pBO-2GP-3B: A Batch Parallel Known/Unknown Constrained Bayesian optimization with Feasibility Classification and its Applications in Computational Fluid Dynamics

Anh Tran[a,c], Jing Sun[b], John M. Furlan[c], Krishnan V. Pagalthivarthi[c], Robert J. Visintainer[c], Yan Wang[a,*]

[a] *George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332*
[b] *Deparment of Population Health Sciences, Augusta University, Augusta, GA 30912*
[c] *GIW Industries Inc., Grovetown, GA 30813*

Abstract

In this work, we present a constrained batch-parallel Bayesian optimization (BO) framework, termed pBO-2GP-3B, to accelerate the optimization process for high-dimensional and computationally expensive problems, with known and unknown constraints. Two Gaussian processes (GPs) are simultaneously constructed: one models the objective function, whereas the other models the unknown constraints. The known constraint is penalized directly into the acquisition function. For every iteration, three batches are built in sequential order: the first two are the acquisition hallucination and the exploration batches for the objective GP, respectively, and the third one is the exploration batch for the classification GP. The pBO-2GP-3B optimization framework is demonstrated with three synthetic examples (2D and 6D), as well as a 33D multi-phase solid-liquid computational fluid dynamics (CFD) model for the design optimization of a centrifugal slurry pump impeller.

Keywords: Bayesian optimization, batch parallelization, known constraint, unknown constraint, feasibility classification, computational fluid dynamics

1. Introduction

Bayesian optimization (BO) is a surrogate-based black-box optimization technique that models and updates the response surfaces sequentially as the optimization process advances. It has proven to be successful in many different applications including machine learning [1] and design optimization [2] [3]. Several notable advantages of BO, compared to other optimization techniques, include derivative-free, active learning, uncertainty quantification, and mathematical robustness in high-dimensional continuous design space. Yet, the application of classical BO in engineering simulation-based optimization problems is often limited by several factors. First, the computational cost of the

*Corresponding author

traditional BO is expensive, where each functional evaluation is a single simulation, which can take hours to finish. Combined with the sequential sampling approach in the classical BO, the turnaround time significantly increases. Second, the successful rate of simulations can be well estimated, but occasionally the simulation will crash due to some unforeseeable factors, e.g. mesh problems, solver issues, or an ill-conditioned matrix. This is a common behavior in the development of complex simulations. Therefore, improving these two aspects of simulation-based optimization problems is critical to the optimization problem of complex engineering simulations.

In order to solve optimization problems with complex engineering simulations, in concert with the growth of high-performance computing (HPC) infrastructure, the classical BO has been extended toward different possible directions: batch parallelization, constrained, multi-objective, multi-fidelity, scalable, and mixed-integer optimization [4]. Batch parallelization for BO is the implementation of selecting a batch, where several concurrent functional evaluations can be performed simultaneously in order to obtain the optimal solution in the shortest wall-clock time. Constrained BO includes constraints in the classical BO formulation. The constraints can be conceptually divided into two main categories: *known* constraints, where the constraints are imposed beforehand and can be evaluated without running the functional evaluator, and *unknown* constraints, where the constraints are unknown *a priori*, and only known once the functional evaluator is revoked. For example, in engineering applications, the known constraints can be thought of as a collection of inequality constraints, where physics-based knowledge is injected to form a set of inequalities upon the design variables. The unknown constraints, on the other hand, can be established once the functional evaluator or the modeling simulation fails to obtain a convergent solution. Progress on improving BO is elaborated in Section 2.

With the introduction of the known and unknown constraints, the design space is naturally divided into two regions: feasible and infeasible. The feasible region includes inputs which do not violate any constraints, whereas the infeasible region includes inputs which violate any of these constraints. In order to search for a feasible and optimal input, one needs to predict whether the input is feasible, before attempting to evaluate the functional value. As a result, the binary classification for feasibility is needed to make such a prediction. Here we propose a novel algorithm pBO-2GP-3B to batch parallelize the classical BO for the general constrained BO problem and demonstrate its applications to complex modeling and simulation methods, such as multi-phase computational fluid dynamics (CFD). The known constraint is penalized directly in the acquisition function of the classical BO, whereas the unknown constraints are learned through a classification Gaussian process (GP). A novel acquisition function formulation is devised and generalized to in-

corporate the known and unknown constraints, as well as the trade-off between exploitation and exploration in the classical BO. It is noteworthy that the new acquisition function is applicable for any choice of commonly used acquisition functions, such as probability of improvement, expected improvement, upper-confidence bound. The novel methodology is demonstrated with three synthetic two-dimensional and six-dimensional examples (2D and 6D) and an engineering CFD (33D) example.

2. Related work

The BO formulation can be briefly summarized as follows. Given the unknown objective function $y = f(\boldsymbol{x})$, we wish to find the optimal solution that maximizes the objective function $\boldsymbol{x}_{\text{opt}} \in \text{argmax}\, f(\boldsymbol{x})$. Instead of directly optimizing this unknown black-box function, the GP model is constructed as a response surface. For the classical BO method, the next sample is sequentially chosen to maximize the acquisition function, which can be computed numerically based on the constructed GP response surface. The acquisition function dictates the location of the next sampling design site by reconciling the trade-off between exploration (navigating to the most uncertain region) and exploitation (driving the solution to the best-so-far) in the optimization process. While recently there have been significant advances in devising other acquisition functions, the most popular acquisition functions are probability of improvement (PI) [5], expected improvement (EI) [6] [7] [8], and upper-confidence bound (UCB) [9, 10].

Brochu et al. [11], Shahriari et al. [4], Frazier [12], and Jones et al. [13] provide comprehensive and critical reviews of the classical sequential BO method, including a different acquisition function, as well as its applications. Here we adopt the notation from Shahriari et al. [4], and briefly summarize the GP formulation.

Assume that f is a function of $\boldsymbol{x}$, where $\boldsymbol{x} \in \mathcal{X}$ is the d-dimensional input. A $GP(\mu_0, k)$ is a nonparametric model over functions f, which is fully characterized by the prior mean functions $\mu_0(x) : \mathcal{X} \mapsto \mathbb{R}$ and the positive-definite kernel, or covariance function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. In GP regression, it is assumed that $\boldsymbol{f} = f_{1:N}$ is jointly Gaussian, and the observation y is normally distributed given f, leading to

$$\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K}), \tag{1}$$

$$\boldsymbol{y}|\boldsymbol{f}, \sigma^2 \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I}), \tag{2}$$

where $m_i := \mu(\boldsymbol{x}_i)$, and $K_{i,j} := k(\boldsymbol{x}_i, \boldsymbol{x_j})$. Equation 1 describes the prior distribution induced by the GP.

The covariance kernel k is a choice of modeling covariance between inputs. One of the most widely used kernel is the squared exponential kernel, which can be described as

$$\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_0^2 \exp\left(-\frac{r^2}{2}\right), \tag{3}$$

where $r^2 = (\boldsymbol{x} - \boldsymbol{x}')\boldsymbol{\Gamma}(\boldsymbol{x} - \boldsymbol{x}')$, and $\boldsymbol{\Gamma}$ is a diagonal matrix of d squared length scale θ_i. The exponential kernel is also commonly used, which can be described as

$$\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_0^2 \exp\left(-r\right). \tag{4}$$

Let the dataset $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$ denote a collection of N noisy observations and $\boldsymbol{x}$ denote an arbitrary input of dimension d. Under the formulation of GP, given the dataset $\mathcal{D}_n$, the prediction for an unknown arbitrary point is characterized by the posterior Gaussian distribution, which can be described by the posterior mean and posterior variance functions, respectively as

$$\mu_n(\boldsymbol{x}) = \mu_0(\boldsymbol{x}) + \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{m}), \tag{5}$$

and

$$\sigma_n^2 = k(\boldsymbol{x}, x) - \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{k}(\boldsymbol{x})$ is the covariance vector between the query point $\boldsymbol{x}$ and $\boldsymbol{x}_{1:N}$.

Estimating hyper-parameters θ of the objective GP involves optimization of the likelihood and computation of the determinant and the inverse of the covariance matrix, resulting in $\mathcal{O}(N^3)$ algorithmic complexity. One of the significant drawbacks for the classical BO is the computational bottleneck in optimizing the maximum likelihood function as the number of observations N increases. In section 2.1, several common acquisition functions in BO are reviewed. In section 2.2, the relevant work on constrained BO and its incorporation to classical BO is presented. In section 2.3, the related work on batch parallel BO is summarized and discussed.

2.1. Acquisition function

In the classical BO, a GP model is constructed and updated sequentially as the optimization process advances. Consider a dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where N is the number of observations, $\boldsymbol{x}_i$ is the input, and y_i is the actual observation in the classical BO settings. Denote $\mu(\boldsymbol{x})$, $\sigma^2(\boldsymbol{x})$, and $\theta(\boldsymbol{x})$ as the posterior mean, the posterior variance, and the hyper-parameters of the GP model, respectively. In practice, the hyper-parameters θ are found by maximizing the log likelihood estimation (MLE) at

every iteration over a plausible range between lower and upper bounds of θ. Let $\boldsymbol{x}_{\text{best}} \in \underset{\boldsymbol{x}_i}{\text{argmax}}\, f(\boldsymbol{x}_i)$ be the best sample achieved so far during sequential sampling for a maximization problem, and $\phi(\cdot)$ and $\Phi(\cdot)$ be the probability density function and cumulative distribution function of the standard normal distribution respectively.

The acquisition function for probability of improvement (PI) [5] is defined as

$$a_{\text{PI}}(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) = \Phi(\gamma(\boldsymbol{x})), \tag{7}$$

where

$$\gamma(\boldsymbol{x}) = \frac{\mu(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) - f(\boldsymbol{x}_{\text{best}})}{\sigma(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta)}, \tag{8}$$

indicates the deviation away from the best sample. The acquisition function for expected improvement (EI) scheme [6] [7] [8] [14] is defined as

$$a_{\text{EI}}(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) = \sigma(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) \cdot (\gamma(\boldsymbol{x})\Phi(\gamma(\boldsymbol{x})) + \phi(\gamma(\boldsymbol{x})) \tag{9}$$

The acquisition function for the upper-confidence bounds (UCB) scheme [9, 10] is defined as

$$a_{\text{UCB}}(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) = \mu(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta) + \kappa\sigma(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^N, \theta), \tag{10}$$

where κ is a hyper-parameter describing the acquisition exploitation-exploration balance.

Other forms of acquisition functions include predictive entropy search GP-PES [15] that has been extended to constraint BO [16] and multi-objective BO [17], entropy-search GP-ES [18], and estimation strategy GP-EST [19].

2.2. Constrained BO

Constrained BO is a important and natural extension of classical BO. Digabel and Wild [20] summarized and provided a systematic classification taxonomy, QRAK, for derivative-free simulation-based black-box optimization problems. For *known* constraints, the constraints can be realized by either multi-fidelity approach, where the constraint evaluator is considered as a low-fidelity functional evaluator [21], or is penalized directly within the acquisition function [22] inside the sequential sampling search loop for the next query point. Schonlau et al. [23] introduced a generalized EI acquisition function that balances global-local search by incorporating an exponential power with a known constraint penalization scheme. Parr et al. [24] reviewed and compared different constraint handling schemes using synthetic and real-world engineering applications. Picheny et al. [25] proposed an augmented Lagrangian scheme to convert a constrained problem to an unconstrained optimization problem, handling both equality and inequality constraints.

However, the problem is more challenging if the constraints are *unknown*. Several methods have been proposed to incorporate the probability of constraint violation into the acquisition function. Basudhar et al. [26] utilized support vector machines to calculate the feasible probability which is later incorporated in the EI acquisition function. Sacher et al. [27] extended the method of Basudhar et al. [26] by combining classification probability from the least squares support vector machine with the EI acquisition function and proposed another sampling strategy to locate the next sampling point in a sequential manner. Gelbart et al. [28] proposed an entropy search criterion and used the expected improvement acquisition to search for the next sampling location. Hernández-Lobato et al. [16] suggested an alternative mathematical form of acquisition function, to maximize the expected reduction in the differential entropy of the posterior. Gramacy and Lee [29] introduced an integrated expected conditional improvement and increased the number of hyper-parameters to accommodate the class labels. Lee et al. [30] proposed to couple the random forest classifier and maximize the expected constrained improvement, where the new acquisition function is the product of old acquisition function and the predicted probability of the random forest classifier. The main difference between our work and other works such as Basudhar et al. [26], Sacher et al. [27], and Lee et al. [30], is two-fold. First, our proposed method is batch parallel, whereas other methods are sequential. Second, our proposed acquisition function in Equation 13 is generalized and applicable for any probabilistic binary classifier, for example, kNN [31], AdaBoost [32], RandomForest [33], support vector machine [34] (SVM), least squares support vector machine (LSSVM) [35], and GP.

2.3. Batch parallel BO

Typically, to accelerate the optimization process, there are mainly two approaches: accelerating using multiple processor parallel implementation for one simulation, or batch parallelization where each core handles one simulation. One essential observation is that according to Amdahl's law, the former approach is prone to diminishing returns, where the modeling simulation cannot be accelerated after a certain threshold. Thus the later approach, batch parallel BO is more cost-effective and appealing toward computationally expensive engineering models and simulations.

The core idea of batch parallel BO is simply based on the concept of active learning, in which the optimization history is used to reconstruct the response surface and approximate the objective function statistically. Therefore, it is possible to perform multiple functional evaluations at the same time for a single batch, and update the history once the batch is finished. To search for the new points with unknown objective functions, one can rely on the Gaussian probability prediction of the outputs and condition the acquisition function on this Gaussian distribution. For example, Snoek

et al. [14] proposed an integrated acquisition function based on Monte Carlo sampling, called GP-EI-MCMC, to construct a parallel batch, which is conditioned on the Gaussian probability of the observation. Ginsbourger et al. [36] [37] [38], Roustant et al. [39] suggested a multi-points expected improvement q-EI, in which the EI function is conditioned on the Gaussian observation. Marmin et al. [40] [41] provided analytical formula for gradient-based q-EI extension. Letham et al. [42] extended the q-EI framework toward noisy and constrained problems and performed the experiments at Facebook. Wang et al. [43] and Wu and Frazier [44] (q-KG) both suggested a more numerically efficient approach, based on infinitesimal perturbation analysis, to compute the acquisition function of the gradient-based q-EI extension. Shah and Ghahramani [45] extended the predictive entropy search GP-PES [15] toward parallel batch selection policy GP-PPES. Azimi et al. [46] [47] proposed a simulation matching scheme GP-SM [46] and coordinated matching scheme GP-BCM [47] to select input batches that closely match their expected behavior (GP-SM) or the sequential distribution after a certain amount of step (GP-BCM). Azimi et al. [48] also proposed an alternative switching between sequential and parallel mode for BO and analyzed the theoretical bound for the hybrid batch BO method. Desautels et al. [49] introduced a hallucination scheme for batch selection policy GP-BCUB and its adaptive variant GP-AUCB, where the main acquisition is GP-UCB [9] [10], and showed the theoretical bound of the proposed method. The hallucination scheme for a single GP is based on the assumptions of the posterior mean as the prediction during the current batch parallel iteration, and the posterior variance as zero at the particular input location of the batch. The hallucination scheme is essentially the same with the kriging believer heuristic in Ginsbourger et al [37]. Contal et al. [50] proposed a modified version of GP-BUCB, termed GP-UCB-PE, where only one location is selected via GP-UCB acquisition function, and the rest of the batch is configured to locations where the updated posterior variance is maximized. Both GP-BUCB and GP-UCB-PE are constructed based on a crucial observation that the posterior variance only depends on the input locations, but not the actual output. The theoretical bound of GP-UCB-PE is proven to be better than that of GP-BUCB by an factor of $\sqrt{B_{\text{batch}}}$, where B_{batch} is the size of the batch. González et al. [51] proposed GP-BBO-LP method, which assumes the objective function is Lipschitz continuous, infers the Lipschitz constant directly from the GP, and modifies the acquisition function with a local penalizer and a differentiable transformation functions. Kathuria et al. [52] and Wang et al. [53] offered a batch selection policy via determinantal point process GP-DPP, and proved the expected regret bound of DPP-SAMPLE is less than the regret bound of GP-UCB-PE [50]. Kathuria et al. [52] showed that GP-UCB-PE is a special case as DPP-MAX, where the maximization rule is done via a greedy selection rule, and suggested that GP-PPES [45] performs better than GP-BUCB [49]

and GP-UCB-PE [50] approach, and that GP-UCB-PE [50] performs better than GP-SM approach [46]. There are also concerns that GP-UCB-PE and GP-BUCB are too greedy in the batch selection process, and thus prone to be non-optimal with respect to the "immediate overconfidence" measure [52]. Rontsis et al. [54] proposed an alternative acquisition function GP-OEI, where the lower and upper bounds are computationally tractable in high-dimensional space and showed its numerical robustness over q-EI. Nguyen et al. [55] proposed budgeted batch BO, termed GP-B3O, which utilizes the infinite Gaussian mixture model to automatically identify the number of peaks in the underlying acquisition function, and adapts the batch size accordingly based on the approximated acquisition function. Daxberger and Low proposed a novel distributed batch GP-UCB, dubbed DP-GP-UCB [56], to jointly optimize a batch of inputs, as opposed to selecting the inputs of a batch one at a time, and still preserve the scalability in the batch size.

3. Methodology

In this paper, we propose a two-GP batch-parallelization approach to solve the problem, in which one GP models the objective functions, which is referred to as the objective GP, whereas the other models the binary output of the classification, which is referred to as the classification GP. For the objective GP model, the infeasible data points are also included by the interpolation that occurs at each iteration, where interpolation process is performed in two steps. In the first step, *only* the feasible data points are used to construct the objective GP model. In the second step, the constructed objective GP is used to predict the output at infeasible data points. Then the GP model uses the predicted posterior mean $\mu_{\text{objective}}(\boldsymbol{x})$ at those infeasible data points to update again to reflect the true posterior variance in the objective GP, i.e. $\sigma^2_{\text{objective}}(\boldsymbol{x}) = 0$ at the infeasible locations $\boldsymbol{x}$. This interpolation process occurs for every iteration.

Parallel in BO is performed through batch parallelization method, in which a batch of size B_{batch} in every iteration is decomposed into three batches: the first batch for the acquisition hallucination batch of size $B_{\text{acquisition}}$ for the objective GP, the second batch for the exploration of size B_{explore} for the objective GP, and the third batch for the binary classification size B_{classif} of the classification GP, *sequentially* with respect to the batch order. All three batches are constructed by hallucinating the GP models at each iteration, sequentially with respect to each sampling point in the batch, where hallucination means that the observation is assumed to be exactly the posterior mean of the objective GP $\mu_{\text{objective}}(\boldsymbol{x})$ for one iteration, but later on will be corrected once the batch is finished. Also, the feasible probability of the input location $\boldsymbol{x}$ is assumed to be 1, and the posterior variances for both GPs are updated on the chosen sampling location $\boldsymbol{x}$, i.e. $\sigma^2_{\text{objective}}(\boldsymbol{x}) = \sigma^2_{\text{classification}}(\boldsymbol{x}) = 0$.

The input locations in the exploration batch are selected where the updated posterior variance σ^2 is maximal for both GPs. The infeasible data points are assigned as an interpolated value and subjected to change after each iteration. To incorporate the feasibility classifier, the acquisition is reformulated to condition on the probabilistic prediction, and thus become the expected acquisition function.

The technical contribution of this paper is three-fold. First, the advantage of batch BO is extended to incorporate the constrained optimization problems, based on the premise of a HPC infrastructure, using an expected acquisition function that is conditioned on the predicted feasibility from the probabilistic feasibility classifier. Second, the feasibility classifier is continuously improved by the pure exploration batch, so that the feasibility classifier is forced to learn in the unknown regions to improve its predictability performance. Third, a new acquisition is proposed to combine two GPs models.

3.1. Constraints and Feasibility Classification

In this paper, we are concerned with both types of constraints: *known* inequality constraints, which are known in advance and usually physics-based modeled, and *unknown* constraints, which are only known when the functional evaluation actually ocurrs. The infeasibility function of a design can be represented as a set of inequalities $\lambda(\boldsymbol{x}) \leq \boldsymbol{c}$, which in turn can be interpreted as a constrained optimization problem. To penalize the infeasible design, the feasibility checking function is embedded within the acquisition function, to assign zero improvement to all infeasible points if the sampling location $\boldsymbol{x}$ does not satisfy the *known* constraints $\lambda(\boldsymbol{x}) \leq \boldsymbol{c}$. Mathematically, $a(\boldsymbol{x}) = 0$ if there exists an index i such that one of the constraints is unsatisfied $\lambda_i(\boldsymbol{x}) > c_i$. In classical BO at each optimization iteration, the next sampling point is chosen to maximize the acquisition

$$\boldsymbol{x}_{\text{next}} \in \underset{\lambda(x) \leq \boldsymbol{c}}{\operatorname{argmax}}\, a(\boldsymbol{x}; \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}, \theta), \tag{11}$$

where N is the number of observations, and θ is the hyper-parameters of the metamodel.

We propose a variant acquisition function conditioned on the probabilistic prediction of binary GP classification. The acquisition function is updated according to the probabilistic classification augmented in the classification model to obtain the EI function

$$\mathbb{E}[a(\boldsymbol{x})] = 0 \cdot Pr(\text{clf}(\boldsymbol{x}) = 0) + a(\boldsymbol{x}) \cdot Pr(\text{clf}(\boldsymbol{x}) = 1) = a(\boldsymbol{x}) \cdot Pr(\text{clf}(\boldsymbol{x}) = 1), \tag{12}$$

where $\text{clf}(\cdot)$ denotes the binary probabilistic classifier, and $Pr(\cdot)$ denotes the probability mass function (pmf) of the design variables, $Pr(\text{clf}(\boldsymbol{x}) = 0)$ is the probability that the design variable $\boldsymbol{x}$ is infeasible, and $Pr(\text{clf}(\boldsymbol{x}) = 1)$ is the probability that the design variable $\boldsymbol{x}$ is feasible.

3.2. Batch parallelization

A part of this high-throughput BO framework is built on the premise of a HPC. The idea is to bring down the computational runtime of the optimization of expensive high-fidelity simulations through increased parallelism in HPC. For better computational resource allocation in HPC usage, in the extreme case, the diminishing return in the multi-core simulations can be avoided by parallelizing all the simulation in such a serial/sequential manner. That is, m processors can be used to perform m different simulations, where each processor corresponds to a simulation with different input parameters, yielding no diminishing return in Amdahl's law [57]. The calculations of posterior variances σ^2's do not depend on the observations for both GPs, therefore the exploration sampling points are determined after the acquisition hallucination. Desautels et al. [49] hallucination scheme is adopted during the batch selection process.

The interpolation process can be considered as hallucinating the objective GP at infeasible locations, by assigning the posterior means observations, after constructing the objective GP using data only at feasible locations. The purpose of introducing the posterior means as observations per iteration is to truly reflect the posterior variance $\sigma^2_{\text{objective}}(\boldsymbol{x})$ of the objective GP, particularly at infeasible locations. That is, the objective GP is aware of the locations where the functional evaluator fails to obtain observations. In order to do so, the objective GP assumes the posterior mean $\mu_{\text{objective}}(\boldsymbol{x})$ as the observation per iteration at infeasible location, for the same hyper-parameters θ of the objective GP. As the optimization advances in a batch sequential manner, the hyper-parameters θ of the objective GP changes accordingly, and the interpolation process is repeated again. The mechanism does not interfere in locating the next sampling point, because Equation 13 is used to do so, where the feasible probability is predicted using an alternative binary classifier. Otherwise, the posterior variance $\sigma^2_{\text{objective}}(\boldsymbol{x})$ surface is considerably large at infeasible locations, thus $\sigma^2_{\text{objective}}(\boldsymbol{x})$ is not truly reflected at those locations. The incorrect $\sigma^2_{\text{objective}}(\boldsymbol{x})$ would have an impact on the second term of Equation 13, misleading the algorithm into infeasible regions, which we seek to avoid.

3.2.1. Acquisition hallucination batch for objective GP

We describe the point selection process for the first acquisition hallucination batch in three batches, denoted as $B_{\text{acquisition}}$. The "hallucination" term is chosen to explain the effect of assuming an observation in the current batch, which is actually unknown, but is assumed to be known, right at the moment of the batch construction, and this observation will be further corrected at the end of the current batch iteration before moving to the next batch iteration. Technically, all PI, EI, and UCB acquisition functions feature a trade-off between exploitation and exploration within their

analytical formulations. In this process, the point is selected by maximizing the acquisition function, which could be PI, EI, or UCB function. Then, the objective GP is hallucinated to temporarily register the selected point $\boldsymbol{x}$ and its posterior mean $\mu_{\text{objective}}(\boldsymbol{x})$ as the actual observation until the end of this iteration. The posterior variance of the objective GP is updated to reflect the posterior variance $\sigma^2_{\text{objective}}(\boldsymbol{x}) = 0$ at the selected point $\boldsymbol{x}$. Also, the classification GP is hallucinated to realize the sampling point as feasible until the end of this iteration. This hallucination process happens sequentially within the first acquisition hallucination batch, and stops when the number of selected points reaches $B_{\text{acquisition}}$.

3.2.2. Pure exploration batch for objective GP

After the acquisition hallucination batch is constructed, we shift our focus to the second batch, which is the pure exploration batch for the objective GP. The second batch is referred to as B_{explore}. Using the same hallucination approach, the objective GP and the classification GP register the sampling point and the posterior mean μ as the actual observation until the end of the iteration. The points are selected where the posterior variance σ^2 of the objective GP are maximized. After each point in the second batch is selected, both of the hallucinated GPs are updated to reflect the new posterior variances $\sigma^2_{\text{objective}}$ and $\sigma^2_{\text{classification}}$. The process repeats until the number of selected points in the second batch reaches B_{explore}.

3.2.3. Pure exploration batch for classification GP

The last batch focuses on the convergence of the classification GP and is denoted as $B_{\text{exploreClassif}}$. Classification in high-dimensional space generally suffers from the curse of dimensionality, and thus, this batch is devised to force the classification GP to learn in the most uncertain region. In this batch, the sampling points are chosen where the posterior variance $\sigma^2_{\text{classification}}$ of the classification GP are maximized. After each point in the third batch is selected, the classification GP is updated to reflect the new posterior variance $\sigma^2_{\text{classification}}$. The process repeats until the number of selected points in the third batch reaches $B_{\text{exploreClassif}}$.

3.3. Acquisition function

The final form of the acquisition function is modified as

$$a^*(\boldsymbol{x}) = \mathbb{E}[a(\boldsymbol{x})] = I(\lambda(\boldsymbol{x}) \leq \boldsymbol{c}) \cdot a(\boldsymbol{x}) \cdot Pr(\text{clf}(\boldsymbol{x}) = 1), \tag{13}$$

conditioned on the classifier prediction probability and inequality constraint $\lambda(\boldsymbol{x}) \leq \boldsymbol{c}$, where $I(\cdot)$ is the *known* constraint indicator function, which is embedded within the acquisition function, $a(\boldsymbol{x})$

is the typical acquisition function, which can be PI, EI, UCB, or other types. $Pr(\text{clf}(\boldsymbol{x}) = 1)$ is the predicted probability in which the input $\boldsymbol{x}$ is feasible. The novel acquisition function is composed as the product of the classical acquisition function and two feasibility constraints: one is known, and the other is unknown. It is noteworthy that Equation 13 is applicable to any probabilistic binary classifier, i.e. the classifier is not restricted to be GP. The proposed algorithm pBO-2GP-3B seeks next sampling point by maximizing this acquisition function.

3.4. Algorithm outline

Algorithm 1 presents the overview of the pBO-2GP-3B formulation, in which a batch is selected based on two criteria: acquisition hallucination batch and pure exploration batch. The later criteria applies for two GPs, respectively, thus form a three-batch approach.

Algorithm 1 pBO-2GP-3B algorithm.

Input: dataset $\mathcal{D}_n$ consisting of input, observation, feasibility $(\boldsymbol{x}, y_i, c_i)_{i=1}^n$

Input: objective GP $(\boldsymbol{x}, y_i)_{i=1}^n$, and classification GP $(\boldsymbol{x}, c_i)_{i=1}^n$

for $n = 1, 2, \ldots,$ **do**
  construct the objective GP
    collect feasible data subset $(\boldsymbol{x}_i, y_i, c_i = \text{feasible})\}_{i=1}^N$
    construct the objective GP, $\mathcal{GP}_{\text{objective}}(\boldsymbol{x}_i, y_i | c_i = \text{feasible})$, for feasible points
    hallucinate the objective GP, i.e. $y_i \leftarrow \mu_i$, at infeasible points $c_i = \text{infeasible}$
    reconstruct the objective GP using both feasible and infeasible points
  construct the classification GP, $\mathcal{GP}_{\text{classif}}(\boldsymbol{x}_i, c_i)$
  select a batch $\mathcal{B}$ of size $B_{\text{batch}} = B_{\text{acquisition}} + B_{\text{exploration}} + B_{\text{classif}}$ points
    acquisition hallucination: hallucinate 2GPs and select $B_{\text{acquisition}}$ points
    exploration: hallucinate 2GPs and select B_{explore} points where $\sigma^2_{\text{objective}}$ is maximized
    classification: hallucinate $\mathcal{GP}_{\text{classif}}$ and select B_{classif} points where $\sigma^2_{\text{classification}}$ is maximized
  query objective function for objective $y_{n+1}^{(B)}$ and feasibility $c_{n+1}^{(B)}$ for the current batch
  augment dataset $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\boldsymbol{x}_{n+1}^{(B)}, y_{n+1}^{(B)}, c_{n+1}^{(B)})\}$
end for

4. Synthetic examples

Technically, handling the known constraints is much easier than handling the unknown constraints. In the pBO-2GP-3B formulation, both of them would yield zero value in the acquisition

function, but the penalization scheme for the known constraints would make the optimization of the acquisition function more complex. Therefore, only unknown constraints are used in these three synthetic examples. Covariance matrix adaptation evolution strategy (CMA-ES) [58] is used as an auxiliary optimizer to find the location where the acquisition function is maximized.

4.1. 2D Three-hump camel function

In this example, the proposed algorithm is tested on a 2D three-hump camel function on the domain $[-2, 2] \times [-2, 2]$, where the function is

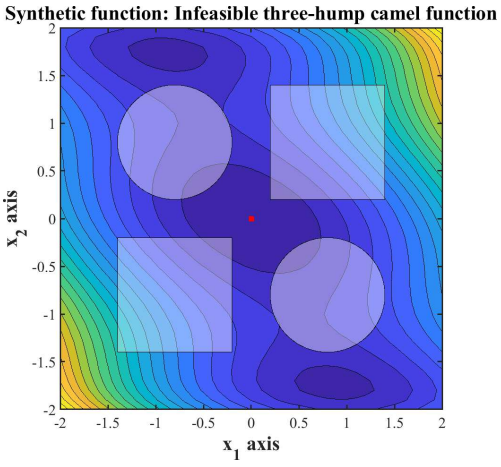
$$f(\boldsymbol{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2. \tag{14}$$

The global minimum of this function is $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0, 0)$. There are four disjoint infeasible regions, including two circles and two squares. The radius of the circle is $\sqrt{0.60} = 0.7746$ for both circles. The dimension of the squares are 1.20 for both squares. There is no known constraint in this numerical problem. Figure 1a shows the contour plot of the unknown constrained three-hump camel function, where the infeasible regions, including two circles and two squares, are shaded. The global minimum is also plotted denoted by a red dot on Figure 1a. Figure 1b presents a batch selection during iteration 5. Readers are referred to color version online. Four sampling locations of the first batch, $B_{\text{acquisition}}$, are plotted as (black) stars. Four sampling locations of the second batch, B_{explore}, are plotted as (green) diamonds. Four sampling locations of the third batch, $B_{\text{exploreClassif}}$, are plotted as (magenta) squares in Figure 1b. As expected, the optimization convergence generally advances due to the sampling locations of the first batch $B_{\text{acquisition}}$. In this case, the second sampling location of the batch $B_{\text{acquisition}}$ is very close with the global optimum of the function.

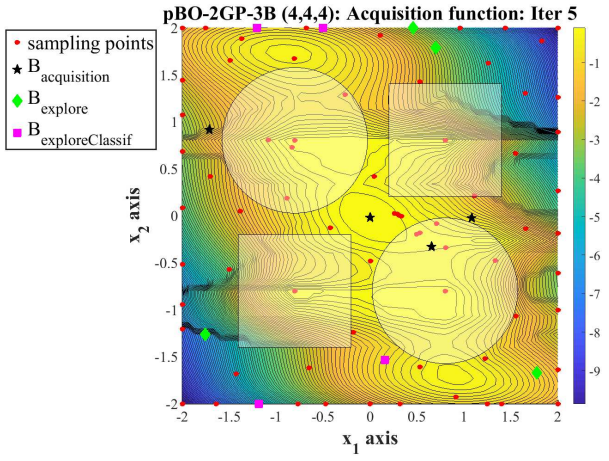
Figure 2a and Figure 2b show the posterior variance of the objective GP $\sigma^2_{\text{objective}}$, before and after the interpolation process, respectively. The $\sigma^2_{\text{objective}}$ is significantly lower after the interpolation process, by hallucinating the objective GP at infeasible locations. Without the interpolation process, the posterior variance $\sigma^2_{\text{objective}}$ is large at infeasible regions, promoting the acquisition function to explore more at these regions. As a result, the algorithm would be misguided into infeasible regions.

Figure 3 shows the convergence of the binary classification problem using the GP classifier. Figure 3a shows the probability of feasibility with 10 data points, whereas Figure 3b shows the probability of feasibility with 300 data points. In order to successfully solve the unknown constrained optimization problem using the binary classifier, the binary classifier must converge to a certain extent, so that the sampling locations can converge to the global optimum.

For the three-hump camel function, the $B_{\text{acquisition}}$, B_{explore}, $B_{\text{exploreClassif}}$ are set to 4, 4, and 4, respectively. Figure 4 shows the convergence plot of pBO-2GP-3B method, where 12 iterations

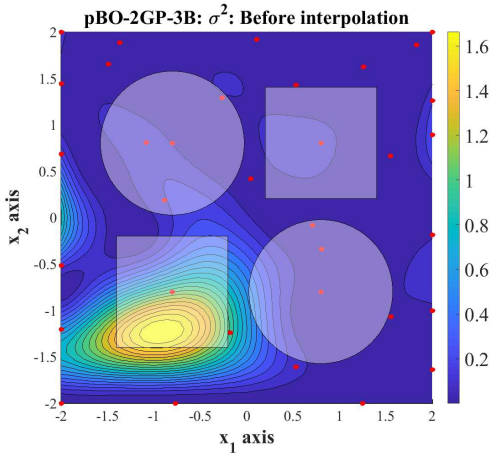


(a) Contour plot and filled infeasible regions of the 2D unimodal three-hump camel function domain. The infeasible regions include two circles and two squares, and are filled to distinguish with the feasible regions. The global minimum is plotted denoted by a red dot.

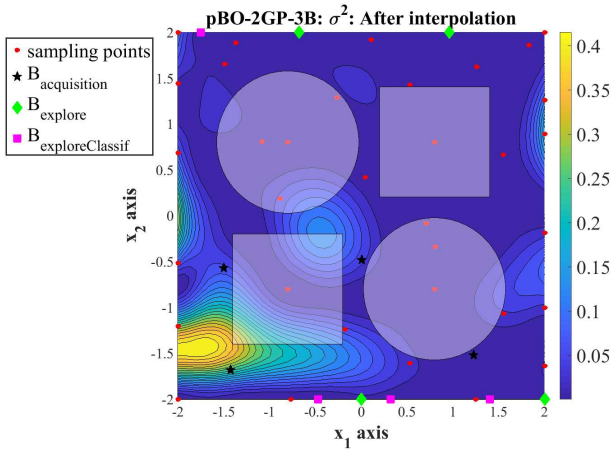


(b) Sampling locations during a batch selection process at iteration 5 for the unknown constrained three-hump camel function, showing 4 sampling locations in each batch.

Figure 1: Three-hump camel function and its infeasible-feasible domain to test pBO-2GP-3B algorithm.

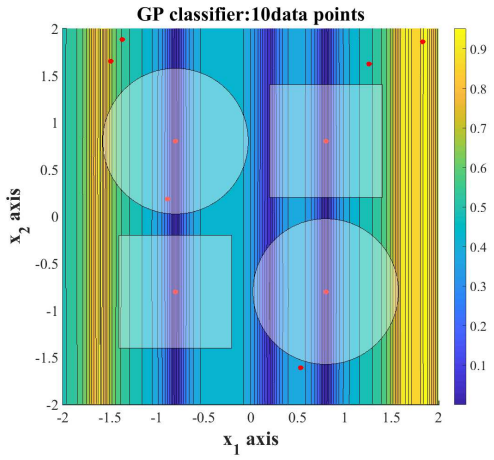


(a) Posterior variance of the objective GP, $\sigma^2_{\text{objective}}$, before interpolation process for infeasible sampling locations.

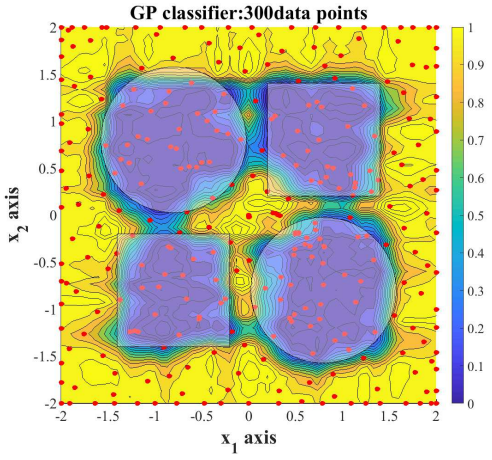


(b) Posterior variance of the objective GP, $\sigma^2_{\text{objective}}$, after interpolation process for infeasible sampling locations.

Figure 2: Illustration of the interpolation process, which hallucinates the objective GP at infeasible sampling locations, to truly reflect the posterior variance $\sigma^2_{\text{objective}}$ at the infeasible locations. Without the interpolation process, the posterior variance $\sigma^2_{\text{objective}}$ is large at infeasible regions, thus promoting sampling in infeasible regions, which is undesirable.



(a) GP classification with 10 data points.



(b) GP classification with 300 data points.

Figure 3: Convergence of GP classifier for binary classification problem: feasible or infeasible.

with batch parallelization are performed, in addition to 6 initial sampling points. The connecting solid line denotes the best solution so far as the optimization advances. The UCB acquisition function is used as the main acquisition $a(\boldsymbol{x})$ in the Equation 13. For the objective GP, the initial hyper-parameters θ are set as $(1, 1)$, whereas the lower and upper bounds for θ are $(10^{-2}, 10^{-2})$ and $(1, 1)$, respectively. The Gaussian kernel is used to construct the objective GP. For the classification GP, the initial hyper-parameters θ is set as $(1, 1)$, whereas the lower and upper bounds for θ are $(10^{-2}, 10^{-2})$ and $(2{\cdot}10^{1}, 2{\cdot}10^{1})$, respectively. The hyper-parameters θ are obtained by the maximum likelihood estimation method. The remaining learning parameters of the toolbox were left to the default value. The exponential kernel is used to construct the classification GP. In iteration 73, pBO-2GP-3B converges to the solution of $f(0.00208764, 0.00180051) = 0.00001572$, showing the fast convergence rate where both the GPs approximate the unknown function very well.

Figure 5 compares the proposed pBO-2GP-3B method with other sequential BO algorithms, including GP-EI-LSSVM by Sacher et al. [27], GP-EI-SVM by Basudhar et al. [26], and GP-EI-RandomForest by Lee et al [30]. Many other binary classifiers, including kNN [31], AdaBoost [32], RandomForest [33], support vector machine [34] (SVM), and least squares support vector machine (LSSVM) [35], are implemented to compare the numerical performance to the proposed pBO-2GP-BO method. Other parallel, unknown constrained BO algorithms are not readily available. Thus, we limit the scope of comparison to other sequential BO methods, which are easy to implement. It is shown that compared to other sequential BO methods, the proposed parallel pBO-2GP-3B converges faster in terms of the computational runtime because of the increasing parallelism, which is controlled by the size of the batches. It is also shown that the UCB acquisition function performs

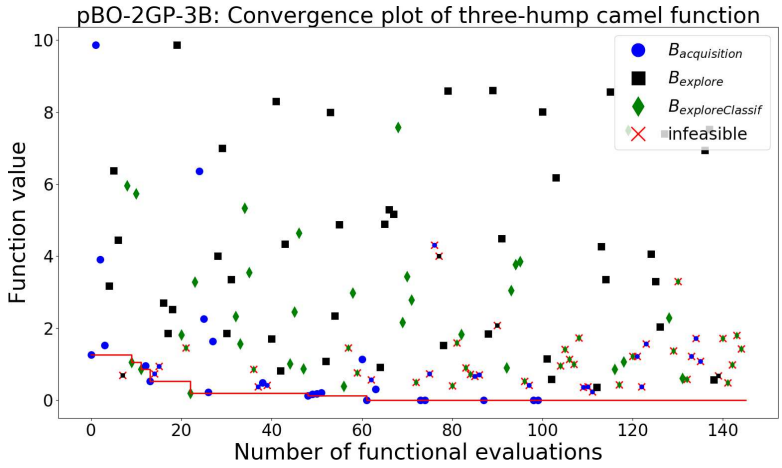


Figure 4: Convergence plot of pBO-2GP-3B on the 2D unimodal three-hump camel function. The feasible data points are denoted as solid circles, whereas infeasible data points are denoted as red crosses. The functional value of infeasible data points are not available, but are evaluated in this plot to aid the visualization.

better, compared to EI acquisition function in the three-hump camel function.

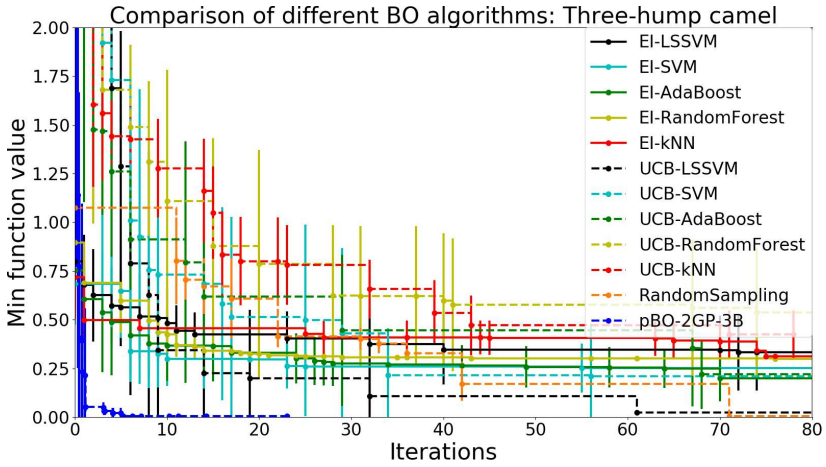


Figure 5: Comparison of different BO algorithms using two acquisition functions and various binary classifiers: three-hump camel function.

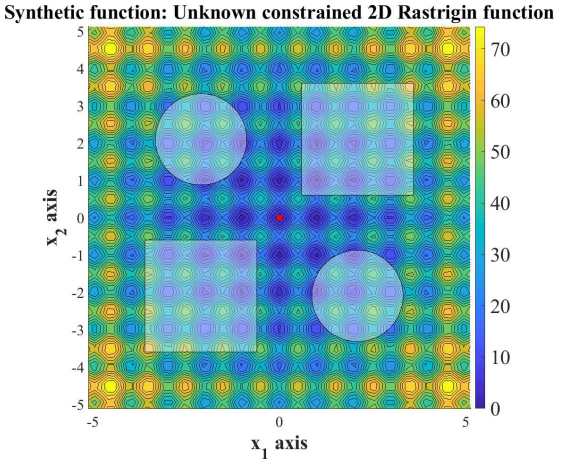
4.2. 2D Rastrigin function

In this example, the proposed algorithm pBO-2GP-3B is tested on a 2D highly multi-modal Rastrigin function on the domain $[-5.12, 5.12] \times [-5.12, 5.12]$, where the objective function is defined as

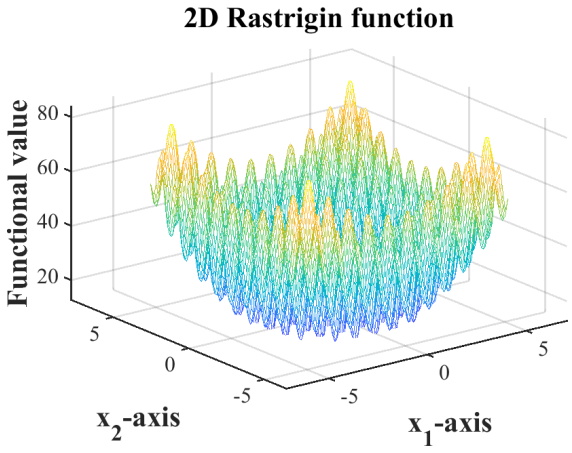
$$f(\boldsymbol{x}) = 20 + \sum_{i=1}^{2} [x_1^2 - 10\cos(2\pi x_i)]. \tag{15}$$

The global minimum of the objective function is $f((0,0)) = 0$. The feasible and infeasible regions are defined similarly to the previous testing example. The dimensions of the squares are 1.50, whereas

the diameters of the circles are $\sqrt{1.50} = 1.225$. Figure 6a presents the contour plot with infeasible regions shaded with the global minimum denoted by a red dot, whereas Figure 6b shows the 3D visualization of the highly multi-modal 2D Rastrigin function.



(a) Feasible and infeasible regions of the 2D Rastrigin function domain. The infeasible regions include two circles and two squares, and are filled to distinguish with the feasible regions. The global minimum is plotted denoted by a red dot.



(b) 3D visualization of 2D Rastrigin function.

Figure 6: 2D Rastrigin function and its infeasible-feasible domain to test pBO-2GP-3B algorithm.

The $B_{\text{acquisition}}$, B_{explore}, $B_{\text{exploreClassif}}$ are set to 5, 1, and 1, respectively. Figure 7 shows the convergence plot, where 12 iterations with batch parallelization are performed, in addition to 6 intial sampling points. The connecting solid line denotes the best solution so far as the optimization advances. The EI acquisition function is used as the acquisition function $a(\boldsymbol{x})$ in the Equation 13. For the objective GP, the initial hyper-parameters θ is set as $(1, 1)$, whereas the lower and upper bounds for θ are $(10^{-2}, 10^{-2})$ and $(1, 1)$, respectively. The Gaussian kernel, mathematically expressed as [59] $k_{\text{Gaussian}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\theta_j(\boldsymbol{x}_j - \boldsymbol{x}'_j)^2\right)$, is used to construct the objective GP. For the classification GP, the initial hyper-parameters θ is set as $(1, 1)$, whereas the lower and upper bounds for θ are $(10^{-2}, 10^{-2})$ and $(2 \cdot 10^1, 2 \cdot 10^1)$, respectively. At the beginning phase of the optimization process, pBO-2GP-3B encounters a substantial number of functional evaluations in infeasible regions, and gradually converges after that. At the iteration 53, pBO-2GP-3B converges to $f(-0.97776279, 0.98550945) = 2.06611835$, showing an acceptable convergence rate for complex functions, as in the case of 2D Rastrigin function.

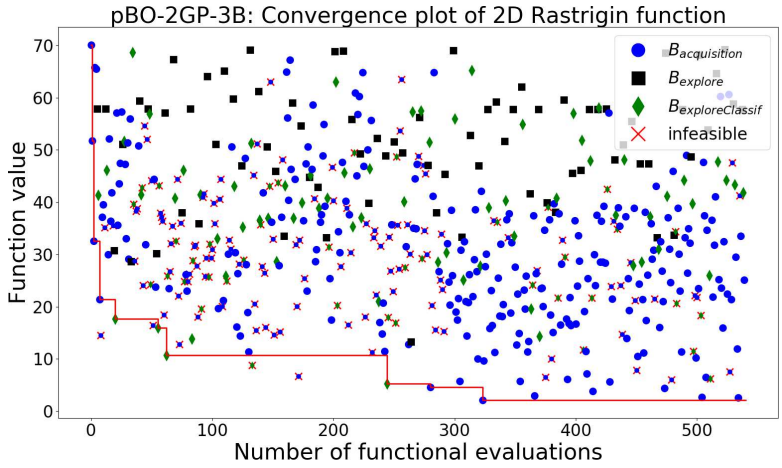


Figure 7: Convergence plot of pBO-2GP-3B on the 2D highly multi-modal Rastrigin function. The feasible data points are denoted as solid circles, whereas infeasible data points are denoted as red crosses. The functional value of infeasible data points are not available, but are evaluated in this plot to aid the visualization.

Figure 8 compares the proposed parallel pBO-2GP-3B, again with other BO methods. The used binary classifiers are kNN, AdaBoost, RandomForest, SVM, and LSSVM. Both acquisition functions EI and UCB are utilized. The comparison shows that because of the parallelism, the proposed pBO-2GP-3B method performs relatively well compared to other sequential BO methods.

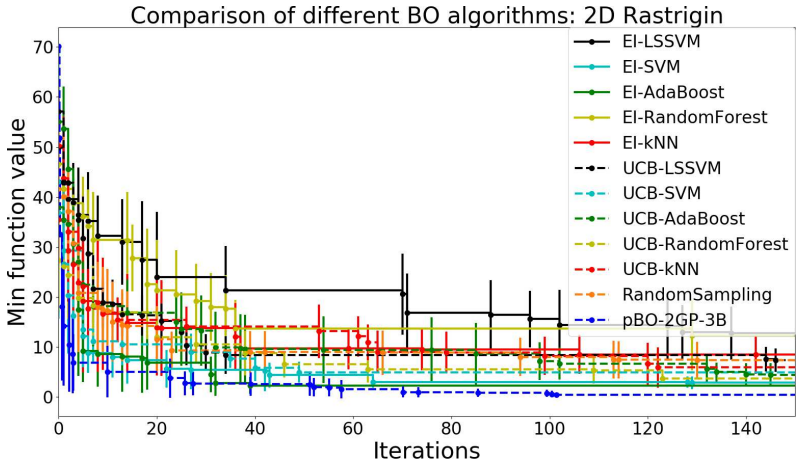


Figure 8: Comparison of different BO algorithms using two acquisition functions and various binary classifiers: 2D Rastrigin function.

4.3. 6D Rastrigin

In this example, the proposed algorithm is tested on 6D Rastrigin function on the domain $x_i \in [-5.12, 5.12]$ for all $i = 1, 2, \cdots, 6$. The global minimum of this function is $f(x^*) = 0$, where $x^* = (0, 0, 0, 0, 0, 0)$. No known constraint is imposed, whereas six unknown constraints with l_2-norm

are embedded blindly to the functional evaluators. Mathematically, they can be described as

$$g_i(\boldsymbol{x}) = \|\boldsymbol{x} - 2.56\boldsymbol{v}_i\|_2 \geq 5, \quad i = 1, \cdots, 6, \tag{16}$$

where $\boldsymbol{v}_i = [-1, \cdots, 1 \cdots, -1]$ is a vector where i-index element is 1, and other elements are -1.

Figure 9 shows the convergence plot of pBO-2GP-3B for the 6D Rastrigin function. The connecting solid line denotes the best solution so far as the optimization advances. EI acquisition function is used to find the next sampling points in the hallucination batch. The $B_{\text{acquisition}}$, B_{explore}, $B_{\text{exploreClassif}}$ parameters in this example are set to 6, 6, and 6, respectively. In iteration 18 (functional evaluation 308), pBO-2GP-3B converges to $f(-0.97884957, 1.97082269, 1.03444007, 1.88768059, 0.19720165, 2.0$ 24.56607326, again, showing an acceptable convergence rate for intermediate dimensionality and a highly complex function.

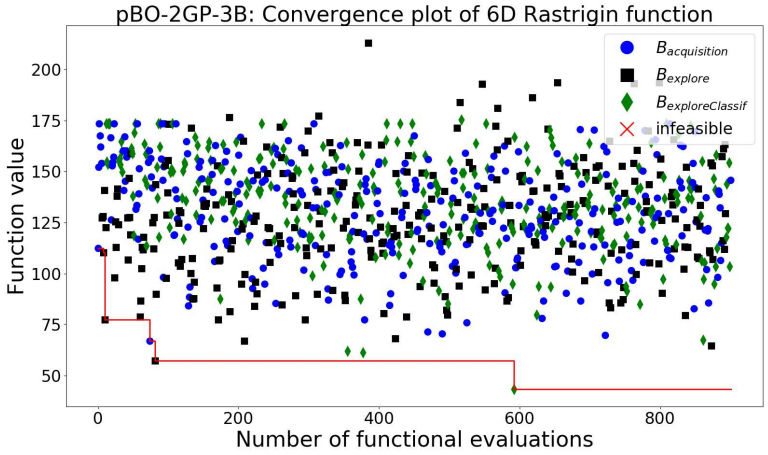


Figure 9: Convergence plot of pBO-2GP-3B on 6D Rastrigin function. The feasible data points are denoted as solid circles, whereas infeasible data points are denoted as red crosses.

Figure 10 presents the comparison plot of different BO algorithms for the 6D Rastrigin function, showing that the pBO-2GP-3B algorithm performs on a par with other sequential BO algorithms. The same classification methods and acquisition functions are used in this comparison.

5. Design optimization of slurry pump impeller

In this section, a case study of CFD simulation to assess the erosion wear rate of the impeller in the slurry pump [60] is used to demonstrate the functionality of the proposed pBO-2GP-3B method in design optimization. Here, a multi-phase solid-liquid CFD model is treated as a black-box function mapping from the design space to the predicted wear performance at certain operating conditions.

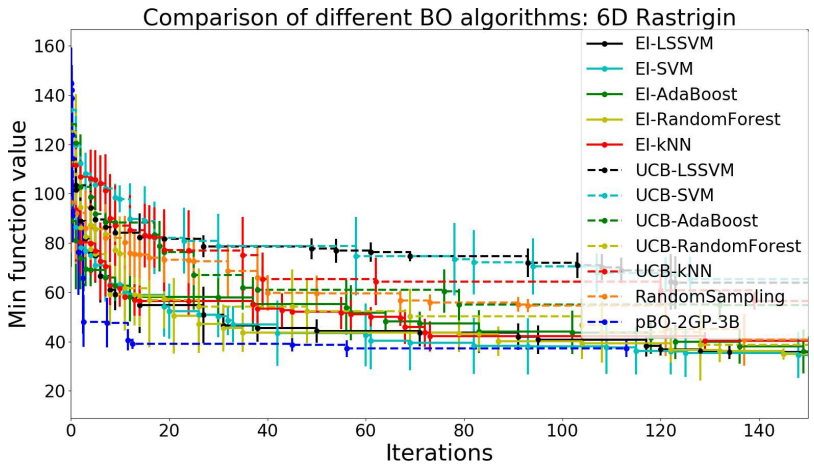


Figure 10: Comparison of different BO algorithms using two acquisition functions and various binary classifiers: 6D Rastrigin function.

In the design space, the geometry of the impeller is modeled using two Bézier patches, where the 3D cylindrical $z-$, $r-$, and $\theta-$coordinates of the Bézier control points are the inputs of the black-box function. For each functional evaluation, a novel geometric model of the impeller is constructed using Bézier patch formulation. Then the CFD erosion wear model is simulated as the functional evaluator. Finally, in the post-process, an average wear quantity is computed on the impeller vanes (suction and pressure sides) to assess the average wear rate.

The CFD erosion wear model for slurry pump impeller is extended and further developed from previous work [61] to capture the multi-size particulate flow. An Eulerian-Eulerian mixture model is used, where volume and time averaged governing equations are derived for the continuity and momentum of the mixture and the individual species in the particle size distribution.

To that end, the goal of this engineering example is to search for the optimal design of slurry pump impeller, which minimizes the predicted erosion wear rate, which is obtained by the post-process of the predictive CFD erosion wear simulation.

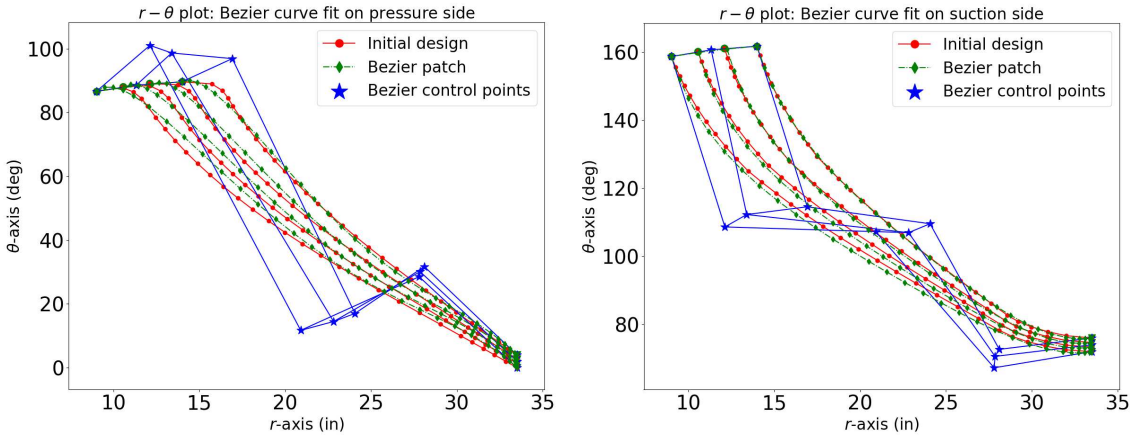
5.1. Parameterization of design variables

The geometry of the impeller vane is discretized and modeled using two Bézier patches, one for the pressure face, and the other for the suction face,

$$\vec{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u) B_j^n(v) \vec{b}_{i,j}, \tag{17}$$

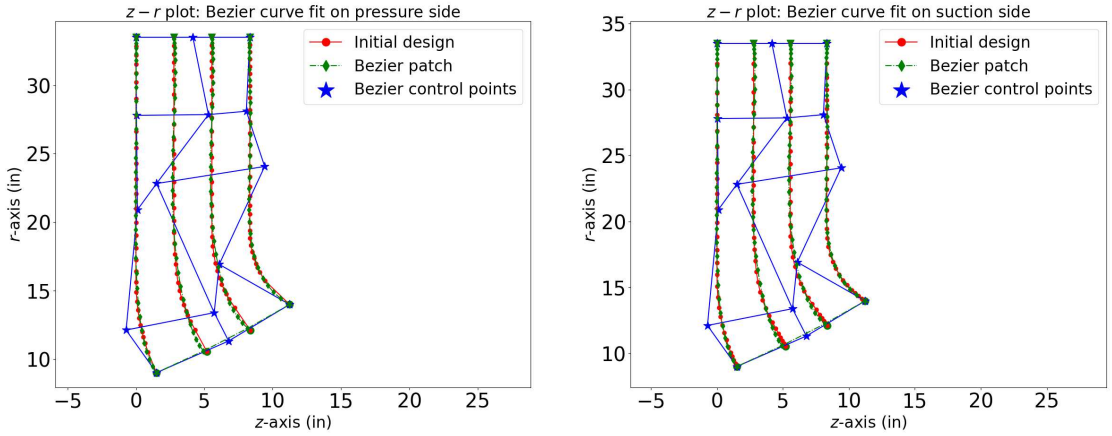
where $B_i^m(u)$ is the Bernstein polynomial of degree m evaluated at $0 \leq u \leq 1$, and $\vec{b}_{i,j}$ is the (i,j)-th control point' coordinates of the Bézier patch. 3D cylindrical coordinate system is used instead of Cartesian coordinate system due to the nature of angular rotation in impeller. Along

the vane direction, m Bézier control points are used, whereas in the transverse direction, n Bézier control points are used. This approach results in $2(m+1)(n+1)$ control points for both pressure and suction faces of the impeller vanes. Figure 11a and Figure 11b show the approximated Bézier curves for $r-\theta$ of the pressure and suction vanes, respectively. Figure 12a and Figure 12b show the approximated Bézier curves for $z-r$ of the pressure and suction vanes, respectively. In Figure 11 and Figure 12, the parameters used are $m=4$ and $n=2$. Thus, the corresponding size of the Bézier patch is 5×3.



(a) $r-\theta$ plot on pressure side. (b) $r-\theta$ plot on suction side

Figure 11: $r-\theta$ plot of impeller meridional plot and its fitted Bézier curve.



(a) $z-r$ plot on pressure side. (b) $z-r$ plot on suction side

Figure 12: $z-r$ plot of impeller meridional plot and its fitted Bézier curve.

The Bézier control points of the pressure and suction vanes share the same $z-$ and $r-$coordinates,

in which some of them are held constant throughout the optimization process due to other physical constraints. The Bézier control points between the pressure and suction vanes vary most in $\theta-$coordinate, because the vane thickness is controlled through $r-\theta$ plot. The initial guess for Bézier control points is constructed from the original design of the impeller. In this study, the $z-$, $r-$, and $\theta-$coordinates of the Bézier control points are the design variables. For each functional evaluation, a set of Bézier control points are sampled, and a novel geometry of impeller is constructed, and evaluated using the predictive CFD impeller model.

5.2. 3D CFD model for slurry pump impellers

In this section, we summarize the previous work of Pagalthivarthi et al. [60] in developing a CFD wear rate prediction for an industrial centrifugal slurry pump impeller. Figure 13 shows the 3D computational domain and boundary conditions of a centrifugal pump impeller in Cartesian coordinate system. Due to the angular symmetry and rotational invariance, only the region between pressure and suction sides of two vanes in the impeller are considered. The computational domain includes a region bounded by two vanes, two shrouds of the impeller, as well as the extensions downstream and upstream of the vanes. A set of governing equations is then derived based on 3D cylindrical coordinates with respect to a reference frame rotating with the impeller. Spalart-Allmaras model [62] is utilized to solve for the turbulent eddy viscosity. Comparison between $k-\varepsilon$ and Spalart-Allmaras turbulence models in 3D multi-size particulate flows is discussed in Pagalthivarthi et al [63]. The inlet velocity boundary condition is applied at the inlet surface B1. The stress free boundary condition is applied at the outlet surface B2. The blade surfaces B3 and B4, the hub surface B5, and the shroud surface B6 are treated as a wall, where Spalding wall functions [64] are utilized. On the surfaces B7, B8, B9, and B10, periodic boundary conditions are applied.

The Streamwise Upwind Petrov Galerkin [65] is utilized to formulate the finite element problem. The system of nonlinear equations for the mixture momentum, solids momentum, and solid concentration are iteratively solved with Newton's method where the under-relaxation factors are added to support the numerical convergence of the solutions. In the implementation, the system of algebraic equations are solved by Intel PARDISO solver [66], which is a shared memory parallel linear solver based on OpenMP. Only the solution of the linear system of equations is parallelized. All inner and outer iterations are repeated until the infinity norms of the absolute error, i.e. the difference between new solution and old solution, of all field variables reach values less than 10^{-5}.

After the CFD solution is obtained, a constitutive material model is utilized to predict the

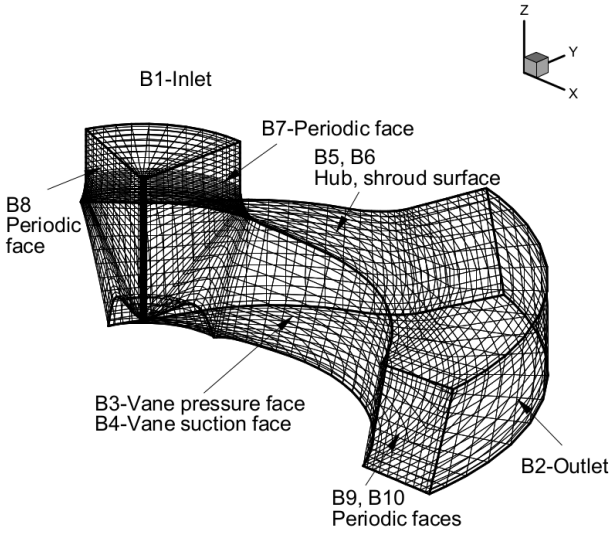


Figure 13: Three-dimensional pump impeller with mesh and its boundary conditions [60].

erosion wear rate [67]. The constitutive wear model is constructed empirically based on experimental measurements and approximates the total wear rate as the sum of impact and sliding wear rates. The impact wear rate is considered as a function of the particle size, impingement angle, local concentration, solids density, and velocity magnitude of the solid particles. The sliding wear rate is expressed as a function of the local concentration, solids shear stress, solids tangential velocity, and friction velocity. The wear coefficients are determined experimentally by curve fitting with respect to experimental wear measurement for a specific wear-resistant alloy. The total wear rate normal to the surface, calculated in the units of $\mu m/hr$, gives the local erosion wear rate.

5.3. Feasibility classification

The unknown constraints in this case study are mainly from two sources. Both unknown constraints can only be assessed using the CFD erosion impeller wear model. The first type of unknown constraints is associated with the non-convergent behavior of the CFD impeller wear model. It is hypothesized that if the design is not plausible, then the CFD impeller wear model would not converge. The second type of unknown constraints occurs when the numerical solution is beyond the physical range. That means, given the design, the CFD numerical solver is ill-conditioned and has converged to an unstable solution.

Different from the unknown constraints, the known constraints can be fairly easy to assess once the inputs, i.e. the coordinates of Bézier control points are given. The known constraints are primarily physics-based validations of an impeller design, which are divided into two known constraints. The first constraint is that the impeller vane must have positive thickness. The second constraint

is that the $r-\theta$ plot must be monotonically decreasing with respect to r. In the implementation, the second constraint is relaxed and not strictly enforced, as long as the deviation is within a user-defined tolerance.

To implement the unknown constraints, each simulation is attached to a wall-clock timer. If the predictive CFD wear model fails to converge to a solution, the design is considered infeasible in the classification GP. The timing threshold is chosen in such a way that a typical impeller wear simulation should converge, assuming a plausible design. For the second type of unknown constraints, a threshold wear rate is imposed. If the predicted wear rate is higher than this threshold, then the design is classified as infeasible in the classification GP.

For the known constraints, the feasibility checking function is implemented based on the formulation of Bézier patch, and is embedded within the acquisition as the known constraint indicator function $I(\lambda(\boldsymbol{x}) \leq \boldsymbol{c})$ in Equation 13. Infeasible designs which fail the feasibility checking function are assigned zero value in the acquisition function.

5.4. Simulations and Results

In this case study, the degrees of Bézier surface patch are $m = 4$ and $n = 2$, resulting in 5×3 Bézier patch for the impeller pressure and suction vanes, respectively. The CFD simulation assumes a constant particle size (or mono-size) and thus the number of species in the particle size distribution is simplified to one. Considering all three $z-$, $r-$, and $\theta-$coordinates, a 33-dimensional input $\boldsymbol{x}$ is formed for each CFD simulation. The optimization procedure is carried out for pump assembly Z0534, at the input operating conditions of $Q = 89637.900$ gpm, $H = 50$ m, $N = 849.000$ RPM, $\eta = 82.400$, $d_{50} = 300\mu$m, $d_{85} = 690\mu$m, $d_{\text{eff}} = 495\mu$m, $C_v = 20\%$, and %BEPQ = 99.6%, where Q is the volumetric flow rate, N is the impeller angular speed, η is the hydraulic efficiency. d_{50}, d_{85} are the 50th and 85th percentile of the particle size distribution. $d_{\text{eff}} = 495\mu$m is the effective particle size, which is calculated as the average of the d_{50} and d_{85} and used as an input for mono-size species in the CFD simulation. $\%BEPQ$ is the percentage of best efficiency point flow rate. The design impeller diameter is 1.7018m, the shroud diameter is 1.7780m, the suction diameter is 0.6604m, and the discharge diameter is 0.6096m. The pump specific speed N_s in US units in 1425.6.

EI acquisition function is used to find the next sampling point in the acquisition hallucination batch. Gaussian and exponential kernels are used for the objective and classification GPs, respectively. A careful lower and upper bounds are chosen for the hyper-parameter of two GPs. The parameters $B_{\text{acquisition}}$, B_{explore}, and B_{classif} are set to 7, 5, 3, respectively. It indicates that at each optimization iteration, 15 CFD models are simulated concurrently in a parallel manner. Covariance

matrix adaptation evolution strategy (CMA-ES) [58] is used as an auxiliary optimizer to find the location where the acquisition function is maximized. A threshold wear rate of $1000\mu m/hr$ is imposed to classify the feasibility of the design. Also, a wall-clock timer of 7 hours is attached to each simulation, by which if the CFD model fails to converge, then the design is classified as infeasible. A fine mesh settings is used during the optimization process, such that the same mesh is used for all simulations.

For each simulation, a post-processing script is devised to extract the quantity of interest as the objective functional value, which is the average wear of the suction and pressure vanes of the impeller for a particular design. The implementation is performed on an Intel Xeon CPU E5-2637 v2 @3.50 GHz with 8 cores, 16 logical processors, and 128 GB RAM on Linux Ubuntu 16.04 platform. Fortran is used as a main language for CFD simulation, where parallelization within the CFD simulation is enabled via PARDISO solver as described above. MATLAB is used as the main programming language to implement the BO based on DACE toolbox [59] [68]. Python and Shell programming languages are used to develop the interface between the pBO-2GP-3B optimizer and the simulation, whereas Fortran with OpenMP, parallelized by the open-source PARDISO [66], is used as the main language for the CFD simulation. 480 initial random sampling points using Monte Carlo method are simulated concurrently to construct the initial GPs in $d = 33$ dimensional design space, where the original design and simulation is the first functional evaluation.

The optimization process is carried out for 89 optimization iterations, where each iteration corresponds to 15 parallel simulations. This results in a total of 1815 simulations of the 3D CFD impeller wear model. The average computational time to construct a batch of size 15 for each iteration is approximately 3-5 hours, depending on the settings of the auxiliary optimizer, which is CMA-ES in this case.

Figure 14 presents the convergence plot of pBO-2GP-3B for the design optimization of the slurry pump impeller within the feasible range. The infeasible designs are classified and assigned zero objective functional value to visualize. Technically, for infeasible designs, the objective functional values either do not exist because the simulations have failed to converge, or lie outside the physically plausible range, which is imposed beforehand. Non-convergent simulations are scattered throughout the optimization process, as the optimizer explores unknown regions. All the best-so-far designs are found by the exploitation batch, as the goal of the exploitation batch is to improve previous designs.

The first functional evaluation denotes the original design, which is predicted at $0.7770\mu m/hr$ for the wear performance. The first iteration starts counting at iteration 480, because there are 480 initial sampling points and the starting index is 0. The optimal design, which is evaluated

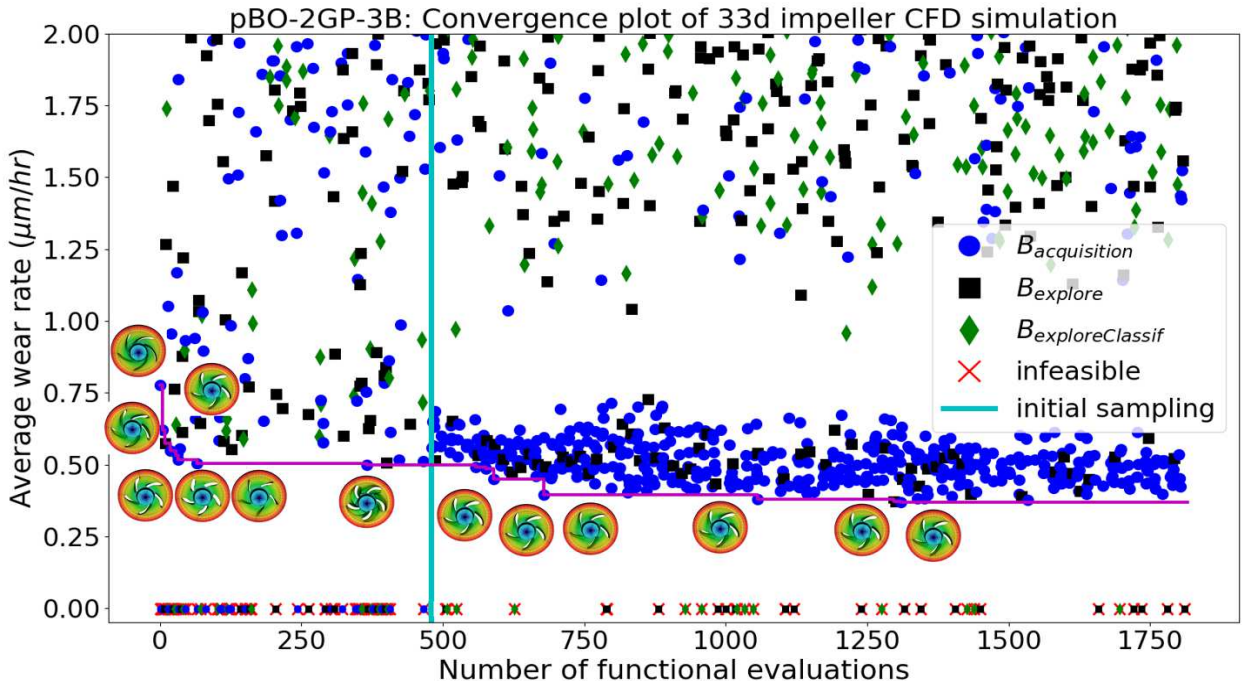
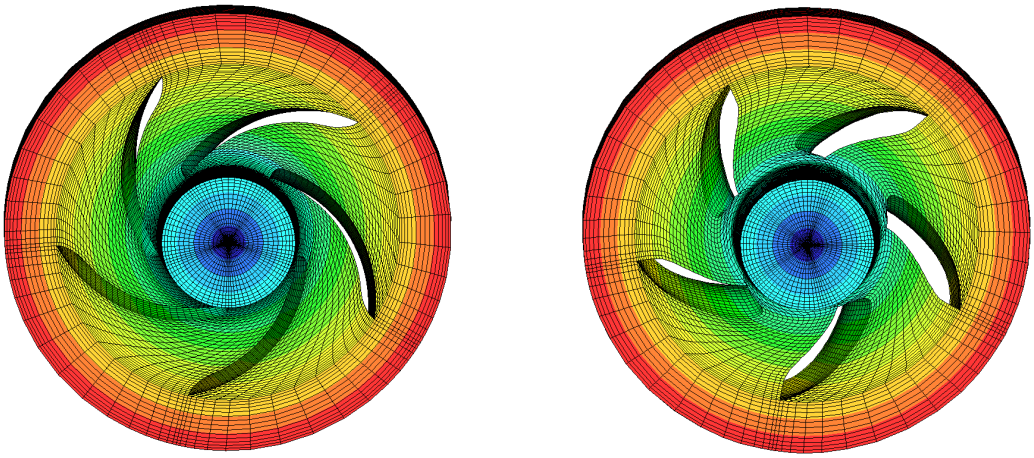


Figure 14: The convergence plot of pBO-2GP-3B for the impeller design optimization problem. The infeasible designs are assigned as zero for the objective GPs and are marked as crosses. The batch sizes are set as $B_{acquisition} = 7$, $B_{explore} = 5$, $B_{exploreClassif} = 3$. 15 CFD simulations are ran concurrently for each iteration.

0.1527 $\mu m/hr$, is found at the iteration 536 in the first batch, at the fifth sampling point according to the hallucination scheme. The average suction side and pressure side wear on the vane, which is the quantity of interest and the objective functional value, is assessed through the multiphase CFD simulation. This value starts dropping from 0.7770 to the following values in the best-so-far solution, sequentially: 0.6203, 0.5732, 0.5485, 0.5316, 0.5181, 0.50141 0.4991, 0.4898, 0.4844, 0.4503, 0.3937, 0.3792, 0.3718, 0.3683 $\mu m/hr$.

Figure 15 presents two meshes, where Figure 15a depicts the original design, and Figure 15b depicts the optimal design of the impellers. The pressure and suction vane designs are structurally different because the Bézier control points are chosen as design inputs, which have been optimized against the average wear rate. In the optimal design, the vane thickness increases in a non-uniform manner, and achieves its maximum thickness near the trailing edge of the vane, before being tapered together at the trailing edge. It is found that the sweep angle in the optimal design is lower compared to the original design. As a result, the length of the vane decreases substantially. The twist angle increases and the outlet angle slightly increases in the optimal design, compared to the original design.

Figure 16 presents the comparison of wear performance between the original and optimal vane



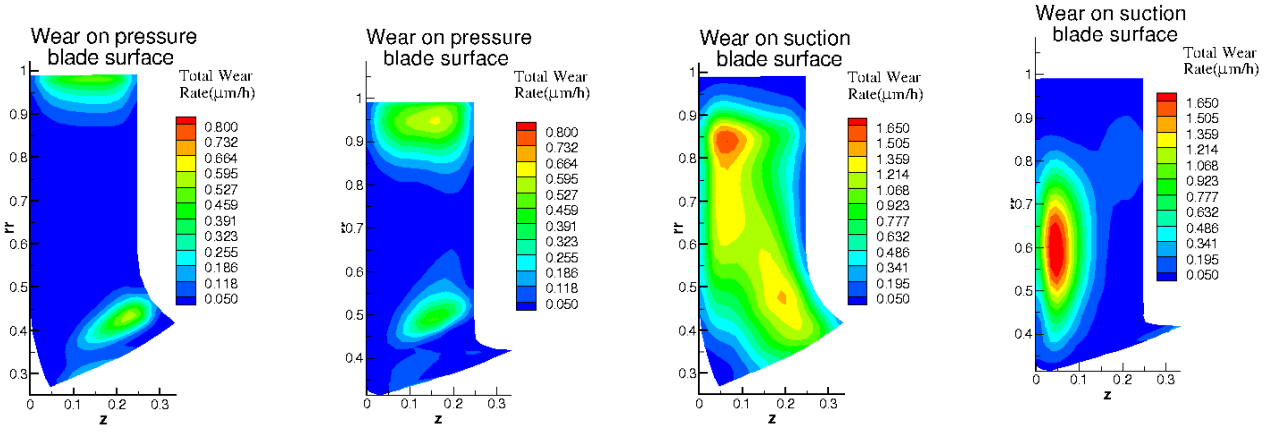
(a) Original design. (b) Optimal design.

Figure 15: Comparison between the CFD meshes of the original and optimal impeller designs.

designs. Figure 16a and Figure 16b compare the total wear plot in $z-r$ coordinates of the original and optimal impeller designs on the pressure side of the vanes. Figure 16c and Figure 16d compare the total wear plot in $z-r$ coordinates of the original and optimal impeller designs on the suction side of the vanes. The result indicates that in the original design, the suction side of the vane is associated with higher wear rates. By optimizing and changing the flow pattern, the suction vane shows significant improvement in wear performance. While the average wear on the pressure vane is essentially similar in terms of magnitude, the average wear on the suction vane is significantly lower because the wear hot spot is localized. Furthermore, the magnitude of the wear hot spot is the same with the wear hot spot in the original design. The average wear rate reduces from $0.7770\mu m/hr$ to $0.3683\mu m/hr$. The optimal design reduces 51.70% of the average total wear compared to the original design. There is a possibility of the BEPQ being shifted as a result of the change in the design, which will be investigated in future studies.

The predicted hydraulic mixture efficiency of the optimal design is 92.49%, compared to 94.18% that of the original design, indicating 2% drop in predicted efficiency. The predicted head on slurry is 56.95 m for the optimal design, compared to 58.18 m for the original design. On the suction side of the vane in the original design, a highly localized maximum wear rate is predicted near the hub, near $rr = [0.8, 0.9]$ scaled region because of the local increase in the particle velocities, as shown in Figure 16c. In the optimal design, the flow is altered so that the wear region is localized on the suction vane with the same magnitude as shown in Figure 16d.

Figure 17 shows the comparison of the mixture velocity and concentration fields on the hub shroud and front shroud of the original and optimal design. In the original design (Figure 17a),



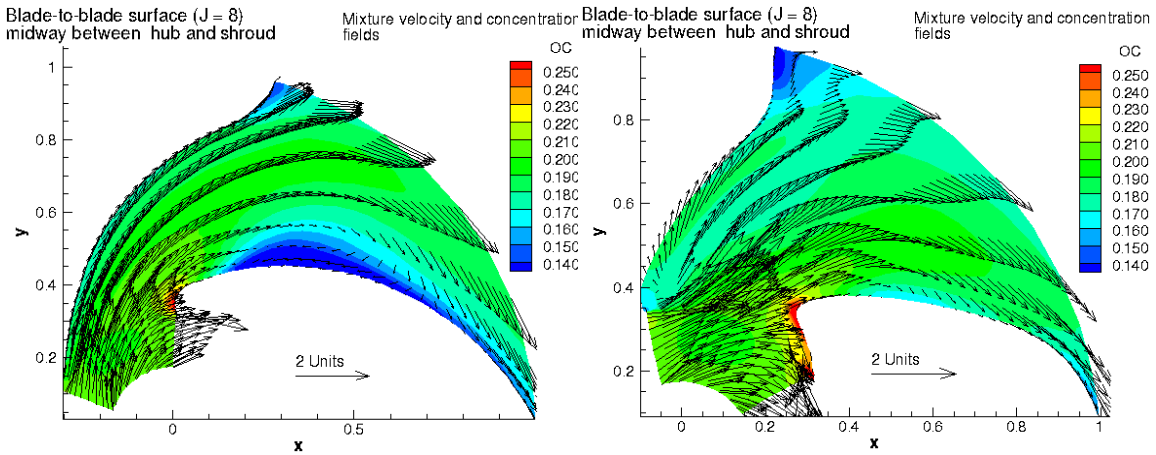
(a) Vane pressure side of original design

(b) Vane pressure side of optimal design

(c) Suction side of vane of original design

(d) Suction side of vane of optimal design

Figure 16: Comparison of total wear between the original and the optimal designs on both sides of the impeller vanes.

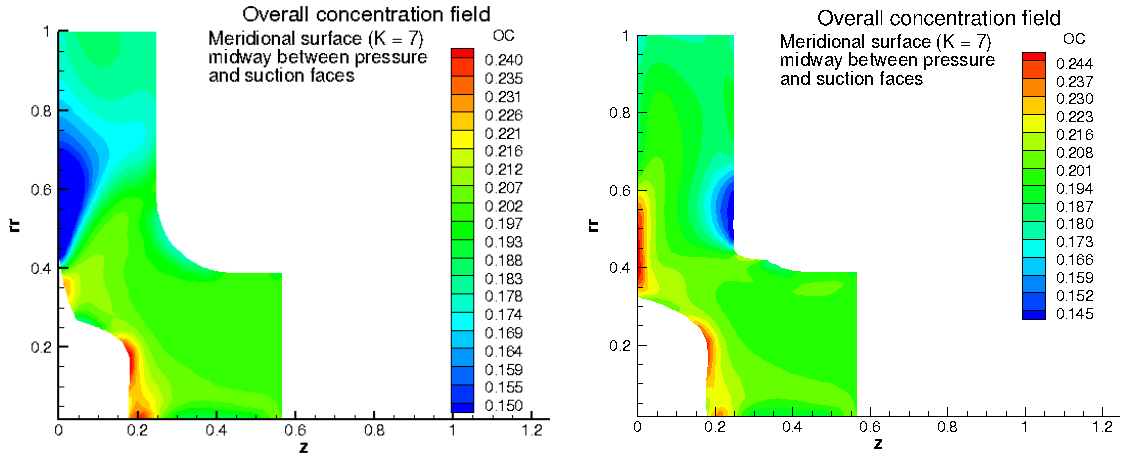


(a) Original design.

(b) Optimal design.

Figure 17: Comparison of mixture velocity and solids concentration between hub shroud and front shroud of the original and optimal impeller designs indicates that in the original design, the particles' maximum velocities occur near the suction vane and produce significant wear on the suction side. On the other hand, in the optimal design, the flow field pattern has been alternated such that the particles achieve its maximum velocities between the pressure and vane. Therefore, the total wear on the suction vane is reduced significantly.

the particles accelerate within the impeller flow field and achieve the maximum velocity near the suction side of the vane, close to the trailing edge. In the optimal design (Figure 17b), the velocity field indicates higher velocities midway between the vanes.



(a) Original design.

(b) Optimal design.

Figure 18: Comparison of overall solids concentration between hub shroud and front shroud of the original and optimal impeller designs indicates the overall concentration is not uniform near the outlet in the original design. In contrast, in the optimal design, the overall concentration is more uniform near the outlet, although there are still some local regions with high concentrations within the impeller.

Figure 18 shows the comparison of the overall concentration in a meridional surface of the original and optimal design in the cylindrical coordinate system, in Figure 18a and Figure 18b, respectively. In the original design (Figure 18a), the overall concentration is less uniform near the outlet and near the back shroud of the impeller, $rr = [0.4 - 0.8], z = [0.0 - 0.1]$. In the optimal design (Figure 18b), the overall concentration field indicates a more uniform behavior, although there is still a small region with low concentration near the front shroud at $rr = [0.45 - 0.60], z = 0.2$. Comparing the region with low concentration in the optimal design (Figure 18b) to that in the original design (Figure 18a), it is obvious that the region is smaller and the concentration field is more uniform in the optimal design.

Figure 19 presents a comparison between two different batch settings, batch-15 and batch-70, after 5 iterations performance. The former one is associated with a total batch size of 15, where the first, second, and third batch sizes are 7, 5, 3, respectively. The later one is associated with a total batch size of 70, where the first, second, and third batch sizes are 40, 15, 15, respectively. The significant improvement of batch-70 over batch-15 in 5 iterations, as shown in Figure 19, demonstrates the efficiency of the proposed pBO-2GP-3B method in HPC environment. It agrees

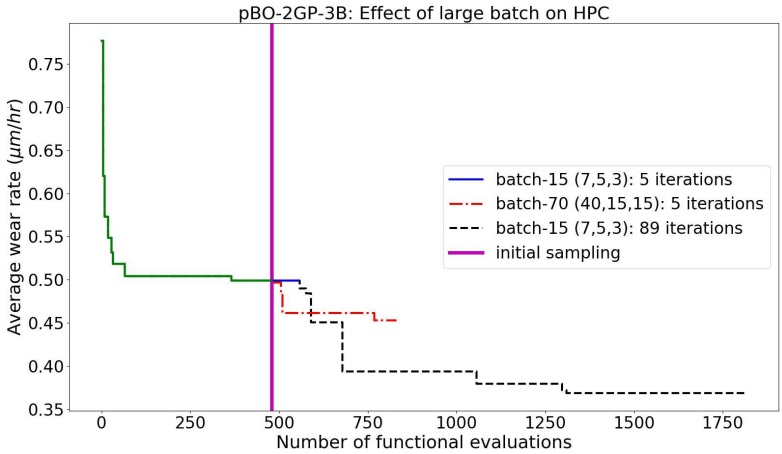


Figure 19: Comparison between different batch settings on the convergence plot shows a significant advantage with large batch size. The initial sampling process occurs up to 480 functional evaluation and is denoted by a solid vertical line. pBO-2GP-3B with batch-15 (7,5,3) stagnates with a solid (blue) line in 5 iterations, whereas pBO-2GP-3B batch-70 (40,15,15) make considerable improvement in 5 iterations.

with the intuition that larger batch is more effective in solving the parallel optimization problem, where the batch size limit depends on the size of the HPC.

However, the improvement is only with respect to time. After 830 functional evaluations, batch-15 is better with respect to observations.

We also note that there is a computational bottleneck in the batch construction process, as the sampling point is selected sequentially within a batch, which may reduce the actual efficiency of the proposed method in practical settings.

6. Discussion

Even though several advanced features have been included, pBO-2GP-3B also has some limitations. One of the drawbacks of the current approach pBO-2GP-3B is the scalability of the algorithms, which suffers in both GPs, as the number of data increases more than 10,000. To cope with the scalability of GPs, a possible solution is to decompose a large dataset into smaller ones and formulate as distributed GPs and weighted linear prediction as in our previous work [69]. Another drawback is the lack of an asynchronous feature, which means that the batch of simulations must be finished, before the optimization process can move on to the next batch. In the CFD engineering example, this idea is implemented by enforcing a maximum time running for of the CFD simulation. The third drawback of the current approach is the sequential nature of the sampling points within a batch. This is particularly important for a HPC, for example, with hundreds to thousands processors available, because the current approach would take more time to construct a batch than simulate it.

This begs for further development for a method which searches the sampling points concurrently, as proposed by Daxberger and Low [56], or an adaptive approach that does not wait until a batch is finished, but rather simulates as the inputs are ready. We also note that too large batch would indeed lead to low optimization efficiency because the limit scale of GPs would be reached in less wall-clock time, yet both GPs do not have enough batch to sequentially control the location of next sampling points. The last drawback of the proposed pBO-2GP-3B method is the simple exploring scheme that maximizes the posterior variance of the objective GP in the second batch B_{explore}, which tends to sample on the border of the domain. The issue is more profound in high-dimensional problem. This can be corrected by more elaborate variance-based sampling scheme, such as Integrated Mean Squared Error (IMSE) [70, 71, 72].

The development time is minimized and thus suitable for fast deployment in industrial applications. Particularly, the development of classification GP for blind constraints is user-friendly, and thus attractive for engineers and designers, who are typically the users of the optimization package in engineering simulation-based problems. Known constraints which are ignored by some users can conveniently become unknown constraints. Such actions reduce the development time for code deployment, yet does not severely affect the functionality of the algorithm, and thus pBO-2GP-3B can be considered as user-friendly. Implementing pBO-2GP-3B in HPC where a job scheduler is available is even easier, since once the design is readily available, one can simply submit the job, and search for the next sampling point while the simulation is in the submission queue.

One remarkable feature of pBO-2GP-3B is the adaptive interpolation of the infeasible data in the objective GP. After each iteration, the objective GP is first reconstructed using only the feasible data points in the dataset, then updated later, assuming the posterior mean as actual observations. The main purpose of this interpolation process is to truly estimate the variance $\sigma^2_{\text{objective}}$ for uncertainty quantification purposes. This is because the computation of the variance $\sigma^2_{\text{objective}}$ only depends on all the locations $\boldsymbol{x}$'s of the dataset and its covariance model, instead of the actual observation y's. Thus by hallucinating the infeasible data points, the variance $\sigma^2_{\text{objective}}$ is accurately quantified. Thus the goal of two pure exploration batches in Algorithm 1 is assured. The interpolation scheme sometimes leads to false optimum in infeasible region. However, the acquisition function is modified in such a way that the binary GP classifier would penalize more in the acquisition function, and thus pBO-2GP-3B would move away from infeasible region.

A critical point in using pBO-2GP-3B algorithm is that the wall-clock time is not simply just the time to perform parallel simulations, but also include others, such as queue time on the HPC if the computational resource is shared, time to construct different batches within one iteration,

time to download and upload data from different sources, etc. Particularly, larger batch size takes longer time to prepare an input to evaluate. If that amount of time is somewhat comparable to the simulation time, there is a combination of batch sizes which yields optimal wall-clock time performance, but the exact answer depends on the computational time of a specific simulation. Therefore, in practice, one is only beneficial from increasing the batch size if the asynchronous feature is enabled. This point, again, refers back to co-optimize and find the sample points within a batch concurrently.

Because of the nature of binary classification problem in the classification GP, the algorithm pBO-2GP-3B performs well if the initial sampling dataset contains at least one feasible and one infeasible data points. Otherwise, the classification GP is not well calibrated, and thus would not yield a good prediction at the beginning phase of optimization process. However, as the optimization advances, the classification GP should converge relatively fast because of the imposed exponential covariance kernel, and thus the estimation of the acquisition function is more accurate once the performance of the classification GP improves.

There are two main differences between our acquisition function described in Equation 13 and Equation (4.2) in Schonlau et al [23]. First, in Schonlau et al. [23], the constraints are unknown and quantifiable, in the sense that the objective GP returns a real value for the model, from which the probability of satisfying the constraints can be estimated. Schonlau et al. [23] method does not handle the case where the functional evaluator crashes, or does not return any response, i.e. non-quantifiable constraints. For our test problems, if the response does not exist in the infeasible domain, the Schonlau et al. [23] method is not applicable. Should an artificial value be used, as in the constant liar heuristic in Ginsbourger et al. [37], the objective GP model would changes sequentially to reflect the constant liar heuristic. As a result, the objective GP model would fail to predict the boundary between infeasible and feasible region, because of the underlying smoothness assumption of the GP formulation. Indeed, it is one of the challenges in engineering domain, where the response does not exist. Second, Equation 13 in our paper handles this problem by borrowing another binary classifier, which classifies whether the domain is feasible or infeasible. Introducing a probabilistic classifier allows the discontinuity between the feasible and infeasible regions, because the objective GP model and the classifier model are independent from each other. A taxonomy of constrained optimization problems is discussed in Digabel and Wild [20] for classifying constrained optimization problems. In this work, the GP is chosen to be the probabilistic classifier, with exponential kernel, instead of Gaussian kernel. However, if the constraints are always quantifiable, then the regression methods to quantify constraints, such as the method of Schonlau et al. [23], should be used to avoid

information loss.

Indeed, one of the most important aspects in solving simulation-based engineering optimization problem is the imbalance dataset, because the classification GP is still functioning as a binary classifier. Too many divergent cases would deteriorate the performance of the classification GP and exhaust the optimization algorithm. Furthermore, since pBO-2GP-3B is a GP-based approach, it also suffers from the scalability. Thus, design the support domain for input variables such that there would be more convergent simulations is recommended. This action can be achieved through examining the results of the initial sampling cases.

There are two ways to obtain a better optimization results. The first one is to expand the support intervals that defines the lower and upper bounds for the input. As the design space grows larger, the global optimum result is guaranteed to be better. However, for simulations that are sensitive to input design variables, such as CFD, this would lead to more divergent cases and exhaust the optimization algorithm, as discussed above. The second one is to increase the number of inputs or degrees of freedom. For example, in the engineering CFD example above, one could also increase m and n parameters, which control the size of the Bézier patch to obtain a better result. This direction is impacted by the curse-of-dimensionality, and thus one should be cautious regarding the trade-off between the approximation error and the curse-of-dimensionality.

Since the method can be thought as a natural extension of GP-BUCB [49] and GP-UCB-PE [50], assuming the batch size of the third batch, the pure exploration for the classification GP, is zero, i.e. $B_{\text{classif}} = 0$, the cumulative regrets is thus bounded by the maximum upper bound of these methods, which is GP-BUCB because GP-UCB-PE has been proved to be better than GP-UCB by a ratio of $\sqrt{B_{\text{acquisition}} + B_{\text{explore}}}$. Indeed, the pure exploration batch of the classification GP plays an important role in preventing convergence to local minima when the infeasible space dominates the high-dimensional input space. While the acquisition function is designed in such a way that the algorithm only samples at feasible regions, the pure exploration batch of the classification GP forces the algorithm to also explore other uncertain regions, with the hope to find some missing feasible regions that have not been found previously. The careful theoretical convergence analysis study remains unsolved and open for future work.

A GP binary classifier is competitive in high-dimensional space binary classification problem with relatively fast convergence rate. Many other binary classifiers, including kNN [31], AdaBoost [32], RandomForest [33], support vector machine [34], and least squares support vector machine [35], are implemented to compare the numerical performance to the proposed pBO-2GP-BO method. However, all of them are cursed by the high-dimensionality of the optimization problems. The GP

classifier is an excellent choice of classifier for two reasons. First, the uncertainty of the GP classifier is naturally quantified by the posterior variance σ^2. Second, because of the uncertainty quantified, the GP classifier is then forced to learn at the most unknown regions. We note that there is not so many binary or multi-class classifiers have uncertainty quantification feature in the context of machine learning.

It is noted that the comparison study is rather limited, since the comparison study only concerns with the convergence as a function of iterations, yet many other norms are also equivalently important. One of the most realistic norms is the actual computational runtime, i.e. wall-clock time, which can be used to measure the efficiency of different optimization algorithms. There are many factors which can alternate the comparison results, including computational platforms, operating systems, hardware, implementation, as well as objective functions, initial designs, constraints, random number generators. To truly compare one optimization algorithm with others, a large scale benchmark study is needed, and thus it is left to future work.

The proposed pBO-2GP-3B method can be easily extended to solve for equality constraints problem. For example, the objective function can be modified to incorporate the equality constraints to a penalized objective function, in which the penalty is applied directly on the unsatisfied equality constraints, as in Picheny et al [25]. GP formulation accounts for some intrinsic noise by assuming that the observations are jointly Gaussian [4]. However, for complex noisy problems where the model response is stochastic, rather than deterministic, a more advanced approach is needed, such as stochastic kriging [73]. The model responses could be the objective functional evaluator, as well as the feasibility condition. This problem remains unsolved and is left to future work.

In the previous numerical and engineering examples, the batch sizes are held constant throughout the optimization process. However, it can be adaptively tuned based on a fixed computational budget. For example, more computational efforts can be focused on reducing the variance σ^2 in the first phase until σ^2 hits a critical number, then more computational efforts can be spent on the acquisition hallucination batch in the second phase for exploitation purpose. This opens up another research question for the dynamic computational resource allocation in the context of HPC, which is a NP-hard problem and will be addressed in future work. Another possible extension is the asynchronous BO, where if a simulation is finished, then a new simulation is readily dispatched.

7. Conclusion

In this paper, we present a novel BO method pBO-2GP-3B, which is aimed at computationally expensive and high-fidelity engineering optimization problems. Two GPs are utilized: the objective

GP for the objective function, and the classification GP for the probabilistic classification. The proposed pBO-2GP-3B significantly brings down the computational cost, based on the premise of available HPC resources. By massively parallelizing the simulation, i.e. each processor handles a single simulation, the diminishing return described in Amdahl's law is avoided, representing a better use for computational resources, which in turn further reduces the wall-clock time.

Additionally, pBO-2GP-3B supports both known and unknown constraints in simulation-based engineering optimization problems, in which the known constraints are defined beforehand, and the unknown constraints are only known once the simulations have been ran. The known constraints are penalized directly into the acquisition using a constraint indicator function, which assigns zero value if the known constraints are violated. On the other hand, the unknown constraints are learned through the classification GP, and the predicted feasibility is then coupled to the acquisition function, using a simple product rule.

The proposed method is demonstrated using a 2D three-hump camel, 2D Rastrigin function, and 6D Rastrigin functions, showing good convergence rate for both numerical problems. It is then applied to a real-world engineering problem for the design optimization of a slurry pump impeller. The predicted result indicates 52.60% reduction in average wear performance, and denotes the potential application of the proposed method in HPC systems.

Acknowledgment

Anh Tran would like to thank Mohamed Garman at GIW Industries for many thoughtful discussions. The project is supported in part by U.S. National Science Foundation under Grant No. CMMI-1306996. The authors is grateful to three anonymous reviewers for improving the paper. This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

References

[1] C. E. Rasmussen, Gaussian processes in machine learning, in: Advanced lectures on machine learning, Springer, 2004, pp. 63–71.

[2] H.-G. Beyer, B. Sendhoff, Robust optimization–a comprehensive survey, Computer methods in applied mechanics and engineering 196 (33) (2007) 3190–3218.

[3] D. Bertsimas, D. B. Brown, C. Caramanis, Theory and applications of robust optimization, SIAM review 53 (3) (2011) 464–501.

[4] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proceedings of the IEEE 104 (1) (2016) 148–175.

[5] 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) (1964) 97–106.

[6] J. Mockus, On Bayesian methods for seeking the extremum, in: Optimization Techniques IFIP Technical Conference, Springer, 1975, pp. 400–404.

[7] J. Mockus, The Bayesian approach to global optimization, System Modeling and Optimization (1982) 473–481.

[8] A. D. Bull, Convergence rates of efficient global optimization algorithms, Journal of Machine Learning Research 12 (Oct) (2011) 2879–2904.

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

[10] N. Srinivas, A. Krause, S. M. Kakade, M. W. Seeger, Information-theoretic regret bounds for Gaussian process optimization in the bandit setting, IEEE Transactions on Information Theory 58 (5) (2012) 3250–3265.

[11] E. Brochu, V. M. Cora, N. De Freitas, A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599.

[12] P. I. Frazier, A tutorial on Bayesian optimization, arXiv preprint arXiv:1807.02811.

[13] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global Optimization 13 (4) (1998) 455–492.

[14] J. Snoek, H. Larochelle, R. P. Adams, Practical Bayesian optimization of machine learning algorithms, in: Advances in neural information processing systems, 2012, pp. 2951–2959.

[15] J. M. Hernández-Lobato, M. W. Hoffman, Z. Ghahramani, Predictive entropy search for efficient global optimization of black-box functions, in: Advances in neural information processing systems, 2014, pp. 918–926.

[16] J. M. Hernández-Lobato, M. Gelbart, M. Hoffman, R. Adams, Z. Ghahramani, Predictive entropy search for Bayesian optimization with unknown constraints, in: International Conference on Machine Learning, 2015, pp. 1699–1707.

[17] D. Hernández-Lobato, J. Hernández-Lobato, A. Shah, R. Adams, Predictive entropy search for multi-objective Bayesian optimization, in: International Conference on Machine Learning, 2016, pp. 1492–1501.

[18] P. Hennig, C. J. Schuler, Entropy search for information-efficient global optimization, Journal of Machine Learning Research 13 (Jun) (2012) 1809–1837.

[19] Z. Wang, B. Zhou, S. Jegelka, Optimization as estimation with gaussian processes in bandit settings, in: Artificial Intelligence and Statistics, 2016, pp. 1022–1031.

[20] S. L. Digabel, S. M. Wild, A taxonomy of constraints in simulation-based optimization, arXiv preprint arXiv:1505.07881.

[21] Q. Zhou, Y. Wang, S.-K. Choi, P. Jiang, X. Shao, J. Hu, L. Shu, A robust optimization approach based on multi-fidelity metamodel, Structural and Multidisciplinary Optimization 57 (2) (2018) 775–797.

[22] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, J. P. Cunningham, Bayesian optimization with inequality constraints., in: ICML, 2014, pp. 937–945.

[23] M. Schonlau, W. J. Welch, D. R. Jones, Global versus local search in constrained optimization of computer models, Lecture Notes-Monograph Series (1998) 11–25.

[24] J. Parr, A. Keane, A. I. Forrester, C. Holden, Infill sampling criteria for surrogate-based optimization with constraint handling, Engineering Optimization 44 (10) (2012) 1147–1166.

[25] V. Picheny, R. B. Gramacy, S. Wild, S. Le Digabel, Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian, in: Advances in Neural Information Processing Systems, 2016, pp. 1435–1443.

[26] A. Basudhar, C. Dribusch, S. Lacaze, S. Missoum, Constrained efficient global optimization with support vector machines, Structural and Multidisciplinary Optimization 46 (2) (2012) 201–221.

[27] M. Sacher, R. Duvigneau, O. Le Maître, M. Durand, É. Berrini, F. Hauville, J.-A. Astolfi, A classification approach to efficient global optimization in presence of non-computable domains, Structural and Multidisciplinary Optimization (2018) 1–21.

[28] M. A. Gelbart, J. Snoek, R. P. Adams, Bayesian optimization with unknown constraints, arXiv preprint arXiv:1403.5607.

[29] R. B. Gramacy, H. K. H. Lee, Optimization under unknown constraints, arXiv preprint arXiv:1004.4027.

[30] H. Lee, R. Gramacy, C. Linkletter, G. Gray, Optimization subject to hidden constraints via statistical emulation, Pacific Journal of Optimization 7 (3) (2011) 467–478.

[31] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[32] T. Hastie, S. Rosset, J. Zhu, H. Zou, Multi-class AdaBoost, Statistics and its Interface 2 (3) (2009) 349–360.

[33] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[34] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intelligent Systems and their applications 13 (4) (1998) 18–28.

[35] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural processing letters 9 (3) (1999) 293–300.

[36] D. Ginsbourger, R. Le Riche, L. Carraro, A multi-points criterion for deterministic parallel global optimization based on Gaussian processes.

[37] D. Ginsbourger, R. Le Riche, L. Carraro, Kriging is well-suited to parallelize optimization, Computational Intelligence in Expensive Optimization Problems 2 (2010) 131–162.

[38] C. Chevalier, D. Ginsbourger, Fast computation of the multi-points expected improvement with applications in batch selection, in: International Conference on Learning and Intelligent Optimization, Springer, 2013, pp. 59–69.

[39] O. Roustant, D. Ginsbourger, Y. Deville, DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization, Journal of Statistical Software 51 (1) (2012) 54p.

[40] S. Marmin, C. Chevalier, D. Ginsbourger, Differentiating the multipoint expected improvement for optimal batch design, in: International Workshop on Machine Learning, Optimization and Big Data, Springer, 2015, pp. 37–48.

[41] S. Marmin, C. Chevalier, D. Ginsbourger, Efficient batch-sequential Bayesian optimization with moments of truncated Gaussian vectors, arXiv preprint arXiv:1609.02700.

[42] B. Letham, B. Karrer, G. Ottoni, E. Bakshy, Constrained Bayesian optimization with noisy experiments, arXiv preprint arXiv:1706.07094.

[43] J. Wang, S. C. Clark, E. Liu, P. I. Frazier, Parallel Bayesian global optimization of expensive functions, arXiv preprint arXiv:1602.05149.

[44] J. Wu, P. Frazier, The parallel knowledge gradient method for batch Bayesian optimization, in: Advances in Neural Information Processing Systems, 2016, pp. 3126–3134.

[45] A. Shah, Z. Ghahramani, Parallel predictive entropy search for batch global optimization of expensive objective functions, in: Advances in Neural Information Processing Systems, 2015, pp. 3330–3338.

[46] J. Azimi, A. Fern, X. Z. Fern, Batch Bayesian optimization via simulation matching, in: Advances in Neural Information Processing Systems, 2010, pp. 109–117.

[47] J. Azimi, A. Fern, X. Zhang-Fern, G. Borradaile, B. Heeringa, Batch active learning via coordinated matching, arXiv preprint arXiv:1206.6458.

[48] J. Azimi, A. Jalali, X. Fern, Hybrid batch Bayesian optimization, arXiv preprint arXiv:1202.5597.

[49] T. Desautels, A. Krause, J. W. Burdick, Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization, The Journal of Machine Learning Research 15 (1) (2014) 3873–3923.

[50] E. Contal, D. Buffoni, A. Robicquet, N. Vayatis, Parallel Gaussian process optimization with upper confidence bound and pure exploration, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 225–240.

[51] J. González, Z. Dai, P. Hennig, N. Lawrence, Batch Bayesian optimization via local penalization, in: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016, pp. 648–657.

[52] T. Kathuria, A. Deshpande, P. Kohli, Batched Gaussian process bandit optimization via determinantal point processes, in: Advances in Neural Information Processing Systems, 2016, pp. 4206–4214.

[53] Z. Wang, C. Li, S. Jegelka, P. Kohli, Batched high-dimensional Bayesian optimization via structural kernel learning, arXiv preprint arXiv:1703.01973.

[54] N. Rontsis, M. A. Osborne, P. J. Goulart, Distributionally robust optimization techniques in batch Bayesian optimization, arXiv preprint arXiv:1707.04191.

[55] V. Nguyen, S. Rana, S. K. Gupta, C. Li, S. Venkatesh, Budgeted batch Bayesian optimization, in: Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, 2016, pp. 1107–1112.

[56] E. A. Daxberger, B. K. H. Low, Distributed batch Gaussian process optimization, in: International Conference on Machine Learning, 2017, pp. 951–960.

[57] M. D. Hill, M. R. Marty, Amdahl's law in the multicore era, Computer 41 (7) (2008) 33–38.

[58] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary computation 11 (1) (2003) 1–18.

[59] H. B. Nielsen, S. N. Lophaven, J. Søndergaard, DACE, a MATLAB Kriging toolbox, Vol. 2, Citeseer, 2002.

[60] K. V. Pagalthivarthi, J. M. Furlan, R. J. Visintainer, Wear rate prediction in multi-size particulate flow through impellers, in: ASME 2013 Fluids Engineering Division Summer Meeting, American Society of Mechanical Engineers, 2013.

[61] K. Pagalthivarthi, G. Addie, Prediction methodology for two-phase flow and erosion wear in slurry impellers, in: 4th International Conference on Multiphase Flow, 2001.

[62] P. R. Spalart, S. R. Allmaras, et al., A one equation turbulence model for aerodynamic flows, Recherche Aerospatiale-French Edition (1994) 5–5.

[63] K. Pagalthivarthi, R. Visintainer, Finite element prediction of multi-size particulate flow through three-dimensional channel: Code validation, The Journal of Computational Multiphase Flows 5 (1) (2013) 57–72.

[64] F. White, Viscous Fluid Flow 2nd Edition, McGraw-Hill New York, 1991.

[65] T. J. Hughes, A. N. Brooks, A theoretical framework for petrov-galerkin methods with discontinuous weighting functions: Application to the streamline-upwind procedure, Finite elements in fluids (1982) 47–65.

[66] O. Schenk, K. Gärtner, Solving unsymmetric sparse systems of linear equations with PARDISO, Future Generation Computer Systems 20 (3) (2004) 475–487.

[67] H. H. Tian, G. R. Addie, K. V. Pagalthivarthi, Determination of wear coefficients for erosive wear prediction through coriolis wear testing, Wear 259 (1-6) (2005) 160–170.

[68] S. N. Lophaven, H. B. Nielsen, J. Søndergaard, Aspects of the matlab toolbox DACE, Citeseer, 2002.

[69] A. Tran, L. He, Y. Wang, An efficient first-principles saddle point searching method based on distributed kriging metamodels, ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering 4 (1) (2018) 011006.

[70] B. Gauthier, L. Pronzato, Convex relaxation for IMSE optimal design in random-field models, Computational Statistics & Data Analysis 113 (2017) 375–394.

[71] R. T. Silvestrini, D. C. Montgomery, B. Jones, Comparing computer experiments for the Gaussian process model using integrated prediction variance, Quality Engineering 25 (2) (2013) 164–174.

[72] B. Gauthier, L. Pronzato, Spectral approximation of the IMSE criterion for optimal designs in kernel-based interpolation models, SIAM/ASA Journal on Uncertainty Quantification 2 (1) (2014) 805–825.

[73] B. Ankenman, B. L. Nelson, J. Staum, Stochastic kriging for simulation metamodeling, Operations research 58 (2) (2010) 371–382.