Diffusion Models as Constrained Samplers for Optimization with Unknown Constraints

Lingkai Kong*[1], **Yuanqi Du***[2], **Wenhao Mu***[1], **Kirill Neklyudov**[3,4]
Valentin De Bortoli[5], **Dongxia Wu**[6], **Haorui Wang**[1], **Aaron Ferber**[2]
Yi-An Ma[6], **Carla P. Gomes**[2], **Chao Zhang**[1]

[1]Georgia Tech [2]Cornell University [3]Université de Montréal [4]Mila
[5]ENS, CNRS, PSL University, [6] UCSD

Abstract

Optimization with unknown constraints is a challenging, yet unsolved problem. Overlooking these constraints can lead to spurious solutions that are unrealistic in practice. To deal with such unknown constraints, we propose to perform optimization within the data manifold using diffusion models. To constrain the optimization process to the data manifold, we reformulate the original optimization problem as a sampling problem from the product of the Boltzmann distribution defined by the objective function and the data distribution learned by the diffusion model. Depending on the differentiability of the objective function, we propose two different sampling methods. For differentiable objectives, we propose a two-stage framework that begins with a guided diffusion process for warm-up, followed by a Langevin dynamics stage for further correction. For non-differentiable objectives, we propose an iterative importance sampling strategy using the diffusion model as the proposal distribution. Comprehensive experiments on a synthetic dataset, six real-world black-box optimization datasets, and a multi-objective molecule optimization dataset show that our method achieves better or comparable performance with previous state-of-the-art baselines.

*Equal contribution, authors agreed ordering can be changed for their respective interests.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

1 Introduction

Optimization problems are ubiquitous in real-world applications when approaching search problems (Bazaraa et al., 2011), partial differential equations (Isakov, 2006), molecular design (Sanchez-Lengeling and Aspuru-Guzik, 2018). While significant advancements have been made in resolving a broad spectrum of abstract optimization problems with analytically known objective functions and constraints (Boyd and Vandenberghe, 2004; Rao, 2019; Petropoulos et al., 2023), optimization in real-world scenarios remains challenging since the exact nature of the objective is often unknown, and access to constraints is limited (Conn et al., 2009). For example, it is challenging to incorporate the closed-form constraints on a molecule to be synthesizable or design an objective function for target chemical properties.

Previous studies have identified problems with unknown objective functions as black-box optimization problems (Conn et al., 2009; Alarie et al., 2021). In such scenarios, the only way to obtain the objective value is through running a simulation (Larson et al., 2019) or conducting a real-world experiment (Shields et al., 2021), which might be expensive and non-differentiable. A prevalent approach to this challenge involves learning a surrogate model with available data to approximate the objective function which can be implemented in either an online (Snoek et al., 2012; Shahriari et al., 2015; Srinivas et al., 2010) or offline manner (Trabucco et al., 2021a; 2022).

However, there is a significant lack of research focused on scenarios where analytic constraints are absent. The only works that deal with unknown constraints are from the derivative-free optimization community (Audet and Dennis Jr, 2004; 2006; Nguyen and Balasubramanian, 2023). However, these methods can only be applied to simple low-dimensional problems and cannot be applied to more complex problems such as molecule and protein

optimization. In practice, overlooking these feasibility constraints during optimization can result in spurious solutions. For instance, the optimization process might yield a molecule with the desired chemical property but cannot be physically synthesized (Gao and Coley, 2020; Du et al., 2022), which would require restarting the optimization from different initializations (Krenn et al., 2020; Jain et al., 2023).

To restrict the search space to the set of feasible solutions, we propose to perform optimization within the support of the data distribution or the data manifold. Indeed, in practice, one usually has an extensive set of samples satisfying the necessary constraints even when the constraints are not given explicitly. For instance, the set of synthesizable molecules can be described by the distribution of natural products (Baran, 2018; Voršilák et al., 2020). To learn the data distribution, we focus on using diffusion models, which recently demonstrated the state-of-the-art performance in image modeling (Ho et al., 2020; Song et al., 2020), video generation (Ho et al., 2022), and 3D synthesis (Poole et al., 2022). Moreover, (Pidstrigach, 2022; De Bortoli, 2022) theoretically demonstrated that diffusion models can learn the data distributed on a lower dimensional manifold embedded in the representation space, which is often the case of the feasibility constraints.

To constrain the optimization process to the data manifold, we reformulate the original optimization problem as a sampling problem from the product of two densities: i) a Boltzmann density with energy defined by the objective function and ii) the density of the data distribution. The former concentrates around the global minimizers in the limit of zero temperature (Hwang, 1980; Gelfand and Mitter, 1991), while the latter removes the non-feasible solutions by yielding the zero target density outside the data manifold. Depending on whether the objective function is differentiable or non-differentiable, we propose two different sampling methods. When the objective function is differentiable, we propose a two-stage sampling strategy: (i) a guided diffusion process acts as a warm-up stage to provide initialization of data samples on the manifold, and (ii) we ensure convergence to the target distribution via Markov Chain Monte Carlo (MCMC). When the objective is non-differentiable, we propose an iterative importance sampling strategy using diffusion models to gradually improve the proposal distribution.

The main contributions of this work are: (1) We reformulate the problem of optimization under unknown constraints as a sampling problem from the product of the data distribution and the Boltzmann distribution defined by the objective function. (2) We propose two different sampling methods with diffusion models, depending on the differentiability of the objective function. (3) Empirically, we validate the effectiveness of our proposed framework on a synthetic toy example, six real-world offline black-box optimization tasks as well as a multi-objective molecule optimization task. We find that our method, named DiffOPT, can outperform state-of-the-art methods.

2 Background

Problem definition. Consider an optimization problem with *objective function* $h : \mathbb{R}^d \to \mathbb{R}$. Additionally, we consider a *feasible set* C, which is a subset of $\mathbb{R}^d$.

Our goal is to find the set of minimizers $\{x_i^\star\}_{i=1}^M$ of the objective h within the feasible set C. This can be expressed as the following constrained optimization problem

$$\{x_i^\star\}_{i=1}^M = \arg\min_{x \in C} h(x).$$

However, in our specific scenario, the explicit formulation of the feasible set C is *unavailable*. Instead, we can access a set of points $\mathcal{D} = \{x_i\}_{i=1}^N$ sampled independently from the feasible set C.

Optimization via sampling. If we do not consider the constraints, under mild assumption, the optimization process is equivalent to sample from a Boltzmann distribution $q_\beta(x) \propto \exp[-\beta h(x)]$, in the limit where $\beta \to \infty$, where β is an *inverse temperature* parameter. This is the result of the following proposition which can be found in (Hwang, 1980, Theorem 2.1).

Proposition 1. *Assume that $h \in \mathrm{C}^3(\mathbb{R}^d, \mathbb{R})$. Assume that $\{x_i^\star\}_{i=1}^M$ is the set of minimizers of h. Let p be a density on $\mathbb{R}^d$ such that there exists $i_0 \in \{1, \dots, M\}$ with $p(x_{i_0}^\star) > 0$. Then Q_β the distribution with density w.r.t the Lebesgue measure $\propto q_\beta(x)p(x)$ weakly converges to Q_∞ as $\beta \to \infty$ and we have that*

$$Q_\infty = \textstyle\sum_{i=1}^M a_i \delta_{x_i^\star} / \sum_{i=1}^M a_i,$$

with $a_i = p(x_i^\star) \det(\nabla^2 h(x_i^\star))^{-1/2}$.

Based on this proposition, (Gelfand and Mitter, 1991) proposed a *tempered* method for global optimization. However, the proposed temperature schedule scales logarithmically with the number of steps; hence, the total number of iterations scales exponentially, hindering this method's straightforward application. In practice, we can sample from the density with the target high β, see (Raginsky et al., 2017; Ma et al., 2019; De Bortoli and Desolneux, 2021).

Product of Experts. To constrain our optimization procedure to the feasible set C, we propose to model the target density as a product of experts (Hinton, 2002), a modeling approach representing the "unanimous vote" of independent models. Given the density models of $m \in \mathbb{N}$ "experts" $\{q_i(x)\}_{i=1}^m$, the target density of their

product is defined as

$$\pi(x) \propto \prod_{i=1}^{m} q_i(x)\,.$$

Hence, if one of the experts yields zero density at x, the total density of the product at x is zero.

Diffusion Models. Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N \sim p_{\text{data}}(x)^{\otimes N}$ with p_{data} concentrated on the feasible set C, we will learn a generative model p such that $p \approx p_{\text{data}}$ using a diffusion model (Sohl-Dickstein et al., 2015; Song and Ermon, 2019; Ho et al., 2020). The density p will then be used in our product of experts model to enforce the feasibility constraints, see Section 3.1. In diffusion models, we first simulate a *forward* noising process starting from data distribution $p_0(x) = p_{\text{data}}(x)$ which converges to the standard Gaussian distribution $p_T(x) \approx \mathcal{N}(0, \text{Id})$ as $T \to \infty$. The forward process is defined by the following SDE

$$\mathrm{d}\vec{\mathbf{x}}_t = f(\vec{\mathbf{x}}_t, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t,\ \vec{\mathbf{x}}_0 \sim p_{\text{data}}(x),\ 0 \le t \le T\,,$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued drift function, $g(t) : \mathbb{R} \to \mathbb{R}$ is a scalar-valued diffusion coefficient, and $(\mathbf{w}_t)_{t\ge 0}$ is a d-dimensional Brownian motion. Then, under mild assumptions, the reverse process that generates data from normal noise follows the backward SDE (Haussmann and Pardoux, 1986; Anderson, 1982)

$$\mathrm{d}\overleftarrow{\mathbf{x}}_t = \left[-f(\overleftarrow{\mathbf{x}}_t, \tau) + g^2(\tau)\nabla_x \log p_\tau(\overleftarrow{\mathbf{x}}_t)\right]\mathrm{d}t + g(\tau)\mathrm{d}\mathbf{w}_t,$$

where $\tau = T - t$ and $\nabla_x \log p_\tau(x)$ is the score function which is modeled by a time-dependent neural network via the score matching objective

$$\mathbb{E}_t\left[\lambda(t)\mathbb{E}_{\vec{\mathbf{x}}_0}\mathbb{E}_{\vec{\mathbf{x}}_t|\vec{\mathbf{x}}_0}\left[\|s_\theta(\vec{\mathbf{x}}_t, t) - \nabla_x \log p_{t|0}(\vec{\mathbf{x}}_t|\vec{\mathbf{x}}_0)\|_2^2\right]\right],$$

where $p_{t|0}(\vec{\mathbf{x}}_t|\vec{\mathbf{x}}_0)$ is the conditional density of the forward SDE starting at $\vec{\mathbf{x}}_0$, and $\lambda(t) > 0$ is a weighting function. Under assumptions on f and g, there exists α, σ such that for any $t \in [0, T]$, $\vec{\mathbf{x}}_t = \alpha_t \vec{\mathbf{x}}_0 + \sigma_t \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \text{Id})$, and therefore one does not need to integrate the forward SDE to sample $(\vec{\mathbf{x}}_0, \vec{\mathbf{x}}_t)$. Recent works have shown that diffusion models can detect the underlying data manifold supporting the data distribution (Pidstrigach, 2022; De Bortoli, 2022). This justifies the use of the output distribution of a diffusion model as a way to identify the feasible set.

3 Proposed Method

In this section, we present our method, DiffOPT. First, we formulate the optimization process as a sampling problem from the product of the data distribution concentrated on the manifold and the Boltzmann distribution defined by the objective function. Then we propose two sampling methods with diffusion models for different types of objective functions.

3.1 Constrained Optimization as Sampling from Product of Experts

We recall that $q_\beta(x) \propto \exp[-\beta h(x)]$, where h is the objective function. While it is possible to sample directly from $q_\beta(x)$, the generated samples may fall outside of the feasible set C, defined by the dataset $\mathcal{D}$. To address this, we opt to sample from the product of the data distribution and the Bolzmann distribution induced by the objective function. Explicitly, our goal is to sample from a distribution $\pi_\beta(x)$ defined as

$$\pi_\beta(x) \propto p(x) q_\beta(x). \tag{1}$$

Using Proposition 1, we have that π_β concentrates on the *feasible* minimizers of h as $\beta \to +\infty$.

It should be noted that Equation (1) satisfies the following properties: (a) it assigns high likelihoods to points x that simultaneously exhibit sufficiently high likelihoods under both the base distributions $p(x)$ and $q_\beta(x)$; (b) it assigns low likelihoods to points x that display close-to-zero likelihood under either one or both of these base distributions. This ensures that the generated samples not only remain within the confines of the data manifold but also achieve low objective values.

In practice, the objective function h can be either differentiable or non-differentiable. In the following sections, we will propose two sampling methods using diffusion models, for each type respectively.

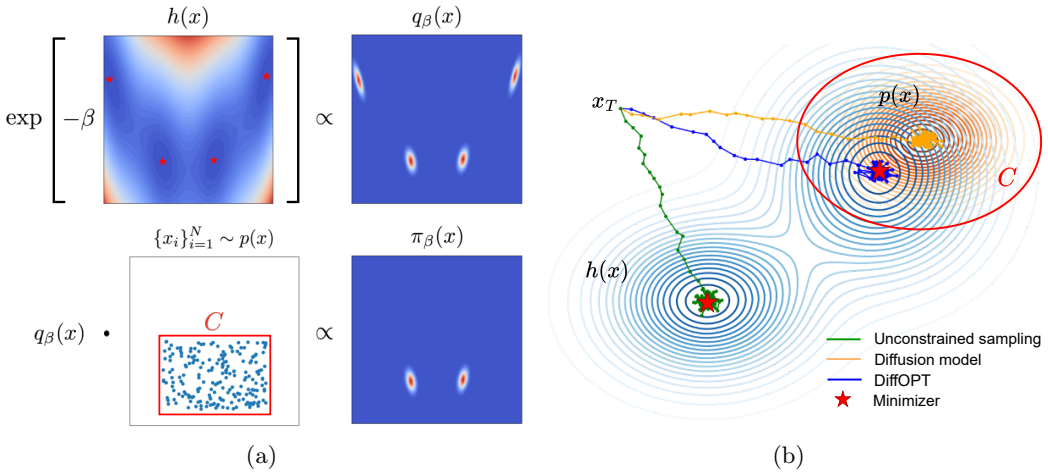
3.2 Differentiable Objective: Two-stage Sampling with Optimization-guided Diffusion

Under mild assumptions, the following SDE converges to π_β *w.r.t.* the total variation distance (Roberts and Tweedie, 1996)

$$\mathrm{d}\mathbf{x}_t = \nabla_{\mathbf{x}} \log \pi_\beta(\mathbf{x}_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{w}_t. \tag{2}$$

where the gradient of the unnormalized log-density can be conveniently expressed as the sum of the scores, *i.e.*, $\nabla_x \log \pi_\beta(x) = \nabla_x \log p(x) + \nabla_x \log q_\beta(x)$. Furthermore, one can introduce a Metropolis-Hastings (MH) correction step to guarantee convergence to the target distribution when using discretized version of Equation (2) (Durmus and Moulines, 2022), which is known as the Metropolis-Adjusted-Langevin-Algorithm (MALA) (Grenander and Miller, 1994). We provide more details of both algorithms in Appendix C.

Theoretically, sampling from Equation (1) can be done via MALA. However, in practice, the efficiency significantly depends on the choice of the initial distribution and the step size schedule. The latter is heavily linked with the Lipschitz constant of $\log q_\beta$ which controls the stability of the algorithm. Large values of β, necessary



(a) (b)

Figure 1: Constrained optimization as a sampling from the product of densities. That is, we minimize the objective function $h(x)$ (red stars denote the minimizers) within the feasible set C, which is given by samples $\{x_i\}_{i=1}^N \sim p(x)$. This problem is equivalent to sampling from the density $\pi_\beta(x) \propto p(x)\exp[-\beta h(x)]$, which concentrates around minimizers of $h(x)$ within the feasible set C. The distribution we sample from is shown on the left and the trajectory we take to sample is shown on the right.

to get accurate minimizers, also yield large Lipschitz constants which in turn impose small stepsizes. Moreover, the gradient of the log-density can be undefined outside the feasibility set C.

To circumvent these practical issues, we propose sampling in two stages: a warm-up stage and a sampling stage. The former aims to provide a good initialization for the sampling stage. The sampling stage follows the Langevin dynamics for further correction. The pseudocode for both stages is provided in Appendix D.

Stage I: Warm-up with guided diffusion. In imaging inverse problems, it is customary to consider *guided* diffusion models to enforce some external condition, see (Chung et al., 2022a;b; Song et al., 2022; Wu et al., 2023). In our setting, we adopt a similar strategy where the guidance term is given by βh, *i.e.*, we consider

$$\begin{aligned} \mathrm{d}\mathbf{x}_t^\epsilon =& [-f(\mathbf{x}_t^\epsilon, \tau) + g^2(\tau) s_\theta(\mathbf{x}_t^\epsilon, \tau) - \beta \nabla h(\mathbf{x}_t^\epsilon)]\mathrm{d}t \\ &+ g(\tau)\mathrm{d}\mathbf{w}_t, \quad \tau = T - t. \end{aligned} \tag{3}$$

Theorem 1. *Under assumptions on p, h, we denote p_{T-t}^β the distribution of Equation (3) at time t and there exists $C > 0$ such that for any $x \in \mathbb{R}^d$*

$$(1/C)\tilde{p}_0^\beta(x) \le p_0^\beta(x) \le C\tilde{p}_0^\beta(x),$$

where p_0^β is the output of the warm-up guided diffusion process and $\tilde{p}_0^\beta(x) = p_0(x)\exp[\log(\beta_0)W_0^\beta(x)]$, with $W_0^\beta(x) = \Delta h(x) + \langle \nabla \log p_0^\beta(x), \nabla h(x)\rangle$ and β_0 is the inverse temperature at the end of the process

The proof is postponed to Appendix H. As an immediate consequence of Theorem 1, we have that $\lim_{\beta_0\to\infty} p_0^\beta(x_\star) = +\infty$ for every local strict minimizer $x^\star$ of h *within* the support of $p_0(x)$, and $\lim_{\beta_0\to\infty} p_0^\beta(x_\star) = 0$ for $x^\star$ *outside* the support of $p_0(x)$, see Appendix H. That is, p_0^β concentrates on the *feasible* local minimizers of h as $\beta_0 \to +\infty$.

Theorem 1 indicates that the guided diffusion process in the first stage yields a more effectively initialized distribution within the data manifold, bounded both above and below by a product of experts related to the original constrained optimization problem. However, it is known that guided diffusion cannot accurately sample from the product of experts q_β (Du et al., 2023b; Garipov et al., 2023), see more details in Appendix E. While additional contrastive training of a surrogate objective has been proposed (Lu et al., 2023), in this work, we do not consider such complex corrections. Instead, we rely on ideas from MCMC literature to ensure convergence.

Stage II: Further correction with Langevin dynamics. In the second stage, we further use Langevin dynamics for accurate sampling from $\pi_\beta(x)$. The gradient of the log-density of the data distribution $\nabla \log p(x)$ can be obtained by setting the time of the score function to 0, *i.e.*, $s_\theta(x, 0)$. The unadjusted Langevin algorithm is then given by

$$x^{k+1} = x^k + (s_\theta(x^k, 0) - \beta\nabla_x h(x^k))\Delta t + \sqrt{2\Delta t}z,$$

where Δt is the step size and z comes from a Gaussian distribution. In practice, we find that a constant β is enough for this stage. When using the score-based parameterization $s_\theta(x, t)$, we cannot access the unnormalized log-density of the distribution. Therefore, we cannot use the MH correction step. Although the sampler is not exact without MH correction, it performs well in practice.

To further incorporate the MH correction step, we can adopt an energy-based parameterization following (Du et al., 2023b): $E_\theta(x,t) = -\frac{1}{2}\|\mathrm{NN}_\theta(x,t)\|^2$, where $E_\theta(x,t)$ is the energy function of the data distribution, and $\mathrm{NN}_\theta(x,t)$ is a vector-output neural network. An additional benefit of MH correction is that it can enforce hard constraints. The vanilla Langevin dynamics includes a Brownian motion term, whose support spans the entire space (similar to a Gaussian distribution). As a result, samples may occasionally fall outside the constrained region. However, with a MH correction step, the acceptance ratio $p(x^k+1)/p(x^k)$ becomes zero if the proposed sample x^{k+1} lies outside the support of the distribution, ensuring that such samples are disregarded. We provide more details of this energy-based parameterization in Appendix G.

3.3 Non-differentiable Objective: Iterative Importance Sampling with Diffusion Models

When gradients of the objective function are unavailable, we can use self-normalized importance sampling (SNIS) (Rubinstein and Kroese, 2016), which is easy and fast to implement. SNIS first proposes several particles from a proposal distribution. Then, resample the particles according to their weights, calculated as the ratio of the target density to the proposal density.

However, the performance of SNIS is heavily determined by the proposal distribution. A good proposal distribution should be close to the target distribution. To address this, we employ an iterative importance sampling with diffusion models to improve the initial proposal distribution. The pseudocode of our derivative-free sampling algorithm is provided in Appendix F.

Initialization with SNIS. We begin by randomly sampling S particles $\{x_s^0\}_{s=1}^S$ from the diffusion model $p_\theta(x)$, assigning each particle a weight $w_s^0 = \frac{\pi_\beta(x_s^0)}{p_\theta(x_s^0)} = q_\beta(x_s^0)$. We then resample S particles based on the normalized weights $\tilde{w}_s^0 = \frac{w_s^0}{\sum_{j=1}^S w_j^0}$ using multinomial sampling with replacement.

Diffusion-guided proposal for iterative importance sampling. At the k-th iteration, we have S particles $\{x_s^{k-1}\}_{s=1}^S$ resampled from the previous iteration. The key insight of our method is that these resampled particles form a proposal distribution closer to the target distribution. However, these resampled particles tend to be repetitive and lack diversity. To address this, we use a diffusion model to diversify the resampled particles, ensuring that the proposal distribution remains diverse and closely approximates the target distribution. Henceforth, we will use x_s^k and $x_{s,0}^k$ interchangeably to refer to the particle at time 0. We also omit the arrows denoting forward/reverse process of diffusion models for simplicity.

The diversify process works as follows: we first add noise to the particles $x_{s,0}^{k-1}$ through a forward diffusion process until a randomly sampled time t to get $x_{s,t}^{k-1} \sim p_{t|0}(x_{s,t}^{k-1}|x_{s,0}^{k-1})$. Then we denoise them back to obtain new particles $x_{s,0}^k \sim p_{0|t}(x_{s,0}^k|x_{s,t}^{k-1})$. This proposal can be written as $Q(x_s^k|x_s^{k-1}) = \int p_{t|0}(x_{s,t}^{k-1}|x_{s,0}^{k-1}) p_{0|t}(x_{s,0}^k|x_{s,t}^{k-1})\, dx_{s,t}^{k-1}$. We use the Monte Carlo method to compute the marginalization by drawing J samples $x_{s,t,j}^{k-1}$ for each particle: $Q(x_s^k|x_s^{k-1}) \approx \frac{1}{J}\sum_{j=1}^J p_{0|t}(x_{s,0}^k|x_{s,t,j}^{k-1})$. Sampling $x_{s,t,j}^{k-1}$ is straightforward as the forward process in diffusion models admits a closed-form conditional Gaussian distribution: $p_{t|0}(x_{s,t}^{k-1}|x_{s,0}^{k-1}) = \mathcal{N}(x_{s,t}^{k-1}|\alpha_t x_{s,0}^{k-1}, \sigma_t^2 \mathrm{Id})$. Following (Song et al., 2023), we approximate $p(x_{s,0}^k|x_{s,t}^{k-1})$ as $\mathcal{N}(\frac{x_{s,t,j}^{k-1}+\sigma_t s_\theta(x_{s,t,j}^{k-1},t)}{\alpha_t}, \frac{\sigma_t^2}{1+\sigma_t}\mathrm{Id})$. The marginalization $Q(x_s^k)$ is computed by enumerating all the particles from last iteration, *i.e.*, $\frac{1}{S}\sum_{s'=1}^S Q(x_s^k|x_{s'}^{k-1})$. Each particle is assigned a weight $w_s^k = \frac{\pi_\beta(x_s^k)}{Q(x_s^k)}$. We then resample S particles based on the normalized weights. This process is iterated for K steps.

Note that by involving only resampling and diffusion models in each iteration, we can ensure staying within the data manifold, thereby satisfying hard constraints.

4 Related Work

Diffusion models and data manifold. Diffusion models have demonstrated impressive performance in various generative modeling tasks, such as image generation (Song et al., 2020; Ho et al., 2020), video generation (Ho et al., 2022), and protein synthesis (Watson et al., 2023; Gruver et al., 2023). Several studies reveal diffusion models implicitly learn the data manifold (Pidstrigach, 2022; De Bortoli et al., 2021; Du et al., 2023a; Wenliang and Moran, 2023). This feature of diffusion models has been used to estimate the intrinsic dimension of the data manifold in (Stanczuk et al., 2022). Moreover, the concentration of the samples on a manifold can be observed through the singularity of the score function. This phenomenon is well-understood from a theoretical point of view and has been acknowledged in (De Bortoli, 2022; Chen et al., 2022).

Optimization as sampling problems. Numerous studies have investigated the relationship between optimization and sampling (Ma et al., 2019; Stephan et al., 2017; Trillos et al., 2023; Wibisono, 2018; Cheng, 2020; Cheng et al., 2020). Sampling-based methods have been successfully applied in various applications of stochastic optimization when the solution space is too large to search exhaustively (Laporte et al., 1992) or when the objective function exhibits noise (Branke

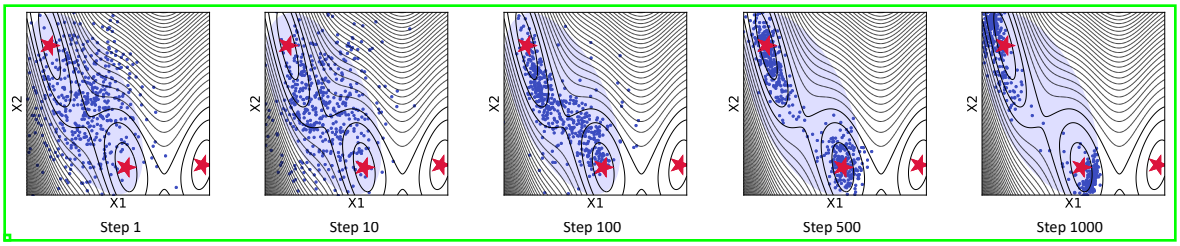


Figure 2: Sampling trajectory of DiffOPT in the synthetic Branin experiment with unknown constraints. Red stars denote the minimizes, and the blue region denotes the feasible space from which training data is sampled. DiffOPT can effectively navigate towards the two feasible minimizers.

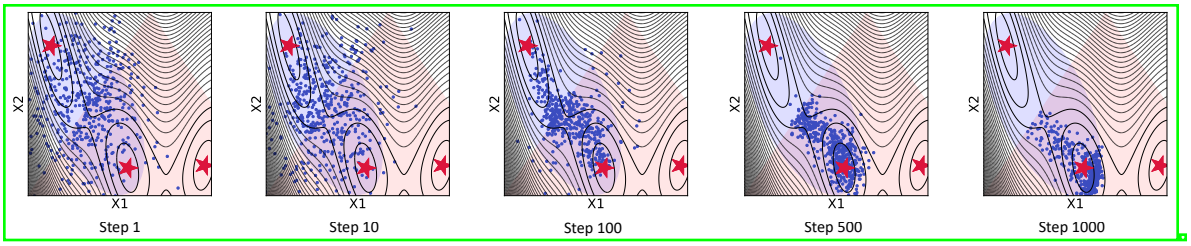


Figure 3: Sampling trajectory of DiffOPT in the synthetic Branin experiment with additional known constraints. Red stars denote the minimizers, the blue region denotes the feasible space from which training data is sampled and the pink region denotes the feasible space defined by the added given constraints. DiffOPT can effectively navigate towards the unique minimizer at the intersection of the two feasible spaces.

and Schmidt, 2004) or countless local optima (Burke et al., 2005; 2020). A prominent solution to global optimization is through sampling with Langevin dynamics (Gelfand and Mitter, 1991), which simulates the evolution of particles driven by a potential energy function. Furthermore, simulated annealing (Kirkpatrick et al., 1983) employs local thermal fluctuations enforced by Metropolis-Hastings updates to escape local minima (Metropolis et al., 1953; Hastings, 1970). More recently, (Zhang et al., 2023) employs generative flow networks to amortize the cost of the sampling process for combinatorial optimization with both closed-form objectives and constraints.

Learning for optimization. Recently, there has been a growing trend of adopting machine learning methods for optimization tasks. The first branch of work is model-based optimization, which focuses on learning a surrogate model for the black-box objective function. This model can be developed in either an online (Snoek et al., 2012; Shahriari et al., 2015; Srinivas et al., 2010; Zhang et al., 2021), or an offline manner (Yu et al., 2021; Trabucco et al., 2021b; Fu and Levine, 2020; Chen et al., 2023; Yuan et al., 2023). Additionally, some research (Kumar and Levine, 2020; Krishnamoorthy et al., 2023; Kim et al., 2023) has explored the learning of stochastic inverse mappings from function values to the input domain, utilizing generative models such as generative adversarial nets (Goodfellow et al., 2014) and diffusion models (Song et al., 2020; Ho et al., 2020).

The second branch, known as "Learning to Optimize", involves training a neural network to address fully specified optimization problems. In these works, a model is trained using a distribution of homogeneous instances, to achieve generalization on similar unseen instances. Various learning paradigms have been used in this context, including supervised learning (Li et al., 2018; Gasse et al., 2019), reinforcement learning (Li and Malik, 2016; Khalil et al., 2017; Kool et al., 2018), unsupervised learning (Karalias and Loukas, 2020; Wang et al., 2022; Min et al., 2023), and generative modeling (Sun and Yang, 2023; Li et al., 2023). In contrast to our approach, these works typically involve explicitly defined objectives and constraints.

Optimization under unknown constraints. Existing work on optimization under unknown constraints are based on derivative-free optimization methods, such as generalized pattern search (Audet and Dennis Jr, 2004), mesh adaptive direct search (Audet and Dennis Jr, 2006), line search (Fasano et al., 2014; Liuzzi et al., 2016), the Frank-Wolfe algorithm (Usmanova et al., 2019), and stochastic zeroth-order constraint extrapolation (Nguyen and Balasubramanian, 2023).

However, these works differ from ours and are not directly comparable. First, the settings are different—they require oracle evaluation of whether a proposed solution violates constraints during the optimization process, whereas we do not need this. We only require data samples from the feasible set. Second, these works are hardly applied to the high-dimensional problems that we focus on, such as molecule optimization, protein design, and robot morphology optimization.

Baseline	TFBind8	TFBind10	Superconductor	Ant	D'Kitty	ChEMBL	Mean Rank
Dataset Best	0.439	0.00532	74.0	165.326	199.231	383.7e3	-
CbAS	0.958±0.018	0.761±0.067	83.178±15.372	468.711±14.593	213.917±19.863	389.0e3±0.5e3	6.33
GP-qEI	0.824±0.086	0.675±0.043	92.686±3.944	480.049±0.000	213.816±0.000	388.1e3±0.0	7.17
CMA-ES	0.933±0.035	0.848±0.136	90.821±0.661	**1016.409**±906.407	4.700±2.230	388.4e3±0.4e3	5.17
Gradient Ascent	<u>0.981</u>±0.010	0.770±0.154	93.252±0.886	−54.955±33.482	226.491±21.120	390.1e3±2.0e3	4.33
REINFORCE	0.959±0.013	0.692±0.113	89.469±3.093	−131.907±41.003	−301.866±246.284	388.4e3±2.1e3	7.33
MINs	0.938±0.047	0.770±0.177	89.027±3.093	533.636±17.938	<u>272.675</u>±11.069	<u>391.0e3</u>±0.2e3	4.50
COMs	0.964±0.020	0.750±0.078	78.178±6.179	540.603±20.205	**277.888**±7.799	390.2e3±0.5e3	4.50
DDOM	0.971±0.005	<u>0.885</u>±0.367	<u>103.600</u>±8.139	<u>548.227</u>±11.725	250.529±10.992	388.0e3±1.1e3	<u>3.67</u>
Ours	**0.987** ±0.014	**0.924**±0.224	**113.545**±5.322	493.191±18.165	261.673±3.486	**391.1e3**±3.4e3	**2.00**

Table 1: Results of offline black-box optimization on DesignBench. We report the mean and standard deviation across five random seeds. The best results are **bolded**, and the second best is <u>underlined</u>.

	Top-1				Top-10				Invalidity ↓
	QED ↑	SA ↓	GSK3B ↑	Sum ↑	QED ↑	SA ↓	GSK3B ↑	Sum ↑	
Dataset Mean	0.598	0.204	0.045	0.439	0.598	0.204	0.045	0.439	0
Dataset Sum Best	0.846	0.159	0.99	1.677	0.771	0.129	0.877	1.519	0
Dataset Best	0.947	0.030	0.99	1.907	0.945	0.204	0.947	1.688	0
DDOM	0.790±0.023	0.124±0.007	<u>0.856</u>±0.046	<u>1.521</u>±0.063	0.747±0.033	0.141±0.006	<u>0.695</u>±0.021	<u>1.301</u>±0.037	63.60±3.61
Gradient Ascent	<u>0.834</u>±0.025	0.130±0.024	0.784±0.152	1.487±0.150	0.674±0.043	0.134±0.010	0.678±0.047	1.218±0.091	80.00±19.20
GP-qEI	0.784±0.059	0.149±0.034	0.551±0.106	1.186±0.089	0.743±0.032	0.147±0.010	0.370±0.057	0.966±0.028	63.80±7.62
MINs	0.795±0.156	0.163±0.027	0.466±0.249	1.097±0.200	**0.838**±0.059	0.145±0.029	0.273±0.149	0.966±0.108	<u>38.20</u>±24.66
REINFORCE	**0.865**±0.047	**0.083**±0.013	0.062±0.069	0.843±0.027	<u>0.816</u>±0.056	**0.079**±0.009	0.085±0.072	0.822±0.026	53.40±106.80
CbAS	0.762±0.119	0.138±0.046	0.681±0.077	1.305±0.049	0.687±0.035	0.153±0.010	0.593±0.078	1.126±0.059	46.80±13.90
CMA-ES	0.446±0.011	0.207±0.122	0.012±0.004	0.250±0.128	0.435±0.033	0.230±0.169	0.008±0.001	0.212±0.205	880.00±59.25
Ours	0.798±0.023	<u>0.100</u>±0.031	**0.944**±0.023	**1.641**±0.018	0.786±0.003	<u>0.100</u>±0.006	**0.866**±0.035	**1.552**±0.031	**35.80**±1.46

Table 2: Results on the multi-objective molecule optimization task. Sum denotes the total objective (QED + GSK3B − SA). SA is normalized from the range 1-10 to 0-1. We report the mean and standard deviation across five random seeds. The best results are **bolded**, and the second best is <u>underlined</u>. Top-1 denotes the best solution found, and Top-10 denotes the average of the best ten solutions found. Invalidity denotes the number of invalid molecules in the 1000 generated samples.

5 Experiment

In this section, we conduct experiments on (1) a synthetic Branin task, (2) six real-world offline black-box optimization tasks, and (3) a multi-objective molecule optimization task. Finally, we do ablation studies.

5.1 Synthetic Branin Function Optimization

We first validate our model on a synthetic Branin task. The Branin function (Dixon, 1978) is given as

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s,$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, $t = \frac{1}{8\pi}$. The function has three global minimas, $(-\pi, 12.275), (\pi, 2.275)$, and $(9.42478, 2.475)$.

Optimization with unknown constraints. To assess the capability of DiffOPT in optimizing functions under unknown constraints, we generate a dataset of 6,000 points, uniformly distributed within the feasible domain shaped like an oval. An effective optimizer is expected to infer the feasible space from the dataset and yield solutions strictly within the permissible region, *i.e.*, $(-\pi, 12.275)$ and $(\pi, 2.275)$. We train a diffusion model with Variance Preserving (VP) SDE (Song et al., 2020) on this dataset. More details of the experimental setup are provided in Appendix I. Figure 2 illustrates the sampling trajectory of DiffOPT, clearly demonstrating its capability in guiding the samples towards the optimal points confined to the feasible space.

Compatible with additional known constraints. Our framework is also adaptable to scenarios with additional, known constraints C'. In such instances, we introduce an extra objective function whose Boltzmann density is uniform within the constraint bounds and 0 otherwise, *i.e.*, $q'(x) \propto \exp[\beta' \cdot I(x \in C')]$, with $I(\cdot)$ being the indicator function. To demonstrate this capability, we incorporate a closed-form linear constraint alongside the implicit constraint represented by the dataset. This new constraint narrows the feasible solutions to only $(\pi, 2.275)$. As depicted in Figure 3, DiffOPT effectively navigates towards the sole viable minimizer within the constrained space delineated by the data-driven and explicitly stated constraints. This feature is particularly beneficial in practical applications, such as molecular optimization, where imposing additional spatial or structure constraints might be necessary during optimizing binding affinities with different protein targets (Du et al., 2022).

5.2 Offline Black-box Optimization

We further evaluate DiffOPT on the offline black-box optimization task, wherein a logged dataset is utilized to train a surrogate model that approximates the ob-

jective function. The surrogate model is trained on finite data and may have large fitness errors beyond the data distribution. At inference time, if we directly apply gradient ascent on the surrogate objective, it may produce out-of-distribution designs that "fool" the learned surrogate model into outputting a high value. Therefore, we need to constrain the optimization process within the data distribution because the trained surrogate is only reliable within this range. However, this data distribution constraints cannot be expressed analytically, and therefore this task can be viewed as an instance of optimization with unknown constraints.

Following (Krishnamoorthy et al., 2023), we conduct evaluation on six tasks of DesignBench (Trabucco et al., 2022). **Superconductor** is to optimize for finding a superconducting material with high critical temperature. **Ant** nad **D'Kitty** is to optimize the robot morphology. **TFBind8** and **TFBind10** are to find a DNA sequence that maximizes binding affinity with specific transcription factors. **ChEMBL** is to optimize the drugs for a particular chemical property.

Baselines. We compare DiffOPT with multiple baselines, including gradient ascent, Bayesian optimization (GP-qEI) (Krishnamoorthy et al., 2023), REINFORCE (Sutton et al., 1999), evolutionary algorithm (CAM-ES) (Hansen), and recent methods like MINS (Kumar and Levine, 2020), COMs (Trabucco et al., 2021a), CbAS (Brookes et al., 2019) and DDOM (Krishnamoorthy et al., 2023). We follow (Krishnamoorthy et al., 2023) and set the sampling budget as 256. More details of the experimental setups are provided in Appendix I.

Results. Table 1 shows the performance on the six datasets for all the methods. As we can see, DiffOPT achieves an average rank of 2.0, the best among all the methods. We achieve the best result on 4 tasks. Particularly, in the Superconductor task, DiffOPT surpasses all baseline methods by a significant margin, improving upon the closest competitor by 9.6%. The exceptional performance of DiffOPT is primarily due to its application of a diffusion model to learn the valid data manifold directly from the data set, thus rendering the optimization process significantly more reliable. In contrast, the gradient ascent method, which relies solely on optimizing the trained surrogate model, is prone to settle on suboptimal solutions. Moreover, while DDOM (Krishnamoorthy et al., 2023) employs a conditional diffusion model to learn an inverse mapping from objective values to the input space, its ability to generate samples is confined to the maximum values present in the offline dataset. This limitation restricts its ability to identify global maximizers within the feasible space. The experimental results also demonstrate that DiffOPT can consistently outperform DDOM except for the Ant Dataset.

5.3 Multi-objective Molecule Optimization

An additional advantage of incorporating the data distribution constraints for offline black-box optimization is their direct impact on enhancing the validity of generated solutions. However, it's worth noting that DesignBench lacks a specific metric for assessing validity. Therefore, we further test on a multi-objective molecule optimization task and extend our evaluation to include validity. In this task, we have three objectives: the maximization of the quantitative estimate of drug-likeness (QED), and the activity against glycogen synthase kinase three beta enzyme (GSK3B), and the minimization of the synthetic accessibility score (SA). Following (Jin et al., 2020; Eckmann et al., 2022), we utilize a pre-trained autoencoder from (Jin et al., 2020) to map discrete molecular structures into a continuous low-dimensional latent space and train neural networks as proxy functions for predicting molecular properties to guide the optimization. Detailed experimental setups are provided in the Appendix I.

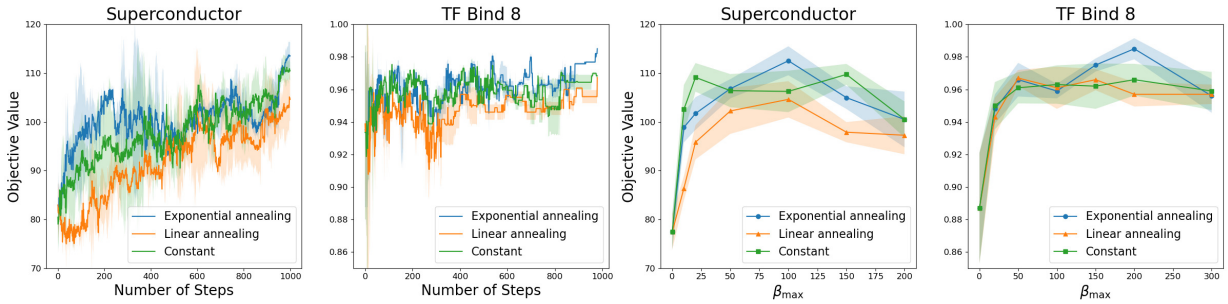
Results. For each method, we generate 1,000 candidate solutions, evaluating the three objective metrics solely on those that are valid. As we can see from Table 2, DiffOPT can achieve the best validity performance among all the methods. In terms of optimization performance, DiffOPT can achieve the best overall objective value. We further report the average of the top 10 solutions found by each model and find that DiffOPT is also reliable in this scenario. This multi-objective optimization setting is particularly challenging, as different objectives can conflict with each other. The superior performance of DiffOPT is because we formulate the optimization problem as a sample problem from the product of experts, which is easy for compositions of various objectives.

5.4 Derivative-free Optimization

	Hypervolume ↑	Invalidity ↓
EA	0.558±0.029	37.6±8.73
IS	0.493±0.108	**0.0**±0.0
Ours	**0.590**±0.060	**0.0**±0.0

Table 3: Results of derivative-free optimization on the multi-objective molecule optimization task.

In this subsection, we explore the scenario where the objective function is not differentiable and examine the effectiveness of the proposed iterative importance sampling with diffusion models. We continue using the molecule optimization task. Here, we do not train a surrogate objective using offline data but instead directly use the oracle function from Rdkit (RDKit, online). We compare our method with evolutionary algorithm (EA; (Hansen)), which is a common technique in derivative-free multi-objective optimization.



(a) Impact of annealing strategies (b) Impact of $\beta_{\max}$

Figure 4: Impact of annealing strategies and $\beta_{\max}$ in the guided diffusion stage. $\beta_{\max}$ is the value of β at the end of annealing.

Additionally, we compare our method with one-step importance sampling (IS) using diffusion models as the proposal distribution. Table 3 provides the performance of all the methods. As we can see, DiffOPT can achieve the best hypervolume and validity among all the methods.

5.5 Ablation Study

	Superconductor	TFBind8
Best Baseline	103.600 ± 8.139	0.981 ± 0.010
Only Stage I	112.038 ± 6.783	0.984 ± 0.012
Only Stage II	92.432 ± 8.635	0.951 ± 0.028
Stage I + Stage II	113.545 ± 5.322	0.987 ± 0.014
Stage I + Stage II + MH	$\mathbf{114.945} \pm 3.615$	$\mathbf{0.989} \pm 0.021$

Table 4: Ablation study on the two-stage sampling.

Impact of two-stage sampling. Table 4 shows the impact of two-stage sampling on performance. Our findings reveal that even after the initial stage, DiffOPT outperforms the top-performing baseline on both datasets. Relying solely on Langevin dynamics, without the warm-up phase of guided diffusion, results in significantly poorer results. This aligns with our discussion in Section 3.2, where we attributed this failure to factors such as the starting distribution, the schedule for step size adjustments, and the challenges posed by undefined gradients outside the feasible set. Integrating both stages yields a performance improvement as the initial stage can provide a better initialization within the data manifold for the later stage (Theorem 1). Adding the MH correction step further enhances results, leading to the best performance observed.

Impact of annealing strategies. We study the influence of different annealing strategies for β during the guided diffusion stage, focusing on the superconductor and TFBind8 datasets. We explore three strategies: constant, linear annealing, and exponential annealing. Figure 4(a) presents the performance across various diffusion steps. We find that our method is not particularly sensitive to the annealing strategies. However, it is worth noting that exponential annealing exhibits a marginal performance advantage over the others.

We also investigate how the value of β at the end of annealing, denoted as $\beta_{\max}$, affects model performance in Figure 4(b). We find that increasing $\beta_{\max}$ initially leads to better performance. However, beyond a certain threshold, performance begins to decrease. It is noteworthy that the optimal value varies across different annealing strategies. Particularly, at $\beta_{\max} = 0$, the model reverts to a pure diffusion process, exhibiting the lowest performance due to the lack of guidance from the objective function.

Sample efficiency. We explore the sample efficiency of DiffOPT at both training and testing stages. Figure 8 (in Appendix K) shows the performance of various methods versus the ratio of training data on Superconductor, TFBind8 and multi-objective molecule optimization. As we can see, on all the three datasets, DiffOPT can outperform all. Our method is also sample efficient during inference. Figure 9 (in Appendix K) shows the performance versus number of samples at inference stage. Notably, on both Superconductor and TFBind8, DiffOPT consistently outperforms all the baseline methods for various sample sizes during inference. It is also important to highlight that our method consistently achieves much greater sample efficiency than DDOM at both training and inference stages, despite both approaches leveraging diffusion models.

6 Conclusion

In this paper, we propose DiffOPT to solve optimization problems where analytic constraints are unavailable. We learn the unknown feasible space from data using a diffusion model and then reformulate the original problem as a sampling problem from the product of (i) the density of the data distribution learned by the diffusion model and (ii) the Boltzmann density defined by the objective function. For differentiable objectives, we propose a two-stage framework consisting of a guided diffusion stage for warm-up and a Langevin dynamics stage for further correction. For non-differentiable objectives, we propose an iterative importance sampling method with diffusion models. Our experiments validate the effectiveness of DiffOPT.

Acknowledgement

This work was supported in part by NSF IIS-2008334, IIS-2106961, IIS-2403240, and CAREER IIS-2144338. Yuanqi Du and Carla P. Gomes was supported by the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Futures program; the National Science Foundation (NSF), the Air Force Office of Scientific Research (AFOSR); the Department of Energy; and the Toyota Research Institute (TRI). Dongxia Wu and Yi-an Ma was supported by the U.S. Department Of Energy, Office of Science, and NSF Grant 2112665 (TILOS), CDC-RFA-FT-23-0069 and DARPA AIE FoundSci.

Y.D. would like to thank Yunan Yang, Guan-horng Liu, Ricky T.Q. Chen, Ge Liu and Yimeng Zeng for their helpful discussions.

References

Stéphane Alarie, Charles Audet, Aïmen E Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9: 100011, 2021.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Charles Audet and John E Dennis Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14(4): 980–1010, 2004.

Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

Maximilian Baladat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. *Advances in Neural Information Processing Systems*, 2020.

Phil S Baran. Natural product total synthesis: as exciting as ever and here to stay, 2018.

Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Linear convergence bounds for diffusion models via stochastic localization. *arXiv preprint arXiv:2308.03686*, 2023.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Jürgen Branke and Christian Schmidt. Sequential sampling in noisy environments. In *International Conference on Parallel Problem Solving from Nature*, pages 202–211. Springer, 2004.

David H Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. *International Conference on Machine Learning*, 2019.

James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. *Numerical nonsmooth optimization: State of the art algorithms*, pages 201–225, 2020.

Patrick Cattiaux, Giovanni Conforti, Ivan Gentil, and Christian Léonard. Time reversal of diffusion processes under a finite entropy condition. In *Annales de l'Institut Henri Poincaré (B) Probabilités et Statistiques*, volume 59, pages 1844–1881. Institut Henri Poincaré, 2023.

Can Chen, Yingxue Zhang, Xue Liu, and Mark Coates. Bidirectional learning for offline model-based biological sequence design. *arXiv preprint arXiv:2301.02931*, 2023.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

Xiang Cheng. *The Interplay between Sampling and Optimization*. University of California, Berkeley, 2020.

Xiang Cheng, Dong Yin, Peter Bartlett, and Michael Jordan. Stochastic gradient and langevin processes. In *International Conference on Machine Learning*, pages 1810–1819. PMLR, 2020.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022a.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022b.

Giovanni Conforti, Alain Durmus, and Marta Gentiloni Silveri. Score diffusion models without early stopping: finite fisher information is all you need. *arXiv preprint arXiv:2308.12240*, 2023.

Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*, 2022.

Valentin De Bortoli and Agnès Desolneux. On quantitative laplace-type convergence results for some exponential probability measures, with two applications. *arXiv preprint arXiv:2110.12922*, 2021.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

Laurence Charles Ward Dixon. The global optimization problem: an introduction. *Towards Global Optimiation 2*, pages 1–15, 1978.

Weitao Du, He Zhang, Tao Yang, and Yuanqi Du. A flexible diffusion model. In *International Conference on Machine Learning*, pages 8678–8696. PMLR, 2023a.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023b.

Yuanqi Du, Tianfan Fu, Jimeng Sun, and Shengchao Liu. Molgensurvey: A systematic survey in machine learning models for molecule design. *arXiv preprint arXiv:2203.14500*, 2022.

Alain Durmus and Éric Moulines. On the geometric convergence for mala under verifiable conditions. *arXiv preprint arXiv:2201.01951*, 2022.

Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation. In *International Conference on Machine Learning*, pages 5777–5792. PMLR, 2022.

Giovanni Fasano, Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. A linesearch-based derivative-free approach for nonsmooth constrained optimization. *SIAM journal on optimization*, 24(3):959–992, 2014.

Justin Fu and Sergey Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2020.

Wenhao Gao and Connor W Coley. The synthesizability of molecules proposed by generative models. *Journal of chemical information and modeling*, 60(12):5714–5723, 2020.

Timur Garipov, Sebastiaan De Peuter, Ge Yang, Vikas Garg, Samuel Kaski, and Tommi S. Jaakkola. Compositional sculpting of iterative generative processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=w79RtqIyoM.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Saul B Gelfand and Sanjoy K Mitter. Recursive stochastic algorithms for global optimization in r^d. *SIAM Journal on Control and Optimization*, 29(5):999–1018, 1991.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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.

Nate Gruver, Samuel Don Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=MfiK69Ga6p.

Nikolaus Hansen. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, pages 75–102.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Ulrich G Haussmann and Etienne Pardoux. Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Advances in neural information processing systems*, 2021.

Chii-Ruey Hwang. Laplace's method revisited: weak convergence of probability measures. *The Annals of Probability*, pages 1177–1182, 1980.

Victor Isakov. *Inverse problems for partial differential equations*, volume 127. Springer, 2006.

Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR, 2020.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Minsu Kim, Federico Berto, Sungsoo Ahn, and Jinkyoo Park. Bootstrapped training of score-conditioned generator for offline design of biological sequences. *arXiv preprint arXiv:2306.03111*, 2023.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Peter E Kloeden, Eckhard Platen, Peter E Kloeden, and Eckhard Platen. *Stochastic differential equations*. Springer, 1992.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Siddarth Krishnamoorthy, Satvik Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. *International Conference on Machine Learning*, 2023.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 2020.

Greg Landrum, Paolo Tosco, Brian Kelley, sriniker, gedeck, NadineSchneider, Riccardo Vianello, Ric, Andrew Dalke, Brian Cole, AlexanderSavelyev, Matt Swain, Samo Turk, Dan N, Alain Vaucher, Eisuke Kawashima, Maciej Wójcikowski, Daniel Probst, guillaume godin, David Cosgrove, Axel Pahl, JP, Francois Berenger, strets123, JLVarjo, Noel O'Boyle, Patrick Fuller, Jan Holst Jensen, Gianluca Sforna, and DoliathGavid. rdkit/rdkit: 2020_03_1 (q1 2020) release, March 2020. URL `https://doi.org/10.5281/zenodo.3732262`.

Gilbert Laporte, Francois Louveaux, and Hélène Mercure. The vehicle routing problem with stochastic travel times. *Transportation science*, 26(3):161–170, 1992.

Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

Ke Li and Jitendra Malik. Learning to optimize. In *International Conference on Learning Representations*, 2016.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 2023.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018.

Giampaolo Liuzzi, Stefano Lucidi, and Francesco Rinaldi. A derivative-free approach to constrained multiobjective nonsmooth optimization. *SIAM Journal on optimization*, 26(4):2744–2774, 2016.

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.

Yi-An Ma, Yuansi Chen, Chi Jin, Nicolas Flammarion, and Michael I Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. *arXiv preprint arXiv:2303.10538*, 2023.

Anthony Nguyen and Krishnakumar Balasubramanian. Stochastic zeroth-order functional constrained optimization: Oracle complexity and applications. *INFORMS Journal on Optimization*, 5(3):256–272, 2023.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Fotios Petropoulos, Gilbert Laporte, Emel Aktas, Sibel A Alumur, Claudia Archetti, Hayriye Ayhan, Maria Battarra, Julia A Bennell, Jean-Marie Bourjolly, John E Boylan, et al. Operational research: Methods and applications. *Journal of the Operational Research Society*, pages 1–195, 2023.

Jakiw Pidstrigach. Score-based generative models detect manifolds. *Advances in Neural Information Processing Systems*, 35:35852–35865, 2022.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR, 2017.

Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

RDKit, online. RDKit: Open-source cheminformatics. http://www.rdkit.org. [Online; accessed 11-April-2013].

Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

Tim Salimans and Jonathan Ho. Should ebms model the energy or the score? In *Energy Based Models Workshop-ICLR 2021*, 2021.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2022.

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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*, 2020.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference*

on International Conference on Machine Learning, pages 1015–1022, 2010.

Jan Stanczuk, Georgios Batzolis, Teo Deveney, and Carola-Bibiane Schönlieb. Your diffusion model secretly knows the dimension of the data manifold. *arXiv preprint arXiv:2212.12611*, 2022.

Mandt Stephan, Matthew D Hoffman, David M Blei, et al. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JV8Ff0lgVV.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 1999.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Conservative objective models for effective offline model-based optimization. *International Conference on Machine Learning*, 2021a.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021b.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

N García Trillos, B Hosseini, and D Sanz-Alonso. From optimization to sampling through gradient flows. *NOTICES OF THE AMERICAN MATHEMATICAL SOCIETY*, 70(6), 2023.

Ilnura Usmanova, Andreas Krause, and Maryam Kamgarpour. Safe convex learning under uncertain constraints. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2106–2114. PMLR, 2019.

Milan Voršilák, Michal Kolář, Ivan Čmelo, and Daniel Svozil. Syba: Bayesian estimation of synthetic accessibility of organic compounds. *Journal of cheminformatics*, 12(1):1–13, 2020.

Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. *Advances in Neural Information Processing Systems*, 35:31444–31458, 2022.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620 (7976):1089–1100, 2023.

Li Kevin Wenliang and Ben Moran. Score-based generative models learn manifold-like structures with constrained mixing. *arXiv preprint arXiv:2311.09952*, 2023.

Andre Wibisono. Sampling as optimization in the space of measures: The langevin dynamics as a composite optimization problem. In *Conference on Learning Theory*, pages 2093–3027. PMLR, 2018.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in neural information processing systems*, 29, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016b.

Luhuan Wu, Brian L Trippe, Christian A Naesseth, David M Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *arXiv preprint arXiv:2306.17775*, 2023.

Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34:4619–4631, 2021.

Ye Yuan, Can Chen, Zixuan Liu, Willie Neiswanger, and Xue Liu. Importance-aware co-teaching for offline model-based optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Dinghuai Zhang, Jie Fu, Yoshua Bengio, and Aaron Courville. Unifying likelihood-free inference with black-box optimization and beyond. In *International Conference on Learning Representations*, 2021.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with GFlownets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=sTjW3JHs2V.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix for DiffOPT

A Limitations and Future Work

We discuss limitations and possible extensions of DiffOPT. (i) *Manifold preserving*. The guided diffusion may deviate from the manifold that the score network is trained, leading to error accumulations. One approach to mitigate this is to incorporate manifold constraints during the guided diffusion phase (Chung et al. 2022b; He et al., 2023). (ii) *Online learning*. We have applied DiffOPT in the offline black-box optimization (BBO) setting. Considering the unknown constraints not only benefits the offline setting but also helps the online BBO. In the context of online BBO, we propose molecules and subsequently receive evaluations from the ground-truth simulator at each iteration to train the surrogate objective. Proposing a higher proportion of valid molecules can significantly increase the sampling efficiency of training the surrogate objective.

B Broader Impacts

Optimization techniques can be used to solve a wide range of real-world problems, from decision making (planning, reasoning, and scheduling), to solving PDEs, and to designing new drugs and materials. The method we present in this paper extends the scope of the previous study to a more realistic setting where (partial) constraints for optimization problems are unknown, but we have access to samples from the feasible space. We expect that by learning the feasible set from data, our work can bring a positive impact to the community in accelerating solving real-world optimization problems and finding more realistic solutions. However, care should be taken to prevent the method from being used in harmful settings, such as optimizing drugs to enhance detrimental side effects.

C (Metropolis-adjusted) Langevin Dynamics.

Langevin dynamics is a class of Markov Chain Monte Carlo (MCMC) algorithms that aims to generate samples from an unnormalized density $\pi(x)$ by simulating the differential equation

$$\mathrm{d}\mathbf{x}_t = \nabla_{\mathbf{x}} \log \pi(\mathbf{x}_t)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{w}_t. \tag{4}$$

Theoretically, the continuous SDE of Equation (4) is able to draw exact samples from $\pi(x)$. However, in practice, one needs to discretize the SDE using numerical methods such as the Euler-Maruyama method (Kloeden et al., 1992) for simulation. The Euler-Maruyama approximation of Equation (4) is given by

$$x_{t+\Delta t} = x_t + \nabla \log \pi(x) + \sqrt{2\Delta t}z, \qquad z \sim \mathcal{N}(0,1), \tag{5}$$

where Δt is the step size. By drawing x_0 from an initial distribution and then simulating the dynamics in Equation (4), we can generate samples from $\pi(x)$ after a 'burn-in' period. This algorithm is known as the Unadjusted Langevin Algorithm (ULA) (Roberts and Tweedie, 1996), which requires $\nabla \log \pi(x)$ to be L-Lipschitz for stability.

The ULA always accepts the new sample proposed by Equation (5). In contrast, to mitigate the discretization error when using a large step size, the Metropolis-adjusted Langevin Algorithm (MALA) (Grenander and Miller, 1994) uses the Metropolis-Hastings algorithm to accept or reject the proposed sample. Specifically, we first generate a proposed update $\hat{x}$ with Equation (4), then with probability $\min(1, \frac{\pi(\hat{x})\mathcal{N}(x|\hat{x}+\Delta t\cdot\nabla_{\hat{x}}\log\pi(\hat{x}),2\Delta t)}{\pi(x)\mathcal{N}(\hat{x}|x+\Delta t\cdot\nabla_x\log\pi(x),2\Delta t)})$, we set $x_{t+\Delta t} = \hat{x}$, otherwise $x_{t+\Delta} = x_t$. We provide the pseudocode of both algorithms in Algorithm 1.

Algorithm 1 Sampling via the (Metropolis-Adjusted) Langevin dynamics

Require: unnormalized density $\pi(x)$, step size Δt
$x \sim$ initial distribution
$\mathcal{X} = \varnothing$
for number of iterations **do**
  $\hat{x} = x + \nabla_x \log \pi(x)\Delta t + \sqrt{2\Delta t}\cdot z, \quad z \sim \mathcal{N}(0,1)$
  if applying Metropolis-Hastings test **then**
    $u \sim \mathrm{Uniform}[0,1]$
    $\log P_{\text{accept}} = \log \frac{\pi(\hat{x})\mathcal{N}(x|\hat{x}+\Delta t\cdot\nabla_{\hat{x}}\log\pi(\hat{x}),2\Delta t)}{\pi(x)\mathcal{N}(\hat{x}|x+\Delta t\cdot\nabla_x\log\pi(x),2\Delta t)}$
    if $\log P_{\text{accept}} > \log u$, then $x \leftarrow \hat{x}$
  else
    $x \leftarrow \hat{x}$
  end if
  $\mathcal{X} \leftarrow \mathcal{X} \cup x$
end for
Return $\mathcal{X}$

D Pseudocode of the Two-Stage Sampling

The pseudocode for the proposed two-stage sampling method is provided in Algorithm 2.

E Illustration of Why Guided Diffusion Cannot Sample from the Product of Experts

The primary limitation of relying solely on Stage I is its inability to theoretically sample from our desired true target distribution, the product of distributions $\pi_\beta \propto p(x)q_\beta(x)$. This is because the score of the diffused marginal distribution does not directly correspond to the aggregate of the scores from each individual distribution,

$$\begin{aligned}\nabla_x \log \pi_\beta^t(x_t) &= \nabla_x \log \int p_0(x_0)q_\beta(x_0)p_{t|0}(x_t|x_0)\mathrm{d}x_0 \\ &\neq \nabla_x \log q_\beta(x_t) + \nabla_x \log \underbrace{\int p_0(x_0)p_{t|0}(x_t|x_0)\mathrm{d}x_0}_{p_t(x_t)}, \quad t > 0,\end{aligned}$$

Algorithm 2 Sampling via DIFFOPT for differentiable objective

Require: inverse temperature schedule $\beta(t)$, diffusion volatility schedule $g(t)$ and drift $f(x,t)$, score model $s_\theta(x_t,t)$, energy function of the data distribution $E_\theta(x,t)$ if applying the MH correction.

$\mathcal{X} \leftarrow \varnothing$
Sample $x_0 \sim \mathcal{N}(0, \mathrm{Id})$
// Stage I: *Warm-up with guided diffusion.*
for $t = 0, \ldots, T$ **do**
  Draw $z \sim \mathcal{N}(0, \mathrm{Id})$, define $\tau = T - t$
  $x_{t+\Delta t} \leftarrow x_t + [-f(x_t,\tau) + g^2(\tau)s_\theta(x_t,\tau)$
  $\qquad -\beta(\tau)\nabla_{x_t} h(x_t)]\Delta t + g(\tau)\sqrt{\Delta t}z$
end for
// Stage II: *Further correction with Langevin dynamics.*
for $t = T, \ldots, T'$ **do**
  Draw $z \sim \mathcal{N}(0, \mathrm{Id})$
  $\hat{x} \leftarrow x + [s_\theta(x,0) - \beta\nabla h(x)]\Delta t + \sqrt{2\Delta t}z$
  if applying Metropolis-Hastings test **then**
    $u \sim \mathrm{Uniform}[0,1]$
    $\ell_\theta(\hat{x}) = E_\theta(\hat{x},0) - \beta h(\hat{x})$
    $\ell_\theta(x) = E_\theta(x,0) - \beta h(x)$
    $\ell(\hat{x},x) = -\|x - \hat{x} - \Delta t[s_\theta(\hat{x},0) - \beta\nabla h(\hat{x})]\|^2$
    $\ell(x,\hat{x}) = -\|\hat{x} - x - \Delta t[s_\theta(x,0) - \beta\nabla h(x)]\|^2$
    $\ell_{\mathrm{acc}} = \ell_\theta(\hat{x}) - \ell_\theta(x) + (\ell(\hat{x},x) - \ell(x,\hat{x}))/(2\Delta t)$
    if $\ell_{\mathrm{acc}} > \log(u)$, then $x \leftarrow \hat{x}$
  else
    $x \leftarrow \hat{x}$
  end if
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$
end for
Return $\mathcal{X}$

where $p_{t|0}(x_t|x_0)$ is the conditional density of the forward SDE starting at x_0.

Therefore we have to include Langevin dynamics (Stage II) as an optional stage to correct the bias introduced in Stage I. This stage can provide a theoretical guarantee for drawing exact samples from the product of distributions, despite the empirical observation that it offers only marginal performance improvements.

It's important to note that Stage I is essential because its output focuses on feasible minimizers under specific conditions, offering an improved initialization for Langevin dynamics. This has been demonstrated in our ablation study.

F Pseudocode of the Proposed Derivative-free Sampling

The pseudocode of our derivative-free sampling algorithm is provided in Algorithm 3.

G Energy-based Parameterization

In a standard diffusion model, we learn the score of the data distribution directly as $s_\theta(x,t) = \nabla \log p_t(x)$. This parameterization can be used for ULA, which only requires gradients of log-likelihood. However, to incorporate the Metropolis-Hastings (MH) correction step, access to the unnormalized density of the data distribution is necessary to calculate the acceptance probability.

To enable the use of MH correction, we can instead learn the energy function of the data distribution, *i.e.*, $p(x,t) \propto e^{E_\theta(x,t)}$. The simplest approach is to use a scalar-output neural network, denoted as $\mathrm{NN}_\theta(x,t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, to parameterize $E_\theta(x,t)$. By taking the gradient of this energy function with respect to the input x, we can derive the score of the data distribution. However, existing works have shown that this parameterization can cause difficulties during model training (Salimans and Ho, 2021). Following the approach by (Du et al., 2023b), we define the energy function as $E_\theta(x,t) = -\frac{1}{2}|\mathrm{NN}_\theta(x,t)|_2^2$, where $\mathrm{NN}_\theta(x,t)$ is a vector-output neural network mapping from $\mathbb{R}^d \times \mathbb{R}$ to $\mathbb{R}$. Consequently, the score of the data distribution is represented as $s_\theta(x,t) = -\mathrm{NN}_\theta(x,t)\nabla_x \mathrm{NN}_\theta(x,t)$.

Algorithm 3 Sampling via DIFFOPT for non-differentiable objective

Require: inverse temperature schedule $\beta(t)$, diffusion volatility schedule $g(t)$ and drift $f(x,t)$, score model $s_\theta(x_t,t)$.

// Initialization
Sample S particles $\{x_s^0\}_{s=1}^S$ from diffusion model
Compute $w_s^0 = \frac{\pi_\beta(x_s^0)}{p_\theta(x_s^0)} = q_\beta(x_s^0)$ for each particle
Normalize the weight $\tilde{w}_s^0 = \frac{w_s^0}{\sum_{s=1}^M w_s^0}$ for each particle
Resample S particles $\{x_s^0\}_{s=1}^S$ according to the weights
// Iterative Importance Sampling
for $k = 1, \cdots, K+1$ **do**
  Sample $t \sim \mathcal{U}[0,T]$
  for $s = 1, \cdots, S$ **do**
    // $x_s^k \equiv x_{s,0}^k$
    Add noise to the particle by forward diffusion until time t: $x_{s,t}^{k-1} \sim p_{t|0}(x_{s,t}^{k-1}|x_{s,0}^{k-1})$
    Denoise the particle through backward diffusion: $x_{s,0}^k \sim p_{0|t}(x_{s,0}^k|x_{s,t}^{k-1})$
    Sample $x_{s,t,j}^{k-1} \sim p_{t|0}(x_{s,t}^{k-1}|x_{s,0}^{k-1}) = \mathcal{N}(x_{s,t}^{k-1}|\alpha_t x_{s,0}^{k-1}, \frac{\sigma_t^2}{1+\sigma_t}\mathrm{Id})$ for J times
    $Q(x_s^k|x_s^{k-1}) \approx \frac{1}{J}\sum_{j=1}^J \mathcal{N}\left(\frac{x_{s,t,j}+\sigma_t s_\theta(x_{s,t,j}^{k-1},t)}{\alpha_t}, \frac{\sigma_t^2}{1+\sigma_t}\mathrm{Id}\right)$
    Marginalize to get $Q(x_s^k) \approx \frac{1}{S}\sum_{s=1}^S Q(x_s^k|x_s^{k-1})$
  end for
  Compute $w_s^k = \frac{\pi_\beta(x_s^k)}{Q(x_s^k)}$ for each particle
  Normalize the weight for each particle: $\tilde{w}_s^k = \frac{w_s^k}{\sum_{j=1}^S w_j^k}$
  Resample M particles $\{x_s^k\}_{s=1}^S$ according to the weights
end for
Return $\mathcal{X} = \{x_s^{K+1}\}_{s=1}^S$

H End Distribution of the Warm-up Stage

In this section, we study in further detail the warm-up stage of DIFFOPT. We recall that we consider a process of the following form

$$\mathrm{d}\mathbf{y}_t^\beta = [-f(\mathbf{y}_t^\beta, 1-t) + g(1-t)^2\nabla \log p_{1-t}(\mathbf{y}_t^\beta) - \beta(1-t)\nabla h(\mathbf{y}_t^\beta)]\mathrm{d}t + g(1-t)\mathrm{d}\mathbf{w}_t, \qquad \mathbf{y}_0^\beta \sim \mathbf{x}_T \tag{6}$$

where $T=1$ and p_t is the density w.r.t. Lebesgue measure of the distribution of $\mathbf{x}_t$ where

$$\mathrm{d}\mathbf{x}_t = f(t,\mathbf{x}_t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t, \qquad \mathbf{x}_0 \sim p_0. \tag{7}$$

We recall that under mild assumption (Cattiaux et al., 2023), we have that $(\hat{\mathbf{y}}_t)_{t\in[0,1]} = (\mathbf{x}_{1-t})_{t\in[0,1]}$ satisfies

$$\mathrm{d}\hat{\mathbf{y}}_t = [-f(\hat{\mathbf{y}}_t, 1-t) + g(1-t)^2\nabla \log p_{1-t}(\hat{\mathbf{y}}_t)]\mathrm{d}t + g(1-t)\mathrm{d}\mathbf{w}_t, \qquad \hat{\mathbf{y}}_0 = \mathbf{x}_T.$$

Let us highlight some differences between (6) and the warm-up process described in Algorithm 2. First, we note that we do not consider an approximation of the score but the real score function $\nabla \log p_t$. In addition, we do not consider a discretization of (6). This difference is mainly technical. The discretization of diffusion processes is a well-studied topic, and we refer to (De Bortoli et al., 2021; Benton et al., 2023; Conforti et al., 2023; Chen et al., 2022) for a thorough investigation. Our contribution to this work is orthogonal as we are interested in the role of h on the distribution. Our main result is Proposition 4 and details how the end distribution of the warm-up process concentrates on the minimizers of h, which also support the data distribution p_0.

We first show that under assumptions on h, q_T^β, the density w.r.t. the Lebesgue measure of $\mathbf{y}_T$ has the same support as p_0. We denote by $\mathrm{supp}(p_0)$ the support of p_0. We consider the following assumption.

Assumption 1. *We have that for any $t \in [0,1]$, $g(t) = g(0)$ and $f(t,x) = -\gamma_0 x$ with $\gamma_0 > 0$. Assume that p_0 has bounded support, i.e. there exists $R > 0$ such that $\mathrm{supp}(p_0) \subset \mathrm{B}(0,R)$ with $\mathrm{B}(0,R)$ the ball with center 0. In addition, assume that h is Lipschitz.*

Then, we have the following proposition.

Proposition 2. *Assume* **A1**. *Then, we have that for any* $\beta > 0$, $\mathbf{y}_T^\beta \in \mathrm{supp}(p_0)$.

Proof. This directly applies to the results of (Pidstrigach, 2022). First, we have that (Pidstrigach, 2022, Assumption 1, Assumption 2) are satisfied using A1. (Pidstrigach, 2022, Lemma 1) and the second part of (Pidstrigach, 2022, Theorem 2). We conclude using the first part of (Pidstrigach, 2022, Theorem 2). □

In Proposition 2, we show that the guided reconstructed scheme used for warm-up (6) cannot discover minimizers outside the support of p_0. In Proposition 4, we will show that we concentrate on the minimizers inside the support of p_0 under additional assumptions.

Next, we make the following assumption, which is mostly technical. We denote q_t^β the distribution of $\mathbf{y}_t^\beta$ for any $t \in [0,1]$. We also denote $(p_{1-t}^\beta)_{t\in[0,1]} = (q_t^\beta)_{t\in[0,1]}$.

Assumption 2. *We have that* $h \in \mathrm{C}^\infty(\mathbb{R}^d, \mathbb{R})$. *In addition,* $C > 0$ *exists such that for any* $x \in \mathbb{R}^d$, *we have a.s.*

$$|\int_0^1 \beta_t (W_t^\beta(\mathbf{z}_t) - W_0^\beta(\mathbf{z}_0))\mathrm{d}t| \leq C, \tag{8}$$

with $\mathrm{d}\mathbf{z}_t = \{f_t - g_t^2 \nabla \log p_t\rangle \mathrm{d}t + \mathrm{d}\mathbf{w}_t$, $\mathbf{z}_0 = x$, $W_t = \langle \nabla \log p_t^\beta, \nabla h\rangle + \Delta h$ *and* $\beta_t \equiv \beta(1-t)$. *For* $t \in [0,1]$ *and* $x \in \mathbb{R}^d$

$$V_t = \mathrm{div}(f_t - g_t^2 \nabla \log p_t), \qquad W_t^\beta = \langle \nabla h, \nabla \log p_t^\beta \rangle + \Delta h.$$

Assume that V *and* W^β *are continuous and bounded on* $[0,1] \times \mathbb{R}^d$. *Assume that* $(p_t)_{t\in[0,1]}$ *and* $(p_t^\beta)_{t\in[0,1]}$ *are strong solutions to their associated Fokker-Planck equations.*

We do not claim that we verify this hypothesis in this paper. Proving A2 is out of the scope of this work, and we mainly use it to 1) control high-order terms and 2) provide sufficient conditions to apply the Feynman-Kac theorem (Oksendal, 2013, Theorem 7.13) and the Fokker-Planck equation. The bound in (8) controls the regularity of the $W_t^\beta(\mathbf{z}_t)$. Given that (under some mild regularity assumption), $(W_t^\beta(\mathbf{z}_t))_{t\in[0,1]}$ satisfies a Stochastic Differential Equation we expect that $\mathbb{E}[\|(W_t^\beta(\mathbf{z}_t) - W_0^\beta(\mathbf{z}_0))\|] \leq C\sqrt{t}$ for any $t \in [0,1]$ and some constant $C > 0$ (independent of t). Therefore, we get that (8) is true in expectation under some regularity assumption (a. Δh is Lipschitz. bThe diffused distribution p_t is smooth with bounded derivatives.). We conjecture that the almost sure bound we require is unnecessary and that moment bounds should be enough. We leave this study for future work.

Proposition 3. *Assume* **A2**. *For any* $x \in \mathbb{R}^d$, *let* $\tilde{p}_0^\beta(x)$ *be given by*

$$\tilde{p}_0^\beta(x) = p_0(x) \exp[\log(\beta_0)\{\Delta h(x) + \langle \nabla \log p_0^\beta(x), \nabla h(x)\rangle\}],$$

where p_0^β *is the distribution of* $\mathbf{y}_T^\beta$ *and* β_0 *is the inverse temperature at the end of the process.*

Then there exists $C_0 > 0$ *such that for any* $x \in \mathbb{R}^d$

$$(1/C_0)\tilde{p}_0^\beta(x) \leq p_0^\beta(x) \leq C_0 \tilde{p}_0^\beta(x).$$

Proof. First, for any $t \in [0,1]$ we denote p_t the density of $\mathbf{x}_t$ where $(\mathbf{x}_t)_{t\in[0,1]}$ is given by (7). Similarly, we denote q_t^β the distribution of $\mathbf{y}_t^\beta$ for any $\tau_0 > 0$ and $t \in [0,1]$. Finally, we denote $(p_{1-t}^\beta)_{t\in[0,1]} = (q_t^\beta)_{t\in[0,1]}$ for any $\tau_0 > 0$. In what follows, we fix $t_0 > 0$. Using A2 we have that for any $t \in [0,1]$

$$\partial_t p_t = -\mathrm{div}(f_t p_t) + (g_t^2/2)\Delta p_t.$$

Therefore, we have that for any $t \in [0,1]$

$$\partial_t p_t + \mathrm{div}(f_t p_t) - (g_t^2/2)\Delta p_t = 0.$$

This can also be rewritten as

$$\partial_t p_t + \langle f_t - g_t^2 \nabla \log p_t, \nabla p_t\rangle + (g_t^2/2)\Delta p_t + \mathrm{div}(f_t - g_t^2 \nabla \log p_t) p_t = 0, \tag{9}$$

where we have used that $\mathrm{div}(\nabla \log p_t p_t) = \Delta p_t$. Similarly, we have that

$$\partial_t q_t^\beta = -\mathrm{div}(\{-f_{1-t} + g_{1-t}^2 \nabla \log p_{1-t} - \beta_{1-t}\nabla h\} q_t^\beta) + (g_{1-t}^2/2)\Delta q_t^\beta.$$

We have that

$$\partial_t p_t^\beta = \mathrm{div}(\{-f_t + g_t^2 \nabla \log p_t - \beta_t \nabla h\} p_t^\beta) - (g_t^2/2)\Delta p_t^\beta.$$

This can also be rewritten as

$$\partial_t p_t^\beta + \langle f_t - g_t^2 \nabla \log p_t^\beta, \nabla p_t^\beta \rangle + \mathrm{div}(f_t - g_t^2 \nabla \log p_t) p_t^\beta + (g_t^2/2)\Delta p_t^\beta + \mathrm{div}(\beta_t \nabla h p_t^\beta) = 0. \tag{10}$$

Finally, this can be rewritten as

$$\partial_t p_t^\beta + \langle f_t - g_t^2 \nabla \log p_t, \nabla p_t^\beta \rangle + (g_t^2/2)\Delta p_t^\beta + \mathrm{div}(f_t - g_t^2 \nabla \log p_t) p_t^\beta + \beta_t \{\langle \nabla h, \nabla \log p_t^\beta \rangle + \Delta h\} p_t^\beta = 0.$$

In what follows, we denote

$$\mu_t = f_t - g_t^2 \nabla \log p_t, \qquad V_t = \mathrm{div}(f_t - g_t^2 \nabla \log p_t), \qquad W_t^\beta = \langle \nabla h, \nabla \log p_t^\beta \rangle + \Delta h. \tag{11}$$

Hence, using (10) we have

$$\partial p_t^\beta + \langle \mu_t, \nabla \log p_t^\beta \rangle + (g_t^2/2)\Delta p_t^\beta + V_t p_t^\beta + \beta_t W_t^\beta p_t^\beta = 0. \tag{12}$$

Similarly, using (9) we have

$$\partial p_t + \langle \mu_t, \nabla \log p_t \rangle + (g_t^2/2)\Delta p_t + V_t p_t = 0. \tag{13}$$

Therefore, combining (13), A2 and (Oksendal, 2013, Theorem 7.13) we get that for any $x \in \mathbb{R}^d$

$$p_0(x) = \mathbb{E}[\exp[\textstyle\int_0^1 V_t(\mathbf{z}_t)\mathrm{d}t] p_T(\mathbf{z}_T) \mid \mathbf{z}_0 = x],$$

with $\mathrm{d}\mathbf{z}_t = \mu_t \mathrm{d}t + g_t \mathrm{d}\mathbf{w}_t$ and $\mathbf{z}_0 = x$. Similarly, combining (12), A2 and (Oksendal, 2013, Theorem 7.13) we get that for any $x \in \mathbb{R}^d$

$$p_0^\beta(x) = \mathbb{E}[\exp[\textstyle\int_0^1 V_t(\mathbf{z}_t)\mathrm{d}t] \exp[\int_0^1 \beta_t W_t^\beta(\mathbf{z}_t)\mathrm{d}t] p_T(\mathbf{z}_T) \mid \mathbf{z}_0 = x]. \tag{14}$$

Using (11), we have that

$$\textstyle\int_0^1 \beta_t W_t^\beta(\mathbf{z}_t)\mathrm{d}t = \int_0^1 \beta_t \{\langle \nabla h, \nabla \log p_t^\beta \rangle + \Delta h\}(\mathbf{z}_t)\mathrm{d}t.$$

Hence, we have

$$\textstyle\int_0^1 \beta_t W_t^\beta(\mathbf{z}_t)\mathrm{d}t = \log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta \rangle + \Delta h\}(\mathbf{z}_0) + \int_0^1 \beta_t (W_t^\beta(\mathbf{z}_t) - W_0^\beta(\mathbf{z}_0))\mathrm{d}t.$$

Using A2 we have that

$$-C \leq \textstyle\int_0^1 \beta_t W_t^\beta(\mathbf{z}_t)\mathrm{d}t - \log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta \rangle + \Delta h\}(\mathbf{z}_0) \leq C\,.$$

Hence,

$$-C + \log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta \rangle + \Delta h\}(\mathbf{z}_0) \leq \textstyle\int_0^1 \beta_t W_t^\beta(\mathbf{z}_t)\mathrm{d}t \leq C + \log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta \rangle + \Delta h\}(\mathbf{z}_0)\,.$$

Combining this result with (14) we get that for any $x \in \mathbb{R}^d$

$$\begin{aligned} p_0^\beta(x) &\leq \mathbb{E}[\exp[\textstyle\int_0^1 V_t(\mathbf{z}_t)\mathrm{d}t] p_T(\mathbf{z}_T) \mid \mathbf{z}_0 = x] \exp[\log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta + \Delta h \rangle\}(x)] \exp[C] \\ &= p_0(x) \exp[\log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta + \Delta h \rangle\}(x)] \exp[C]. \end{aligned}$$

Similarly, we have for any $x \in \mathbb{R}^d$

$$p_0^\beta(x) \geq p_0(x) \exp[\log(\beta_0)\{\langle \nabla h, \nabla \log p_0^\beta \rangle + \Delta h\}(x)] \exp[-C],$$

which concludes the proof. □

In Proposition 3, we show that the output distribution of the warm-up process is upper and lower bounded by a product of experts comprised of (i) p_0 which ensures the *feasibility* conditions (ii) $\exp[\log(\beta_0) W_0^\beta]$ related to the *optimization* of the objective. While Proposition 3 gives an explicit form for p_0^β, it does not provide insights on the properties of this distribution. However, we can still infer some limiting properties.

Proposition 4. *If $x^\star$ is a local strict minimizer of h in the support of p_0 then $\lim_{\beta_0 \to \infty} p_0^\beta(x^\star) = +\infty$. If $x^\star$ is a local strict minimizer of h not in the support of p_0 then $\lim_{\beta_0 \to \infty} p_0^\beta(x^\star) = 0$.*

Proof. The case where $x^\star$ is not in support of p_0 is trivial using Proposition 3. We now assume that $x^\star$ is in the support of p_0. Using Proposition 3 and that $\nabla h(x^\star) = 0$ and $\Delta h(x^\star) > 0$ since the local minimizer $x^\star$ is strict, we get that $\lim_{\beta_0 \to \infty} p_0^\beta(x^\star) = +\infty$, which concludes the proof. □

In particular, Proposition 4 shows that the limit distribution of $\mathbf{y}_T^\beta$ concentrates around the minimizers of h, which is the expected behavior of increasing the inverse temperature. What is also interesting is that we only target the minimizers of h, which are inside the support of p_0. This is our primary goal, which is *constrained* optimization of h.

I Experiment Details

I.1 Computing Infrastructure

System: Ubuntu 18.04.6 LTS; Python 3.9; Pytorch 1.11. CPU: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz. GPU: GeForce GTX 2080 Ti.

I.2 Synthetic Branin Experiment

We consider the commonly used Branin function as a synthetic toy example that takes the following form (illustrated in Figure 5):

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s,$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, $t = \frac{1}{8\pi}$.

The Branin function $f(x_1, x_2)$ has three global minimas located at points $(-\pi, 12.275)$, $(\pi, 2.275)$, and $(9.42478, 2.475)$ with a value of 0.397887.

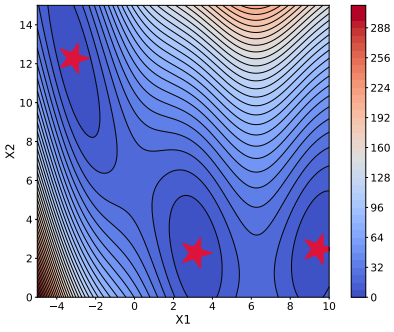


Figure 5: Branin function.

Dataset details. We curate 6,000 data points by sampling uniformly in an ellipse region with center $(-0.2, 7.5)$ and semi-axis lengths $(3.6, 8.0)$ as training data (the blue region in Figure 2). It is tilted counterclockwise by 25 degrees, ensuring that it covers two minimizers of the function: $(-\pi, 12.275)$ and $(\pi, 2.275)$. It is worth noting that sampling points (x_1, x_2) to construct the training dataset are irrelevant to the objective value $f(x_1, x_2)$. For the experiment with additional known constraints, we introduce two constraints $x_2 \leq \frac{3}{2}x_1 + \frac{15}{2}$ and $x_2 \leq -\frac{3}{2}x_1 + 15$ (the pink region in Figure 2) to further narrow down the feasible solution to $(\pi, 2.275)$. We split the dataset into training and validation sets by 9:1.

Implementation details. We build the score network s_θ of the diffusion model with a 2-layer MLP architecture with 1024 hidden dimensions and ReLU activation function. The forward process is a Variance Preserving (VP) SDE (Song et al., 2020). We set the minimum and maximum values of noise variance to be 0.01 and 2.0, respectively. We employ a fixed learning rate of 0.001, a batch size of 128, and 1000 epochs for model training. At test time, we sample 500 candidate solutions. We use a constant inverse temperature $\beta = 5$ for the Boltzmann distribution induced by the objective function. For the distribution induced by the additional known constraints, we set $\beta' = 10$.

I.3 Offline Black-box Optimization

Dataset details. DesignBench (Trabucco et al., 2022) is an offline black-box optimization benchmark for real-world optimization tasks. Following (Krishnamoorthy et al., 2023), we use three continuous tasks: Superconductor, D'Kitty Morphology and Ant Morphology, and three discrete tasks: TFBind8, TFBind10, and ChEMBL. Consistent with (Krishnamoorthy et al., 2023), we exclude NAS due to its significant computational resource demands. We also exclude Hopper as it is known to be buggy (see Appendix C in (Krishnamoorthy et al., 2023)). We split the dataset into training and validation sets by 9:1.

- Superconductor: materials optimization. This task aims to search for materials with high critical temperatures. The dataset contains 17,014 vectors with 86 components representing the number of atoms of each chemical element in the formula. The provided oracle function is a pre-trained random forest regression model.
- D'Kitty Morphology: robot morphology optimization. This task aims to optimize the parameters of a D'Kitty robot, such as size, orientation, and location of the limbs, to make it suitable for a specific navigation task. The dataset size is 10,004, and the parameter dimension is 56. It uses MuJoCO (Todorov et al., 2012), a robot simulator, as the oracle function.
- Ant Morphology: robot morphology optimization. Similar to D'Kitty, this task aims to optimize the parameters of a quadruped robot to move as fast as possible. It consists of 10,004 data, and the parameter dimension is 60. It also uses MuJoCo as the oracle function.
- TFBind8: DNA sequence optimization. This task aims to find the DNA sequence of length eight with the maximum binding affinity with transcription factor SIX6_REF_R1. The design space is the space of sequences of nucleotides represented as categorical variables. The size of the dataset is 32,898, with a dimension of 8. The ground truth serves as a direct oracle since the affinity for the entire design space is available.
- TFBind10: DNA sequence optimization. Similar to TFBind8, this task aims to find the DNA sequence of length ten that has the maximum binding affinity with transcription factor SIX6_REF_R1. The design space consists of all possible designs of nucleotides. The size of the dataset is 10,000, with a dimension of 10. Since the affinity for the entire design space is available, it uses the ground truth as a direct oracle.
- ChEMBL: molecule activity optimization. This task aims to find molecules with a high MCHC value when paired with assay CHEMBL3885882. The dataset consists of 441 samples of dimension 31.

Baselines. We compare with eight baselines on DesignBench tasks. The results of all the baselines are from (Krishnamoorthy et al., 2023). **Gradient ascent** learns a surrogate model of the objective function and generates the optimal solution by iteratively performing gradient ascent on the surrogate model. **CbAS** learns a density model in the design space coupled with a surrogate model of the objective function. It iteratively generates samples and refines the density model on the new samples during training. **GP-qEI** fits a Gaussian Process on the offline dataset. It employs the quasi-Expected-Inprovement (qEI) acquisition function from Botorch (Baladat et al., 2020) for Bayesian optimization. **MINS** learns an inverse map from objective value back to design space using a Generative Adversarial Network (GAN). It then obtains optimal solutions through conditional generation. **REINFORCE** parameterizes a distribution over the design space and adjusts this distribution in a direction that maximizes the efficacy of the surrogate model. **COMS** learns a conservative surrogate model by regularizing the adversarial samples. It then utilizes gradient ascent to discover the optimal solution. **CMAES** enhances a distribution over the optimal design by adapting the covariance matrix according to the highest-scoring samples selected by the surrogate model. **DDOM** learns a conditional diffusion model to learn an inverse mapping from the objective value to the input space.

Implementation details. We build the score network s_θ using a simple feed-forward network. This network consists of two hidden layers, each with a width of 1024 units, and employs ReLU as the activation function. The forward process is a Variance Preserving (VP) SDE (Song et al., 2020). We set the noise variance limits to a minimum of 0.01 and a maximum of 2.0.

For the surrogate models, we explore various network architectures tailored to different datasets, including Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), Gaussian Fourier Network, and Deep Kernel Learning (DKL) (Wilson et al., 2016b,a). LSTM network uses a single-layer LSTM unit with a hidden dimension of 1024, followed by 1 hidden layer with a dimension of 1024, utilizing ReLU as the activation function. In the Gaussian Fourier regressor, Gaussian Fourier embeddings (Tancik et al., 2020) are applied to the inputs x and t. These embeddings are then processed through a feed-forward network with 3 hidden layers, each of 1024 width, utilizing Tanh as the activation function. This regressor is time-dependent, and its training objective follows the method used by (Song et al., 2020) for training time-dependent classifiers in conditional generation. For DKL, we use the ApproximateGP class in Gpytorch[1], which consists of a deep feature extractor and a Gaussian process (GP). The feature extractor is a simple feed-forward network consisting of 2 hidden layers with a width of 500 and 50, respectively, and ReLU activations. The GP uses radial basis function (RBF) kernel.

[1] https://docs.gpytorch.ai/en/stable/_modules/gpytorch/models/approximate_gp.html#ApproximateGP

We use a fixed learning rate of 0.001 and a batch size of 128 for both the diffusion and surrogate models. During testing, we follow the evaluation protocol from the (Krishnamoorthy et al., 2023), sampling 256 candidate solutions. We apply different annealing strategies for different datasets. Specifically, we apply exponential annealing for TFBind8, superconductor, D'Kitty, and ChEMBL. The exponential annealing strategy is defined as $\beta(\tau) = \beta_{\max}[1 - \exp(-100(T - \tau))]$, where $\tau = T - t$, and a constant β for Ant and TFBind10. Though exponential annealing usually exhibits better performance, we leave the exploration of exponential annealing on TFBind10 and D'Kitty for future work due to time limit. The step size Δt is 0.001 for the first stage, and 0.0001 for the second stage.

Detailed hyperparameters and network architectures for each dataset are provided in Table 5.

	Annealing strategy	$\beta_{\max}$	Surrogate model
TFBind8	Exponential	200	Gaussian Fourier
TFBind10	Constant	20	Gaussian Fourier
Superconductor	Exponential	100	Gaussian Fourier
Ant	Exponential	30	Gaussian Fourier
D'Kitty	Constant	$3e4$	DKL
ChEMBL	Exponential	100	LSTM

Table 5: Implementation details on design-bench. $\beta_{\max}$ is the value of β at the end of the annealing process.

I.4 Multi-objective Molecule Optimization

Dataset details. We curate the dataset by encoding 10000 molecules (randomly selected) from the ChEMBL dataset (Gaulton et al., 2012) with HierVAE (Jin et al., 2020), a commonly used molecule generative model based on VAE, which takes a hierarchical procedure in generating molecules by building blocks. Since validity is important for molecules, we ensure HierVAE can decode all the randomly selected encoded molecules. We split all the datasets into training and validation sets by 9:1.

Oracle details. We evaluate three commonly used molecule optimization oracles including *synthesis accessibility (SA)*, *quantitative evaluation of drug-likeness (QED)* and activity again target *GSK3B* from RDKit (Landrum et al., 2020) and TDC (Huang et al., 2021). All three oracles take as input a SMILES string representation of a molecule and return a scalar value of the property. The oracles are non-differentiable.

Implementation details. We build the score network s_θ of the diffusion model using a 2-layer MLP architecture. This network features 1024 hidden dimensions and utilizes the ReLU activation function. The forward process adheres to a Variance Preserving (VP) SDE proposed by (Song et al., 2020). We calibrate the noise variance within this model, setting its minimum at 0.01 and maximum at 2.0.

For the surrogate model of the objective function, we use the ApproximateGP class in Gpytorch[2], which consists of a deep feature extractor and a Gaussian process. The feature extractor is a simple feed-forward network with two hidden layers, having widths of 500 and 50, respectively, and both employ ReLU activation functions. Regarding model optimization, we apply a fixed learning rate of 0.001 for the diffusion model and 0.01 for the surrogate model. Additionally, we set a batch size of 128 and conduct training over 1000 epochs for both models. For the sampling process, we use a consistent inverse temperature $\beta = 10^4$ for all the three objectives. The step size Δt is 0.001 for the first stage, and 0.0001 for the second stage.

We sample 1000 candidate solutions at test time for all the methods. For DDOM (Krishnamoorthy et al., 2023), we use their implementation [3]. For other baselines, we use the implementations provided by DesignBench[4]. We tune the hyper-parameters of all the baselines as suggested in their papers.

For derivative-free optimization, the number of iterations is set to 100 for both the evolutionary algorithm and DiffOPT. The number of particles for all methods remains the same as before, *i.e.*, 1000.

[2] https://docs.gpytorch.ai/en/stable/_modules/gpytorch/models/approximate_gp.html#ApproximateGP
[3] https://github.com/siddarthk97/ddom
[4] https://github.com/brandontrabucco/design-bench

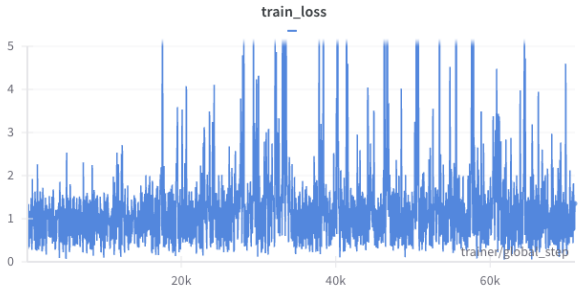


Figure 6: Training loss of the surrogate objective on Ant dataset

J Analysis on Ant Dataset

As shown in Table 1, DiffOPT can only achieve subpar performance on the Ant dataset. This underperformance is primarily due to the difficulty in training the surrogate objective function. An illustration of this challenge is provided in Figure 6, where the training loss of the surrogate objective is displayed. The training loss fluctuates throughout the training process. Although we investigated different network architectures, the issue still remains. Building an effective surrogate model for this dataset may require a more sophisticated architecture design. We leave this for future work.

K Ablation Studies

	Superconductor	TFBind8
Best Baseline	103.600 ± 8.139	0.981 ± 0.010
Only Stage I	112.038 ± 6.783	0.984 ± 0.012
Only Stage II	92.432 ± 8.635	0.951 ± 0.028
Stage I + Stage II	113.545 ± 5.322	0.987 ± 0.014
Stage I + Stage II + MH	$\mathbf{114.945} \pm 3.615$	$\mathbf{0.989} \pm 0.021$

Table 6: Ablation study on the two-stage sampling.

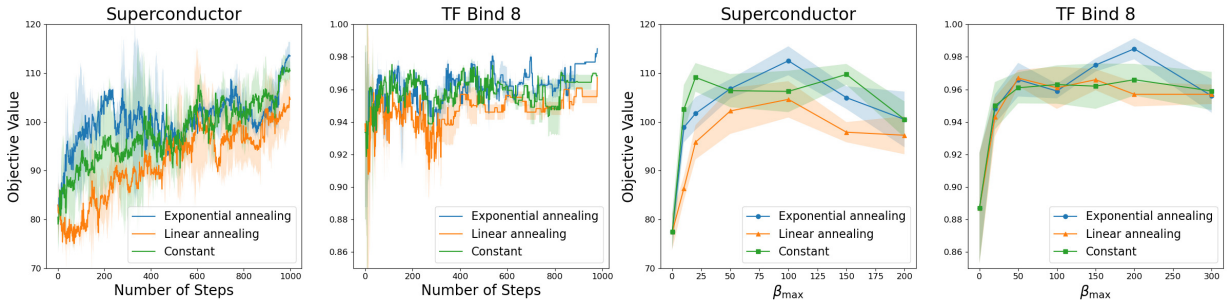
Impact of two-stage sampling. Table 6 shows the impact of two-stage sampling on performance. Our findings reveal that even after the initial stage, DiffOPT outperforms the top-performing baseline on both datasets. Relying solely on Langevin dynamics, without the warm-up phase of guided diffusion, results in significantly poorer results. This aligns with our discussion in Section 3.2, where we attributed this failure to factors such as the starting distribution, the schedule for step size adjustments, and the challenges posed by undefined gradients outside the feasible set. Integrating both stages yields a performance improvement as the initial stage can provide a better initialization within the data manifold for the later stage (Theorem 1). Adding the MH correction step further enhances results, leading to the best performance observed.

Impact of annealing strategies. We study the influence of different annealing strategies for β during the guided diffusion stage, focusing on the superconductor and TFBind8 datasets. We explore three strategies: constant, linear annealing, and exponential annealing. Figure 7(a) presents the performance across various diffusion steps. We find that our method is not particularly sensitive to the annealing strategies. However, it is worth noting that exponential annealing exhibits a marginal performance advantage over the others.

We also investigate how the value of β at the end of annealing, denoted as $\beta_{\max}$, affects model performance in Figure 7(b). We find that increasing $\beta_{\max}$ initially leads to better performance. However, beyond a certain threshold, performance begins to decrease. It is noteworthy that the optimal value varies across different annealing strategies. Particularly, at $\beta_{\max} = 0$, the model reverts to a pure diffusion process, exhibiting the lowest performance due to the lack of guidance from the objective function.

Sample efficiency We explore the sample efficiency of DiffOPT at both training and testing stages. Figure 8 shows the performance of various methods versus the ratio of training data on Superconductor, TFBind8 and multi-objective molecule optimization. As we can see, on all the three datasets, DiffOPT can outperform all.

Our method is also sample efficient during inference. Figure 9 shows the performance versus number of samples at inference stage. Notably, on both Superconductor and TFBind8, DiffOPT consistently outperforms all the baseline methods for various sample sizes during inference.



(a) Impact of annealing strategies (b) Impact of $\beta_{\max}$

Figure 7: Impact of annealing strategies and $\beta_{\max}$ in the guided diffusion stage. $\beta_{\max}$ is the value of β at the end of annealing.

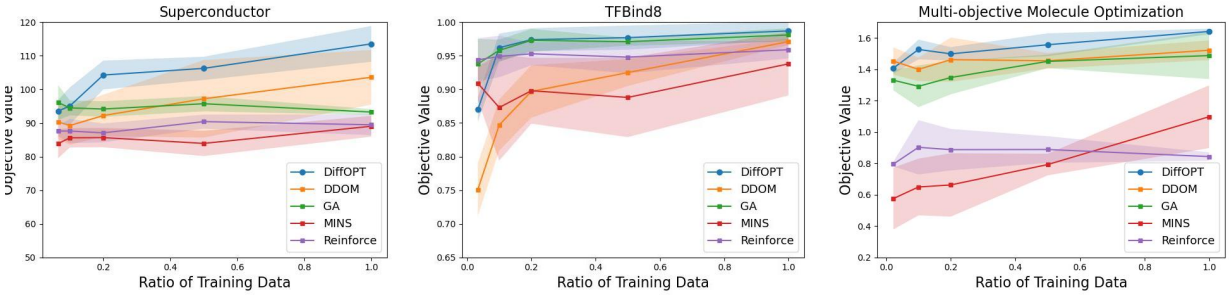


Figure 8: Impact of the number of training data.

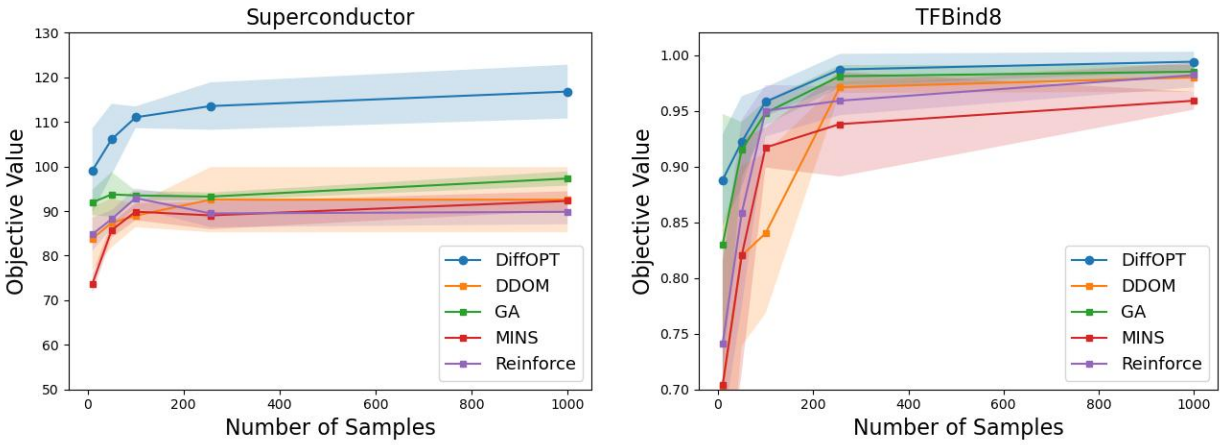


Figure 9: Impact of the number of samples at testing time.

It is also important to highlight that our method consistently achieves much greater sample efficiency than DDOM at both training and inference stages, despite both approaches leveraging diffusion models.