-II algorithm to propose a new set of observations that are likely to be nondominated. We plot the predicted hypervolume from the NN surrogate model after each batch of measurements in Fig. 7(a).

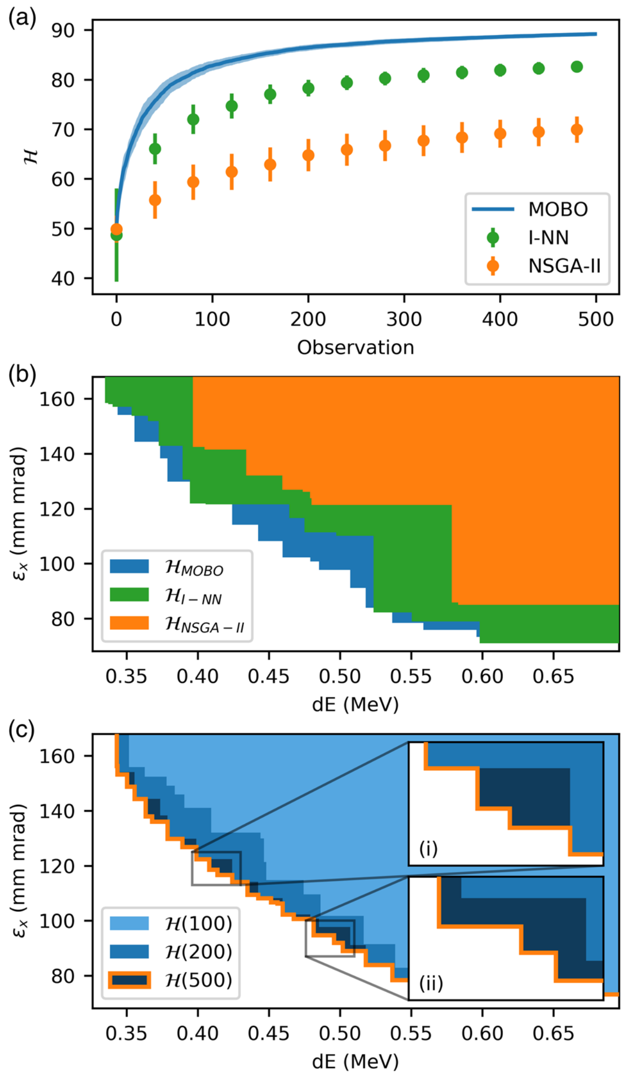

FIG. 7. (a) Average Pareto front hypervolume $\mathcal{H}$ of ten multiobjective optimization runs of the AWA example using MOBO, NSGA-II and iterated neural network (I-NN) algorithms. Shading and error bars denote 1 sigma variance. (b) Projected hypervolume onto energy spread (dE) vs transverse emittance (ε_x) subspace after 200 observations for each optimization algorithm shown in (a). (c) Projected hypervolume onto dE vs ε_x subspace after 100 (light blue), 200 (blue) and 500 (dark blue with orange outline) observations using the MOBO algorithm. Inset zoom (i) shows an increase in hypervolume due to increasing the Pareto front resolution, while inset zoom (ii) shows an increase in hypervolume due to finding new points that dominate old observations in projected space.

From this comparison, we clearly see that MOBO reaches convergence much faster than both the NSGA-II and I-NN algorithms. While not shown in Fig. 7(a) it took about 17,500 NSGA-II observations to reach the same hypervolume that MOBO reached after 500 observations, roughly a factor of 35 times slower.

In Fig. 7(b) we show the 2D projected Pareto front on the energy spread dE and horizontal beam emittance ε_x

objective space from each of these optimization algorithms after 200 observations. We observe that the Pareto front generated by NSGA-II is far from optimal and contains few points. This is a direct result from NSGA-II's inefficient sampling behavior. The points that NSGA-II chooses to observe are frequently dominated by previous observations due to the randomized heuristic used to generate observation proposals. The I-NN algorithm improves over NSGA-II by including a neural network surrogate model that directs NSGA-II towards observing nondominated points. Neither of these algorithms includes a direct calculation of the hypervolume increase for each proposed observation point, and thus does not optimally increase the hypervolume at each observation step. As a result the Pareto front generated by MOBO is larger and has a higher resolution than the Pareto fronts generated by either NSGA-II or I-NN. Generally, MOBO shows similar improvements in optimization speed when used in solving a variety of different optimization problems with varying input and objective spaces [38,39].

In Fig. 7(c) we show the 2D projected Pareto fronts generated by MOBO after 100, 200 and 500 observations. Since the objective space is high dimensional, it takes a large number of observations (100–200) for the algorithm to build up a well-defined Pareto front. Once the front is loosely meshed, the acquisition function can increase the hypervolume in one of two ways, by either finding points in objective space in between prior observations, in order to improve the Pareto front resolution [Fig. 7(c)(i)], or by finding points in objective space that dominate initial observations [Fig. 7(c)(ii)]. We see that most of the points present in the Pareto front after 200 observations remain on the Pareto front after 500 observations, as the majority of new observations lie in between prior observations in objective space. We conclude that in this optimization run, the algorithm often chooses to sample points on the true Pareto front, leading to a gain in optimization efficiency. This is in contrast to the heuristic methods we compare MOBO with, where only a small fraction of the observed points are actually on the true Pareto front.

B. Constrained optimization

We now investigate the effect of preferential or constrained treatment of an objective on accelerator optimization. First, we consider a case where we want to optimize the same objectives as the previous problem but wish to only find solutions where the energy spread satisfies $dE < 0.52$ MeV ($dE < -0.25$ in normalized coordinates). To judge how this modification effects optimization, we compare the observed points projected onto the 2D energy spread dE and horizontal emittance ε_x objective space, after 300 iterations in Fig. 8. We see in Fig. 8(a) projected observations when no constraints or preferences are added. It is important to note that a large majority of the observations plotted here are on the 7D Pareto front, even though only a small fraction make up the projected front in 2D space. When preferential treatment is added to the acquisition function [Fig. 8(b)], the algorithm observes almost no points that violate this preference. Furthermore, since the effective volume of the objective space is significantly reduced, the optimizer finds a higher quality 2D Pareto front in the same number of steps as the unconstrained case.

Second, we consider a case where we relax this preference, removing the energy spread minimization objective entirely and replacing it with the inequality constraint $dE < 0.52$ MeV. The resulting distribution of observations appears significantly different in this case [Fig. 8(c)]. Here, more observations are made that violate the constraint, which is necessary to accurately model the constraining function near the boundary. Furthermore, the optimizer allows the energy spread to increase up to the constraint value, instead of attempting to minimize it, in order to better optimize the six remaining objectives. We see this effect in Fig. 8(d) where the projected Pareto front for a different set of objectives, σ_x and ε_x, is better when the energy spread preference is relaxed to a constraint. The 2D front then improves again when the constraint is completely

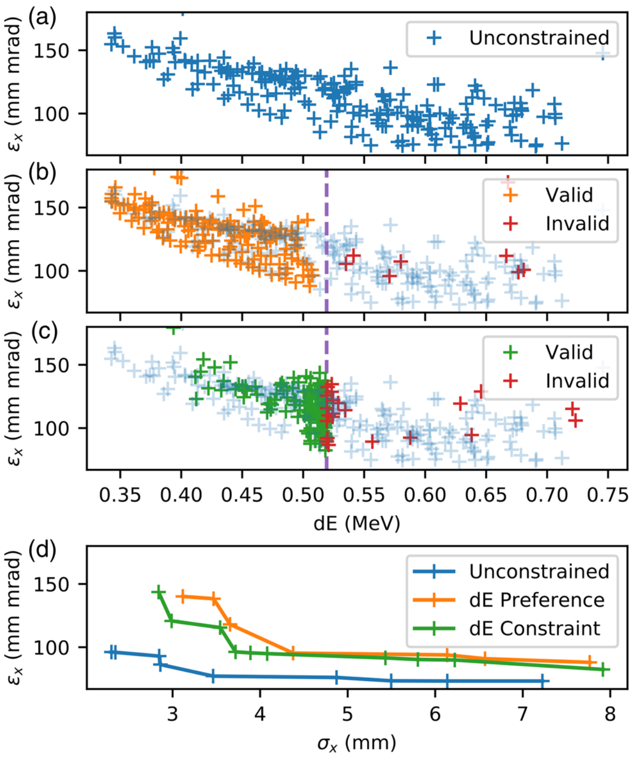

FIG. 8. Plots showing energy spread (dE) and horizontal beam emittance (ε_x) observations after 300 observations taken by MOBO algorithms. (a) MOBO with no constraints. (b) MOBO with an optimization preference of $dE < 0.52$ MeV. (c) MOBO with an inequality constraint of $dE < 0.52$ MeV. (d) Projected Pareto fronts for σ_x vs ε_x for each case above. The dotted lines in (b) and (c) denote the preference/constraint limit.

removed, owing to the fact that dE is allowed to increase further when all constraints are removed.

C. Proximal optimization

Finally, we demonstrate the use of P-UCB-HVI on optimizing the AWA problem. We start with the same set of ten initial sets of observations as in Sec. IV A with the same hyperparameter training schedule. However this time we run MOBO optimization using the P-UCB-HVI acquisition function, with an isotropic precision matrix [see Eq. (10)] $\mathbf{\Lambda} = 4\mathbf{I}$ defined in normalized input space.

Results from these optimization runs are presented in Fig. 9. We observe that during optimization, when the UCB-HVI acquisition function is used, the solenoid strength parameter K_1 is wildly varied to increase the hypervolume as much as possible each step. However, when the proximal term is added to the acquisition function, the frequency and amplitude of large jumps in parameter space are both decreased. While not shown here, this change in behavior is mirrored in each of the other five input parameters. Furthermore, the use of P-UCB-HVI acquisition function over the generic UCB-HVI function only minimally reduces the overall speed at which the method maximizes the Pareto front hypervolume [Fig. 9(c)].

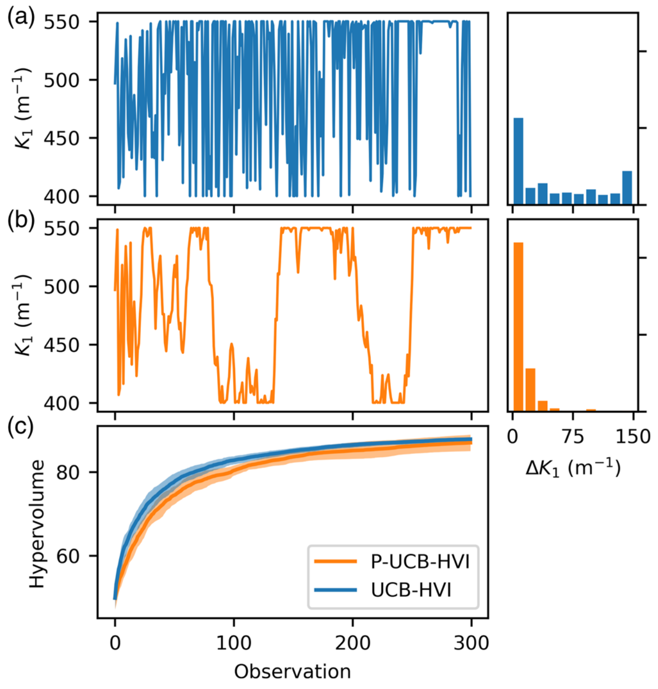

FIG. 9. Comparison between UCB-HVI and the proximal UCB-HVI acquisition functions when used to perform optimization of the AWA photoinjector. (a) Solenoid 1 strength parameter over 300 observations using UCB-HVI (left) and corresponding distribution of ΔK_1 for each step (right). (b) Solenoid 1 strength parameter and travel distance distribution when P-UCB-HVI is used. (c) Average Pareto front hypervolume of ten optimization runs for UCB-HVI and P-UCB-HVI with identical random initialization sets. Shading denotes one sigma variance.

V. CONCLUSION

In this paper we have demonstrated that the MOBO framework can be used to solve online multiobjective optimization accelerator physics problems. This method efficiently finds the Pareto front in a serialized manner, which makes it viable for use in online accelerator optimization. The framework also allows the user to explicitly specify objective preferences and constrain the objective space through the use of GPs. Finally, we demonstrated that adding a proximal term to the acquisition function effectively restricts the MOBO algorithm to prioritize moving through input space in a proximal, disciplined manner, which is especially important for practical use in accelerator facilities.

Our results also demonstrate the reasons why MOBO is ideal for online multiobjective accelerator optimization. Optimization takes place after every observation step, as opposed to methods designed for parallel use, which are not sample efficient when used in a serialized context. Second, observation points proposed by MOBO directly incorporate learned information about the objective function instead of using a model independent heuristic to generate potential solutions. As a result, MOBO strategically proposes solutions which optimally increase Pareto front hypervolume and improve Pareto front resolution. While this comes with an increase in computational complexity, the extra computation time needed (estimated to be <5 s for most problems) is small relative to the reduction in optimization time associated with faster convergence to the Pareto front.

Practical online multiobjective optimization is useful primarily as an experimental beam line characterization tool that can identify accelerator working points. These working points are often predetermined through conducting multiobjective optimization on simulated versions of the beam line. However, simulations rarely capture the full behavior of the beam in the real accelerator. As a result, any working points produced by conducting multiobjective optimization on a simulation might not be ideal in reality. MOBO can be used to find the realistic, ideal trade-off between given objectives, and identify the input parameters that are Pareto optimal. Once a trade-off has been selected by specifying an explicit weighting of the objectives and the correct input parameters have been determined, single objective optimization strategies can take over during regular operation to do local optimization in response to noise and/or temporal drift. If we wish to scan over a given objective, such as the longitudinal bunch length in a photoinjector, we can maintain optimal settings without repeating optimization, since MOBO characterizes the entire Pareto front. This assumes that the Pareto front varies on a sufficiently long time scale that any changes are negligible. Applying this algorithm towards solving time-dependent multiobjective optimization problems is a source of future study. Finally, MOBO can be repeated periodically to monitor changes in machine performance over time and benchmark simulated Pareto fronts to experimental results.

While the goal of this work is to apply the MOBO framework towards online optimization of accelerators, this technique can also be applied to solve computational accelerator physics optimization problems. High fidelity simulations of particle accelerator physics (beam lines, cavities, magnets etc.) are also a resource intensive process, often requiring time on computational clusters which have limited availability. The recently developed q-expected hypervolume improvement (qEHVI) [40] algorithm is a parallelized extension of EHVI. In contrast to EHVI, which proposes a single point at each optimization step, qEHVI proposes multiple q points that are likely to increase the Pareto front hypervolume for each optimization step. These points can be evaluated in a batched parallel process on a computing cluster, which significantly reduces overall optimization time while maintaining the sampling efficiency advantages of EHVI. Furthermore, multifidelity approaches to MOBO have also been proposed to reduce optimization time [41], by incorporating low-cost, approximate simulations as a temporary stand-in for expensive high fidelity simulations. The application of these methods to solving computational accelerator problems has the potential to dramatically reduce resource requirements for those in the field. Results from using these computational tools can also be integrated back into an experimental MOBO algorithm in order to speed up serialized optimization.

ACKNOWLEDGMENTS

We would like to thank Nicole Neveu for her involvement in the design of the simulation data set we used [25] and running simulations used to create the data set. This work was supported by the U.S. National Science Foundation under Grant No. PHY-1549132, the Center for Bright Beams and the U.S. Department of Energy, Office of Science, Contracts No. DE-AC02-76SF00515 and No. DE-AC02-06CH11357, and DOE Office of Basic Energy Sciences under FWP 100637.

APPENDIX A: GAUSSIAN PROCESS REGRESSION

A GP regression model works by representing the function value at a given input point via a random variable drawn from a multivariate Gaussian distribution $f(\mathbf{x}) \sim \mathcal{GP}[\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')]$, where $\mu(\mathbf{x})$ is the mean and $k(\mathbf{x}, \mathbf{x}')$ is the covariance [8]. We start with a prior belief that $\mu(\mathbf{x}) = 0$ (without loss of generality) and use Bayes rule to update this belief to a new one (known as the posterior), conditioned on the observed dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ and the covariance function. The prediction of the function at a test point $\mathbf{x}_*$ is given by $f_* = f(\mathbf{x}_*)$. This random variable is then drawn from the conditional Gaussian,

$$p(f_*|\mathcal{D}) \sim \mathcal{N}(\mu_*, \sigma_*^2) \tag{A1}$$

$$\mu_* = \mathbf{k}^T[K + \sigma_n^2 \mathbf{I}]^{-1}\mathbf{y} \tag{A2}$$

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T[K + \sigma_n^2 \mathbf{I}]^{-1}\mathbf{k} \tag{A3}$$

$$\mathbf{k} = [k(\mathbf{x}_*, \mathbf{x}_1), k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N)] \tag{A4}$$

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \tag{A5}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$ and σ_n is the noise hyperparameter.

The covariance function $k(\mathbf{x}, \mathbf{x}')$ is specified based on prior knowledge of the target functions' behavior. A common kernel is the radial-basis function (RBF) given by

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda}(\mathbf{x} - \mathbf{x}')\right], \tag{A6}$$

where Λ is known as the precision matrix. In the isotropic case, the matrix is specified by $\mathbf{\Lambda} = \mathbf{I}/\lambda^2$ where I is the identity matrix and λ is referred to as the length-scale hyperparameter. This hyperparameter describes the characteristic length scale at which the function varies. For large λ the function is expected to be smooth; decreasing λ causes the function to vary quickly over a short distance. In a case where the function is expected to vary at different length scales along each input axis we can define an anisotropic kernel, where the matrix is specified by a length scale vector $\mathbf{l}$ where $\mathbf{\Lambda} = \mathrm{diag}(\mathbf{l})^{-2}$ where the diagonal elements specify a length scale for each input dimension. Further, the entire precision matrix can be specified via $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^T + \mathrm{diag}(\mathbf{l})$ where $\mathbf{L}$ is an upper triangle matrix to ensure that $\mathbf{\Lambda}$ is positive self-definite. Generally hyperparameters can be trained by maximizing the marginal log likelihood, which optimizes them to best fit the observed dataset while minimizing the regression's functional complexity [8]. However, they can also be determined in a localized region by calculating the function's Hessian matrix at a given point [9].

APPENDIX B: CODE AVAILABILITY

This research used only open source Python software libraries, including GPFlow [42] and TensorFlow [43]. The algorithms developed are contained in a repository [44].

The surrogate model of the AWA photoinjector is available upon request.

[1] X. Huang, J. Corbett, J. Safranek, and J. Wu, An algorithm for online optimization of accelerators, Nucl. Instrum. Methods Phys. Res., Sec. A **726**, 77 (2013).

[2] J. A. Nelder and R. Mead, *A Simplex Method for Function Minimization* (Oxford Academic, 1965), Vol. 7, pp. 308–313.

[3] M. J. D. Powell, The BOBYQA algorithm for bound constrained optimization without derivatives, Cambridge NA Report No. DAMTP 2009/NA06, University of Cambridge, Cambridge, 2009.

[4] N. Neveu, J. Larson, J. G. Power, and L. Spentzouris, Photoinjector optimization using a derivative-free, model-based trust-region algorithm for the Argonne wakefield accelerator, in the *8th International Particle Accelerator Conference (IPAC'17), Copenhagen, Denmark, 2017* (JACOW, Geneva, Switzerland, 2017), pp. 4100–4103.

[5] S. Appel and S. Reimann, Beam line optimization using derivative-free algorithms, J. Phys. **1350**, 012104 (2019).

[6] 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, Proceedings of the IEEE (2016), Vol. **104**, p. 148.

[7] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, Bayesian optimization for adaptive experimental design: A review, IEEE Access **8**, 13937 (2020).

[8] C. E. Rasmussen and . K. I. Williams, *Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning* (MIT Press, Cambridge, MA, 2006).

[9] A. 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:1911.01538.

[10] M. McIntire, T. Cope, S. Ermon, D. Ratner *et al.*, Bayesian optimization of FEL performance at LCLS, in *Proceedings of the 7th International Particle Accelerator Conference* (IOP Publishing, Bristol, England, 2016).

[11] J. Duris, D. Kennedy, A. Hanuka, J. Shtalenkova, A. Edelen, A. Egger, T. Cope, and D. Ratner, *Bayesian Optimization of a Free-Electron Laser* (APS, Maryland, 2020).

[12] J. Kirschner, M. Mutný, N. Hiller, R. Ischebeck, and A. Krause, Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces, in Proceedings of the 36th International Conference on Machine Learning (PMLR, 2019), pp. 3429–3438, http://proceedings.mlr.press/v97/kirschner19a/kirschner19a.pdf.

[13] N. Neveu, L. Spentzouris, A. Adelmann, Y. Ineichen, A. Kolano, C. Metzger-Kraus, C. Bekas, A. Curioni, and P. Arbenz, Parallel general purpose multiobjective optimization framework with application to electron beam dynamics, Phys. Rev. Accel. Beams **22**, 054602 (2019).

[14] Y. Li, W. Cheng, L. H. Yu, and R. Rainer, Genetic algorithm enhanced by machine learning in dynamic aperture optimization, Phys. Rev. Accel. Beams **21**, 054601 (2018).

[15] M. Emmerich, K. Yang, A. Deutz, H. Wang, and C. M. Fonseca, A multicriteria generalization of Bayesian global optimization, *Advances in Stochastic and Deterministic Global Optimization*, edited by P. M. Pardalos, A. Zhigljavsky, and J. Źilinskas, Springer Optimization and Its Applications (Springer International Publishing, New York), pp. 229–242.

[16] A. Scheinker, S. Hirlaender, F. M. Velotti, S. Gessner, G. Z. Della Porta, V. Kain, B. Goddard, and R. Ramjiawan, Online multiobjective particle accelerator optimization of the AWAKE electron beam line for simultaneous emittance and orbit control, AIP Adv. **10**, 055320 (2020).

[17] E. Cropp and A. Edelen (to be published).

[18] L.. R. Zuhal, P. Satria Palar, and K. Shimoyama, A comparative study of multiobjective expected improvement for aerodynamic design, Aerosp. Sci. Technol. **91**, 548 (2019).

[19] B. Naujoks, N. Beume, and M. Emmerich, Multiobjective optimization using s-metric selection: Application to three-dimensional solution spaces, in *2005 IEEE Congress on Evolutionary Computation* (IEEE, Piscataway, NJ, 2005), Vol. 2, pp. 1282–1289.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation, Piscataway, NJ* (2002), Vol. 6, p. 182.

[21] J. Kennedy and R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95—International Conference on Neural Networks* (IEEE, New York, 1995), Vol. 4, pp. 1942–1948.

[22] X. Huang and J. Safranek, Nonlinear dynamics optimization with particle swarm and genetic algorithms for SPEAR3 emittance upgrade, Nucl. Instrum. Methods Phys. Res., Sect. A **757**, 48 (2014).

[23] X. Pang and L. J. Rybarcyk, Multiobjective particle swarm and genetic algorithm for the optimization of the LANSCE linac operation, Nucl. Instrum. Methods Phys. Res., Sect. A **741**, 124 (2014).

[24] X. Huang, M. Song, and Z. Zhang, Multiobjective multigeneration Gaussian process optimizer for design optimization, Report No. SLAC-PUB, Menlo Park, 2019.

[25] A. Edelen, N. Neveu, M. Frey, Y. Huber, C. Mayes, and A. Adelmann, Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems, Phys. Rev. Accel. Beams **23**, 044601 (2020).

[26] D. R. Jones, M. Schonlau, and W. J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. **13**, 455 (1998).

[27] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, Gaussian process optimization in the bandit setting: No regret and experimental design, in *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10* (OmniPress, Madison, WI, 2010), pp. 1015–1022.

[28] K. Yang, M. Emmerich, A. Deutz, and T. Bäck, *Efficient Computation of Expected Hypervolume Improvement Using Box Decomposition Algorithms* (Springer, New York, 2019).

[29] L. While, L. Bradstreet, and L. Barone, *A Fast Way of Calculating Exact Hypervolumes* (IEEE Transactions on Evolutionary Computation, New York, 2012), Vol. 16, p. 86.

[30] W. Tang, H.-L. Liu, L. Chen, K. C. Tan, and Y.-M. Cheung, Fast hypervolume approximation scheme based on a segmentation strategy, Inf. Sci. **509**, 320 (2020).

[31] K. Yang, A. Deutz, Z. Yang, T. Back, and M. Emmerich, Truncated expected hypervolume improvement: Exact computation and application, in *2016 IEEE Congress on Evolutionary Computation (CEC)* (IEEE, Piscataway, NJ, 2016), pp. 4350–4357.

[32] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, Bayesian optimization with inequality constraints, in *ICML* (2014), Vol. 2014, pp. 937–945.

[33] J. Snoek, H. Larochelle, and R. P. Adams, *Practical Bayesian optimization of machine learning algorithms*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Advances in Neural Information Processing Systems 25 (Curran Associates, Inc., Red Hook, NY, 2012), pp. 2951–2959.

[34] M. E. Conde, S. P. Antipov, D. S. Doran, W. Gai, Q. Gao, and G. Ha, *Research Program and Recent Results at the Argonne wakefield Accelerator Facility (AWA)* (IOP Publishing, Bristol, England, 2017), p. 3.

[35] A. Adelmann, Ch. Kraus, Y. Ineichen, S. Russell, Y. Bi, and J. J. Yang, The object oriented parallel accelerator library (OPAL), design, implementation and application, in *Proceedings of the 23rd Particle Accelerator Conference, Vancouver, Canada, 2009* (IEEE, Piscataway, NJ, 2009).

[36] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[37] F. Biscani and D. Izzo, A parallel global multiobjective framework for optimization: PAGMO, J. Open Source Software **5**, 2338 (2020).

[38] D. Zhan and H. Xing, Expected improvement for expensive optimization: A review, J. Global Optim. **78**, 507 (2017).

[39] R. Allmendinger, M. T. M. Emmerich, J. Hakanen, Y. Jin, and E. Rigoni, Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case, J. Multi-Criteria Decision Analysis **24**, 5 (2017).

[40] S. Daulton, M. Balandat, and E. Bakshy, Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization, arXiv:2006.05078.

[41] S. Belakaria, A. Deshwal, and J. R. Doppa, Multifidelity multiobjective Bayesian optimization: An output space entropy search approach, arXiv:2011.01542.

[42] A. G. de G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, GPFlow: A Gaussian process library using TensorFlow, J. Mach. Learn. Res. **18**, 1 (2017).

[43] M. Abadi *et al.*, TensorFlow: A system for large-scale machine learning, arXiv:1605.08695.

[44] https://github.com/roussel-ryan/Accelerator_MOBO.