Posterior Inference in Latent Space for Scalable Constrained Black-box Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Optimizing high-dimensional black-box functions under black-box constraints is a pervasive task in a wide range of scientific and engineering problems. These problems are typically harder than unconstrained problems due to hard-to-find feasible regions. While Bayesian optimization (BO) methods have been developed to solve such problems, they often struggle with the curse of dimensionality. Recently, generative model-based approaches have emerged as a promising alternative for constrained optimization. However, they suffer from poor scalability and are vulnerable to mode collapse, particularly when the target distribution is highly multi-modal. In this paper, we propose a new framework to overcome these challenges. Our method iterates through two stages. First, we train flow-based models to capture the data distribution and surrogate models that predict both function values and constraint violations with uncertainty quantification. Second, we cast the candidate selection problem as a posterior inference problem to effectively search for promising candidates that have high objective values while not violating the constraints. During posterior inference, we find that the posterior distribution is highly multi-modal and has a large plateau due to constraints, especially when constraint feedback is given as binary indicators of feasibility. To mitigate this issue, we amortize the sampling from the posterior distribution in the latent space of flow-based models, which is much smoother than that in the data space. We empirically demonstrate that our method achieves superior performance on various synthetic and real-world constrained black-box optimization tasks. Our code is publicly available here.

1 Introduction

Optimizing high-dimensional black-box functions under black-box constraints is a fundamental task across numerous scientific and engineering problems, including machine learning [1], drug discovery [2, 3], control [4, 5], and industrial design [6, 7]. In most cases, these problems are much harder than unconstrained problems due to analytically undefined and hard-to-find feasible regions [8].

Bayesian Optimization (BO) has been widely used to solve black-box optimization problems in a sample-efficient manner [9, 10]. While most BO methods focus on unconstrained optimization problems, some works address problems with black-box constraints by developing new acquisition functions [1, 11] or relaxing the constraints [12, 13]. However, even without constraints, BO methods scale poorly to high dimensionality [14]. Moreover, incorporating constraints makes the function landscape highly complex, hindering accurate estimation of surrogate models.

Recently, generative models have emerged as an alternative solution for black-box optimization problems with constraints [15, 16, 17]. For example, we can leverage generative models to sample protein sequences that maximize the binding affinity while preserving the naturalness of the design.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

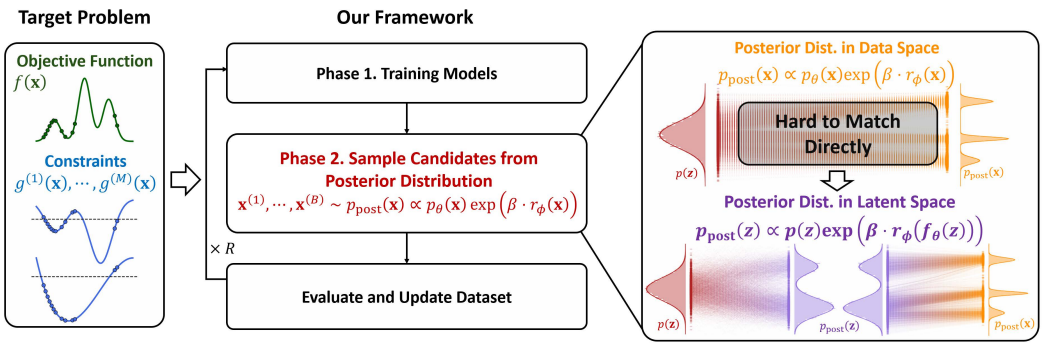


Figure 1: Motivating figure. In a high-dimensional setting, sampling from the posterior distribution is beneficial for selecting candidates. However, the posterior distribution is highly multi-modal and has a large plateau due to the constraints (orange one). We can mitigate this issue by sampling latents from the posterior distribution (purple one) of the latent space and projecting them into the data space.

However, existing methods rely on MCMC-based approaches [15], which limit scalability in high-dimensional spaces. While one can fine-tune pretrained generative models with reward functions [18, 19, 20], naive application of fine-tuning methods is vulnerable to mode collapse when the target distribution is highly multi-modal [21], which leads to a convergence on sub-optimal solutions.

In this paper, we propose a novel generative model-based framework for constrained black-box optimization to overcome the aforementioned limitations. To efficiently explore high-dimensional spaces, we first frame the candidate selection problem as sampling from the posterior distribution, which can be constructed by multiplying the prior distribution with a Lagrangian-relaxed objective. To sample candidates from the posterior distribution, our key idea is to amortize inference in the latent space of a flow-based model using an outsourced diffusion sampler [21], as illustrated in Figure 1. Since the posterior distribution in the latent space is much smoother than that in the data space, we can approximate the distribution more accurately and alleviate the mode collapse problem [22].

Our method iterates through two stages. First, we train a flow-based model to capture the current data distribution and surrogate models to predict the objective value and constraints, respectively. For the surrogate models, we use an ensemble to quantify the uncertainty of the prediction, as we have only a small amount of data that covers a tiny fraction of the whole search space. We treat a trained flow-based model as a prior, and Lagrangian relaxation of the objective as a reward function. Second, we sample candidates from the posterior distribution. As the posterior distribution is highly multi-modal and has a large plateau due to constraints, especially when constraint feedback is given as binary indicators of feasibility, we train a diffusion sampler that amortizes the posterior distribution in the latent space of flow models. Then, we sample latents from the diffusion sampler and project them into data space using a deterministic mapping derived from the trained flow model. By repeating these two stages, we can effectively capture high-scoring regions that satisfy the constraints.

We conduct extensive experiments on three synthetic and three real-world benchmarks to validate the superiority of our method on scalable constrained black-box optimization problems. We also consider a more challenging scenario where the feedback from the constraints is given as a binary value. We empirically show that our method outperforms several competitive baselines across different tasks.

2 Related Works

2.1 Constrained Black-box Optimization

Most scientific and engineering optimization problems involve black-box constraints, such as the synthesizability of molecules in chemical design [2] and safety constraints in robot control policies [4]. Existing BO methods solve this problem by either integrating the constraints directly into the acquisition function (cEI [23], LogcEI [24]) or by employing trust region approaches for scalability (SCBO [8], PCAGP-SCBO [7]). Another line of work utilizes evolutionary algorithms like CMA-ES [25, 26] with an augmented Lagrangian method to navigate constrained spaces. However, the performance of these methods often degrades as dimensionality and the number of evaluations increase, which motivates the need for a more scalable approach.

2.2 Generative Model-based Optimization

There are several attempts to utilize generative models for black-box optimization. In an offline setting, DDOM [27] trains a conditional diffusion model with classifier-free guidance and applies a loss-reweighting to emphasize samples with high objective values. DiffOPT [15] solves a constrained optimization problem. It applies diffusion to capture data distribution, followed by an iterative importance-sampling procedure. In an online setting, DiffBBO [28] and DiBO [29] both leverage diffusion models and incorporate uncertainty estimation during candidate selection. DiBO treats candidate selection as posterior inference to guide sampling toward regions of high reward and uncertainty, while DiffBBO selects conditioning targets by employing an uncertainty-based acquisition function. Unfortunately, constrained black-box optimization in the online setting remains unexplored.

2.3 Amortized Inference in Flow-based and Diffusion Models

Given a diffusion or flow prior $p_\theta(\mathbf{x})$ trained on a dataset and a reward function $r(\mathbf{x})$, sampling from the posterior $p_{\text{post}}(\mathbf{x}) \propto p_\theta(\mathbf{x})r(\mathbf{x})$ has numerous applications in downstream tasks [30, 18, 31, 32, 19, 21]. However, direct sampling from the unnormalized posterior $p_\theta(\mathbf{x})r(\mathbf{x})$ is intractable [18, 33].

To address this problem, some approaches train classifiers directly within intermediate noised spaces [30, 34] while others approximate posterior sampling via Markov Chain Monte Carlo (MCMC) procedures [22, 32, 35, 36]. However, training classifiers in noisy data spaces and employing MCMC methods scale poorly to high dimensionality. On the other hand, several methods utilize reinforcement learning [37, 38] or stochastic optimal control [19] to fine-tune the pretrained model and amortize the posterior sampling. Meanwhile, naive implementations of fine-tuning methods can be prone to mode collapse when the target distribution is highly multi-modal and has a large plateau region [21].

To mitigate this issue, we adopt the outsourced diffusion sampler method proposed by Venkatraman et al. [21]. Matching the distribution within the latent space significantly simplifies the alignment task when the distribution is highly multi-modal and has a large flat region in the original data space.

3 Preliminaries

3.1 Constrained Black-box Optimization

In constrained black-box optimization, our problem is:

$$\begin{aligned} &\text{find } \mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad \text{s.t. } g^{(1)}(\mathbf{x}) \leq 0, \cdots, g^{(M)}(\mathbf{x}) \leq 0 \\ &\text{with } R \text{ rounds of } B \text{ batch of queries} \end{aligned} \tag{1}$$

The objective function $f : \mathcal{X} \to \mathbb{R}$ and constraints $g^{(1)}, \cdots, g^{(M)} : \mathcal{X} \to \mathbb{R}$ are black-box functions. We also consider a more challenging scenario, only access to information on whether we violate constraints or not, i.e., $h^{(m)}(\mathbf{x}) = \mathbb{I}[g^{(m)}(\mathbf{x}) > 0]$. We refer to this as an indicator constraint.

3.2 Flow-based Models

Flow-based models [39, 40, 41] are a class of generative models for approximating a target distribution $q(\mathbf{x})$. Flow-based models are defined via the deterministic ordinary differential equation (ODE):

$$d\mathbf{x}_t = v_\theta(\mathbf{x}_t, t)\, dt \tag{2}$$

where $v_\theta(\mathbf{x}_t, t) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is a parametric velocity field.

For each given velocity field, the corresponding flow $\psi_\theta(\mathbf{x}_0, t) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ satisfies:

$$\frac{d}{dt}\psi_\theta(\mathbf{x}_0, t) = v_\theta(\psi_\theta(\mathbf{x}_0, t), t), \quad \psi_\theta(\mathbf{x}_0, 0) = \mathbf{x}_0. \tag{3}$$

The velocity field $v_\theta(\mathbf{x}_t, t)$ defines a continuous probability path p_t induced by the flow:

$$\mathbf{x}_t = \psi_\theta(\mathbf{x}_0, t) \sim p_t, \quad \text{where} \quad \mathbf{x}_0 \sim p_0. \tag{4}$$

Training Flow-based Models. We use Flow Matching [39] to learn the velocity field v_θ that generates a path interpolating smoothly between an initial distribution $p_0 = p$ and a target distribution $p_1 = q$.

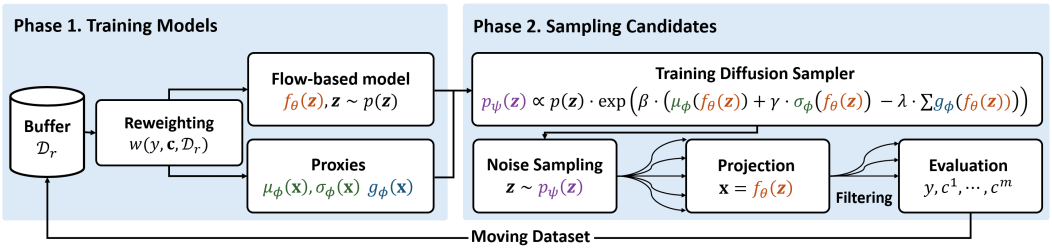


Figure 2: Overview of our method. **Phase 1:** Train flow-based models and proxies for the objective and constraints. **Phase 2:** Sample candidates from the posterior distribution using an outsourced diffusion sampler. After sampling, we utilize filtering to enhance sample efficiency. Then, we evaluate samples, update the dataset, and repeat the process until the evaluation budget is exhausted.

We employ the simplest linear interpolation path $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$, with derivative $\frac{d\mathbf{x}_t}{dt} = \mathbf{x}_1 - \mathbf{x}_0$, following [39]. The Flow Matching loss is expressed as:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0,I),\, \mathbf{x}_1 \sim q(\mathbf{x}),\, t \sim \text{Unif}(0,1)} \left[\| v_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0) \|_2^2 \right]. \tag{5}$$

3.3 Posterior Inference in Flow-based and Diffusion Models

Given a pretrained flow-based prior $p_\theta(\mathbf{x})$, and a reward function $r(\mathbf{x})$, we consistently encounter a situation where we need to sample from the posterior distribution, $p_{\text{post}}(\mathbf{x}) \propto p_\theta(\mathbf{x}) r(\mathbf{x})$. To sample from this intractable [33] distribution, we utilize the outsourced diffusion sampling [21].

We can interpret the sampling process of flow-based models into a noise generation $\mathbf{z} \sim p(\mathbf{z})$, followed by a deterministic transformation $\mathbf{x} = f_\theta(\mathbf{z})$, where $p(\mathbf{z})$ is standard normal and f_θ represents the learned mapping derived by prior. Under this formulation, by Proposition 3.1 of [21], we can sample from the posterior distribution by substituting noise generation as $\mathbf{z} \sim p_{\text{post}}(\mathbf{z}) \propto p(\mathbf{z}) r(f_\theta(\mathbf{z}))$.

To approximate the target distribution $p_\psi(\mathbf{z}) \approx p_{\text{post}}(\mathbf{z})$, we can learn the parameters of diffusion sampler ψ with the trajectory balance (TB) objective [42, 43]:

$$\mathcal{L}_{\text{TB}}(\mathbf{z}_{0:1}; \psi) = \left(\log \frac{Z_\psi p(\mathbf{z}_0) \prod_{i=0}^{T-1} p_F(\mathbf{z}_{(i+1)\Delta t} | \mathbf{z}_{i\Delta t}; \psi)}{p(\mathbf{z}_1) r(f_\theta(\mathbf{z}_1)) \prod_{i=1}^{T} p_B(\mathbf{z}_{(i-1)\Delta t} | \mathbf{z}_{i\Delta t})} \right)^2, \tag{6}$$

where Z_ψ is the parameterized partition estimator, $(\mathbf{z}_0 \to \mathbf{z}_{\Delta_t} \to \cdots \mathbf{z}_1 = \mathbf{z})$ is the discrete time Markov chain of reverse-time stochastic differential equation (SDE) [44] with time increment $\Delta t = \frac{1}{T}$. p_F and p_B are transition kernels of the discretized reverse and forward SDE.

4 Method

In this section, we introduce **CiBO**, a new framework for scalable constrained black-box optimization by leveraging generative models. Our method consists of two iterative stages. First, we train a flow-based model to capture the data distribution and surrogate models to predict objective values and constraints with uncertainty quantification. Next, we sample candidates from the posterior distribution. To accomplish this, we train a diffusion sampler that draws samples from the posterior distribution in the latent space. After sampling, we evaluate candidates, update the dataset, and repeat the process until the evaluation budget is exhausted. Figure 2 illustrates the overview of our method.

4.1 Phase 1. Training Models

In each round r, we have a pre-collected dataset $\mathcal{D}_r = \{\mathbf{x}_i, y_i, \mathbf{c}_i\}_{i=1}^I$, where $y_i = f(\mathbf{x}_i)$, $\mathbf{c}_i = \{c_i^m | c_i^m = g^{(m)}(\mathbf{x}_i),\ \forall m = 1, \ldots, M\}$, and I is the number of data points collected so far.

Training Prior. We first train a prior model p_θ to capture the current data distribution. As the search space is too high-dimensional, it is better to implicitly constrain the search space close to the current data distribution. We use flow-based models to learn this distribution using Equation (5).

Training Surrogates. We also train surrogate models to predict both objective values and constraints. As we are only able to access a small number of data points in the vast search space, we need to

properly quantify the uncertainty of the prediction. To this end, we train an ensemble of proxies to estimate objective values with uncertainty quantification [45]. Specifically, we train an ensemble of K proxies $f_{\phi_1}, \ldots, f_{\phi_K}$ for objective values, and individual proxy $g_\phi^{(1)}, \ldots, g_\phi^{(M)}$ for each constraint.

Reweighted Training. During training, we introduce a reweighted training scheme [27, 46, 47] to focus on promising data points with high objective values while not violating constraints. Specifically, the weight for each data point is computed as follows:

$$l(y, \mathbf{c}) = y - \lambda \sum_{m=1}^{M} \max(0, c^m), \quad w\left(y, \mathbf{c}, \mathcal{D}_r\right) = \frac{\exp\left(l\left(y, \mathbf{c}\right)\right)}{\sum_{(y', \mathbf{c}') \in \mathcal{D}_r} \exp\left(l\left(y', \mathbf{c}'\right)\right)}. \tag{7}$$

Then, our training objective for flow-based models and proxies can be described as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0 \sim \mathcal{N}(0,I),\, (\mathbf{x},y,\mathbf{c}) \in \mathcal{D}_r,\, t \sim \text{Unif}(0,1)} \left[w(y, \mathbf{c}, \mathcal{D}_r) \| v_\theta(\mathbf{x}_t, t) - (\mathbf{x} - \mathbf{x}_0) \|_2^2 \right], \tag{8}$$

$$\mathcal{L}(\phi) = \sum_{(\mathbf{x},y,\mathbf{c}) \in \mathcal{D}_r} w(y, \mathbf{c}, \mathcal{D}_r) \left[\sum_{k=1}^{K} \left(y - f_{\phi_k}\left(\mathbf{x}\right)\right)^2 + \sum_{m=1}^{M} \left(c^m - g_\phi^{(m)}\left(\mathbf{x}\right) \right)^2 \right]. \tag{9}$$

4.2 Phase 2. Sampling Candidates

After training models, we proceed to select candidates for evaluation in the current round. As the search space is high-dimensional, the prediction of surrogate models is likely to be inaccurate in regions that are too far away from the dataset collected so far. Therefore, it is advantageous to sample candidates from the distribution that satisfies the two desiderata: (1) promote exploration towards high-scoring and feasible regions, and (2) prevent sampling candidates that deviate too far from the current data distribution. To accomplish these objectives, we cast the candidate selection problem as sampling from the target distribution p_{post} defined as follows:

$$p_{\text{post}}(\mathbf{x}) = \arg\max_{p \in \mathcal{P}} \mathbb{E}_{\mathbf{x} \sim p}\left[r_\phi(\mathbf{x})\right] - \frac{1}{\beta} \cdot D_{\text{KL}}\left(p \,\|\, p_\theta\right), \tag{10}$$

where $\mathcal{P}$ is the space of all probability distributions over the domain $\mathcal{X}$, and

$$r_\phi(\mathbf{x}) = \mu_\phi(\mathbf{x}) + \gamma \cdot \sigma_\phi(\mathbf{x}) - \lambda \sum_{m=1}^{M} \max(0, g_\phi^{(m)}(\mathbf{x})). \tag{11}$$

$\mu_\phi(\mathbf{x})$ and $\sigma_\phi(\mathbf{x})$ represent the mean and standard deviation from the ensemble of surrogate models for the objective. γ controls exploration-exploitation trade-off, β is an inverse temperature, and λ is a Lagrange multiplier. Based on derivation from [48], our target distribution analytically derived as:

$$p_{\text{post}}(\mathbf{x}) \propto p_\theta(\mathbf{x}) \exp\left(\beta \cdot \left[r_\phi(\mathbf{x})\right]\right). \tag{12}$$

If we treat the flow-based model $p_\theta(\mathbf{x})$ as a prior and the exponential term $\exp(\beta \cdot [r_\phi(\mathbf{x})])$ as a reward $r(\mathbf{x})$, then our objective is to sample from the posterior distribution $p_{\text{post}}(\mathbf{x}) \propto p_\theta(\mathbf{x}) r(\mathbf{x})$.

Amortized Inference in Latent Space. However, directly sampling from this posterior is intractable [33]. Also, the target posterior is highly multi-modal and has a large plateau due to the constraint penalties in (11), making finetuning-based methods [18, 37] susceptible to mode collapse. To this end, we utilize an amortized sampler in the latent space suggested by Venkatraman et al [21].

As introduced in Section 3.3, we can view the sampling procedure of flow-based models as drawing samples from the standard normal distribution $\mathbf{z} \sim p(\mathbf{z})$, followed by the deterministic transformation $\mathbf{x} = f_\theta(\mathbf{z})$. Within this framework, we can generate samples from the posterior distribution $p_{\text{post}}(\mathbf{x})$ by modifying the noise generation distribution as follows:

$$\mathbf{z} \sim p_{\text{post}}(\mathbf{z}) \propto p(\mathbf{z}) r(f_\theta(\mathbf{z})). \tag{13}$$

To sample latents $\mathbf{z}$ from the posterior distribution in the latent space $p_{\text{post}}(\mathbf{z})$, we train a diffusion model $p_\psi(\mathbf{z})$ to amortize $p_{\text{post}}(\mathbf{z})$ with the following Trajectory Balance (TB) objective:

$$\mathcal{L}_{\text{TB}}(\mathbf{z}_{0:1}; \psi) = \left(\log \frac{Z_\psi p(\mathbf{z}_0) \prod_{i=0}^{T-1} p_F(\mathbf{z}_{(i+1)\Delta t} | \mathbf{z}_{i\Delta t}; \psi)}{p(\mathbf{z}_1) r(f_\theta(\mathbf{z}_1)) \prod_{i=1}^{T} p_B(\mathbf{z}_{(i-1)\Delta t} | \mathbf{z}_{i\Delta t})} \right)^2. \tag{14}$$

By training an amortized sampler in the latent space of flow-based models, we can more accurately sample candidates from the target distribution as the posterior distribution in the latent space is smoother than that in the data space. We also adopt off-policy training, detailed in Appendix D.2.1

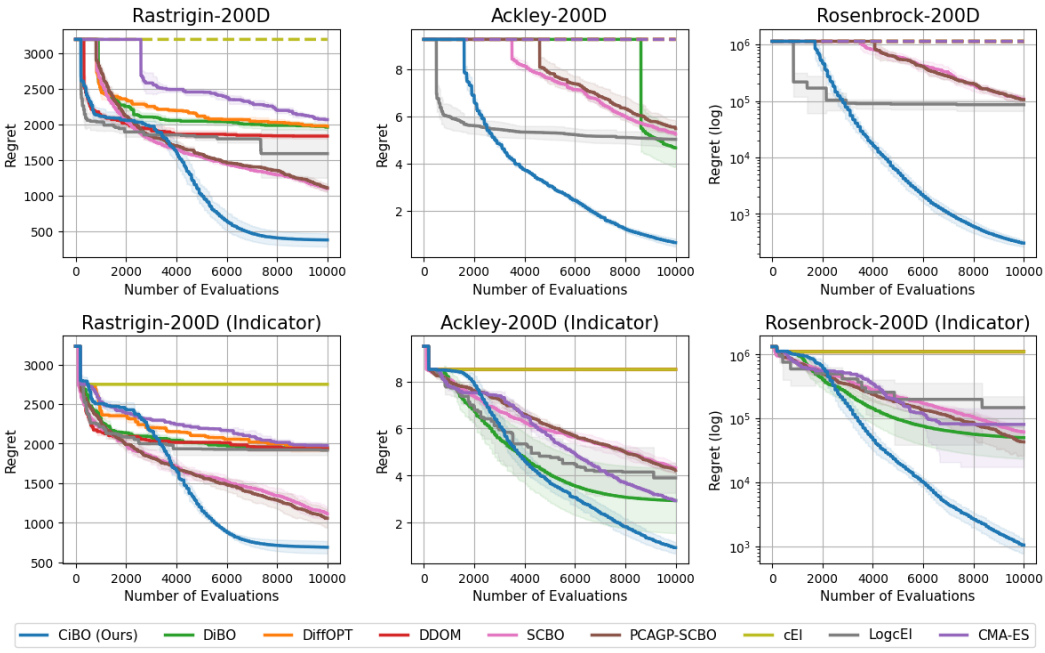


Figure 3: Comparison between our method and baselines in synthetic tasks. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported. A dashed line means that no feasible solutions were found.

4.3 Filtering, Evaluation and Moving Dataset

Filtering. After sampling from the posterior distribution, we need to carefully select candidates for the sample efficiency of the algorithm. To do so, we generate $N \cdot B$ samples from the amortized sampler and select the top-B samples in terms of Lagrangian relaxation of objectives as candidates.

Evaluation and Moving Dataset. We evaluate the values of the objective function and constraint functions for each selected candidate, then update the dataset with new observations. During the update, we empirically find that taking only a subset of total observations is beneficial in terms of computational complexity. We remove the samples with the lowest Lagrangian-relaxed objective if the dataset size is larger than the buffer size L. The pseudocode of our method is in Algorithm 1.

5 Experiments

In this section, we report experimental results for scalable constrained black-box optimization tasks. First, we perform experiments on three 200-dimensional synthetic functions, which are the standard benchmarks in Bayesian Optimization (BO) studies [14]. Furthermore, we assess the performance of our method on a more challenging scenario, where the feedback from constraints is given as binary indicators of feasibility. We refer to this setting as the indicator constraint setting. Finally, we conduct experiments on three real-world optimization tasks: Rover Planning 60D [8, 49], Mopta 124D [6], and Lasso DNA 180D [50]. The detailed description of each task can be found in Appendix A.

For evaluation, we report the minimum regret of feasible solutions over the course of the training, and assign the largest regret found in all algorithms to the infeasible solutions, following [8, 51].

5.1 Baselines

We compare our method with several constrained BO baselines, including cEI [23], LogcEI [24], SCBO [8], PCAGP-SCBO [7], and the evolutionary search algorithm CMA-ES [25]. We also evaluate generative model-based approaches specifically designed for constrained optimization: DiffOPT [15], as well as methods that can be extended to constrained optimization via the Lagrangian relaxation: DDOM [27] and DiBO [29]. Detailed implementations of all baselines are provided in Appendix B.

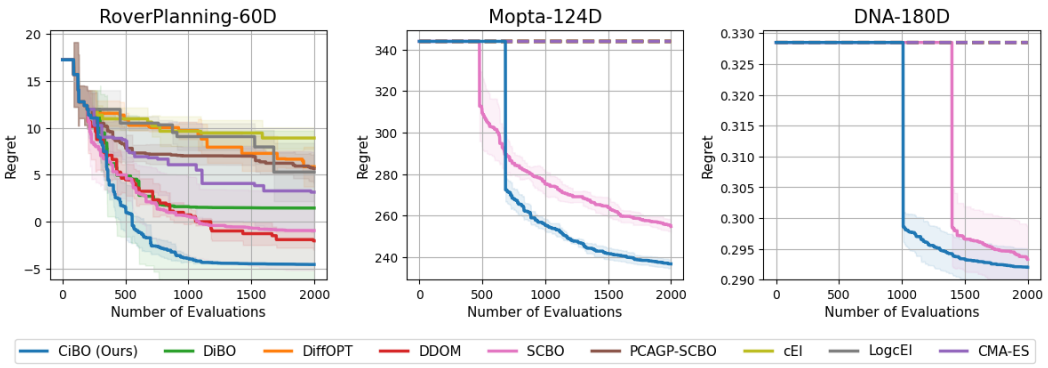


Figure 4: Comparison between our method and baselines in real-world tasks. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported. A dashed line means that no feasible solutions were found.

5.2 Synthetic Experiments

We first conduct experiments on three synthetic functions, Rastrigin-200D, Ackley-200D, and Rosenbrock-200D. For each function, we utilize two inequality constraints proposed by SCBO [8]: $\sum_{d=1}^{200} x_d \leq 0$ and $||\mathbf{x}||_2^2 \leq 30$. We conduct all experiments with an initial dataset size of $|D_0| = 200$, using a batch size of $B = 100$ and a maximum evaluation limit of $10,000$. In the indicator constraints scenarios, as it is too challenging to find an initial feasible solution across all baselines, we sample 10 points within feasible regions during initialization.

As shown in the Figure 3, our method outperforms all baselines across different synthetic tasks, both in the standard and indicator constraints. Generative model-based methods, including DiffOPT and DDOM, struggle to find a feasible solution and fail to improve on indicator constraints. While DiBO achieves better feasibility, its finetuning-based approach suffers from mode collapse and tends to converge to suboptimal solutions. These results show that employing an outsourced diffusion sampler significantly enhances performance in constrained black-box optimization by effectively capturing multi-modal and expansive flat target distributions.

Constrained BO methods (SCBO, PCAGP-SCBO, and LogcEI) successfully identify feasible points but show limited sample efficiency compared to our method across all tasks. The evolutionary search algorithm CMA-ES performs modestly in general but fails to find a feasible solution for some tasks. These results underscore that our approach effectively captures both high-scoring and feasible regions in a sample-efficient manner. Furthermore, compared to other baselines, our method consistently finds feasible solutions throughout the optimization process, which is illustrated in Appendix E.1.

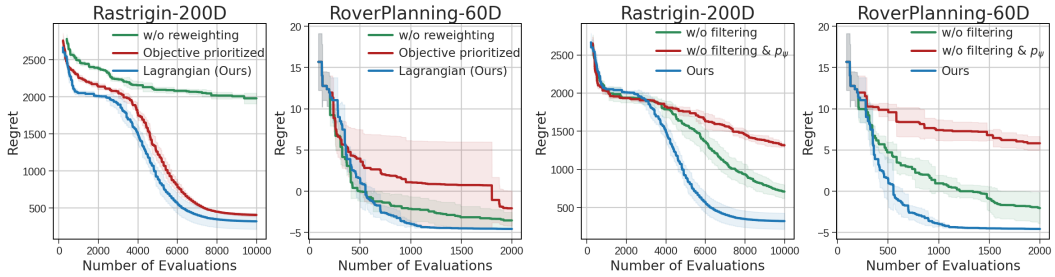
5.3 Real World Experiments

To validate the robustness of our approach, we evaluate our method on three challenging real-world benchmark problems: (1) Rover Planning in 60 dimensions with 15 infeasible square-shaped regions, (2) Mopta in 124 dimensions with 68 constraints, and (3) Lasso DNA in 180 dimensions with 5 constraints. For all experiments, we initialize with $|\mathcal{D}_0| = 200$ data points and limit evaluations to $2,000$. We use a batch size of $B = 50$ for Rover Planning and Lasso DNA, and $B = 20$ for Mopta, as no baseline methods could identify feasible solutions with $B = 50$.

As illustrated in Figure 4, our approach consistently identifies high-quality feasible solutions with superior sample efficiency across all tasks. We observe that the performance gap between our method and other baselines becomes larger on real-world problems and most baselines failed to find any feasible solutions for the challenging Mopta-124D and DNA-180D tasks. While SCBO is the only competing method to achieve feasibility alongside our approach, it exhibits lower sample efficiency. This highlights the robustness of our approach for scalable constrained black-box optimization.

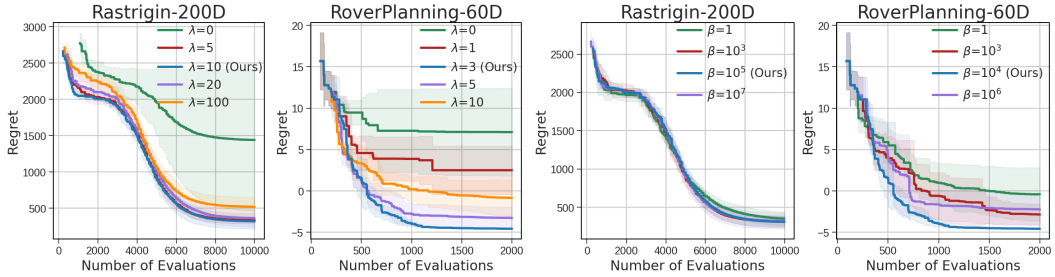
5.4 Additional Analysis

In this section, we conduct a comprehensive analysis of each component of our proposed method through ablation experiments on Rastrigin 200D and Rover Planning 60D.



(a) Reweighted Training

(b) Sampling Procedure



(c) Analysis on λ

(d) Analysis on β

Figure 5: Additional analysis for various components of CiBO. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

Reweighted Training. To investigate the effectiveness of our reweighted training approach suggested in Equation (7), we conduct a comparative analysis of two variants: training without reweighting, applying weights based on the objective values (Objective-prioritized). As shown in Figure 5a, variants without reweighting or using objective-prioritized reweighting exhibit low sample efficiency.

Sampling Procedure. We analyze the effect of each component in candidate sampling. We conduct experiments with two variants: removing filtering, and removing both filtering and the diffusion sampler, thus sampling candidates directly from the prior p_θ. As depicted in Figure 5b, there is a significant performance gap between our method and other variants, validating the effectiveness of each proposed component. We also experiment with the filtering coefficient N in Appendix E.2.

Lagrangian Multiplier λ. We introduce the Lagrangian multiplier λ. As shown in Figure 5c, setting $\lambda = 0$ (eliminating the constraint penalty) significantly degrades performance on both tasks, as it only focuses on high objective values and neglects the feasibility of solutions. Conversely, excessively high λ values diminish the influence of the objective function, resulting in reduced sample efficiency.

Inverse Temperature β. The inverse temperature controls the balance between the prior $p_\theta(\mathbf{x})$ and the reward function $r(\mathbf{x})$. We conduct experiments by varying β values. As shown in Figure 5d, using a moderately high β generally helps to improve sample efficiency. However, if β is too high, the performance is heavily dependent on the accuracy of surrogate models, leading to slow convergence. This validates that incorporating prior distribution is crucial for scalability (Section 4.2).

Further Analysis. To further understand our method, we analyze the impact of the buffer size L (Appendix E.3), batch size B, and initial dataset size $|\mathcal{D}_0|$ (Appendix E.5). We also investigate the effect of off-policy training (Appendix E.4) and runtime scalability of our method (Appendix F).

6 Conclusion

We introduced CiBO, a generative model-based framework for scalable constrained black-box optimization. Our approach formulates candidate selection as posterior inference, leveraging flow-based models to capture the data distribution and surrogate models to predict both objectives and constraints. By amortizing posterior sampling in the latent space with outsourced diffusion samplers, our method effectively addresses the challenges posed by highly multi-modal and flat posterior distributions that arise from incorporating constraints. Extensive experiments across synthetic and real-world benchmarks demonstrate the superiority of our proposed method.

References

[1] Jacob Gardner, Matt Kusner, Kilian Weinberger, John Cunningham, et al. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, pages 937–945. PMLR, 2014.

[2] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

[3] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

[4] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *International conference on robotics and automation (ICRA)*, 2016.

[5] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 112(10):3713–3747, 2023.

[6] MF Anjos and DR Jones. Mopta 2008 benchmark. *URL http://www. miguelanjos. com/jones-benchmark*, 2009.

[7] Hauke F Maathuis, Roeland De Breuker, and Saullo GP Castro. High-dimensional bayesian optimisation with large-scale constraints via latent space gaussian processes. *arXiv preprint arXiv:2412.15679*, 2024.

[8] David Eriksson and Matthias Poloczek. Scalable constrained bayesian optimization. In *International conference on artificial intelligence and statistics*, pages 730–738. PMLR, 2021.

[9] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[10] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[11] José Miguel Hernández-Lobato, Michael Gelbart, Matthew Hoffman, Ryan Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. In *International conference on machine learning*, pages 1699–1707. PMLR, 2015.

[12] Victor Picheny, Robert B Gramacy, Stefan Wild, and Sebastien Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented lagrangian. *Advances in neural information processing systems*, 29, 2016.

[13] Setareh Ariafar, Jaume Coll-Font, Dana Brooks, and Jennifer Dy. Admmbo: Bayesian optimization with unknown constraints using admm. *Journal of Machine Learning Research*, 20(123):1–26, 2019.

[14] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[15] Lingkai Kong, Yuanqi Du, Wenhao Mu, Kirill Neklyudov, Valentin De Bortoli, Dongxia Wu, Haorui Wang, Aaron M Ferber, Yian Ma, Carla P Gomes, and Chao Zhang. Diffusion models as constrained samplers for optimization with unknown constraints. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[16] Wenqian Xing, JungHo Lee, Chong Liu, and Shixiang Zhu. Black-box optimization with implicit constraints for public policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28511–28519, 2025.

[17] Masatoshi Uehara, Xingyu Su, Yulai Zhao, Xiner Li, Aviv Regev, Shuiwang Ji, Sergey Levine, and Tommaso Biancalani. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design. *arXiv preprint arXiv:2502.14944*, 2025.

[18] Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, Alexandre Adam, Jarrid Rector-Brooks, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Amortizing intractable inference in diffusion models for vision, language, and control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[19] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024.

[21] Siddarth Venkatraman, Mohsin Hasan, Minsu Kim, Luca Scimeca, Marcin Sendera, Yoshua Bengio, Glen Berseth, and Nikolay Malkin. Outsourced diffusion sampling: Efficient posterior inference in latent spaces of generative models. In *International Conference on Machine Learning (ICML)*, 2025.

[22] Florentin Coeurdoux, Nicolas Dobigeon, and Pierre Chainais. Normalizing flow sampling with langevin dynamics in the latent space. *arXiv preprint arXiv:2305.12149*, 2023.

[23] Matthias Schonlau, William J Welch, and Donald R Jones. Global versus local search in constrained optimization of computer models. *Lecture notes-monograph series*, pages 11–25, 1998.

[24] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

[25] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102, 2006.

[26] Asma Atamna, Anne Auger, and Nikolaus Hansen. Augmented lagrangian constraint handling for cma-es—case of a single linear constraint. In *International Conference on Parallel Problem Solving from Nature*, pages 181–191. Springer, 2016.

[27] Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *International Conference on Machine Learning (ICML)*, 2023.

[28] Dongxia Wu, Nikki Lijing Kuang, Ruijia Niu, Yian Ma, and Rose Yu. Diff-BBO: Diffusion-based inverse modeling for black-box optimization. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*, 2024.

[29] Taeyoung Yun, Kiyoung Om, Jaewoo Lee, Sujin Yun, and Jinkyoo Park. Posterior inference with diffusion models for high-dimensional black-box optimization. In *International Conference on Machine Learning (ICML)*, 2025.

[30] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[31] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2022.

[32] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Ruiqi Feng, Chenglei Yu, Wenhao Deng, Peiyan Hu, and Tailin Wu. On the guidance of flow matching. In *Forty-second International Conference on Machine Learning*, 2025.

[34] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

[35] Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

[36] Gabriel Cardoso, Sylvain Le Corff, Eric Moulines, et al. Monte carlo guided denoising diffusion models for bayesian linear inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024.

[37] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[38] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[39] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[40] Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[41] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

[42] Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. Improved off-policy training of diffusion samplers. *Advances in Neural Information Processing Systems*, 37:81016–81045, 2024.

[43] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[44] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[45] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[46] Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[47] Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[48] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[49] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754. PMLR, 2018.

[50] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso. In *International Conference on Automated Machine Learning*, pages 2–1. PMLR, 2022.

[51] José Miguel Hern, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, Zoubin Ghahramani, et al. A general framework for constrained bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17(160):1–53, 2016.

[52] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. A supervised learning approach involving active subspaces for an efficient genetic algorithm in high-dimensional optimization problems. *SIAM Journal on Scientific Computing*, 43(3):B831–B853, 2021.

[53] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

[54] Zeji Yi, Yunyue Wei, Chu Xin Cheng, Kaibo He, and Yanan Sui. Improving sample efficiency of high dimensional bayesian optimization with mcmc. In *6th Annual Learning for Dynamics & Control Conference*, pages 813–824. PMLR, 2024.

[55] Zelda B Zabinsky and Robert L Smith. Hit-and-run methods. *Encyclopedia of Operations Research and Management Science*, pages 721–729, 2013.

[56] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive bayesian optimization in nested subspaces. *Advances in Neural Information Processing Systems*, 35:11586–11601, 2022.

[57] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[58] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[60] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[61] Ricky T. Q. Chen. torchdiffeq, 2018.

[62] Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

[63] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[64] J. H. Halton. Sequential monte carlo. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(1):57–78, 1962.

[65] Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[66] John Skilling. Nested sampling for general bayesian computation. 2006.

[67] Pablo Lemos, Nikolay Malkin, Will Handley, Yoshua Bengio, Yashar Hezaveh, and Laurence Perreault-Levasseur. Improving gradient-guided nested sampling for posterior inference. In *International Conference on Machine Learning*, pages 27230–27253. PMLR, 2024.

[68] Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *International Conference on Learning Representations*, 2022.

[69] Francisco Vargas, Will Sussman Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *The Eleventh International Conference on Learning Representations*, 2023.

[70] Julius Berner, Lorenz Richter, and Karen Ullrich. An optimal control perspective on diffusion-based generative modeling. *Transactions on Machine Learning Research*, 2024.

[71] Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. In *The Twelfth International Conference on Learning Representations*, 2024.

[72] Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken. Transport meets variational inference: Controlled monte carlo diffusions. In *The Twelfth International Conference on Learning Representations*, 2024.

[73] Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-Garcıa, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. In *International Conference on Machine Learning*, pages 18269–18300. PMLR, 2023.

[74] Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

[75] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[76] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

[77] Aaron J Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Daniel S. Levine, Brandon M Wood, Bin Hu, Brandon Amos, Brian Karrer, Xiang Fu, Guan-Horng Liu, and Ricky T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. In *Frontiers in Probabilistic Inference: Learning meets Sampling*, 2025.

[78] Minsu Kim, Sanghyeok Choi, Taeyoung Yun, Emmanuel Bengio, Leo Feng, Jarrid Rector-Brooks, Sungsoo Ahn, Jinkyoo Park, Nikolay Malkin, and Yoshua Bengio. Adaptive teachers for amortized samplers. In *The Thirteenth International Conference on Learning Representations*, 2025.

Appendix

A Task Details

A.1 Synthetic Functions

We evaluate three synthetic functions in our constrained black-box optimization experiments: Rastrigin, Ackley, and Rosenbrock. The Rastrigin and Ackley functions are highly multi-modal functions with numerous local minima, whereas the Rosenbrock function features a narrow valley that makes convergence to the global minimum notoriously difficult [52]. Following [53, 54], we define the search domains as Rastrigin: $[-5, 5]^D$, Ackley: $[-5, 10]^D$, and Rosenbrock: $[-5, 10]^D$. All functions are subject to two constraints:

$$\sum_{d=1}^{200} x_d \leq 0 \quad \text{and} \quad ||\mathbf{x}||_2^2 \leq 30$$

Although prior work enforced the tighter bound $||\mathbf{x}||_2^2 \leq 5$, we relax this constraint in our high-dimensional setting. For the indicator constraint experiments, we sample initial feasible points by hit-and-run MCMC [55].

A.2 Rover Trajectory Planning

Rover Trajectory Planning is a trajectory optimization task in a 2D environment introduced by [49]. The objective is to optimize the rover's trajectory, where its trajectory is represented by 30 points defining a B-Spline. We place 15 impassable obstacles o_i and impose collision-avoidance constraints $c_i(\mathbf{x})$ as in [8]:

$$c_i(\mathbf{x}) = \begin{cases} -\,d\big(o_i, \gamma(\mathbf{x})\big) & \text{if } \gamma(\mathbf{x}) \cap o_i = \varnothing, \\ \max\limits_{\alpha \in \gamma(\mathbf{x}) \cap o_i} \; \min\limits_{\beta \in \partial o_i} d(\alpha, \beta) & \text{otherwise.} \end{cases}$$

where $\gamma(\mathbf{x})$ denotes final trajectory, o_i is the region of the obstacle and ∂o_i denotes the boundary of o_i. A trajectory is feasible if and only if it does not intersect any obstacle. We follow the implementation from [49], but since there is no released code for the constraints, we implement the violation metric ourselves. Below is an example of the trajectory found by our method.

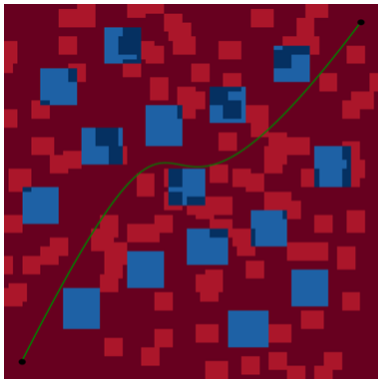


Figure 6: Trajectory found by CiBO, achieving regret of -4.59.

A.3 Vehicle Design with 68 Constraints (MOPTA)

MOPTA is the high-dimensional real-world problem of large-scale multidisciplinary mass optimization [6]. The objective is to minimize a vehicle's mass, which incorporates decisions about materials, gauges, and vehicle shape with 68 performance constraints. The best-known optimum mass is approximately 222.74. We followed the implementation from [56]. [1]

A.4 LassoBench

LassoBench [50] [2] is a high-dimensional benchmark for hyperparameter optimization, specifically designed to tune the hyperparameters of the Weighted LASSO (Least Absolute Shrinkage and Selection Operator) regression model. It includes both synthetic tasks (simple, medium, high, and hard) and real-world tasks (Breast cancer, Diabetes, Leukemia, DNA, and RCV1). In this work, we focus on the DNA task, a microbiology classification problem. It computes the average validation error across all cross-validation folds as an unconstrained objective. We reformulate the problem by retaining the mean validation error as the objective while introducing constraints that the validation error on each fold must not exceed 0.32.

B Baselines Details

In this section, we provide a thorough description of our baseline implementation details and specify the hyperparameter settings used across all experiments.

DiBO [29]: We use the original code[3], and adapt DiBO to handle constrained optimization by reformulating the objective as a Lagrangian, setting the same λ value as our methods for fair comparison.

DiffOPT [15]: As there is no publicly available code, we re-implement this baseline on our own. To approximate the data distribution, we use diffusion models with a similar architecture to our method. To enable accurate sampling from the target distribution, we implement Langevin dynamics as the energy function, which can be constructed by surrogate models in our setting, is differentiable.

DDOM [27]: Building on the original implementation[4], we reconstruct this baseline with network architecture matching our flow-based model. While maintaining the method's specific parameters as specified in the original work, we incorporate a Lagrangian framework and set the same λ as ours.

SCBO [8]: We follow the tutorial code for SCBO provided by `botorch` [5] to reproduce the results.

PCAGP-SCBO [7]: To reproduce PCAGP-SCBO, we follow the code for SCBO and then apply `torch pca`[6] to project high-dimensional data into a reduced latent space with dimension l before fitting GP surrogates for constraints. For all synthetic tasks, we use $l = 2$ and for real-world tasks, we conduct a hyperparameter search on $[2, \lfloor D/2 \rfloor]$ and report the best one.

cEI [23]: We implement cEI acquisition function by using `qExpectedImprovement()` in `botorch` library. We train a GP surrogate model independently for the objective and each constraint.

LogcEI [24]: We implement logcEI acquisition function by using `qLogExpectedImprovement()` in `botorch` library. We train a GP surrogate model independently for the objective and each constraint.

CMA-ES [25]: We employ the `pycma`[7] library [57]. For constraint handling, we formulate the problem using the same Lagrangian approach with the same λ value as ours for each task.

[1] `https://github.com/LeoIV/BAxUS`
[2] `https://github.com/ksehic/LassoBench`
[3] `https://github.com/umkiyoung/DiBO`
[4] `https://github.com/siddarthk97/ddom`
[5] `https://botorch.org/docs/tutorials/scalable_constrained_bo/`
[6] `https://github.com/valentingol/torch_pca`
[7] `https://github.com/CMA-ES/pycma`

C Algorithms

Algorithm 1 CiBO

Input: Initial dataset $\mathcal{D}_0$; Max rounds R; Batch size B; Buffer size L; Number of constraints M; Flow model p_θ; Diffusion sampler p_ψ; Proxies $f_{\phi_1}, \cdots, f_{\phi_K}, g_\phi^{(1)}, \cdots, g_\phi^{(M)}$;
for $r = 0, \ldots, R-1$ **do**
  Initialize $p_\theta, p_\psi, f_{\phi_1}, \cdots, f_{\phi_K}, g_\phi^{(1)}, \cdots, g_\phi^{(M)}$

  Phase 1. Training Models
  Compute weights $w(y, \mathbf{c}, \mathcal{D}_r)$ with Equation (7)
  Train p_θ with Equation (8)
  Train $f_{\phi_1}, \cdots, f_{\phi_K}, g_\phi^{(1)}, \cdots, g_\phi^{(M)}$ with Equation (9)

  Phase 2. Sampling Candidates
  Train p_ψ with Equation (14) using prior p_θ and $\mathbf{z} \sim N(0, \mathbf{I})$
  Sample latent noise with $\{\mathbf{z}_i\}_{i=1}^{NB} \sim p_\psi(\mathbf{z})$
  Projection to data space with learned mapping $\mathbf{x}_i = f_\theta(\mathbf{z}_i) \quad \forall i \in \{1, \cdots, NB\}$

  Filtering
  Select top-B samples $\{\mathbf{x}_b\}_{b=1}^{B}$ with respect to:
  $r_\phi(\mathbf{x}_i) - \lambda \sum_{m=1}^{M} \max(0, g_\phi^{(m)}(\mathbf{x}_i)) \quad \forall i = \{1, \cdots, NB\}$

  Evaluation and Moving Dataset
  Evaluate $y_b = f(\mathbf{x}_b), \quad c_b^m = g^{(m)}(\mathbf{x}_b) \quad \forall m = \{1, \cdots M\} \quad \forall b = \{1, \cdots, B\}$
  Update $\mathcal{D}_{r+1} \leftarrow \mathcal{D}_r \cup \{(\mathbf{x}_b, y_b, \mathbf{c}_b)\}_{b=1}^{B}$
  if $|\mathcal{D}_{r+1}| > L$ **then**
    Remove last $|\mathcal{D}_{r+1}| - L$ samples from $\mathcal{D}_{r+1}$ with respect to: $y - \lambda \sum_{m=1}^{M} \max(0, c^m)$
  end if
end for

D Implementation Details

In this section, we introduce the implementation details of our method **CiBO**. Specifically, model architectures, the training processes employed, the hyperparameter configurations used, and the computational resources required.

D.1 Training Models

D.1.1 Training Proxies

We employ an ensemble of five proxies to model the objective function and a single proxy for each constraint. Each proxy is implemented as a MLP with three hidden layers of 1024 units, using GELU [58] activations. Proxies are trained with the Adam optimizer [59] for 100 epochs per round at a learning rate of 1×10^{-3} and a batch size of 256. All hyperparameters related to the proxy are listed in Table 1.

Table 1: Hyperparameters for Training Proxy

	Parameters	Values
Architecture	Num Ensembles	5
	Number of Layers	3
	Num Units	1024
Training	Batch size	256
	Optimizer	Adam
	Learning Rate	1×10^{-3}
	Training Epochs	100

D.1.2 Training Flow-based Models

We adopt the architecture of [60] for our flow model, comprising three hidden layers with 512 units each. Training is performed using Adam optimizer for 500 epochs per round, with a learning rate of 1×10^{-3} and a batch size of 256. For ODE integration during sampling, we employ the Runge-Kutta 4 method with `torchdiffeq` [61], and set the integration steps as 250. All flow-model hyperparameters are detailed in Table 2.

Table 2: Hyperparameters for Training Flow-based Model

	Parameters	Values
Architecture	Number of Layers	3
	Num Units	512
Training	Batch size	256
	Optimizer	Adam
	Learning Rate	1×10^{-3}
	Training Epochs	500

D.2 Sampling Candidates

D.2.1 Training Diffusion Sampler

Various approaches have been developed to draw samples from a distribution when only an unnormalized probability density or energy function is available. Traditional methods include Markov Chain Monte Carlo (MCMC) techniques [62, 63, 64, 65, 66, 67], though their computational cost increases dramatically in high-dimensional spaces. More recently, amortized variational inference methods, particularly those based on training diffusion samplers [68, 69, 70, 71, 72, 73, 74], have gained widespread adoption as they offer improved scalability for high-dimensional problems.

Following the [21], we adopt [42] to train diffusion sampler to sample from the target:

$$p_{\text{post}}(\mathbf{z}) \propto p(\mathbf{z}) \exp\left(\beta \cdot \left[r_\phi(f_\theta(\mathbf{z})) - \lambda \sum_{m=1}^{M} \max\left(0, g_\phi^{(m)}(f_\theta(\mathbf{z}))\right)\right]\right) \tag{15}$$

Here, the right-hand-side term serves as an unnormalized probability density, which the diffusion sampler amortizes the sampling cost by approximating it.

Off-policy Training of Diffusion Sampler As mentioned in the Section 4.2, we use the Trajectory Balance objective to train the diffusion sampler.

$$\mathcal{L}_{\text{TB}}(\mathbf{z}_{0:1}; \psi) = \left(\log \frac{Z_\psi p(\mathbf{z}_0) \prod_{i=0}^{T-1} p_F(\mathbf{z}_{(i+1)\Delta t} | \mathbf{z}_{i\Delta t}; \psi)}{p(\mathbf{z}_1) r(f_\theta(\mathbf{z}_1)) \prod_{i=1}^{T} p_B(\mathbf{z}_{(i-1)\Delta t} | \mathbf{z}_{i\Delta t})}\right)^2$$

The primary advantage of the TB loss is off-policy training [42, 75]. We can train our model not only from the on-policy trajectories through the reverse SDE $\{\mathbf{z}_0, \cdots, \mathbf{z}_1\} = \tau \sim p_F(\tau)$ but also from the trajectories through the forward SDE conditioned on the generated samples $\tau \sim p_B(\tau|\mathbf{z}_1)$. This proves its effectiveness on mode coverage and credit assignment [42].

Specifically, we repeat two processes. First, we sample trajectories on-policy $\tau \sim p_F(\tau)$, train the model with Equation (14), and collect the samples $\mathbf{z}_1$ into the buffer. Second, from the collected samples $\mathbf{z}_1$, we generate off-policy trajectories through $\tau \sim p_B(\tau|\mathbf{z}_1)$, then train with the Equation (14). During the off-policy training, we prioritize the samples with low energy: $\mathcal{E}(\mathbf{z}_1) = -\log(p(\mathbf{z}_1) r(f_\theta(\mathbf{z}_1)))$ following [42] to make our model focus on the low energy samples. These techniques improve the overall performance of our framework (Appendix E.4).

We use the original code[8] released from [42] for implementation. We also set method-specific hyperparameters with Path Integral Sampler (PIS) [68] architecture, zero initialization, and t-scale to 1 to make sure the initialized $p_F(\mathbf{z}_1)$ starts from the standard normal distribution. Detailed hyperparameters for training the diffusion sampler can be found in Table 3.

Table 3: Hyperparameters for Training Diffusion Sampler

	Parameters	Values
Architecture	Number of Layers	2
	Num Units	256
	Diffusion Time Steps	50
Training	Batch size	256
	Optimizer	Adam
	Learning Rate	1×10^{-3}
	Training Epochs	50

Computational Resources. Our experiments were conducted using NVIDIA RTX 3090 and A6000 GPUs. These resources were sufficient to train our models within a reasonable time for all reported experiments. Details of computational time can be found at Appendix F.

[8] `https://github.com/GFNOrg/gfn-diffusion`

D.3 Hyperparameters

In our formulation of constrained black-box problems, we introduce λ for Lagrangian augmentation. We draw $N \times B$ samples from the posterior distribution, then select B samples during filtering. After evaluation, we update the training set by keeping the top L highest-scoring samples subject to the Lagrangian objective. Table 4 summarizes all hyperparameter values used in candidate selection, and we include additional analysis to assess how each parameter affects overall performance in Section 5.4 and Appendix E.

Table 4: Hyperparameters during sampling candidates

	Lambda λ	Inverse Temperature β	Buffer Size L	Filtering Coefficient N
Ackley 200D	10	10^5	3000	10
Rastrigin 200D	10	10^5	2000	10
Rosenbrock 200D	10	10^5	2000	10
RoverPlanning 60D	3	10^5	1000	10
Mopta 124D	3	10^3	500	10
DNA 180D	5	10^3	1000	15

E Further Analysis

In this section, we provide further analysis on different components of our method that are not included in the main manuscript due to the page limit.

E.1 Analysis on Feasibility Ratio

To further analyze our method's ability to effectively handle constraints, we report the feasibility ratio across optimization batches for the Rastrigin 200D task. Here, the feasibility ratio denotes the number of feasible samples over queried samples.

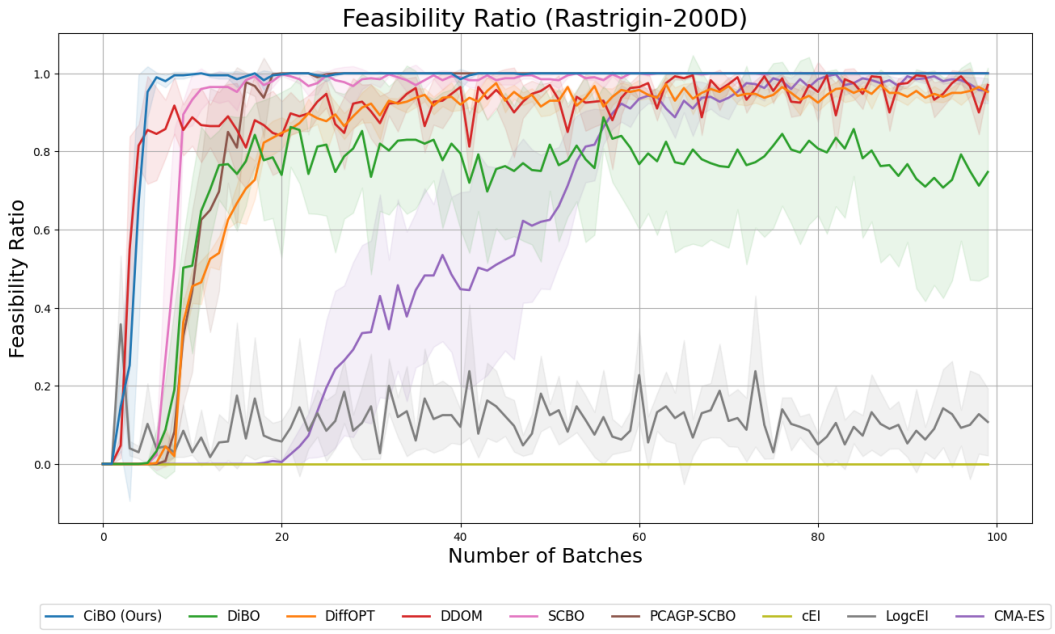


Figure 7: Feasibility ratio over all baselines. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

As shown in Figure 7, CiBO demonstrates superior performance by rapidly achieving the highest feasibility ratio within the first 5-10 batches, significantly faster than all competing methods. While some baselines (SCBO, PCAGP-SCBO) eventually reach high feasibility ratios, they require approximately twice as many batches to achieve comparable performance. Other methods like DiBO and DiffOPT take even longer (around 20 batches), and CMA-ES struggles substantially, only reaching moderate feasibility ratios after 50 batches. Notably, CiBO not only reaches the high feasibility ratio faster but also maintains it consistently throughout the optimization process, demonstrating its robust constraint-handling capabilities in high-dimensional spaces.

E.2 Analysis on Filtering coefficient N

To improve the sample efficiency of our method, we introduce filtering, where we sample $N \times B$ candidates from the posterior distribution, then select the highest B samples with respect to the Lagrangian-relaxed objective function. To analyze the impact of the filtering coefficient N, we experiment with varying N values, including our default $N = 100$.

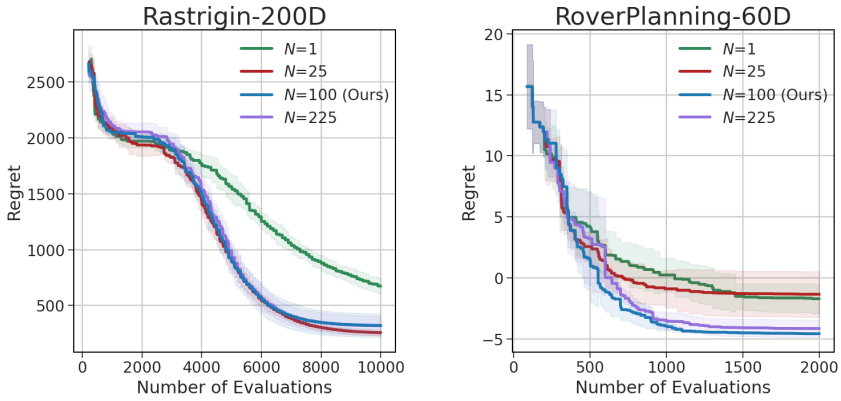


Figure 8: Performance of CiBO in Rastrigin-200D and Rover Planning-60D with varying N. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

As shown in Figure 8, increasing the filtering coefficient improves sample efficiency by concentrating candidate selection in both high objective values and feasible regions. If the coefficient is set too low, we lose its exploitation capability, leading to slower convergence.

E.3 Analysis on Buffer Size L

In each round, we retain the L top-scoring samples with respect to the Lagrangian-relaxed objective function for computational efficiency. To analyze the effect of the buffer size L, we conduct experiments by varying L. As demonstrated in Figure 9, using too small L occasionally gets stuck in a sub-optimal solution while using too large L exhibits a slow convergence rate.

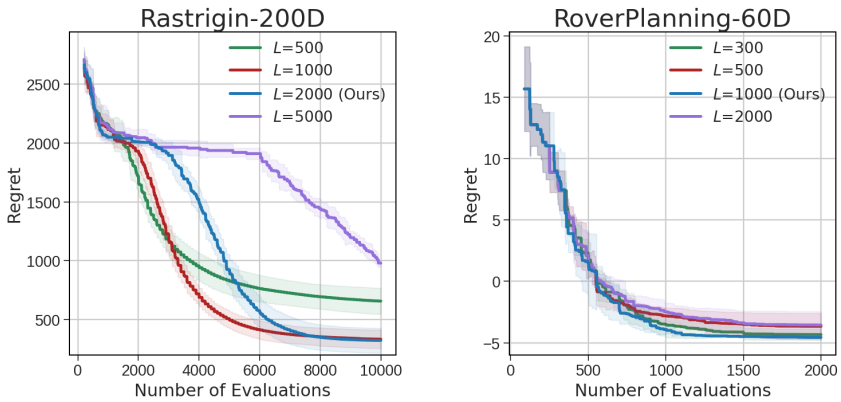


Figure 9: Performance of CiBO in Rastrigin-200D and Rover Planning-60D with varying L. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

E.4 Effect of Off-policy Training in Amortized Inference

We employ off-policy training with the TB loss to train the diffusion sampler as detailed in Section 4.2. To analyze the impact of off-policy training on performance, we conduct ablation studies on different training schemes. As shown in Figure 10, off-policy training consistently outperforms on-policy methods, and the performance gap widens as the number of constraints grows (15 constraints in Rover Planning versus only 2 in Rastrigin). It underlines that training with off-policy samples is crucial for amortizing the posterior distribution with multiple modes and a large plateau.

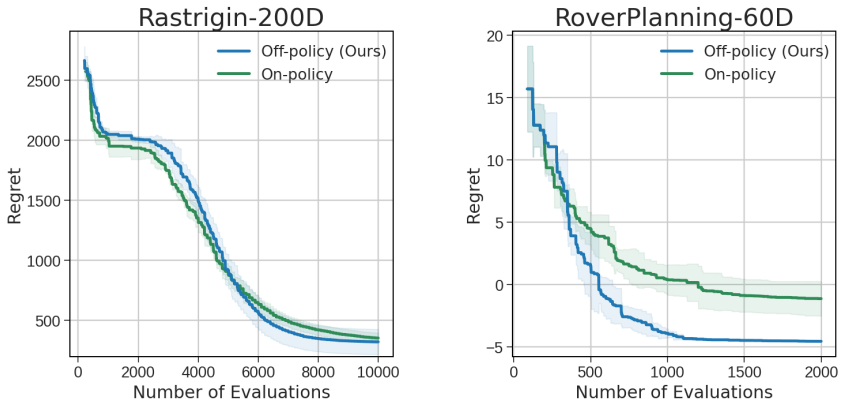


Figure 10: Comparison between off-policy and on-policy in Rastrigin-200D and Rover Planning-60D. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

E.5 Analysis on Initial Dataset size $|D_0|$ and Batch size B

The size of the initial dataset, $|D_0|$, and batch size B play a critical role in the performance of black-box optimization algorithms. When $|D_0|$ is small and B is large, the algorithm must optimize using very limited information, making the search significantly more challenging. To this end, we conduct experiments varying $|D_0|$ and B to demonstrate the robustness of our method on initial data configurations. As shown in Figure 11, our method demonstrates robustness regarding both the initial dataset size $|\mathcal{D}_0|$ and the batch size B.

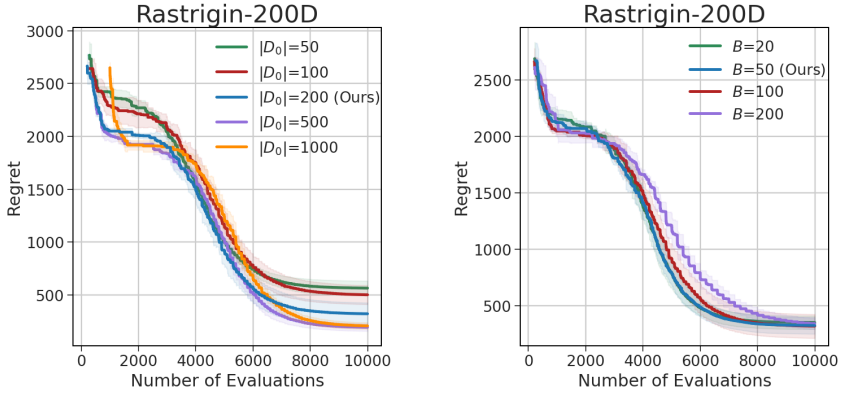


Figure 11: Performance of CiBO in Rastrigin-200D with varying $|D_0|$ and B. Experiments are conducted with four random seeds, and the mean and one standard deviation are reported.

F Runtimes

We report the running time of each method in Table 5. To measure the runtime, we conduct experiments on a single NVIDIA RTX 3090 GPU and Intel Xeon Platinum CPU @ 2.90 GHz. As shown in the table, the running time of our method is similar to other generative model-based approaches, and mostly faster than BO-based methods.

Table 5: Average time (in seconds) for each round in each method.

	Rastrigin-200D	Ackley-200D	Rosenbrock-200D	RoverPlanning-60D	Mopta-124D	DNA-180D
cEI	336.96 ± 47.48	133.12 ± 6.66	489.86 ± 81.38	111.13 ± 5.83	205.66 ± 5.35	133.98 ± 9.16
LogcEI	720.45 ± 56.76	158.38 ± 10.13	593.39 ± 97.93	113.28 ± 3.97	324.27 ± 9.91	161.08 ± 8.21
SCBO	322.81 ± 47.38	117.81 ± 6.50	475.02 ± 80.11	87.83 ± 4.42	270.30 ± 5.19	147.83 ± 12.57
PCAGP-SCBO	327.48 ± 51.20	122.67 ± 3.50	479.06 ± 82.41	17.55 ± 3.05	20.69 ± 0.39	81.34 ± 9.19
CMA-ES	0.08 ± 0.00	0.09 ± 0.00	0.10 ± 0.01	0.61 ± 0.00	5.33 ± 0.15	46.58 ± 3.16
DDOM	26.87 ± 0.28	27.00 ± 0.32	26.96 ± 0.12	3.56 ± 0.02	8.63 ± 0.37	50.99 ± 0.81
DiffOPT	91.00 ± 5.07	111.04 ± 1.51	89.37 ± 8.27	29.48 ± 1.21	105.61 ± 2.71	60.64 ± 1.34
DiBO	73.97 ± 0.56	68.89 ± 0.98	73.83 ± 1.11	29.43 ± 1.51	66.61 ± 4.16	71.85 ± 2.48
CiBO	73.39 ± 2.09	103.43 ± 4.84	82.24 ± 6.50	53.58 ± 5.14	105.43 ± 2.24	81.77 ± 2.37

G Limitations and Future Work

We are interested in improving our method further. First, as we need to train all models with the updated dataset in every round, presenting a framework that can efficiently reuse the trained models from the previous rounds would be beneficial. Furthermore, there are several advancements in the literature on flow-based model training [76] and diffusion samplers [77, 78], which could potentially yield substantial performance gains. We leave them as future work.

H Broader Impact

Advances in real-world design optimization have the potential to drive major innovations, but they also come with potential risks and unintended consequences. For example, optimization techniques in biochemical design may uncover novel compounds with therapeutic potential, but similar methods could also be misused to discover harmful substances. It is essential for researchers to act responsibly and ensure their work serves the public good.