Boundary Exploration for Bayesian Optimization With Unknown Physical Constraints

Yunsheng Tian [1] **Ane Zuniga** [1] **Xinwei Zhang** [2] **Johannes P. Dürholt** [3] **Payel Das** [2] **Jie Chen** [2]
Wojciech Matusik [1] **Mina Konaković Luković** [1]

Abstract

Bayesian optimization has been successfully applied to optimize black-box functions where the number of evaluations is severely limited. However, in many real-world applications, it is hard or impossible to know in advance which designs are feasible due to some physical or system limitations. These issues lead to an even more challenging problem of optimizing an unknown function with unknown constraints. In this paper, we observe that in such scenarios optimal solution typically lies on the boundary between feasible and infeasible regions of the design space, making it considerably more difficult than that with interior optima. Inspired by this observation, we propose *BE-CBO*, a new Bayesian optimization method that efficiently explores the boundary between feasible and infeasible designs. To identify the boundary, we learn the constraints with an ensemble of neural networks that outperform the standard Gaussian Processes for capturing complex boundaries. Our method demonstrates superior performance against state-of-the-art methods through comprehensive experiments on synthetic and real-world benchmarks. Code available at: https://github.com/yunshengtian/BE-CBO

1. Introduction

Many optimization problems involve the need to optimize black-box functions, where the performance of a sample can only be determined through physical experimentation or time-expensive simulation, which may take even on the order of weeks or months per single experiment. Thus, the total number of evaluations that can be conducted is limited. In this scenario, Bayesian optimization (BO) (Jones et al., 1998; Shahriari et al., 2016) has proven to be a successful approach that guides the search for an optimal solution by iteratively proposing which experiment to evaluate that may lead to the highest performance increase.

In addition to optimizing unknown functions, many practical problems include unknown constraints. For untested samples, it is impossible to determine if a particular combination of design parameters will lead to feasible or infeasible design. In such cases, infeasible regions are typically discovered when pushing the limits of what is physically possible, and the optimal solution lies on the boundary of feasible regions.

For example, consider the problem of designing an airplane wing. The goal is to make it as lightweight as possible while also being durable and structurally strong to sustain the forces. Hence, the optimization needs to reduce the amount of used material until the wing starts breaking under applied forces. The optimal design will be found just before we step into the infeasible range. Similarly, imagine the task of moving a linear stage to a desired position by applying an acceleration followed by deceleration. To optimize this process, the optimizer might progressively increase the voltage applied to achieve faster and more efficient movement. However, if the voltage is continuously increased without considering the motor's limitations, it can lead to excessive heat generation, effectively burning the motor and causing performance degradation or failure. Comparable examples can be found in chemistry and materials science, especially in formulation development where often certain amounts/combinations of ingredients are needed to yield feasible materials, and the actual properties of interest can be measured only for feasible materials. This leads to a situation in which the optimizer is proposing a candidate for which no performance feedback can be obtained, resulting in a BO iteration in which no new information is retrieved. Indeed, while analyzing many real-world benchmark problems, we observe that all these problems have the optimal solution on the boundary between feasible and infeasible designs (further details in Section 5.1).

[1]MIT CSAIL, USA [2]MIT-IBM Watson AI Lab, IBM Research, USA [3]Evonik Operations GmbH, Germany. Correspondence to: Yunsheng Tian <yunsheng@csail.mit.edu>, Mina Konaković Luković <minakl@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Motivated by this observation, we argue that a good model of the boundary and efficient search around it can significantly improve efficacy and lead to the discovery of better solutions compared to exploring the entire design space. We propose a novel BO algorithm incorporating a boundary exploration strategy tailored for exploiting this problem structure, which is prevalent in real-world problems with unknown physical constraints. In case of unknown physical constraints we need to model the space of feasible and infeasible designs. In physical experiments a sample is considered infeasible if it fails before being able to measure its performance, therefore such sample does not provide any continuous value in return. Hence, we train a binary classifier to represent all constraints that may appear in the system. Furthermore, it is important to note the difference between imposed constraints and scenarios where the constraints are unknown. In the former, when the constraints are specified by the user, imposed artificially, the optimum solution may be found in the interior of the feasible region. The case of boundary optima is considerably more challenging than that of interior optima, more even so for BO, where gradients are infeasible to evaluate and the number of function evaluations is subject to a limited budget. For unknown constraints, a surrogate needs to be sufficiently accurate (at least in the vicinity of the optimum) so that the identified solution is sufficiently close to the feasible side of the boundary while not trespassing on the other side.

While existing approaches (Gelbart et al., 2014; Eriksson & Poloczek, 2021; Antonio, 2021) have addressed BO with unknown constraints, our method is the first to explicitly consider the boundary issue and it demonstrates superior performance. In summary, our main contributions are the following:

- We introduce BE-CBO (Boundary Exploration for Constrained Bayesian Optimization) for optimizing black-box functions with a limited evaluation budget and unknown constraints. Our key insight is that accurately modeling and efficiently exploring the boundary between feasible and infeasible designs is crucial in discovering better performing designs.
- To model the unknown constraint boundary, we propose using *Deep Ensembles* and demonstrate its superior modeling capability by comparing against the most common method in surrogate modeling for BO, i.e., Gaussian Processes.
- Comprehensive experiments and ablation studies on synthetic functions and real-world benchmark problems showing the efficiency of our method and state-of-the-art performance on practical setups.

2. Related Work

Constrained Bayesian Optimization BO has proven to be a powerful methodology for global optimization of black-box functions with expensive evaluations (Shahriari et al., 2016). It has demonstrated remarkable success in various applications, including robotics (Lizotte et al., 2007), resource allocation (Hickish et al., 2019), hyperparameter tuning (Snoek et al., 2012), experimental design (Srinivas et al., 2010), and clinical drug trials (Yu et al., 2019). In classical BO, the feasible set $\mathcal{X}$ is assumed to be known and easy to evaluate, e.g., they are either a hyper-cubes or a simplex (Močkus, 1975; Frazier, 2018). Recent research faces a more challenging scenario called constrained Bayesian optimization (CBO), where the feasiblility of optimization variables is unknown or hard to evaluate. In these cases, the evaluation of the feasibility of one solution is also time costly as the objective function. Two slightly different settings are considered in CBO: 1) continuous-valued constraint and 2) binary-valued constraint. In continuous-valued CBO, the constraints take the form of inequality constraint $R(x) \geq 0$, where $R : \mathbb{R}^d \to \mathbb{R}$ is a continuous-valued function. It considers the scenario where both the function value and the constraints' values can be obtained from the experiments, even if the sample falls into the infeasible region. In the binary-constraint case, the outcome of the experiment becomes $\mathbf{1}_{R(x)\geq 0} f(x)$, that we only observe the function value when the sample is feasible ($R(x) \geq 0$), and when the sample falls into the infeasible region, we observe a failure with no function value.

Prior works mainly consider continuous-valued constraints. Gardner et al. (2014) proposes EIC that multiples the Expected Improvement (EI) (Močkus, 1975) with the probability of constraint satisfaction as the acquisition function. PESC (Hernández-Lobato et al., 2015) extends Predicted Entropy Search (PES) (Hernández-Lobato et al., 2014) to the constrained case. ADMMBO (Ariafar et al., 2019) uses ADMM to alternatively optimize the objective value and feasibility of the solution. SCBO (Eriksson & Poloczek, 2021) uses a trust region optimizer to scale up to high dimensional problems with constraints. 2-OPT-C (Zhang et al., 2021) applies a multi-step lookahead approach instead of the standard myopic approach, which encourages sampling the boundary between feasible and infeasible regions, sharing a similar motivation as our method. All methods use Gaussian Process (GP) regressors to model the unknown constraints.

The binary-valued CBO is much less considered in the prior works, though the setup is prevalent in practice for optimizing physical systems. Modeling the binary-valued constraint of the feasibility set is fundamentally different from modeling the continuous-valued constraints. Similar to Gardner et al. (2014), Gelbart et al. (2014) proposes CEI that mul-

tiples the EI with the probability of constraint satisfaction. Lindberg & Lee (2015) uses a slightly different formulation by using asymmetric entropy as the feasibility score for efficient exploration. However, these methods use GP classifiers (GPC) for relatively simple and low dimensional constraints and the efficacy on complicated constraints is not demonstrated. A recent work SVM-CBO (Antonio, 2021) proposes a two-stage approach. In the first stage, it trains an SVM classifier to explore the constraint boundary, and in the second stage, the algorithm performs BO in the feasible region captured by the SVM.

Safe Bayesian Optimization In some scenarios, ensuring the safety and reliability of evaluations have become important concerns when running BO. To address these concerns, safe BO techniques have emerged as a promising solution (Sui et al., 2015; Turchetta et al., 2019; Kirschner et al., 2019). A representative application of safe BO can be found in robotics (Baumann et al., 2021; Berkenkamp et al., 2023) due to the concern of robot hardware damage during optimization. However, although constrained BO and safe BO share a similar spirit, the main objective in constrained BO is to achieve higher performance while safe BO puts more priority on being safe, i.e., conservative on violating the constraint and exploring infeasible design space, which will be less effective in terms of finding the optimum.

Neural Networks for Classification Though GPC has been used in CBO algorithms for modeling the constraints (Bachoc et al., 2020; Lindberg & Lee, 2015), they face the issue of high computational complexity in both theory and lack of supporting software packages, and require a careful choice of the GP kernel, or otherwise the resulting constraint boundaries might be too smooth and lack details. In contrast, Neural Networks (NN) have been widely used and have reached great success in classification problems with both low- and high-dimensional data with complex boundaries, including manually selected features (Swain et al., 2012) and raw data such as images (Lu & Weng, 2007), texts (Minaee et al., 2021), and graphs (Zhang et al., 2018). Despite impressive classification accuracies in supervised learning benchmarks, naive NNs are poor at quantifying predictive uncertainty (Lakshminarayanan et al., 2017; Gawlikowski et al., 2023), thus cannot be directly applied to CBO for constraint modeling. Recently, the prediction uncertainty of NNs has been studied in several approaches, including Bayesian NN (Kwon et al., 2020; Tran et al., 2019) and ensemble methods (Lakshminarayanan et al., 2017), have been proposed. Additionally, the ensemble method has shown strong promise in few-shot training (Beluch et al., 2018) where the number of training samples is small.

3. Preliminaries

We are interested in efficiently optimizing black-box functions with costly evaluations and unknown constraints. Bayesian Optimization (BO) is a powerful framework for such scenario (Jones et al., 1998; Shahriari et al., 2016). In the typical setup, BO addresses the challenge of finding the global optimum of an expensive objective function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$, where direct gradient information is unavailable. In addition, the total number of performed function evaluations is often limited to several dozens.

The optimization process begins with an initial set of N evaluations $Y_0 = \{f(x_i)\}_{i=1}^N$, where x_i is sampled at random from the design space $\mathcal{X}$, to explore the function's behavior. A surrogate model $\hat{f}(x|Y_0)$, typically a Gaussian Process (GP), is then used to approximate the unknown function f using existing observations Y_0. The main strength of BO lies in the strategy for selecting the subsequent evaluations by balancing exploration and exploitation. This balancing is defined with an acquisition function q that guides the search by trading off between exploiting promising regions and exploring uncertain regions of the function. Popular acquisition functions include Expected Improvement (EI) (Močkus, 1975), Entropy Search (ES) (Hennig & Schuler, 2012), Predictive Entropy Search (PES) (Hernández-Lobato et al., 2014), and Thompson Sampling (TS) (Thompson, 1933). In this work, we use EI, but the proposed method is straightforward to generalize to other acquisition functions. EI measures the expected amount of improvement over the current best value f' by observing at the sampling point: $\mathrm{EI}(x) = \mathbb{E}[\max\{0, \hat{f}(x|Y_i) - f'\}]$. When $\hat{f}(x)$ is GP, EI has a closed-form expression (Jones et al., 1998), thus widely used in practice. Finally, the sample for next-step evaluation is selected as $x^+ = \arg\max_x \mathrm{EI}(x)$. The set of observed samples is updated as $Y_{i+1} = Y_i \cup \{f(x^+)\}$, and $\hat{f}$ is recomputed on Y_{i+1}.

Another layer of complexity is added to our problem by incorporating unknown constraints. In this work, we are interested in practical applications where design samples can be feasible and infeasible. Unlike other works with constraints that are continuous functions giving a real value even for samples that do not satisfy the constraints (Eriksson & Poloczek, 2021), in our setup, it is impossible to obtain any objective value when the design is infeasible. These designs are impossible to create and evaluate. They either contradict the forces of physics or fail during the evaluation time. Hence, we model the constraints as a binary function $c : \mathcal{X} \to \{0, 1\}$, where x is feasible if $c(x) = 1$ and infeasible otherwise. Our final goal can be formalized as

$$\arg\min_{x \in \mathcal{X}} f(x) \quad \text{s.t.} \quad c(x) = 1,$$

where both f and c are unknown.

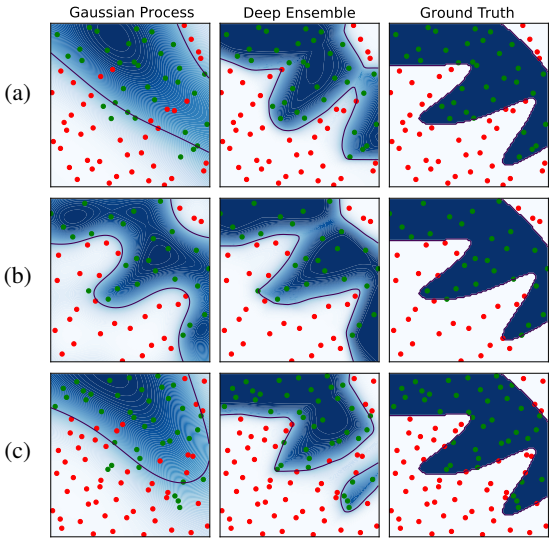

Figure 1: Comparison of modeling constraints between Gaussian Processes (GP) and Deep Ensembles (DE) on the LSQ problem with three sets of random sample evaluations. Green dots represent feasible designs and red dots represent infeasible designs. We observe that DE tends to be more robust than GP in capturing complex boundaries. (a) Failure case of GP; (b) Successful case of GP; (c) Both GP and DE are not fitted well due to the poor sample distribution but DE is closer to the ground truth.

4. Proposed Method

Our proposed method implements the BO pipeline with a few modifications described below. In summary, our method consists of the following steps: We first conduct initial evaluations on a random set of samples. We then fit a GP (as outlined in Rasmussen & Williams (2005)) on the evaluated data to model the objective function. In parallel, we fit another surrogate model to approximate the constraints described in Section 4.1. To select which sample to evaluate next, we optimize an acquisition function EI (see Section 3) with a constrained optimization approach introduced in Section 4.2. Finally, we evaluate the proposed sample and iterate until we reach the budget for the number of function evaluations.

4.1. Modeling Constraints

As described in Section 3, we consider the problem of representing constraints as a binary function, i.e., a classifier that predicts for each design sample whether it is feasible or infeasible. Specifically, we aim to train a classifier $C : \mathcal{X} \subset \mathbb{R}^d \rightarrow [0, 1]$ that measures a probability of sample x being feasible. When $C(x) > 0.5$, sample x is considered more likely to be feasible.

For many practical problems, it is difficult even to find a single feasible solution due to the non-convex nature of the feasible set. Hence, fitting a good surrogate model for the unknown constraints is a delicate task. Many previous works utilize GPs to approximate the constraints based on a given set of evaluations (Eriksson & Poloczek, 2021; Gelbart et al., 2014). However, since we observe that the optimal solution typically lies on the boundary between feasible and infeasible regions (see Section 4.2), having a higher-accuracy surrogate model is crucial. To the best of our knowledge, we are the first to propose the use of *Deep Ensembles (DE)* (Lakshminarayanan et al., 2017; Fort et al., 2019) for this purpose in Bayesian optimization. Please note the difference between ensembles previously used in Bayesian optimization that were in the context of Gaussian Process Ensembles (Wang et al., 2018) and Acquisition Functions Ensembles (Hoffman et al., 2011; Kandasamy et al., 2020).

Deep Ensembles DE is composed of a set of neural networks, often multilayer perceptrons (MLPs), and can be used to model epistemic uncertainty. By training multiple independent MLPs and leveraging the diversity among them, DE enable improved accuracy and enhanced uncertainty estimation. Each MLP in the ensemble is trained on the same dataset but with different random initialization for weights. During inference, predictions from the ensemble are obtained by averaging or combining the individual model predictions. Unlike Bayesian NN, it does not require delicate hyperparameter tuning and long training.

More specifically, in this work, we train N identical MLPs M_{θ_i} independently on all evaluated samples Y_i to model the probability of each sample being feasible or infeasible. The output of each M_{θ_i} is a probability value between 0 and 1 at a given sample point x. Our classifier C is defined as a DE and takes the mean of the predicted value from M_{θ_i}, $\mu_E(x) = \frac{1}{N}\sum_i^N M_{\theta_i^\star}(x)$, where $\theta_i^\star$ are trained weights for each model. In addition, we can compute the variance of the prediction as $\sigma_E(x)^2 = \frac{1}{N-1}\sum_i^N (M_{\theta_i^\star}(x) - \mu_E(x))^2$. To evaluate the feasibility probability $C(x)$ we are essentially computing $C(x) = \mu_E(x)$. See details in Appendix A.1.

Benefits of Deep Ensembles GP-based constraint modeling has several limitations which can be addressed by using DE. First, GPs inherently assume a degree of smoothness in the underlying function they model, as evidenced by the choice of kernel function (e.g., RBF, Matern). This smoothness assumption can lead to difficulties when trying to capture sharp boundaries or discontinuities of the constraint. Second, the kernel structure is chosen a priori and remains fixed throughout the optimization. Although parameters of the kernel can be fitted to data, the form of the kernel itself is fixed and might not be well-suited to represent complex, non-linear boundaries. This limitation

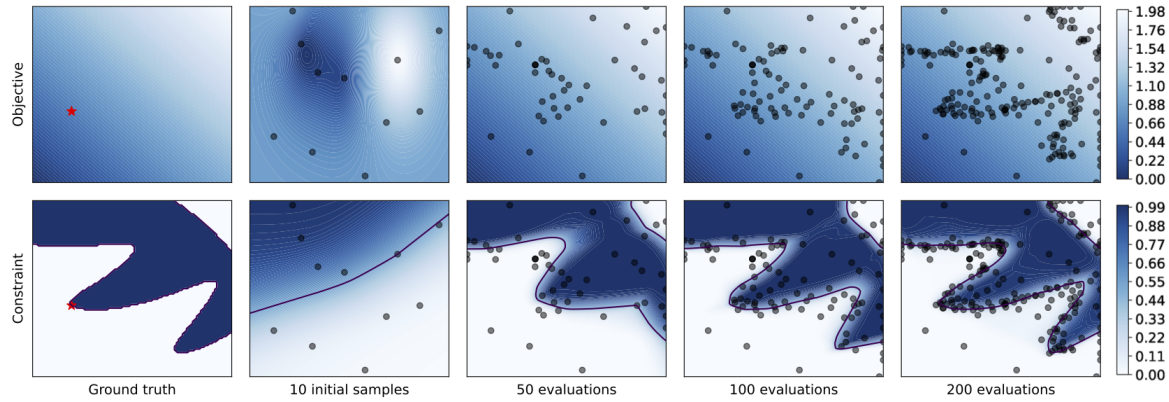

Figure 2: Example evaluation of the LSQ synthetic benchmark problem guided by our algorithm BE-CBO. Top row illustrates the objective function, while the bottom row illustrates the classifier. In the left plot, the ground truth is shown with the boundary and optimal solution. In the second plot, the state of the surrogate is shown for both the constraint and the objective, based on the first initial 10 samples. The following three plots demonstrate the state of the surrogate models for the objective and the constraint after 50, 120, and 200 evaluations.

is discussed in depth in the context of model selection and adaptation (Duvenaud et al., 2013). Thus, compared to GP, DE is more flexible and powerful in representing complex constraint functions (see Figure 1 and (Beluch et al., 2018) for examples). Besides, DE is also fast to train due to easy parallelization and simple architectures (see Section 5 for runtime comparisons). Please refer to Appendix D.1 for more detailed comparison between GP and DE.

Training Deep Ensembles In practice, we find that training DE with the popular maximum likelihood estimation (e.g., using binary cross-entropy loss) as suggested by Lakshminarayanan et al. (2017) leads to poorly calibrated uncertainties and thus deteriorates the BO performance. Instead, we find training DE with variational inference, specifically, evidence lower bound (ELBO), provides better-calibrated uncertainties and improves performance by a large margin, as aligned with observations from Tomczak et al. (2018). This approach requires the model to output continuous latent values that can be transformed to probabilities using Bernoulli likelihood. Please see Appendix A.1.2 for computation details and experimental validations.

4.2. Boundary Exploration

For many practical problems, there is no information about constraints, and discovering the feasible and infeasible regions of space is intertwined with the optimization process. These infeasible regions occur when the system has physical or implicit limitations not artificially imposed by the users. In such scenarios, we discover infeasible regions by pushing the limit of what is feasible as we try to optimize the designs further. This reasoning, and many examples we have seen in practice, lead us to observe that the optimum most frequently (if not always) lies on or very close to the boundary between feasible and infeasible regions.

To ensure exploration around the boundary, we formulate a constrained optimization problem:

$$\arg\max_{x} q(x) \quad \text{s.t.} \quad l(x) \leq C(x) \leq u(x) \tag{1}$$

where q is an acquisition function (see Section 3), C is the classifier (Section 4.1), and l and u are lower and upper bound functions respectively. As discussed in Section 3, we implement EI for the acquisition function, but note that Eq. 1 is not in any way tied to the particular formulation of EI. The proposed method can easily generalize to other acquisition functions. Function l determines how far into the infeasible region can we sample. By allowing the samples to be queried in the infeasible part it encourages pushing the boundary and discovering new regions. However, querying many infeasible samples is especially problematic for a small sampling budget as those samples do not return any value. Hence, we limit l to remain close to the boundary, accounting for uncertainty in the prediction of the boundary, as discussed in the following paragraph. Function u, however, determines how far into the feasible region we can sample. Since all feasible values return additional information for the GP modeling the objective function, we allow exploration of the entire feasible region and set $u(x) = 1$ (recall that the upper bound for C is equal to 1 since it represents the probability). Please note that all the functions are updated in every iteration of BO to fit the new data.

A trivial approach to defining the lower bound around the boundary would be to set a constant value, say $l(x) = 0.4$. This bound leaves a 10% margin on the infeasible side of the boundary to allow some space for exploration and address-

ing the inaccuracy of surrogate model predictions. However, in the initial iterations of BO, the surrogate models for classifier and objective function typically exhibit very high inaccuracy due to the small number of samples for fitting and nonlinear functions to approximate (see, for example, Figure 2). The accuracy can quickly increase with more samples being evaluated. Hence, the bound should also account for this change. It should be wider when the accuracy is low. We propose a *dynamic bound* strategy, where the exploration region around the boundary directly relates to the uncertainty of the fitted classifier. Inspired by UCB (Srinivas et al., 2010), we define the dynamic bound as:

$$l(x) = 0.5 - \sigma_E(x)$$

where $\sigma_E(x)$ is the standard deviation of the Deep Ensembles C for input x.

With this formulation, we incorporate the uncertainty of the classifier into the selection strategy, while the uncertainty of the surrogate of the objective function is captured in the acquisition function (Močkus, 1975). We demonstrate one example of the optimization progress lead by BE-CBO over 200 evaluations in Figure 2. It qualitatively shows that accurate constraint modeling plus active boundary exploration is effective in discovering both the correct constraint boundary and the global optimum.

5. Experimental Evaluation

We conduct comprehensive experiments to evaluate the performance of our methods and compare them to the relevant state-of-the-art methods on both synthetic test functions and real-world benchmark problems.

Algorithms We compare our algorithm to several baseline algorithms described in Section 2 that can be applied to binary constraints: CEI (Gelbart et al., 2014), SCBO (Eriksson & Poloczek, 2021), SVM-CBO (Antonio, 2021), and random search. We implement and compare CEI and SCBO in our Python codebase, built upon the BoTorch (Balandat et al., 2020) BO framework. We conduct SVM-CBO experiments using their framework. Our code will be released with a reproducibility guarantee. See more details in Appendix B.2.

Benchmark Problems Our benchmark includes three synthetic test functions and nine real-world engineering design problems: 2D Townsend function (Townsend, 2014), 2D Simionescu function (Simionescu, 2014), 2D LSQ function (Gramacy et al., 2016), 2D three-bar truss design (Ray & Saini, 2001), 3D tension-compression string design (Hedar et al., 2006), 4D welded beam design (Hedar et al., 2006), 4D gas transmission compressor design (Pant et al., 2009), 4D pressure vessel design (Coello & Montes, 2002), 7D speed reducer design (Lemonge et al., 2010), 9D planetary gear train design (Rao et al., 2012), 10D rolling element bearing design (Gupta et al., 2007), and 30D cantilever beam design (Cheng et al., 2018). Please refer to Appendix B.1 for more detailed descriptions.

5.1. Results and Discussion

Performance We monitor the current best value f' over the number of evaluations performed. For every algorithm, we run experiments with 10 different random seeds and the same 10 initial samples for a total of 200 evaluations. We average the results over these 10 experiments for every algorithm, and plot the mean and the standard deviation across them, as presented in Figure 3.

Test problems are chosen to assess the efficacy of the method in handling a wide range of function characteristics, including concave, convex, disconnected, and varying complexities in design space. As shown in Figure 3, BE-CBO consistently exhibits top performance and low variance compared to other algorithms that oscillates across different problems. Figure 4 shows qualitative comparisons on sample distributions of algorithms at different time steps when evaluating on the Simionescu function. Our method effectively explores the boundary region, classifies the complex constraint landscape well, and discovers both optima of the function. Besides, our method also places samples occasionally on other parts of the boundary to capture other local optima that are potentially better than the best discovered optima. See more qualitative examples in Appendix C.1. We benchmark algorithm runtime in Appendix C.2, which shows that BE-CBO exhibits stable runtime across all problem dimensions.

Overall, BE-CBO performs robustly on all of our benchmark problems, and we have not observed a case in which it completely fails or performs much worse than other baselines. However, we noticed that BE-CBO is slightly outperformed by SCBO on our 30D cantilever beam design problem. We speculate that this is because SCBO's key advantage in high-dimensional optimization is its usage of TURBO (Eriksson et al., 2019), a well-designed optimizer for high-dimensional BO based on the idea of using several local surrogates instead of a single global surrogate, along with its restart strategy when TURBO gets stuck. In BE-CBO, we adopt a relatively standard SLSQP optimizer with a global surrogate for acquisition optimization, which may be suboptimal for practical high-dimensional optimization. Although our core idea of BE-CBO is independent of the choice of acquisition optimizers, we believe that integrating effective ones such as TURBO with BE-CBO is technically feasible and promising for combining the best of both worlds.

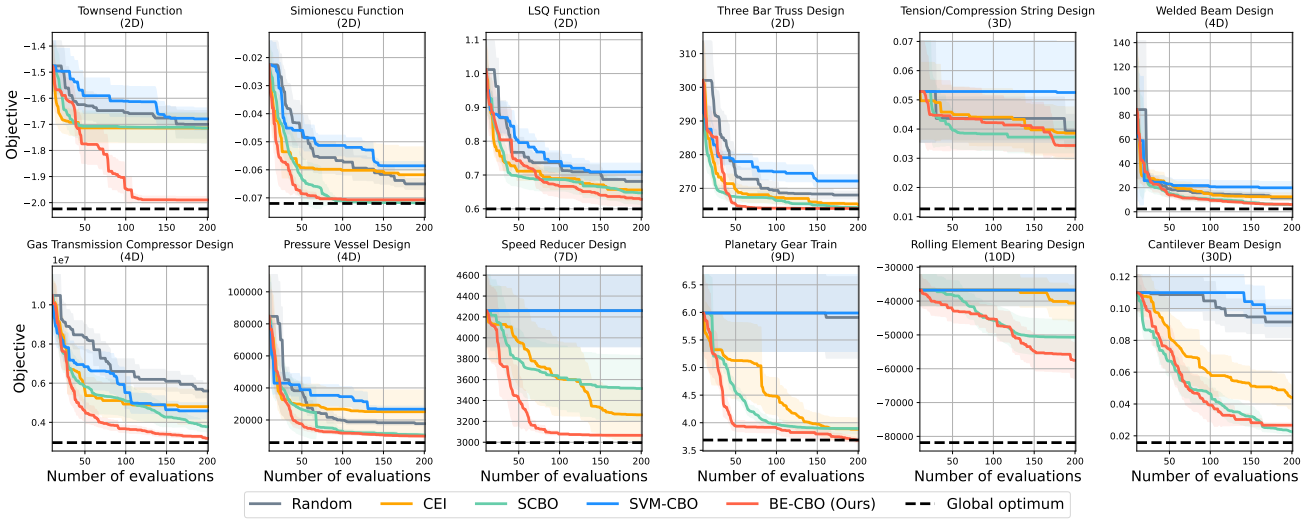

Figure 3: Quantitative comparison of different algorithms including BE-CBO on all benchmark problems. The current best value is shown w.r.t. the number of function evaluations. Each experiment has 10 initial random samples and 200 total evaluations. The curve is averaged over 10 different random seeds and the standard deviation is shown as a shaded region.

Feasibility Ratio We measure the *feasibility ratio* in Figure 5 by comparing the number of feasible and infeasible points sampled by each algorithm. This is to keep track of how many samples did not retrieve any information on the objective value due to falling in the infeasible region. These samples do not improve the objective surrogate model, but they do improve the classifier and push the boundary further.

While the feasibility ratio has not been studied in CBO, there is abundant research on constrained evolutionary algorithms that focuses the search around the boundary and demonstrate improved performance (Isaacs et al., 2008; Ray et al., 2009; Jiao et al., 2019). Jiao et al. (2019) empirically shows that 50% is the most reliable feasibility ratio for evolutionary algorithms on global optimization problems with different function characteristics.

Interestingly, our algorithm outperforms the ones focusing the search in the feasible region (SCBO) or penalizing points less likely to be feasible (CEI). It is worth pointing to the examples where our algorithm is able to discover larger feasible regions and better performing designs due to the improved accuracy in constraint modeling and narrowed search around the boundary on the infeasible side.

Boundary Optimality We focus on practical problems where the constraints are unknown and imposed by a physical system. The design keeps improving until it reaches the limits of the physical system. Hence, the optimum is often found on the boundary of feasible region. In the prior literature, this observation has been exploited in multiple contexts, such as designing effective constrained evolutionary algorithms (Isaacs et al., 2008; Ray et al., 2009; Liu et al., 2021), designing test functions that resemble real-world problems (Sergeyev et al., 2021), and active set methods in constrained optimization (Gratton et al., 2011).

We acknowledge that this property does not hold for every problem, such as synthetically constructed functions with interior optima and problems with user-specified constraints. Therefore, to test the generality of our method, we modified the three synthetic benchmark problems such that their optima are shifted to inside the feasible region. Results in Appendix C.3 shows that BE-CBO successfully reaches the optimal region and exhibits competitive performance.

In addition, we investigate if active constraints exist at the global optima of all nine real-world problems in our benchmark, which cover most of the mechanical design problems with continuous parameters and unknown inequality constraints collected in a real-world non-convex constrained problem suite (Kumar et al., 2020). For problems that do not have a known global optimum, we run CMA-ES (Hansen et al., 2003) with millions of evaluations to approximate a global optimum. As a result, we found that all nine problems have the global optimum on the constraint boundary, i.e., at least one active constraint. In those scenarios, the active constraints represent physical limitations that determine the upper bound of the mechanical design's performance. See more details of these experiments in Appendix B.1.3.

5.2. Ablation Studies

To systematically study how effective each component of BE-CBO is, we conduct comprehensive ablation study ex-

[1]Note that SCBO applies a Gaussian copula transform to the objective function to magnify the optimum region.

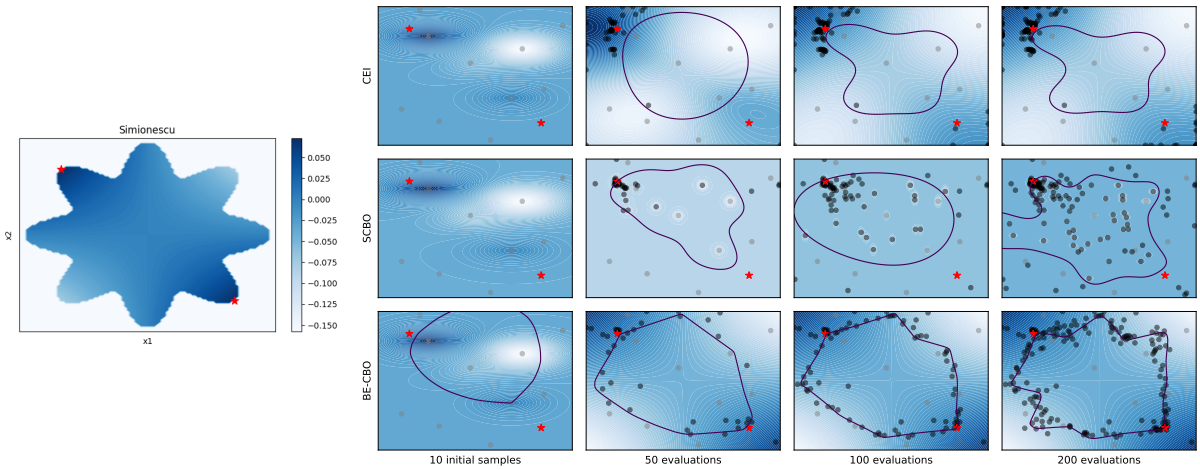

Figure 4: Qualitative comparison of sample distributions from different algorithms on the Simionescu benchmark. Left: The true function landscape, where darker color means a higher objective value and the white region is infeasible. Right: The predicted function landscape (top: CEI, middle: SCBO[1], bottom: BE-CBO) where darker color means a higher objective value and the contour represents the feasibility boundary (feasible inside, infeasible outside). Grey: initial samples, black: evaluated samples guided by each algorithm, and red: the global optima.

periments around the two key contributions of BE-CBO: Deep Ensembles (DE) and boundary exploration. Please see Appendix D for details of each experiment. We aim to answer the following key questions:

1. How does DE improve classification accuracy upon GP? In Appendix D.1.1, we compare the classification performance between DE and GP when being used as the classifier in BE-CBO. By measuring the *Balanced Accuracy* metric from 10K random samples in the space in each iteration, we observe DE's superior and stable performance compared to GP on all benchmark problems.

2. How does DE improve BE-CBO's performance upon GP? Appendix D.1.2 shows that by switching DE to GP in BE-CBO, we observe significant performance drop on half of the benchmark problems. It confirms that having more accurate constraint classification helps improve the BO performance, especially for our boundary exploration strategy where boundary accuracy is important, and for real physical problems where boundary is crucial to explore.

3. Is DE sensitive to hyper-parameters? We find that the overall performance of DE is insensitive to hyperparameters within wide ranges by experimenting different numbers of MLPs, numbers of hidden layers, numbers of neurons in a layer, and learning rates in Appendix D.2.

4. How does BE-CBO perform with other acquisition functions than EI? We use EI as the acquisition function in BE-CBO mainly because of its popularity and good balance between exploration and exploitation. However, BE-CBO is compatible with any standard acquisition functions and is not tied to EI. One can also use upper confidence bound (UCB) when more explicit control over the exploration is preferred. In Appendix D.3, we compare the performance of using EI and UCB for BE-CBO respectively, and the results suggest that they perform similarly well on our benchmark problems overall with some differences in particular problems.

5. How does boundary exploration perform compared to other forms of constrained acquisition functions? We investigate two variants of BE-CBO where similar constraint treatments to CEI and SCBO are applied instead of the proposed boundary exploration. In Appendix D.4, the results show that our boundary exploration is the most robust strategy in coupling the constraints with the acquisition function.

6. Conclusion and Future Work

We introduced a novel CBO method that aims to find a global optimum of an unknown function under unknown constraints. We specifically address the problem in which the infeasible designs cannot be evaluated, hence returning no information for the objective function nor a continuous value for the constraint. Such problems are common, for example, in chemistry and materials science in which specific combinations of synthesis parameters may lead to invalid materials, making it impossible to measure any output quantities. This results in a BO iteration with no information gain which can cause the optimizer to get stuck in the worst case. Coupling the BO optimizer with a classifier that can distinguish the feasible from the infeasible regions allows the optimizer to also draw information from failed experi-

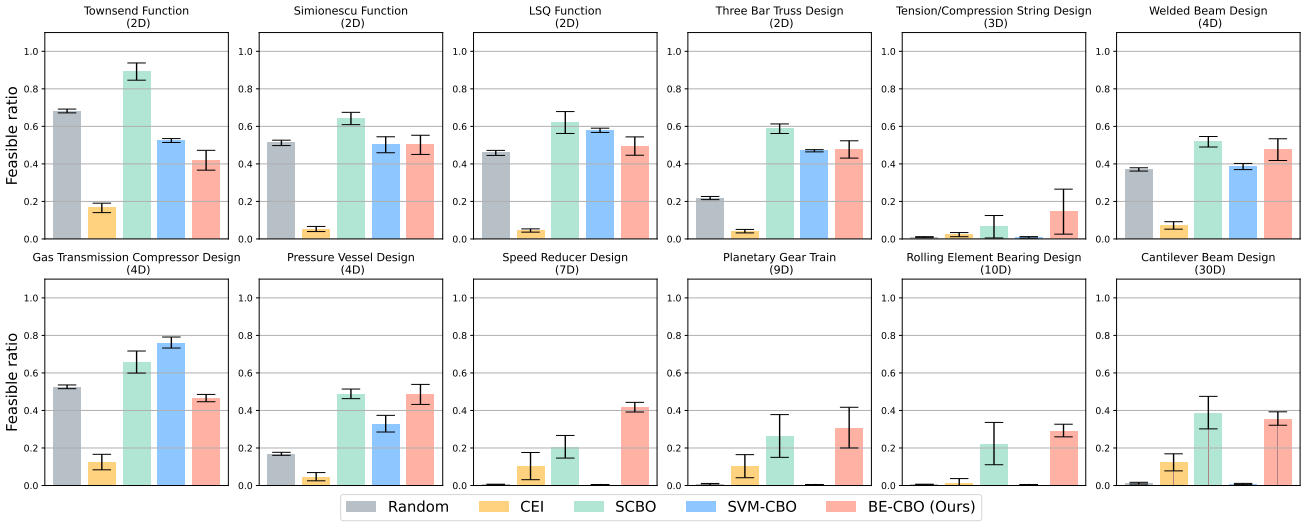

Figure 5: Feasibility ratio comparison of different algorithms including our BE-CBO on synthetic test functions and real-world problems. The error bar charts represent the mean and variance over 10 different random seeds.

ments and to converge quickly to the actual optimum most often located at the constrained boundary, paving the way to more efficient experimentation.

We employed *Deep Ensembles* for the classification of the constraints representing binary feasible/infeasible regions. To the best of our knowledge, we are the first to propose using DE for modeling unknown constraints in BO. DE exhibits improved accuracy in the classification, allowing our method to focus the search for optima on the boundary. However, we share the same observation as Li et al. (2023) that using DE for modeling objectives (i.e., regression) does not lead to improved performance. Finally, we present a boundary exploration strategy that efficiently discovers better designs. We performed extensive tests on both synthetic test functions and real-world problems. We found that our approach outperforms other methods and works particularly well on practical problems.

We acknowledge two main limitations of our work. Firstly, our algorithm was exclusively tested in a continuous design space, focusing on a single objective. To broaden its applicability, a future direction would involve extending the algorithm to handle categorical variables and explore multi-objective BO. Secondly, our work does not account for the potential costs linked to evaluating infeasible samples. For instance, costs associated with material breakage during evaluation or motor damage caused by excessively high voltage are not considered. Furthermore, to the best of our knowledge, we are unable to find theoretical analysis on global convergence rate in the constrained BO literature when unknown constraints are involved due to the added complexity of constraint surrogates and the interplay between constraints and objectives. We leave these exciting directions for future work.

Acknowledgements

We thank the reviewers for their constructive suggestions. This work is supported by the MIT-IBM Watson AI Lab, and its member company, Evonik. This work also received the support of a fellowship from "la Caixa" Foundation (ID 100010434). The fellowship code is LCF/BQ/EU21/11890103.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Antonio, C. Sequential model based optimization of partially defined functions under unknown constraints. *Journal of Global Optimization*, 79(2):281–303, 2021.

Ariafar, S., Coll-Font, J., Brooks, D. H., and Dy, J. G. Admmbo: Bayesian optimization with unknown constraints using admm. *J. Mach. Learn. Res.*, 20(123):1–26, 2019.

Bachoc, F., Helbert, C., and Picheny, V. Gaussian process optimization with failures: classification and convergence proof. *Journal of Global Optimization*, 78:483–506, 2020.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Frame-

work for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Baumann, D., Marco, A., Turchetta, M., and Trimpe, S. Gosafe: Globally optimal safe robot learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4452–4458. IEEE, 2021.

Belegundu, A. D. and Arora, J. S. A study of mathematical programming methods for structural optimization. part i: Theory. *International Journal for Numerical Methods in Engineering*, 21(9):1583–1599, 1985.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9368–9377, 2018.

Berkenkamp, F., Krause, A., and Schoellig, A. P. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 112(10): 3713–3747, 2023.

Cheng, G. H., Gjernes, T., and Gary Wang, G. An adaptive aggregation-based approach for expensively constrained black-box optimization problems. *Journal of Mechanical Design*, 140(9):091402, 2018.

Coello, C. A. C. and Montes, E. M. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203, 2002.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pp. 1166–1174. PMLR, 2013.

Eriksson, D. and Poloczek, M. Scalable constrained bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Frazier, P. I. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pp. 255–278. Informs, 2018.

Gardner, J. R., Kusner, M. J., Xu, Z. E., Weinberger, K. Q., and Cunningham, J. P. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pp. 937–945, 2014.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

Gelbart, M. A., Snoek, J., and Adams, R. P. Bayesian optimization with unknown constraints. In *30th Conference on Uncertainty in Artificial Intelligence, UAI 2014*, pp. 250–259. AUAI Press, 2014.

Gramacy, R. B., Gray, G. A., Le Digabel, S., Lee, H. K., Ranjan, P., Wells, G., and Wild, S. M. Modeling an augmented lagrangian for blackbox constrained optimization. *Technometrics*, 58(1):1–11, 2016.

Gratton, S., Toint, P. L., and Tröltzsch, A. An active-set trust-region method for derivative-free nonlinear bound-constrained optimization. *Optimization Methods and Software*, 26(4-5):873–894, 2011.

Grossmann, I. E. and Sargent, R. W. Optimum design of multipurpose chemical plants. *Industrial & Engineering Chemistry Process Design and Development*, 18(2):343–348, 1979.

Gupta, S., Tiwari, R., and Nair, S. B. Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory*, 42(10):1418–1443, 2007.

Hansen, N., Müller, S. D., and Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Hedar, A.-R., Fukushima, M., et al. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global optimization*, 35 (4):521–550, 2006.

Hennig, P. and Schuler, C. J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

Hernández-Lobato, J. M., Gelbart, M., Hoffman, M., Adams, R., and Ghahramani, Z. Predictive entropy search for bayesian optimization with unknown constraints. In *International conference on machine learning*, pp. 1699–1707. PMLR, 2015.

Hickish, R., Fletcher, D., and Harrison, R. Investigating bayesian optimization for rail network optimization. *International Journal of Rail Transportation*, October 2019. URL http://eprints.whiterose.ac.uk/151331/.

Hoffman, M., Brochu, E., De Freitas, N., et al. Portfolio allocation for bayesian optimization. In *UAI*, pp. 327–336, 2011.

Isaacs, A., Ray, T., and Smith, W. Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 2780–2787. IEEE, 2008.

Jiao, R., Zeng, S., and Li, C. A feasible-ratio control technique for constrained optimization. *Information Sciences*, 502:201–217, 2019.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455, 1998.

Kandasamy, K., Vysyaraju, K. R., Neiswanger, W., Paria, B., Collins, C. R., Schneider, J., Poczos, B., and Xing, E. P. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *The Journal of Machine Learning Research*, 21(1):3098–3124, 2020.

Kelleher, J. D., Mac Namee, B., and D'arcy, A. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT press, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirschner, J., Mutny, M., Hiller, N., Ischebeck, R., and Krause, A. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pp. 3429–3438. PMLR, 2019.

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., and Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56:100693, 2020.

Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142: 106816, 2020.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Lemonge, A. C., Barbosa, H. J., Borges, C. C., and Silva, F. B. Constrained optimization problems in mechanical engineering design using a real-coded steady-state genetic algorithm. *Mecánica Computacional*, 29(95):9287–9303, 2010.

Li, Y. L., Rudner, T. G., and Wilson, A. G. A study of bayesian neural network surrogates for bayesian optimization. *arXiv preprint arXiv:2305.20028*, 2023.

Lindberg, D. V. and Lee, H. K. Optimization under constraints by applying an asymmetric entropy measure. *Journal of Computational and Graphical Statistics*, 24 (2):379–393, 2015.

Liu, Z.-Z., Wang, B.-C., and Tang, K. Handling constrained multiobjective optimization problems via bidirectional coevolution. *IEEE Transactions on Cybernetics*, 52(10): 10163–10176, 2021.

Lizotte, D., Wang, T., Bowling, M., and Schuurmans, D. Automatic gait optimization with gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pp. 944–949, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

Lu, D. and Weng, Q. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28 (5):823–870, 2007.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., and Gao, J. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40, 2021.

Močkus, J. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, pp. 400–404. Springer, 1975.

Pant, M., Thangaraj, R., and Singh, V. Optimization of mechanical design problems using improved differential evolution algorithm. *International Journal of Recent Trends in Engineering*, 1(5):21, 2009.

Paul H., T. Optimal design of an industrial refrigeration system. In *Proc. of Int. Conf. on Optimization Techniques and Applications*, pp. 427–435, 1987.

Rao, R. V., Savsani, V. J., Rao, R. V., and Savsani, V. J. *Advanced Optimization Techniques*. Springer, 2012.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Rathore, A. K., Holtz, J., and Boller, T. Synchronous optimal pulsewidth modulation for low-switching-frequency control of medium-voltage multilevel inverters. *IEEE Transactions on Industrial Electronics*, 57(7):2374–2381, 2010.

Ray, T. and Saini, P. Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Engineering Optimization*, 33(6):735–748, 2001.

Ray, T., Singh, H. K., Isaacs, A., and Smith, W. Infeasibility driven evolutionary algorithm for constrained optimization. *Constraint-handling in evolutionary optimization*, pp. 145–165, 2009.

Sergeyev, Y. D., Kvasov, D. E., and Mukhametzhanov, M. S. A generator of multiextremal test classes with known solutions for black-box-constrained global optimization. *IEEE Transactions on Evolutionary Computation*, 26(6): 1261–1270, 2021.

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

Simionescu, P. A. *Computer-aided graphing and simulation tools for AutoCAD users*, volume 32. CRC Press, 2014.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022, 2010.

Sui, Y., Gotovos, A., Burdick, J., and Krause, A. Safe exploration for optimization with gaussian processes. In *International conference on machine learning*, pp. 997–1005. PMLR, 2015.

Swain, M., Dash, S. K., Dash, S., and Mohapatra, A. An approach for iris plant classification using neural network. *International Journal on Soft Computing*, 3(1):79, 2012.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Tomczak, M. B., Swaroop, S., and Turner, R. E. Neural network ensembles and variational inference revisited. In *1st Symposium on Advances in Approximate Bayesian Inference*, pp. 1–11, 2018.

Townsend, A. Constrained optimization in chebfun. chebfun.org, 2014. Last accessed 2023-05-16.

Tran, D., Dusenberry, M., Van Der Wilk, M., and Hafner, D. Bayesian layers: A module for neural network uncertainty. *Advances in neural information processing systems*, 32, 2019.

Turchetta, M., Berkenkamp, F., and Krause, A. Safe exploration for interactive machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 745–754. PMLR, 2018.

Yu, Z., Ramakrishnan, V., and Meinzer, C. Simulation optimization for bayesian multi-arm multi-stage clinical trial with binary endpoints. *Journal of Biopharmaceutical Statistics*, 29:1–12, 02 2019. doi: 10.1080/10543406.2019.1577682.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhang, Y., Zhang, X., and Frazier, P. Constrained two-step look-ahead bayesian optimization. *Advances in Neural Information Processing Systems*, 34:12563–12575, 2021.

A. Method Implementation Details

A.1. Deep Ensembles

A.1.1. Neural Network Architecture and Parameters

We implement an ensemble of Multi-layer Perceptrons (MLPs) for modeling the unknown constraints. For each MLP in the ensemble, we use a simple and standard structure of 4 fully connected layers with $64\lfloor \log_2(d) \rfloor$ neurons in each hidden layer where d is the problem dimension. The network gets larger as the problem dimension gets higher. We use ReLU nonlinearity between each pair of fully connected layers.

Similar to training GP classifiers using a variational framework (Hensman et al., 2015), we use the variational evidence lower bound (ELBO) to approximate the posterior and marginal likelihood of *Deep Ensemble* classifiers given the Bernoulli likelihood. The ensemble is optimized for maximal marginal log likelihood using the Adam optimizer with a 3×10^{-4} learning rate for 1,000 iterations. Note that we do not apply regularization or dropout as suggested by Lakshminarayanan et al. (2017).

Overall, the architecture we use is very simple and straightforward to implement with almost no hyper-parameters, but works robustly across a wide range of benchmark problems as shown in our experiments.

A.1.2. Mean and Uncertainty Computation

In Section 4.1 of the main paper, we introduced the mean and variance computation following the original formulation of *Deep Ensembles* (Lakshminarayanan et al., 2017). In practice, we empirically observed an alternative implementation of mean and uncertainty computation leads to better performance.

As mentioned in Section 4.1, instead of letting the network ensemble directly output the probability value between $[0, 1]$ for the constraint feasibility and training the ensemble using maximum likelihood estimation (MLE, specifically, binary cross-entropy loss), we treat the network ensemble as a *real-valued latent constraint function* $g(x)$ such that the constraint is satisfied if and only if $g(x) \geq 0$. In other words, we let the network ensemble output a latent Gaussian distribution in a continuous space and later transform the output to a probability between $[0, 1]$ by standard normal CDF Φ, following the approach taken by CEI (Gelbart et al., 2014). The transformed probability is essentially the mean prediction. With such treatment, we can optimize the *Deep Ensembles* in a similar way as training GP classifiers based on a variational inference (VI) framework, as described in Section A.1.1.

For uncertainty computation, since we can easily get the real-valued mean μ_g and standard deviation σ_g from the latent Gaussian distribution, we can transform the one-sigma confidence interval $[\mu_g - \sigma_g, \mu_g + \sigma_g]$ by the standard normal CDF Φ and obtain a corresponding confidence interval in the transformed probability space $[\Phi(\mu_g - \sigma_g), \Phi(\mu_g + \sigma_g)]$ and define the transformed standard deviation as $\sigma_E = (\Phi(\mu_g + \sigma_g) - \Phi(\mu_g - \sigma_g))/2$. In practice, we use this formula to obtain the dynamic bounds as described in Section 4.2 of the main paper.

In practice, we find that models trained by MLE are overly confident, i.e., usually output extremely low uncertainties, which leads to the poor performance. On the contrary, training with VI produces much more reasonable uncertainty estimation. Figure 6 shows empirical comparisons between training DE with VI and MLE respectively for BE-CBO, which shows a clear advantage of VI.

A.2. Constrained Optimization on Acquisition Functions

For numerical optimization of the constrained acquisition function in BE-CBO, we specify the dynamic bound constraint as a nonlinear inequality constraint to the acquisition optimizer in BoTorch (Balandat et al., 2020), which calls an underlying SLSQP optimization from SciPy, a standard and robust choice for handling nonlinear inequality constraints. However, the SLSQP optimizer requires constraint-satisfying initial solutions as input. Since randomly generated initial solutions can easily violate the constraint, to deal with this problem, we use the Adam optimizer (Kingma & Ba, 2014) to find constraint-satisfying samples through gradient descent. We use a standard mean squared error (MSE) loss to measure the difference between the sample's feasibility probability and 0.5. By optimizing such MSE loss, the sample gets closer to the constraint boundary by pushing the feasibility probability to 0.5, thus the sample becomes closer to satisfying the dynamic band constraint.

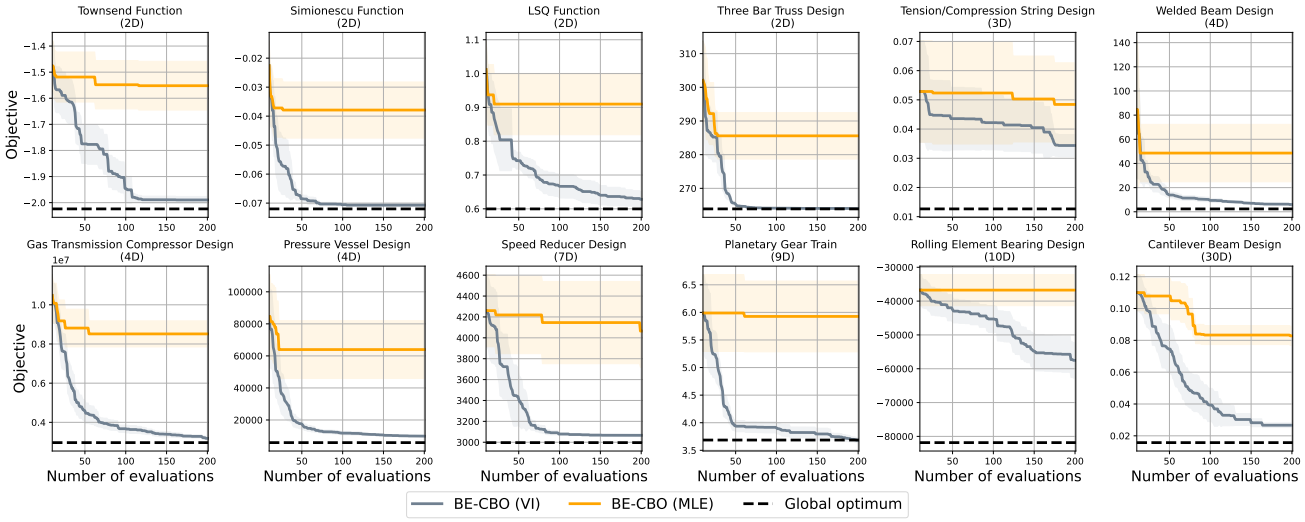

Figure 6: Comparisons between BE-CBO trained with VI and MLE respectively, averaged over 10 random seeds.

B. Experimental Setup

Due to the large number of benchmark problems and random seeds, the experiments are conducted in parallel on a distributed server with Intel Xeon Platinum 8260 CPUs with 4GB RAM per core, where each individual experiment runs on a single CPU thread without GPU.

For surrogate modeling of the objective in BE-CBO and all baseline algorithms, we directly use the implementation of GP regressors in BoTorch (Balandat et al., 2020), where Matern 5/2 kernel is used with their default hyperparameters[1]. For the GP constraint classifiers in all baseline algorithms and also ablation studies in BE-CBO, we leverage the implementation from GPyTorch (Gardner et al., 2018), where RBF kernel is used with their default hyperparameters.[2] We use the standard Bernoulli likelihood in GP classifiers for representing posterior probability distributions, and use the variational evidence lower bound (ELBO) to optimize the GP classifiers (Hensman et al., 2015).

B.1. Benchmark Problems

In this section, we briefly introduce the properties of each problem, including the dimensions of the design space $\mathcal{X} \subset \mathbb{R}^d$. The problem descriptions for 3 synthetic functions and 9 real-world problems are described respectively. We perform 10 independent test runs with 10 different random seeds for each problem on each algorithm. For each test run of one problem, we use the same initial set of samples for every algorithm, which is generated by a scrambled quasirandom Sobol sequence using the same random seed.

We represents all benchmark problems in the form of minimizing $f(x)$ subject to multiple underlying constraints $c_1(x) \geq 0, ..., c_n(x) \geq 0$. Note that we aim to solve the problem where the constraints are unknown and the algorithm only learns whether a design point is feasible or infeasible. Hence, multiple constraints can exist, but their formulas are invisible and they are all captured with one classifier that outputs a binary value.

B.1.1. Synthetic Test Functions

Townsend function (Townsend, 2014) is a trigonometric function $f(x) = -[\cos((x_1 - 0.1)x_2)]^2 - x_1 \sin(3x_1 + x_2)$ constrained by $c(x) = (2\cos t - \frac{1}{2}\cos 2t - \frac{1}{4}\cos 3t - \frac{1}{8}\cos 4t)^2 + (2\sin t)^2 - x_1^2 - x_2^2$ where $t = \arctan2(x_1, x_2)$ and with bounds $-2.25 \leq x_1 \leq 2.25$ and $-2.5 \leq x_2 \leq 1.75$.

[1] https://botorch.org/tutorials/scalable_constrained_bo

[2] https://notebook.community/jrg365/gpytorch/examples/02_Simple_GP_Classification/Simple_GP_Classification

Simionescu function (Simionescu, 2014) is a hyperbolic paraboloid function $f(x) = 0.1x_1x_2$ constrained by $c(x) = (r_T + r_S\cos(n\arctan\frac{x}{y}))^2 - x_1^2 - x_2^2$ where $r_T = 1$, $r_S = 0.2$ and $n = 8$, in the domain $[-1.25, 1.25]^2$.

LSQ function (Gramacy et al., 2016) is a linear objective function $f(x) = x_1 + x_2$ with sinusoidal and quadratic constraints $c_1(x) = x_1 + 2x_2 + \frac{1}{2}\sin(2\pi(x_1^2 - 2x_2)) - \frac{3}{2}$ and $c_2(x) = \frac{3}{2} - x_1^2 - x_2^2$ bounded by $[0, 1]^2$.

B.1.2. Real Test Functions

Three bar truss design (Ray & Saini, 2001) minimizes the volume of the truss structure subject to stress constraints. The analytical formula is given as $f(x) = l(2\sqrt{2}x_1 + x_2)$ with two variables $0 \le x_1, x_2 \le 1$, subject to three constraints:

- $c_1(x) = 2 - \frac{\sqrt{2x_1}+x_2}{\sqrt{2x_1^2+2x_1x_2}}$
- $c_2(x) = 2 - \frac{1}{x_1+\sqrt{2}x_2}$
- $c_3(x) = 2 - \frac{x_2}{\sqrt{2x_1^2+2x_1x_2}}$

Tension-compression string design (Hedar et al., 2006) is a three dimensional design problem where the weight of a tension-compression string, given by $f(x) = (x_1 + 2)x_2x_3^2$, needs to be minimized, where: $2 \le x_1 \le 15$ is the number of active coils (integer), $0.25 \le x_2 \le 1.3$ is the wire diameter, and $0.05 \le x_3 \le 2$ is the mean coil diameter. This minimization is constrained by the minimum deflection, shear stress, surge frequency, and limits on the outside diameter (Coello & Montes, 2002):

- $c_1(x) = \frac{x_2^3x_1}{71785x_3^4} - 1$
- $c_2(x) = 1 - \frac{4x_2^2-x_3x_2}{12566(x_2x_3^3-x_3^4)} - \frac{1}{5108x_3^2}$
- $c_3(x) = \frac{140.45x_3}{x_2^2x_1} - 1$
- $c_4(x) = 1 - \frac{x_2+x_3}{1.5}$

Welded beam design (Belegundu & Arora, 1985) aims to minimize the cost of the beam, $f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$, where $0.125 \le x_1 \le 10$ and $0.1 \le x_2, x_3, x_4 \le 10$ are four design variables referring to physical dimensions of the beam. This optimization is subject to constraints on the shear stress, bending stress in the beam, buckling load on the bar, the end deflection of the beam, and a side constraint (Hedar et al., 2006):

- $c_1(x) = 13000 - \tau(x)$
- $c_2(x) = 30000 - \sigma(x)$
- $c_3(x) = P_c(x) - 6000$
- $c_4(x) = 0.25 - \delta(x)$
- $c_5(x) = x_4 - x_1$

where $\tau(x) = \sqrt{(\tau_1(x))^2 + (\tau_2(x))^2 + \frac{x_2\tau_1(x)\tau_2(x)}{\sqrt{0.25[x_2^2+(x_1+x_3)^2]}}}$, $\tau_1(x) = \frac{6000}{\sqrt{2}x_1x_2}$, $\tau_2(x) = \frac{6000(14+0.5x_2)\sqrt{0.25[x_2^2+(x_1+x_3)^2]}}{2[0.707x_1x_2(x_2^2/12+0.25(x_1+x_3)^2)]}$, $\sigma(x) = \frac{504000}{x_3^2x_4}$, $P_c(x) = 64746.022(1 - 0.0282346x_3)x_3x_4^3$, $\delta(x) = \frac{2.1953}{x_3^3x_4}$.

Gas transmission compressor design (Pant et al., 2009) aims to minimize the total annual cost of a gas pipeline transmission system and its operation, given by $f(x) = (8.61)10^5x_1^{1/2}x_2x_3^{-2/3}x_4^{-1/2} + (3.69)10^4x_3 + (7.72)10^8x_1^{-1}x_2^{0.219} - (765.43)10^6x_1^{-1}$, where $20 \le x_1 \le 50$ is the length between compressor stations, $1 \le x_2 \le 10$ is the compression ratio, $20 \le x_3 \le 50$ is the inside diameter of the pipe, and $0.1 \le x_4 \le 60$ is a non-dimensional parameter, subject to $c(x) = 1 - x_4x_2^{-2} - x_2^{-1}$.

Pressure vessel design (Coello & Montes, 2002) minimize the total cost $f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$, including the cost of the material, forming and welding. There are four design variables: Ts (thickness of the shell), Th (thickness of the head), R (inner radius) and L (length of the cylindrical section of the vessel, not including the head). The constraints are given as:

- $c_1(x) = x1 - 0.0193x_3$
- $c_2(x) = x2 - 0.00954x_3$
- $c_3(x) = \pi x_3^2x_4 + \frac{4}{3}\pi x_3^3 - 1296000$
- $c_4(x) = 240 - x_4$

Speed reducer design (Lemonge et al., 2010) is a seven dimensional problem that seeks to minimize the weight of a speed reducer. The design variables are: the face width ($2.6 \leq x_1 \leq 3.6$), the module of teeth ($0.7 \leq x_2 \leq 0.8$), the number of teeth on pinion (integer) ($17 \leq x_3 \leq 28$), the length of the shaft 1 between the bearings ($7.3 \leq x_4 \leq 8.3$), the length of the shaft 2 between the bearings ($7.3 \leq x_5 \leq 8.3$), the diameter of the shaft 1 ($2.9 \leq x_6 \leq 3.9$), and the diameter of the shaft 2 ($5 \leq x_7 \leq 5.5$).

The weight is given by:

$$f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

and is subject to the following mechanical constraints:

- $c_1(x) = 1 - 27x_1^{-1}x_2^{-2}x_3^{-1}$,
- $c_2(x) = 1 - 397.5x_1^{-1}x_2^{-2}x_3^{-2}$,
- $c_3(x) = 1 - 1.93x_2^{-1}x_3^{-1}x_4^{-3}x_6^{-4}$,
- $c_4(x) = 1 - 1.93x_2^{-1}x_3^{-1}x_5^{-3}x_7^{-4}$,
- $c_5(x) = 1100 - [\frac{745x_4}{x_2x_3}^2 + (16.9)10^6]^{0.5}\frac{1}{0.1x_6^3}$,
- $c_6(x) = 850 - [\frac{745x_5}{x_2x_3}^2 + (157.5)10^6]^{0.5}\frac{1}{0.1x_7^3}$,
- $c_7(x) = 40 - x_2x_3$,
- $c_8(x) = x_1/x_2 - 5$,
- $c_9(x) = 12 - x_1/x_2$,
- $c_{10}(x) = 1 - (1.5x_6 + 1.9)x_4^{-1}$,
- $c_{11}(x) = 1 - (1.1x_7 + 1.9)x_5^{-1}$.

Planetary gear train design (Rao et al., 2012) minimizes the gear ratio errors which can be stated as:

$f(x) = \max|i_k - i_{0k}|$ where $k = \{1, 2, R\}$, $i_1 = \frac{N_6}{N_4}$, $i_{01} = 3.11$, $i_2 = \frac{N_6(N_1N_3+N_2N_4)}{N_1N_3(N_6-N_4)}$, $i_{02} = 1.84$, $i_R = \left(\frac{N_2N_6}{N_1N_3}\right)$, $i_{0R} = -3.11$. The design variables are defined as $\bar{x} = (\rho, N_6, N_5, N_4, N_3, N_2, N_1, m_2, m_1)$. It is also subject to the following constraints:

- $c_1(x) = D_{\max} - m_3(N_6 + 2.5)$
- $c_2(x) = D_{\max} - m_1(N_1 + N_2) - m_1(N_2 + 2)$
- $c_3(x) = D_{\max} - m_3(N_4 + N_5) - m_3(N_5 + 2)$
- $c_4(x) = (N_1 + N_2)\sin\left(\frac{7}{\rho}\right) - N_2 - 2 - \delta_{22}$
- $c_5(x) = (N_6 - N_3)\sin\left(\frac{7}{\rho}\right) - N_3 - 2 - \delta_{33}$
- $c_6(x) = (N_4 + N_5)\sin\left(\frac{7}{\rho}\right) - N_5 - 2 - \delta_{55}$
- $c_7(x) = (N_6 - N_3)^2 - (N_3 + N_5 + 2 + \delta_{35})^2 - (N_4 + N_5)^2 - 2(N_6 - N_3)(N_4 + N_5)\cos\left(\frac{2\pi}{\rho} - \beta\right)$
- $c_8(x) = -N_4 + N_6 - 2N_5 - 2\delta_{56} - 4$
- $c_9(x) = -2N_3 + N_6 - N_4 + 2\delta_{34} + 4$

where $\delta_{22} = \delta_{33} = \delta_{55} = \delta_{35} = \delta_{56} = 0.5$, $\beta = \cos^{-1}\left(\frac{(N_4+N_5)^2+(N_6-N_3)^2-(N_3+N_5)^2}{2(N_6-N_3)(N_4+N_5)}\right)$, and $D_{\max} = 220$, with bounds: $p = (3, 4, 5)$, $m_1 = (1.75, 2.0, 2.25, 2.5, 2.75, 3.0)$, $m_3 = (1.75, 2.0, 2.25, 2.5, 2.75, 3.0)$, $17 \leq N_1 \leq 96$, $14 \leq N_2 \leq 54$, $14 \leq N_3 \leq 51$, $17 \leq N_4 \leq 46$, $14 \leq N_5 \leq 51$, $48 \leq N_6 \leq 124$, and N_i is an integer. We apply continuous optimization methods on this problem by rounding the continuous variables into integer ones.

Rolling element bearing design (Gupta et al., 2007) optimizes the dynamic capacity of a rolling bearing. The design parameter vector can be written as $\bar{x} = (D_m, D_b, Z, f_i, f_o, K_{\text{Dmin}}, K_{\text{Dmax}}, \varepsilon, e, \xi)$ where $\bar{f}_i = \frac{r_i}{D_b}$ and $\bar{f}_o = \frac{r_o}{D_b}$. The bounds of the paramters are $D_m \in [125, 150]$, $D_b \in [10.5, 31.5]$, $Z \in [4, 50]$, $f_i \in [0.515, 0.6]$, $f_o \in [0.515, 0.6]$, $K_{\text{Dmin}} \in [0.4, 0.5]$, $K_{\text{Dmax}} \in [0.6, 0.7]$, $\varepsilon \in [0.3, 0.4]$, $e \in [0.02, 0.1]$, $\xi \in [0.6, 0.85]$. The objective function is defined as $f(x) = C_d$ where

$$C_d = \begin{cases} f_cZ^{2/3}D_b^{1.8}, & D_b \leq 25.4 \text{ mm} \\ 3.647f_cZ^{2/3}D_b^{1.4}, & D_b > 25.4 \text{ mm} \end{cases}$$

$$f_c = 37.91\left[1 + \left\{1.04\left(\frac{(1-\gamma)^{1.72}}{(1+\gamma)^{1.72}}\right)\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3^{0.3}}\right]\left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41}, \gamma = \frac{D_b\cos\alpha}{D_m}.$$

The constraints are defined as:

- $c_1(x) = \frac{\phi_0}{2\sin^{-1}\left(\frac{D_b}{D_m}\right)} - Z + 1$
- $c_2(x) = 2D_b - K_{\text{Dmin}}(D-d)$
- $c_3(x) = K_{\text{Dmax}}(D-d) - 2D_b$
- $c_4(x) = \xi_{B_w} - D_b$
- $c_5(x) = D_m - 0.5(D+d)$
- $c_6(x) = (0.5+e)(D+d) - D_m$
- $c_7(x) = 0.5(D - D_m - D_b) - eD_b$

where

$$\phi_0 = 2\pi - 2\cos^{-1}\left[\frac{\left\{(D-d)/2 - 3(T/4)^2\right\}^2 + \left\{(D/2 - (T/4) - D_b)\right\}^2 - (d/2 + (T/4))^2}{2\{(D-d)/2 - 3(T/4)\}\{(D/2 - (T/4) - D_b)\}}\right], T = D - d - 2D_b.$$

Cantilever beam design (Cheng et al., 2018) minimizes the tip deflection of a stepped cantilever beam subject to constraints. Due to the complexity of formulas, please refer to the original paper for details.

B.1.3. Boundary Optimality of Real-World Problems

Table 1: Details of the 9 real-world design optimization problems. d is the dimension of design variables, g is the number of unknown inequality constraints, g^* is the number of active constraints at the optimum, $f(x^*)$ is the optimal performance value.

Problem Name	d	g	g^*	$f(x^*)$
Three-bar truss design problem	2	3	1	2.6389E+02
Tension/compression spring design	3	3	2	1.2665E-02
Welded beam design	4	5	1	2.4453E+00
Gas transmission compressor design	4	1	1	2.9648E+06
Pressure vessel design	4	4	2	5.8853E+03
Speed reducer design	7	11	2	2.9944E+03
Planetary gear train design	9	9	1	3.6859E+00
Rolling element bearing	10	9	3	8.3918E+04
Cantilever beam design	30	21	1	1.5731E+02

For real-world design optimization problems with unknown physical constraints, the optimal solutions usually lie on the boundary of feasible regions since the performance upper bound is usually constrained by physical limits. Though this is often the experience from practitioners, to further validate whether this observation is true, we check whether the global optimum is on the constraint boundary for each problem in our collected nine benchmark problems. Table 1 summarizes the problem description, the optimum information and active constraints at the optimum. All problems have at least one active constraint at the optimum and some problems have multiple ones.

B.2. Baseline Algorithms

In this section, we provide a description of the other baseline algorithms that we used for comparison with our own algorithm. Specifically, we compare our algorithms to several BO baseline algorithms that are discussed in Results Section of the main paper. These baseline algorithms are designed or have been modified to handle binary unknown constraints. Here we describe the differences between our adoption of these algorithms and their original formulation in the paper.

CEI (Gelbart et al., 2014) We implement this algorithm in BoTorch. The acquisition function remains unchanged to its original form, with one notable modification regarding the constraint classification. In contrast to the paper's setup where individual constraints provide separate responses, our evaluation process only allows for obtaining a single feasibility response. Therefore, we have modified the implementation to utilize a single binary classifier as a constraint surrogate instead of multiple classifiers for multiple constraints as described in the original paper.

SCBO (Eriksson & Poloczek, 2021) We implement this algorithm in BoTorch as there exists a reference implementation in the BoTorch documentation[1]. In contrast to the original setup, where multiple *continuous* responses can be obtained from individual constraints during evaluation, our approach assumes a *binary* feasibility response. Consequently, we employ a single binary classifier in our setup, similar to the adoption of CEI, instead of multiple regressors for multiple constraints as described in the paper. Due to the binary nature of the constraint, we do not apply the Bilog transformation to the constraint values. However, we still apply the Gaussian copula transformation to the objective values and implement the restart of new trust regions according to the paper's descriptions.

SVM-CBO (Antonio, 2021) To conduct the SVM-CBO algorithm experiments, we utilized the code and framework provided by the authors. The experimental setup described in their paper involved 100 evaluations, comprising of 10 initial samples, 60 evaluations in phase 1, and 10 evaluations in phase 2. In our extended evaluations, where we employed 20 initial samples and a total of 200 evaluations, we maintained the same ratio as described in the original paper. Thus, we performed 127 evaluations in phase 1 and 63 evaluations in phase 2.

C. Additional Comparison

C.1. Qualitative Comparison of Sample Distributions

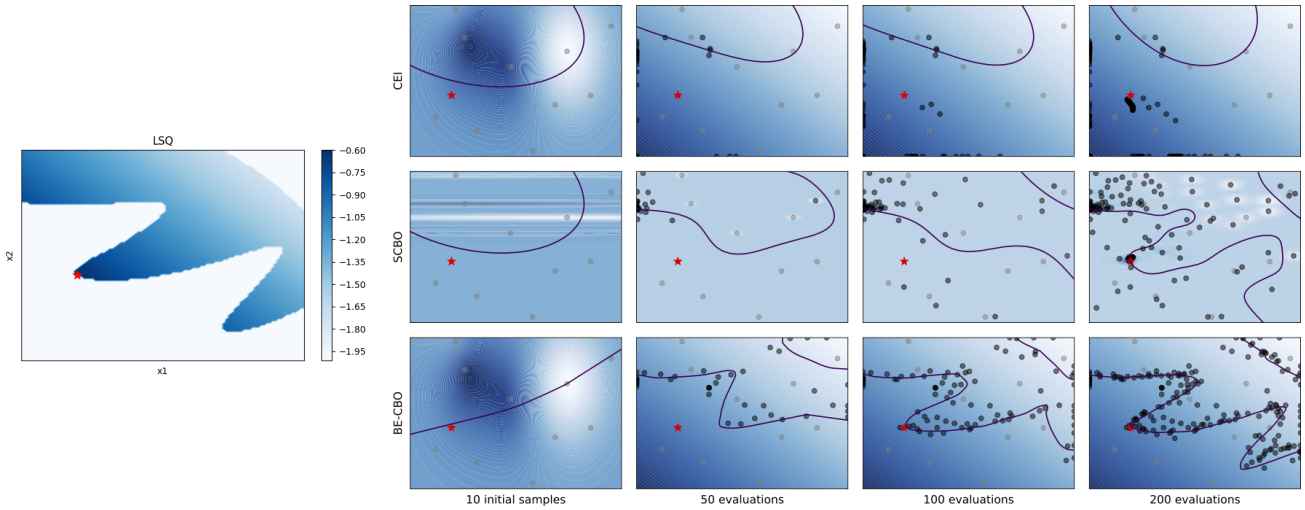

Figure 7: Qualitative comparison of sample distributions from different algorithms on the LSQ benchmark.

We show more qualitative comparison of sample distributions on extra two 2D functions (LSQ and Townsend) in Figure 7 and 8 besides the Simionescu shown in the main paper. Same as the main paper, in each figure, the true function landscape is on the left, where darker color means a higher objective value and the white region means infeasible. On the right is the predicted function landscape (top: CEI, middle: SCBO, bottom: BE-CBO) where darker color means a higher objective value and the contour means the feasibility boundary (feasible inside, infeasible outside). Initial samples (grey), the rest evaluated samples (black) and the global optima (red) are also displayed.

For the LSQ function, the objective function is relatively smooth so all three algorithms discover the global optimum in the end. BE-CBO discovers the global optimum more efficiently at 100 evaluations and also classifies a much more accurate constraint boundary compared to the other two algorithms. For the Townsend function, both CEI and SCBO get stuck in the two local optima in the middle, while BE-CBO sucessfully discovers the true global optimum on the upper right corner and also classifies the constraint boundary well.

C.2. Runtime Comparison

In order to empirically assess the speed efficiency of various algorithms, we collect and analyze the runtime statistics measured in seconds, shown in Figure 9. In general, BE-CBO exhibits stable runtime across different problem dimensions

[1] https://botorch.org/tutorials/scalable_constrained_bo

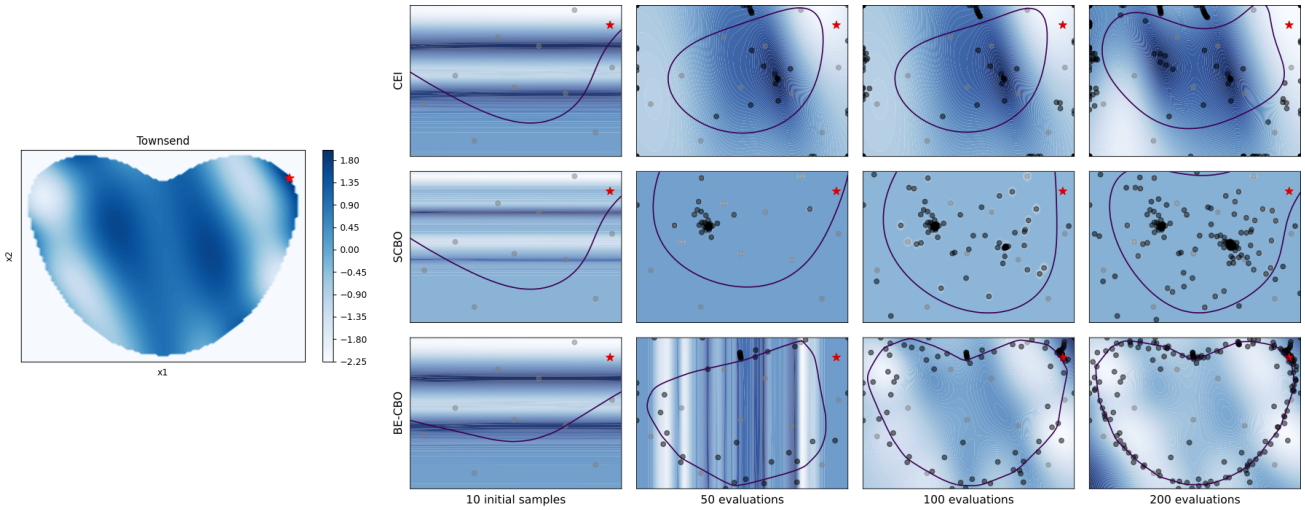

Figure 8: Qualitative comparison of sample distributions from different algorithms on the Townsend benchmark.

with very low variance, at around 5K seconds for 200 iterations. Our speed is close to CEI in many cases and faster than SCBO on average. One might observe that BE-CBO has slower performance than all other algorithms on Tension/Compression String Design, Planetary Gear Train and Rolling Element Bearing Design problems. Though in these cases BE-CBO's speed does not change much, other algorithms operate faster than on other problems. This observation can be actually explained by our proposed feasibility ratio metric shown in Figure 5. In those cases, our algorithm is the most successful in discovering feasible points while other algorithms produce mostly infeasible points. The surrogate classifiers in other algorithms fail to learn useful information thus the fitting stops early, with the exception of SVM-CBO where the SVM fitting time scales badly with the problem dimension.

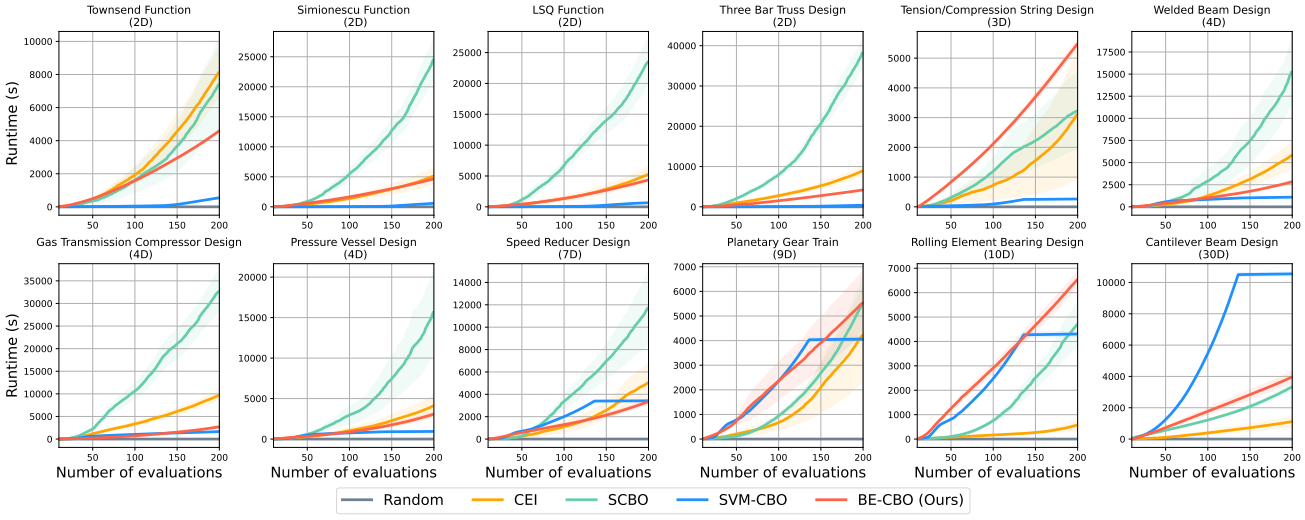

Figure 9: Runtime comparison between algorithms. The experiments are averaged over 10 random seeds. The time is accumulated after every iteration, i.e. after 200 evaluations the runtime is showing the total number of seconds spent to propose and evaluated all 200 samples.

C.3. Synthetic Benchmark Problems With Interior Optima

Although our motivation of developing BE-CBO is the observation that most real-world problems have their optima on the constraint boundary due to physical limits, to test the generality of our method, we construct synthetic functions with global optimum located at the interior of the feasible region. We modified the objective functions of Townsend, Simionescu and

LSQ while leaving their constraint functions unchanged. Figure 10 shows the landscape of the modified functions. The analytical forms of the modified objective functions are as follows:

- Townsend: $f(x) = -[\cos(((x_1 + 1) - 0.1)x_2)]^2 - (x_1 + 1)\sin(3(x_1 + 1) + x_2)$
- Simionescu: $f(x) = 0.1x_1x_2 + 0.1(x_1 - x_2 + 1)^2$
- LSQ: $f(x) = (x_1 - 0.4)^2 + (x_2 - 0.45)^2$

We conduct experiments on these modified functions using CEI, SCBO and BE-CBO. The comparison results are shown in Figure 11. The results show that BE-CBO can effectively discover interior optima in different functions. Though BE-CBO encourages exploration on the constraint boundary, we leverage the underlying Expected Improvement acquisition to decide where is the most promising region considering both the dynamic band and the whole predicted feasible region. In other words, whether to explore the boundary region or the interior feasible region depends on the predicted objective landscape, thus both boundary and interior optima can be discovered.

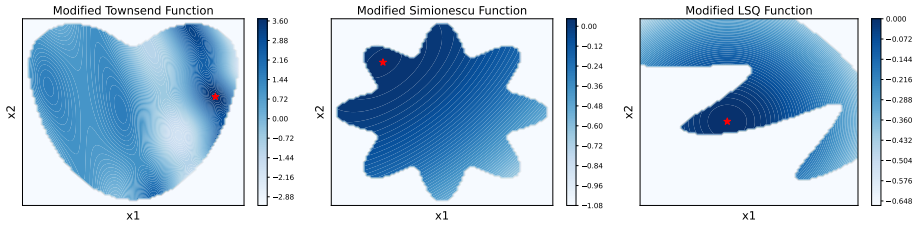

Figure 10: Landscape of the modified Townsend, Simionescu and LSQ functions. The red star indicates the global optimum location, which is inside the feasible region instead of on the constraint boundary in their original forms.

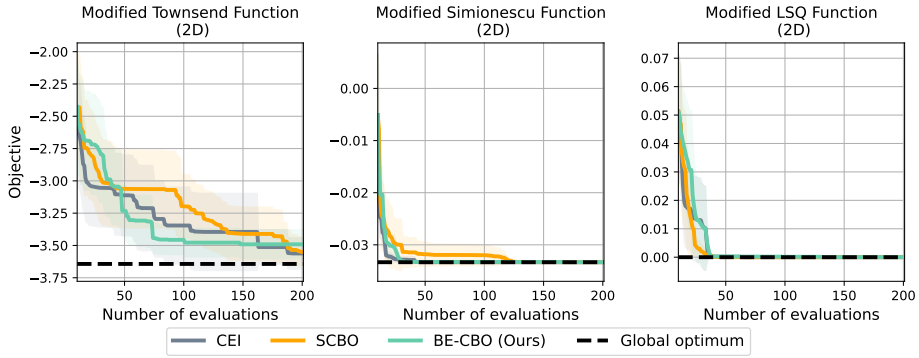

Figure 11: Quantitative comparison of different algorithms including our BE-CBO on shifted synthetic test functions. The current best value is shown w.r.t. the number of function evaluations. Every experiment has 10 initial random samples and 200 evaluations in total. The curve is averaged over 10 different initial random seeds and the standard deviation is shown as a shaded region.

D. Ablation Studies

D.1. Gaussian Processes vs Deep Ensembles for Modeling Unknown Constraints

We use a surrogate model to approximate the unknown constraints. This surrogate model needs to work as a binary classifier that predicts whether a point is feasible since we do not get any additional information when the point is infeasible. Instead of using a popular choice for surrogate models in Bayesian optimization, which are Gaussian Processes, here we show how Deep Ensembles can benefit the overall performance of our Bayesian optimization framework.

D.1.1. Classification Performance Comparison

We demonstrate how the classification accuracy of Deep Ensembles is more stable and in general outperforms the accuracy of Gaussian Processes (GPs). In real-world problems, it is often hard to find any feasible points due to complex constraints, which is reflected in some of the benchmark problems we used to test our approach (see further details in Section B.1). In such cases, we will typically obtain imbalanced data where most of the points are infeasible. Hence, to properly compare the classification accuracy, we use the *Balanced Accuracy* (Kelleher et al., 2020) metric, defined as

$$\frac{TPR + TNR}{2}$$

where $TPR = \frac{\text{true positive}}{\text{total positive}}$ is a true positive rate measuring the sensitivity and $TNR = \frac{\text{true negative}}{\text{total negative}}$ is a true negative rate measuring the specificity.

The experiments are conducted over the same 10 random seeds for both Deep Ensembles and GPs. To test the GPs we simply replace the Deep Ensembles with GPs in our algorithm BE-CBO to train the classifier which models the constraints. All the other steps of our Bayesian optimization framework are identical. Starting from 10 initial random samples, we perform 200 function evaluations in each test run and check the classification accuracy after each evaluation when the classifiers are updated. The results are reported in Figure 12. Note that GPs oscillate in accuracy between different iterations and perform poorly on complex real-world problems, which immediately affects the overall algorithm performance shown in Figure 13.

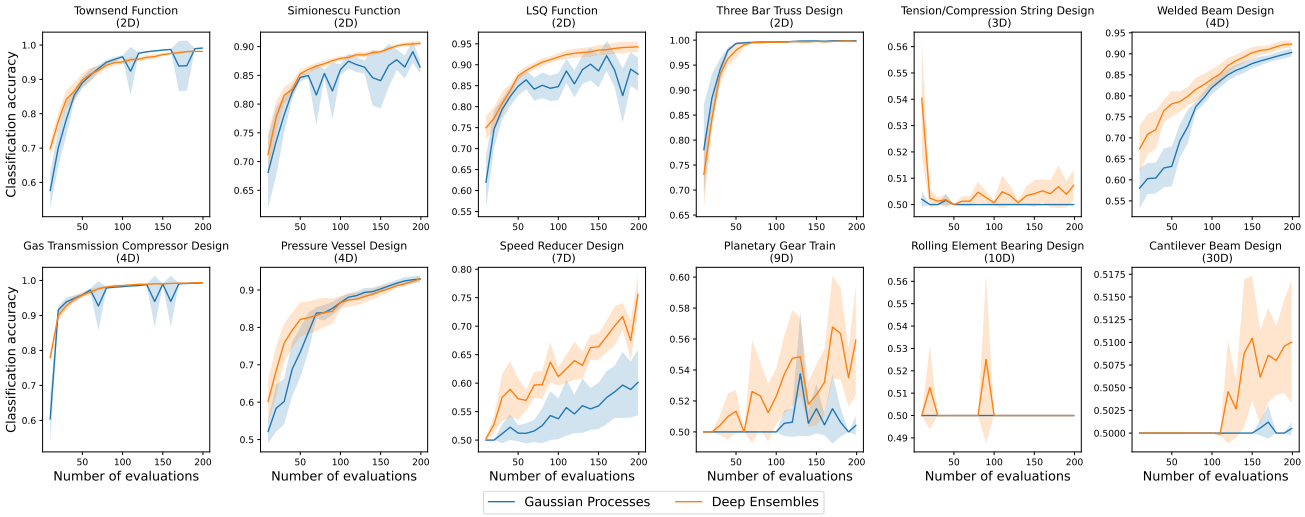

Figure 12: Comparison of classification accuracy between Gaussian Processes and Deep Ensembles when used as classifiers in our Bayesian optimization framework. Classification accuracy is computed with the *Balanced Accuracy* metric. Curves show the average over 10 random seeds and shaded regions represent standard deviation.

D.1.2. Effect of Gaussian Processes vs Deep Ensembles on BE-CBO

We compare the performance of our algorithm BE-CBO when using different surrogate models for the unknown constraints. In one setup, we run our proposed algorithm consisting of Deep Ensembles for modeling the constraints, while in the other setup, we replace the Deep Ensembles with Gaussian Processes (GPs). Both setups are tested on the same 10 initial random samples and we report the average performance and the standard deviation from 10 random seeds in Figure 13. Deep Ensembles and GPs have comparable performance for simpler benchmark problems, while Deep Ensembles demonstrate superior performance in higher dimensions and Tension/Compression String design that is known to be challenging to model and find any feasible designs. Furthermore, we note that GPs perform well only when they are able to closely approximate the true boundary between the feasible and infeasible region of the design space. This finding is reflected in the classification accuracy of different surrogate models, as seen in Section D.1.1.

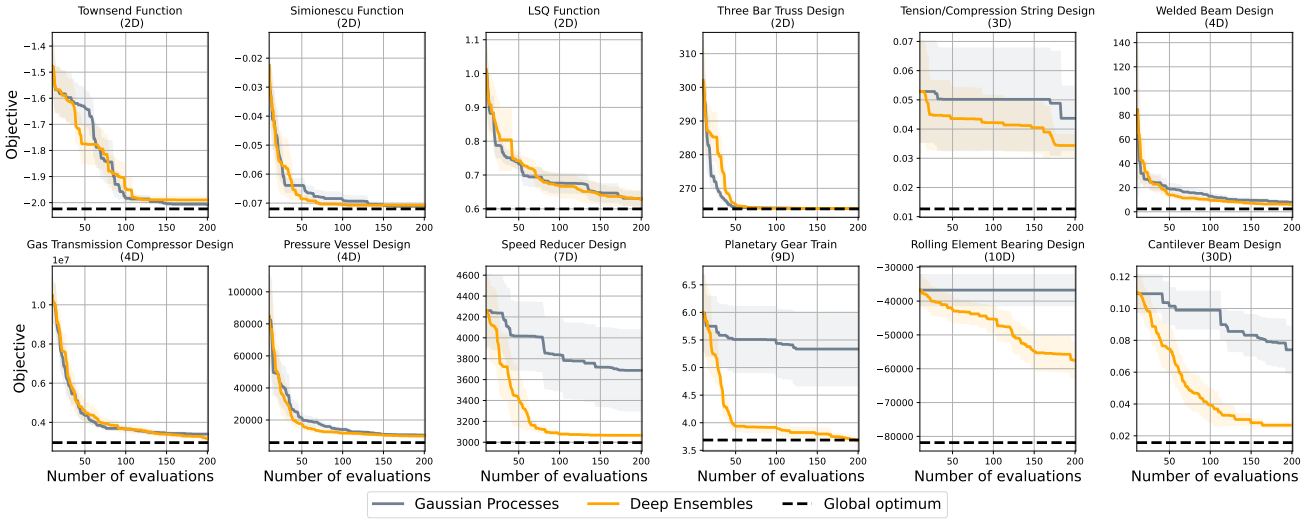

Figure 13: Performance of our algorithm BE-CBO when using Deep Ensembles vs Gaussian Processes for modeling unknown constraints. The current best value found by an algorithm is shown w.r.t. the number of function evaluations. Experiments are run independently from 10 random seeds. The bold curves represent the average in performance over all 10 seeds and shaded areas reflect the standard deviation.

D.2. Hyperparameters of Deep Ensembles

D.2.1. Number of MLPs

One hyperparameter in our network ensemble is the number of MLPs used. However, we observed that varying this parameter does not have much significant effect on the overall performance of our algorithm. In our experiments, we use 5 MLPs in an ensemble for efficiency in memory and computing resources. Our tests show that increasing the number of MLPs up to 8 or decreasing down to 3 mostly has little effect on the overall performance (see Figure 14).

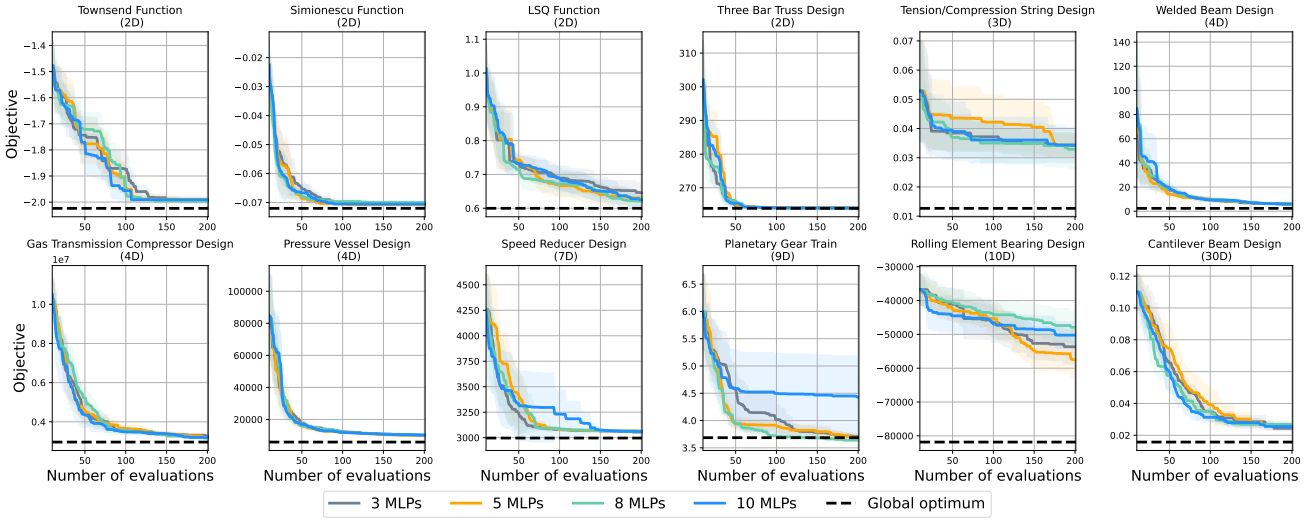

Figure 14: Performance comparison of our algorithm when using different number of MLPs in ensemble. The current best value found by an algorithm is shown w.r.t. the number of function evaluations and the performance is averaged over 10 initial seeds for each experiment.

D.2.2. Number of Hidden Layers

The effect of the number of layers on the algorithm is shown in Figure 15. We tested 1, 2, 3, 4 hidden layers with $64 \log_2 d$ neurons in each layer where d is the problem dimension. The result shows that one hidden layer is insufficient in some cases since the network is too shallow which leads to underfitting. Two or more hidden layers perform similarly in general.

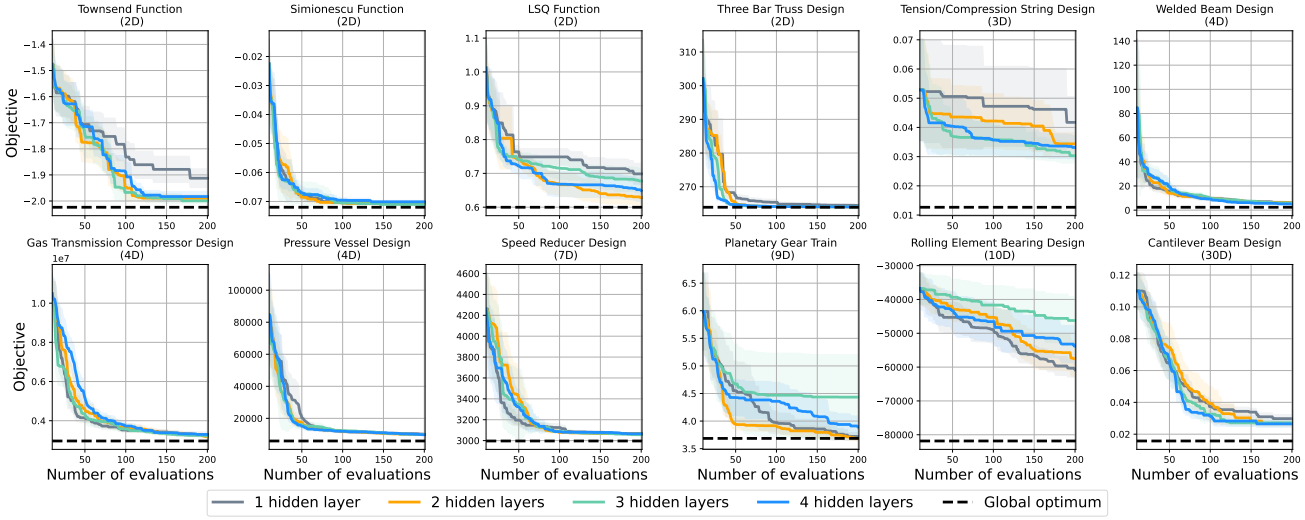

Figure 15: Comparison of BE-CBO with different number of hidden layers for the Deep Ensemble classifier. The current best value is shown w.r.t. the number of function evaluations. The curve is averaged over 10 different initial random seeds. The performance of BE-CBO is not particularly sensitive to the number of layers as long as it has more than one hidden layer.

D.2.3. Number of Neurons in a Layer

The effect of the number of layers on the algorithm is shown in Figure 16. To effectively learn across all problem dimensions, we scale the number of neurons in each layer w.r.t. the problem dimension following a logarithmic formula $N(d) = C \log_2(d)$ where N is the number of neurons, d is the problem dimension and C is a constant factor. We tested $C = 16, 32, 64, 128, 256$ with 2 hidden layers. The results suggest that on most problems, the number of neurons do not matter much, but in some problems, networks with a low capacity ($C = 16$ or 32) are outperformed by networks with more neurons and it seems like $C = 64$ is a stable choice across all problems.

D.2.4. Learning Rate

In the paper, we did experiments with a 3e-4 learning rate, which is a common choice for this hyperparameter. We now test 3e-3, 1e-3, 3e-4, 1e-4, 3e-5 learning rates and show results in Figure 17. Since we set a fixed number of training iterations (1000), small learning rates 1e-4 and 3e-5 are outperformed by larger rates because the network training does not converge. Larger learning rates (3e-3, 1e-3, 3e-4) perform similarly with slight variations on different problems.

D.3. Acquisition Function Choices

To understand the performance of BE-CBO on more acquisition functions, we switch from EI to UCB (with beta = 0.1 as the default value in BoTorch) in our method and run empirical comparisons. As shown in Figure 18, the results suggest that EI and UCB perform similarly well on our benchmark problems overall, with some performance differences in particular problems. In practice, users may select the proper acquisition function to be used in BE-CBO based on the desired properties (such as the explicit control over exploration in UCB).

D.4. Dynamic Adaptation of Bounds for Boundary Exploration

To validate whether the proposed boundary exploration strategy is successful, we modify our BE-CBO with different methods of coupling the constraint into the acquisition function instead of doing boundary exploration. For example, CEI

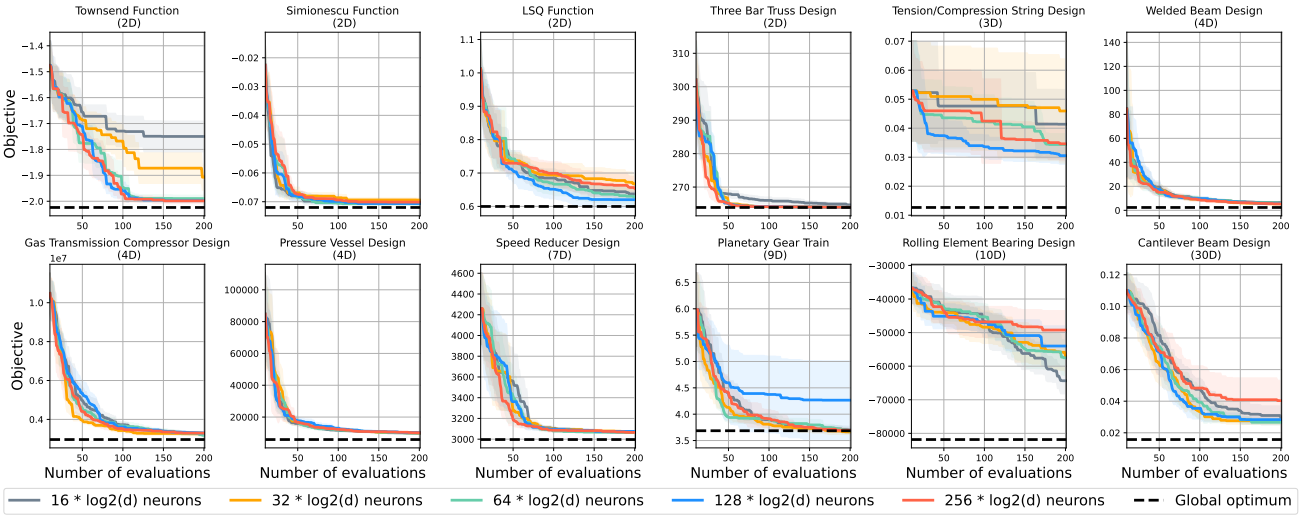

Figure 16: Comparison of BE-CBO with different number of neurons in a layer for the Deep Ensemble classifier. The current best value is shown w.r.t. the number of function evaluations. The curve is averaged over 10 different initial random seeds. The performance of BE-CBO is not sensitive to the number of neurons as long as it has at least 64 neurons in a layer.

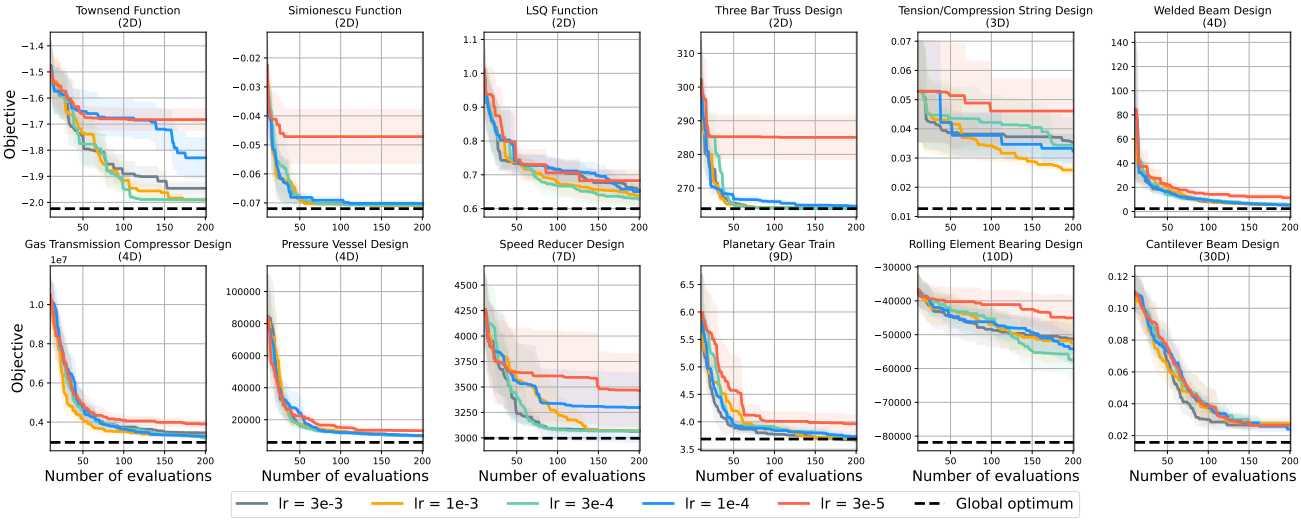

Figure 17: Comparison of BE-CBO with different learning rates for training the Deep Ensemble classifier. The current best value is shown w.r.t. the number of function evaluations. The curve is averaged over 10 different random seeds. Relatively large learning rates perform similarly good as small rates will underfit the network since the number of training iterations is fixed.

proposes to multiply the feasible probability with the acquisition function, and SCBO applies a 0.5 upper bound on the feasible probability as a constraint on top of the acquisition function. Specifically, when ignoring the rest of their approaches, they can be written as optimizing:

$$\text{CEI:} \quad \arg\max_x C(x)q(x)$$

$$\text{SCBO:} \quad \arg\max_x q(x) \quad \text{s.t.} \quad C(x) \geq 0.5$$

For reference, our approach is optimizing:

$$\text{BE-CBO:} \quad \arg\max_x q(x) \quad \text{s.t.} \quad C(x) \geq 0.5 - \sigma_E(x)$$

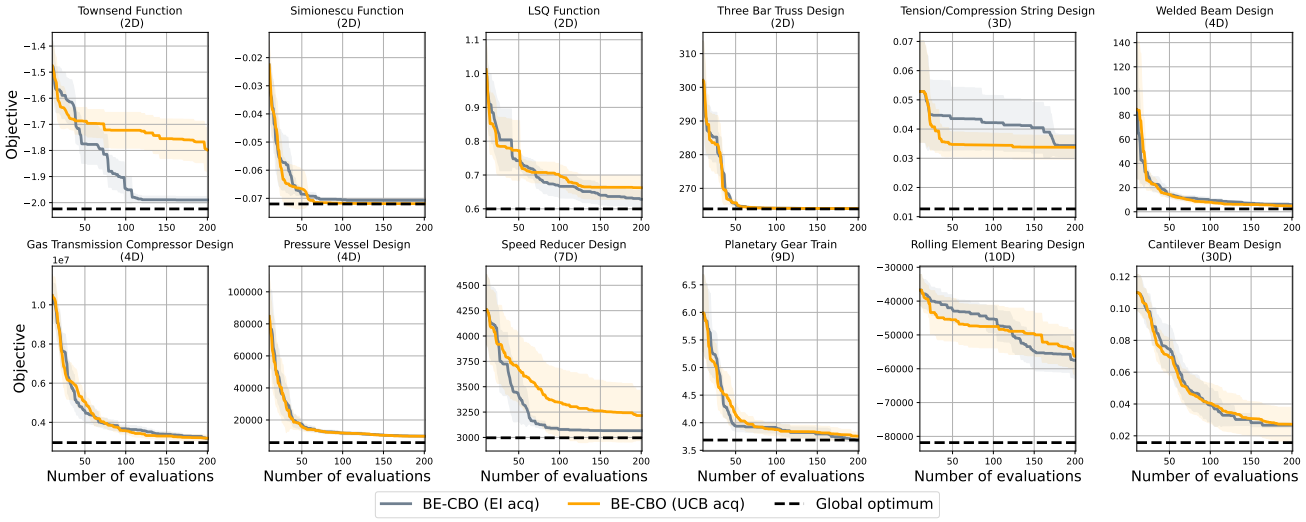

Figure 18: Comparison of BE-CBO with different acquisition functions (EI and UCB). The current best value is shown w.r.t. the number of function evaluations. The curve is averaged over 10 different random seeds.

Thus, we develop two variants of BE-CBO based on CEI and SCBO's optimization objective and call them BE-CBO-M (M stands for multiplication of $C(x)$) and BE-CBO-C (C for the 0.5 cutoff on $C(x)$), while the rest of the algorithm (e.g., surrogate models) share the same design choices as BE-CBO. Figure 19 compares the performance between these two variants and BE-CBO on all benchmark problems.

The results show that BE-CBO-M is clearly outperformed by BE-CBO-C and BE-CBO. Even though the multiplicative form of BE-CBO-M is a reasonable design in theory, in practice, the inaccuracy in constraint modeling can lead to failure cases easily. On one hand, multiplying $C(x)$ with the acquisition may encourage sampling in the safe region instead of exploration; On the other hand, when overestimation of the acquisition value happens in the infeasible region, even with a small probability of $C(x)$, it may still sample very far away in the infeasible region where the predicted $q(x)$ is huge. The algorithm gets stuck in such scenario because the new evaluated infeasible point does not update the classifier much since it already has a low feasibility prediction, then in the next iteration, the algorithm will keep proposing points in the similar region.

BE-CBO-C shares a similar performance with BE-CBO in low dimensional problems, but the performance starts to deteriorate when evaluated on high dimensional problems. Especially on the 30D Cantilever Beam Design problem, having a 0.5 cutoff bound in the objective makes BE-CBO-C explores much slower compared to BE-CBO. For more thorough evaluations, we implemented four additional high-dimensional real-world benchmark problems, including 10D multi-product batch plant design (Grossmann & Sargent, 1979), 10D synchronous optimal pulsewidth modulation control for 9-level and 11-level inverters (Rathore et al., 2010), and 14D industrial refrigeration system design (Paul H., 1987). Besides the dimensionality, we would like to highlight the feasibility ratio (FR) as an important property of our benchmark problems, which indicates the ratio of feasible samples among a massive amount of random samples in the parameter space. As shown in Figure 20, BE-CBO greatly outperforms BE-CBO-C on high-dimensional problems with a low feasibility ratio. To further validate this observation, we conduct controlled experiments on standard synthetic Ackley functions (constrained in the same way following SCBO) with varying dimensions. Figure 21 shows that BE-CBO becomes more advantageous than BE-CBO-C on problems with higher dimensions and lower feasibility ratios, and BE-CBO-C completely fails to improve in challenging cases. In such scenarios, exploration, or the ability to jump out of local minima, is crucial to the algorithm's performance. BE-CBO promotes exploration through our proposed dynamic bound strategy, while the fixed and conservative bound of BE-CBO-C discourages it from effective exploration.

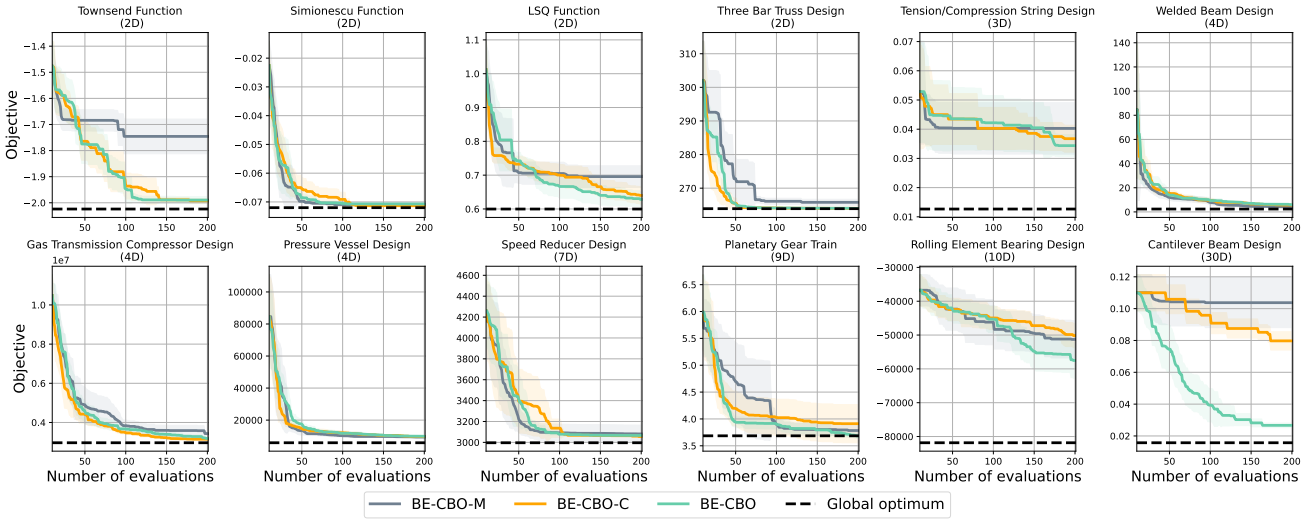

Figure 19: Comparison of BE-CBO with different variants of constraint coupling in the acquisition function. The current best value is shown w.r.t. the number of function evaluations. The curve is averaged over 10 different initial random seeds.

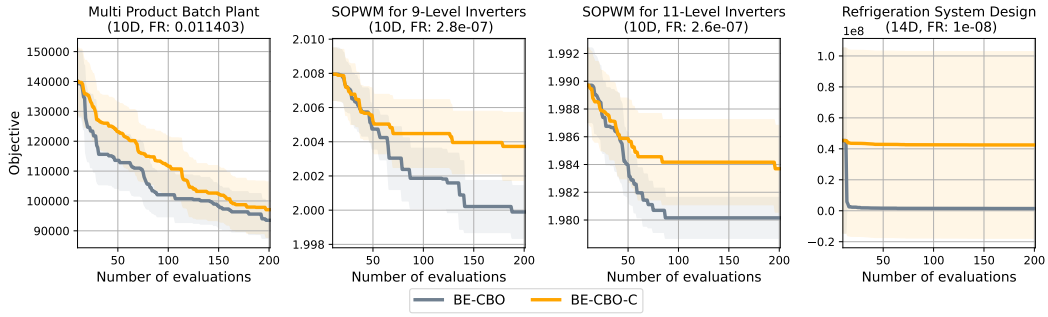

Figure 20: Comparison between BE-CBO and BE-CBO-C on additional high-dimensional real-world problems.

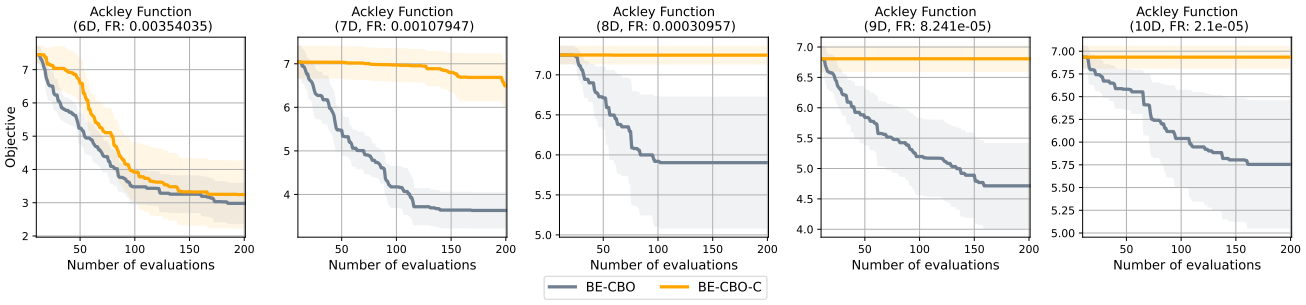

Figure 21: Comparison between BE-CBO and BE-CBO-C on additional synthetic Ackley functions with different dimensionalities.