PROBABILITY MEASURE GRADIENT FLOWS FOR DIFFUSION-BASED AMORTIZED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

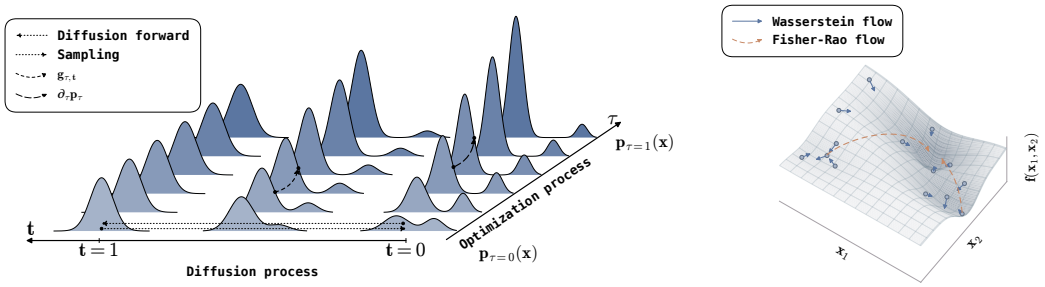
Nonconvex optimization problems are ubiquitous but can be computationally expensive to solve repeatedly. Recent work on amortized optimization accelerates repeated optimization by learning a mapping from problem parameters to their solutions. However, existing methods typically learn a deterministic solution mapping, limiting their ability to represent and explore multiple promising regions of a nonconvex solution landscape. In this work, we introduce DIFLOWOPT, a self-supervised amortized optimization method that models solutions as a distribution parameterized by a diffusion model. We lift amortized optimization to the space of probability measures and study its optimization dynamics under Wasserstein and Fisher–Rao gradient flows. We show that these two geometries induce complementary gradient estimators for training diffusion-based amortized optimizers, leveraging first- and zeroth-order objective information, respectively. Motivated by this connection, DIFLOWOPT updates the diffusion model along a Wasserstein–Fisher–Rao flow that combines both sources of information. To accommodate constraints, we further develop a primal-dual formulation in which instance-specific dual variables are learned by a neural network. Across three standard high-dimensional nonconvex optimization problems and two real-world applications, DIFLOWOPT achieves better solution quality than classical solvers and existing amortized optimization methods, with substantially lower solve time than classical solvers.

1 INTRODUCTION

Nonconvex constrained optimization problems arise in various fields, including control, molecular modeling, power systems, and signal processing. Classical solvers are computationally expensive on these problems and often require multi-start with many iterations to find the global optimum. To address this, self-supervised amortized optimization methods (Park & Van Hentenryck, 2023; Donti et al., 2021; Amos, 2023; Lüken & Lucia, 2026) learn the solution map from problem parameters to the optimal solution with a neural network, and train the network without solutions from classical solvers. This family of methods leverages the shared structure across related problem instances and achieves orders of magnitude speedups compared to classical solvers. However, existing methods have the following limitations: (1) The predicted solution of each instance is represented as a single point and usually updated along the gradient, so the learned solutions can easily get trapped in local minima. (2) The neural network outputs only one deterministic point for each instance, which can lead to mode averaging when the objective landscape is multimodal.

Lifting the optimization variables to the space of probability measures transforms the nonconvex optimization problem into an equivalent linear optimization problem, whose minimizers are the measures supported on the global optima of the original problem (Lasserre, 2001; Hsieh et al., 2019). Population-based global optimization methods such as differential evolution (Storn & Price, 1997), CMA-ES (Hansen et al., 2003) and estimation of distribution algorithms (Larranaga & Bielza, 2025) can be viewed through the lens of probability measure optimization. They maintain a distribution represented by a parametric model or a set of particles, and update the distribution toward the global optima. Another line of work explores sampling as optimization, and uses techniques including variants of Langevin dynamics for global optimization (Geman & Hwang, 1986; Chiang et al., 1987; Wibisono, 2018). These distribution-level methods are more effective in locating the global optima when the objective landscape is complex, but they require a large number of iterations and intensive

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Diffusion-based optimization over the space of probability measures. (b) Optimization dynamics.

Figure 1: Conceptual overview. We lift the optimization problem to the space of probability measures and parametrize the measure with a diffusion model, which is updated so that its induced distribution follows a gradient flow $\partial_\tau p_\tau$. Figure 1b illustrates two representative flows: the Wasserstein gradient flow transports probability mass locally, whereas the Fisher-Rao gradient flow reweights it globally, effectively teleporting mass across basins. Given $\partial_\tau p_\tau$, we derive estimators for the time derivative of the perturbed score, $g_{\tau,t}\left(\mathbf{x}_t\right) = \partial_\tau \nabla_{\mathbf{x}_t} \log p_{\tau,t}\left(\mathbf{x}_t\right)$, and use them to update the diffusion model. At inference, amortized solutions are obtained by standard diffusion sampling from the model at $\tau = T$.

computation, especially when a population of particles is used. Self-supervised amortized optimization at the distribution level is a promising yet largely unexplored direction, and it poses two challenges. The first is how to model the distribution with sufficient flexibility: inaccurate modeling of the intermediate distributions during training can propagate into subsequent updates and lead to suboptimal performance, and the final solutions can also have multiple modes. The second is how to effectively update the model in a self-supervised manner.

To overcome these challenges, in this work we develop a diffusion-based self-supervised amortized optimization method, which models the predicted solution distribution with a diffusion model and optimizes it based on gradient flows in the space of probability measures. Diffusion models provide a highly flexible representation of complex and multimodal distributions, as demonstrated across the domains of image (Rombach et al., 2022), video (Gao et al., 2025) and control (Chi et al., 2025). The scalable regression training loss (Ho et al., 2020; Vincent, 2011) makes them well suited to fitting high-dimensional distributions with known data samples. On the other hand, approaches for optimizing a diffusion model with respect to a functional of its induced distribution have not yet converged on a standard methodology, with each having its own tradeoffs.

Optimizing a diffusion model can be viewed as tracing a path in the space of probability measures from an initial distribution to a target distribution minimizing the objective functional. One can specify such a path through a gradient flow. Each gradient flow and its induced path have different characteristics that shape the optimization behavior. This perspective expands the design space of diffusion optimization algorithms, enables a unified approach for analyzing them, and is naturally formulated in continuous optimization time, which allows for various discretization schemes. As a representative example, the Wasserstein gradient flow (Jordan et al., 1998) exploits local information to transport probability mass, but can get trapped in local optima when the objective landscape is nonconvex. In contrast, the Fisher-Rao gradient flow (Liu et al., 2023b) reweights mass and can reach global optima under proper conditions, but requires the initial distribution to have sufficiently wide support and may fail to use first-order information well. The Wasserstein-Fisher-Rao gradient flow (Liero et al., 2016; Chizat et al., 2018; Ren et al., 2024; Sharrock & Nemeth, 2025) combines mass transport and reweighting, and has shown strong convergence and empirical performance in a broad range of optimization problems.

We further derive a series of novel unbiased estimators for optimizing diffusion models following different gradient flows. Specifically, we focus on the variant based on the Wasserstein-Fisher-Rao gradient flow given its strong fit for various optimization problems. The resulting diffusion optimization loss naturally combines zeroth-order and first-order information, and thus captures both local and global structure of the objective. Building upon this, we propose a practical self-supervised primal-dual learning algorithm for nonconvex constrained optimization that models the primal variable with a diffusion model and the dual variable with a neural network. The diffusion model captures the distribution of the primal variable with strong expressive capability and is up-

dated following the Wasserstein-Fisher-Rao gradient flow. The dual network learns instance-specific constraint multipliers and enables adaptive constraint violation penalties during training.

In summary, the contributions of this paper are:

1. From the probability measure optimization view, we present a unified family of diffusion optimization methods, which recovers existing approaches as special cases and gives rise to new ones. We further derive a series of unbiased estimators for the derivative of the perturbed score following the Wasserstein, Fisher-Rao and Wasserstein-Fisher-Rao gradient flows.
2. We propose a self-supervised primal-dual learning algorithm for amortized optimization of nonconvex constrained problems. The primal variables are modeled with a diffusion model, which provides an expressive representation of their distributions and is effectively updated following the Wasserstein-Fisher-Rao gradient flow. The dual variables are modeled with a neural network, which enables instance-specific constraint violation penalties.
3. We conduct experiments on high-dimensional nonconvex constrained optimization problems. Our proposed algorithm shows significant improvement over classical solvers and existing amortized optimization baselines on both standard and real-world optimization problems.

1.1 Related Work

Amortized optimization approaches use learning to leverage the shared structure across related problem instances, so as to accelerate the solving process and improve solution quality (Amos, 2023). Self-supervised approaches learn directly from the objective and constraint functions, eliminating the need for pre-collected solutions (Donti et al., 2021; Park & Van Hentenryck, 2023; Lüken & Lucia, 2026). Most closely related to ours, DiOpt (Ding et al., 2025) trains a diffusion model with a self-supervised weighted regression loss that supports general objective and constraint formulations, but requires supervised pretraining to perform well. Saravanos et al. learns a flow model as an iterative optimizer, but only for unconstrained problems. Other self-supervised diffusion-based methods are restricted to specific KL-based objectives and constraint formulations (Khalafi et al., 2024; Hadou et al., 2026). In contrast, DiFlowOpt is fully self-supervised, supports general nonconvex objectives and constraints, and is grounded in gradient flows over the space of probability measures. A complete discussion of related work is provided in Section A.

2 Preliminaries

2.1 Parametric Constrained Optimization

We consider solving a family of optimization problems in the form of

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}} \quad & f_{\mathbf{s}}(\mathbf{x}) \\ \text{subject to} \quad & g_{\mathbf{s},i}(\mathbf{x}) \leqslant 0, \quad i=1,\dots,m, \\ & h_{\mathbf{s},j}(\mathbf{x}) = 0, \quad j=1,\dots,n. \end{aligned} \tag{1}$$

Here, $\mathbf{s}\in\mathbb{R}^p$ denotes the problem parameters, and $\mathbf{x}$ is the decision variable to be optimized. Each $\mathbf{s}$ specifies an instance of the optimization problem by determining the functions $f_{\mathbf{s}}$, $g_{\mathbf{s}}$, and $h_{\mathbf{s}}$. We assume the functions $f_{\mathbf{s}}$, $g_{\mathbf{s}}$, and $h_{\mathbf{s}}$ to be differentiable and possibly nonconvex. The primary goal of this work is to learn the solution map $\mathcal{X}^*(\mathbf{s}) = \arg\min_{\mathbf{x}\in\mathcal{F}(\mathbf{s})} f_{\mathbf{s}}(\mathbf{x})$, where $\mathcal{F}(\mathbf{s})$ is the feasible set and $\mathcal{X}^*(\mathbf{s})$ may contain multiple optimal solutions.

2.2 Diffusion-Based Generative Models

Diffusion models (Song et al., 2020; Ho et al., 2020) and flow models (Lipman et al., 2022; Liu et al., 2022) are trained to reverse a forward noising process that transforms a data distribution p_0 to a simple distribution such as a Gaussian. The forward process has the form

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \quad t\in[0,1], \tag{2}$$

where in this paper we assume $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ as in diffusion models. After defining the forward process, this family of models then learns the score function $s_\theta(\mathbf{x}_t,t)$ or the velocity function $v_\theta(\mathbf{x}_t,t)$

that enables sampling from p_0 with the reverse process. Specifically, in the score parametrization, the target score function corresponds to the marginal distribution

$$p_t(\mathbf{x}_t) = \int \mathcal{N}\left(\mathbf{x}_t \mid \alpha_t \mathbf{x}_0,\, \sigma_t^2 \mathbf{I}\right)\, p_0(\mathbf{x}_0)\, d\mathbf{x}_0, \tag{3}$$

and can be learned with $\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\left\| s_\theta(\mathbf{x}_t, t) + \frac{\boldsymbol{\epsilon}}{\sigma_t} \right\|^2\right]$, or in the $\boldsymbol{\epsilon}$-parametrization, $\mathcal{L}_{\mathrm{DM}}(\theta) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\left\| \epsilon_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon} \right\|^2\right]$. With the learned score function or velocity function, sampling can be performed by solving a stochastic or ordinary differential equation (SDE/ODE). For example, in the score parametrization,

$$d\mathbf{x}_t = \left[\frac{\dot{\alpha}_t}{\alpha_t}\mathbf{x}_t - \sigma_t^2\left(\frac{\dot{\sigma}_t}{\sigma_t} - \frac{\dot{\alpha}_t}{\alpha_t}\right) s_\theta(\mathbf{x}_t, t)\right] dt, \tag{4}$$

which is the probability flow ODE corresponding to the forward process in Equation 2.

3 Diffusion-Based Optimization over the Space of Probability Measures

In this work, we study amortized optimization of the parametric constrained optimization problem in Equation 1, especially in the nonconvex setting. Our approach lifts the problem to the space of probability measures and represents the induced distribution with a diffusion model. From this viewpoint, diffusion-based optimization amounts to updating a diffusion model so as to minimize a functional of the distribution it induces. This is naturally cast as optimization over the space of probability measures. The optimization process then traces a path in this space determined by the choice of gradient flow. In this section, we use this perspective to present a unified family of diffusion-based optimization methods. We further derive a class of unbiased estimators for the diffusion model update under different gradient flows.

3.1 Lifting optimization to the space of probability measures

Let $\mathbf{x}$ be the decision variable, and $\mathbf{s} \in \mathbb{R}^p$ be the problem parameters. For simplicity, in this section we consider an unconstrained optimization problem in $\mathbb{R}^d$,

$$\text{(P):} \qquad \min_{\mathbf{x}\in\mathbb{R}^d} \quad f_{\mathbf{s}}(\mathbf{x}), \tag{5}$$

and present the extension to constrained optimization problems in Section 4. Solving Equation 5 in the original space $\mathbb{R}^d$ is complex when $f_{\mathbf{s}}$ is highly nonconvex. Alternatively, one can show that the optimization is equivalent to a linear programming problem in the probability measure space $\mathcal{P}\left(\mathbb{R}^d\right)$,

$$\text{(M):} \qquad \min_{p\in\mathcal{P}(\mathbb{R}^d)} \quad \mathcal{G}(p), \qquad \mathcal{G}(p) \coloneqq \mathbb{E}_{\mathbf{x}\sim p}\left[f_{\mathbf{s}}(\mathbf{x})\right] \tag{6}$$

in the sense that the two formulations have the same optimal value, and every global minimizer $\mathbf{x}^*$ of (P) induces an optimal solution $\delta_{\mathbf{x}^*}$ of (M). Conversely, every optimal solution of (M) is supported on the set of global minimizers of (P). This lifting operation greatly simplifies the objective landscape, converting a nonconvex objective to a linear one, though it introduces difficulty in the solution space since the decision variable is now an infinite-dimensional distribution.

3.2 Solving the lifted problem via gradient flows

We optimize (M) using gradient flows on the probability measure space. A gradient flow is specified by a Riemannian geometry on this space, and different geometries give different evolution paths for the measure. Concretely, the gradient flow is governed by a partial differential equation (PDE) describing how the probability measure evolves with respect to time, which takes the general form

$$\frac{\partial p_\tau}{\partial \tau} = -\operatorname{grad}_m \mathcal{G}(p_\tau). \tag{7}$$

Given the metric m, the gradient flow exhibits distinct characteristics.

Parameterizing the Distribution with Diffusion Models We parametrize the decision variable of the lifted problem (M), i.e., the probability measure p, with a diffusion model. Specifically, we adopt the ϵ-prediction form and learn a neural network to predict the noise ϵ given the instance problem parameter $\mathbf{s}$, the perturbed sample $\mathbf{x}_t$, and the diffusion timestep t as inputs. As we discuss below, the intermediate iterates in amortized optimization training can be complex and multimodal, and a mismatch between the learned and target distributions can propagate through subsequent training, leading to cumulative error. Diffusion models are well suited to this setting since they provide a highly flexible representation of distributions and scale effectively to high-dimensional spaces.

Gradient Flow on the Perturbed Distribution Note that Equation 7 is defined on the clean distribution space, while for diffusion learning we also need access to the formulation on the perturbed distribution space. Since the diffusion forward process remain unchanged for all τ, using equation 3 we obtain

$$\frac{\partial p_{\tau,t}}{\partial \tau} = -\int \mathcal{N}\left(\mathbf{x}_t \mid \alpha_t \mathbf{x}_0,\, \sigma_t^2 \mathbf{I}\right) \operatorname{grad}_m \mathcal{G}\left(p_\tau\right)\left(\mathbf{x}_0\right)\, d\mathbf{x}_0, \tag{8}$$

where $p_\tau(\mathbf{x}_0) = p_{\tau,0}(\mathbf{x}_0)$. We use τ to denote the optimization time, and t for the diffusion time.

We present the following proposition that connects the time derivative of the perturbed score $g_{\tau,t}(\mathbf{x}_t) = \partial_\tau \nabla_{\mathbf{x}_t} \log p_{\tau,t}(\mathbf{x}_t)$ with $\partial_\tau p_\tau$.

Proposition 1 *Let $q_t\left(\mathbf{x}_t|\mathbf{x}_0\right)$ be the forward perturbation kernel, under proper regularity conditions,*

$$g_{\tau,t}\left(\mathbf{x}_t\right) = \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p_\tau(\mathbf{x}_0) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)}. \tag{9}$$

All proofs are deferred to Section B. Plugging the form of $\partial_\tau p_\tau$ under a given gradient flow into the proposition yields practical unbiased estimators for $g_{\tau,t}(\mathbf{x}_t)$, as we will discuss below.

3.3 Unbiased estimators under specific gradient flows

Although Proposition 1 gives a general form for the time derivative of the perturbed score, it is still unclear how to update the distribution parametrized by a diffusion model under a specific gradient flow. To this end, in this section we develop unbiased estimators for $g_{\tau,t}(\mathbf{x}_t)$ under the common Wasserstein and Fisher-Rao gradient flows.

Wasserstein Gradient Flow Under the Wasserstein gradient flow, the probability distribution satisfies the continuity equation

$$\frac{\partial p_\tau}{\partial \tau} - \operatorname{div}\left(p_\tau \nabla_{W_2} \mathcal{G}(p_\tau)\right) = 0, \tag{10}$$

and the particles move according to the velocity field $\dot{\mathbf{x}}_\tau = -\nabla_{W_2} \mathcal{G}(p_\tau)\left(\mathbf{x}_\tau\right)$. In the special case $\mathcal{G}\left(p\right) = \mathbb{E}_{\mathbf{x}\sim p}\left[f(\mathbf{x})\right]$, the Wasserstein gradient is given by $\nabla_{W_2} \mathcal{G}(p_\tau)(\mathbf{x}) = \nabla_{\mathbf{x}} f(\mathbf{x})$. We present the unbiased estimator under Wasserstein gradient flow in the following proposition.

Proposition 2 (Estimator under the Wasserstein Gradient Flow) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function, and assume the involved distribution moments are finite. Denote $m_{\tau,t}\left(\mathbf{x}_t\right) = \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)}\left[\mathbf{x}_0\right]$, and define*

$$W_{\tau,t}\left(\mathbf{x}_0, \mathbf{x}_t\right) = \mathbf{I} + \frac{\alpha_t}{\sigma_t^2}\left(\mathbf{x}_0 - m_{\tau,t}\left(\mathbf{x}_t\right)\right)\left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)^\top. \tag{11}$$

Then, under appropriate regularity conditions, the following identity holds under the Wasserstein flow Equation 10:

$$g^{\mathrm{W}}_{\tau,t}(\mathbf{x}_t) = -\frac{\alpha_t}{\sigma_t^2} \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)}\left[W_{\tau,t}\left(\mathbf{x}_0, \mathbf{x}_t\right) \nabla_{\mathbf{x}_0} f\left(\mathbf{x}_0\right)\right], \tag{12}$$

which gives an unbiased single-sample estimator for $g^W_{\tau,t}(\mathbf{x}_t)$.

In practice, the posterior mean $m_{\tau,t}\left(\mathbf{x}_t\right)$ can be obtained from the current diffusion model via Tweedie's formula, $m_{\tau,t}\left(\mathbf{x}_t\right) = \frac{\mathbf{x}_t - \sigma_t \epsilon_{\bar{\theta}}(\mathbf{x}_t, t)}{\alpha_t}$. Given an accurate $m_{\tau,t}$, the estimator is unbiased with a single sample, and its variance can be reduced by averaging over a batch of particles $\mathbf{x}_0 \sim p_{\tau,t}\left(\mathbf{x}_0|\mathbf{x}_t\right)$.

Fisher-Rao Gradient Flow The Fisher-Rao gradient flow takes the form

$$\frac{\partial p_\tau(\mathbf{x})}{\partial \tau} = p_\tau(\mathbf{x}) \left(\mathbb{E}_{p_\tau} \left[f(\mathbf{x}) \right] - f(\mathbf{x}) \right), \tag{13}$$

which has the closed-form solution $p_\tau(\mathbf{x}) = p_0(\mathbf{x}) \frac{\exp(-\tau f(\mathbf{x}))}{Z_\tau}$. We now present the unbiased estimator in the following proposition.

Proposition 3 (Estimator under the Fisher-Rao Gradient Flow) *Under the regularity conditions of Proposition 2, the following identity holds under the Fisher-Rao flow Equation 13:*

$$g^{\mathrm{FR}}_{\tau,t}(\mathbf{x}_t) = -\frac{\alpha_t}{\sigma_t^2} \mathrm{Cov} \left(\mathbf{x}_0, f\left(\mathbf{x}_0\right) \mid \mathbf{x}_t \right) = -\frac{\alpha_t}{\sigma_t^2} \mathbb{E} \left[\left(\mathbf{x}_0 - m_{\tau,t}\left(\mathbf{x}_t\right) \right) f\left(\mathbf{x}_0\right) \right]. \tag{14}$$

Remark 4 *The widely used KL-regularized diffusion fine-tuning scheme, as well as the diffusion training based on mirror descent, can both be viewed as discretizations of the Fisher-Rao flow.*

3.4 Optimization under the Wasserstein-Fisher-Rao flow

The Wasserstein flow locally transports mass and its estimator (Proposition 2) uses only first-order information, whereas the Fisher-Rao flow globally reweights mass and its estimator (Proposition 3) uses only zeroth-order information. We therefore propose diffusion optimization under the Wasserstein-Fisher-Rao gradient flow, a linear combination of the two:

$$\partial_\tau p_\tau^{\mathrm{WFR}} = a \partial_\tau p_\tau^{\mathrm{W}} + b \partial_\tau p_\tau^{\mathrm{FR}}, \tag{15}$$

where a and b are the coefficients. Its estimator follows from the linearity of $g_{\tau,t}$ in $\partial_\tau p_\tau$.

Proposition 5 (Linearity of the Estimator) *Let p_τ be a probability distribution whose evolution satisfies the linear decomposition*

$$\partial_\tau p_\tau = a \partial_\tau p_\tau^1 + b \partial_\tau p_\tau^2, \tag{16}$$

then the time derivative of the perturbed score satisfies

$$g_{\tau,t}(\mathbf{x}_t) = a g^1_{\tau,t}(\mathbf{x}_t) + b g^2_{\tau,t}(\mathbf{x}_t) \tag{17}$$

Consequently,

$$g^{\mathrm{WFR}}_{\tau,t}(\mathbf{x}_t) = a g^{\mathrm{W}}_{\tau,t}(\mathbf{x}_t) + b g^{\mathrm{FR}}_{\tau,t}(\mathbf{x}_t), \tag{18}$$

which naturally incorporates both first- and zeroth-order information and thus exploits both the local and global structure of the objective, making it well suited to a wide range of optimization problems.

4 Diffusion-Based Primal-Dual Learning

We now apply diffusion-based optimization over the space of probability measures to nonconvex constrained optimization. For Equation 1, the primal-dual problem is

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\lambda^g \geqslant 0, \lambda^h} \mathcal{L}_{\mathbf{s}}\left(\mathbf{x}, \lambda\right) = \min_{\mathbf{x}\in\mathcal{X}} \max_{\lambda^g \geqslant 0, \lambda^h} f_{\mathbf{s}}(\mathbf{x}) + \sum_{i=1}^{m} \lambda^g_{\mathbf{s},i} g_{\mathbf{s},i}(\mathbf{x}) + \sum_{i=1}^{n} \lambda^h_{\mathbf{s},i} h_{\mathbf{s},i}(\mathbf{x}). \tag{19}$$

Since $\mathcal{L}_{\mathbf{s}}(\mathbf{x}, \lambda)$ can be non-convex with respect to x, gradient descent-ascent (GDA) is not guaranteed to find the global minimax points (Lin et al., 2020; Jin et al., 2020).

Lifting to the space of probability distributions We instead lift the primal variable $\mathbf{x}$ to a distribution $p \in \mathcal{P}\left(\mathcal{X}\right)$, giving

$$\begin{aligned} \min_{p\in\mathcal{P}(\mathcal{X})} \quad & \mathbb{E}_{\mathbf{x}\sim p}\left[f_{\mathbf{s}}(\mathbf{x})\right], \\ \text{s.t.} \quad & g_{\mathbf{s},i}(\mathbf{x}) \leqslant 0, \quad i = 1, \ldots, m, \quad p\text{-a.s.} \\ & h_{\mathbf{s},i}(\mathbf{x}) = 0, \quad i = 1, \ldots, n, \quad p\text{-a.s.} \end{aligned} \tag{20}$$

Algorithm 1 DIFLOWOPT

Require: Diffusion network ϵ_θ, dual network λ_ϕ, radii ρ_λ, ρ_s, step sizes $\eta_{\text{imp}}, \eta_{\text{sd}}, \eta_\phi$, rounds R, steps $N_{\text{imp}}, N_{\text{sd}}$.

for $k = 1, 2, \ldots$ **do**
  for $r = 1, \ldots, R$ **do**
    $\bar{\theta} \leftarrow \theta$; sample $\mathcal{D} = \{x_0\}$, $x_0 \sim p_{\bar{\theta},0}$
    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_{\text{dual}}(\phi)$ ▷ Dual update, Equation 23
    Repeat N_{imp} times: sample $x_0 \sim \mathcal{D}$, $\epsilon \sim \mathcal{N}(0, I)$, t; set $x_t = \alpha_t x_0 + \sigma_t \epsilon$; compute $\hat{g}_t$ by Equation 18; $\theta \leftarrow \theta - \eta_{\text{imp}} \nabla_\theta \mathcal{L}_{\text{primal}}(\theta)$ ▷ Primal update, Equation 22
  end for
  $\tilde{\theta} \leftarrow \theta$
  Repeat N_{sd} times: sample $\tilde{x}_0 \sim p_{\tilde{\theta},0}$, $\epsilon \sim \mathcal{N}(0, I)$, t; set $\tilde{x}_t = \alpha_t \tilde{x}_0 + \sigma_t \epsilon$; $\theta \leftarrow \theta - \eta_{\text{sd}} \nabla_\theta \left\| \epsilon_\theta(s, \tilde{x}_t, t) - \epsilon \right\|_2^2$ ▷ Self-distillation
end for

The two problems share the same optimal value: with $\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : g_i(\mathbf{x}) \leqslant 0, i = 1, \ldots, m\}$ the feasible set, any optimal solution $\mathbf{x}^*$ of equation 1 induces an optimal solution $p^* = \delta_{\mathbf{x}^*}$ of equation 20 with $\mathbb{E}_{\mathbf{x} \sim \delta_{\mathbf{x}^*}}[f(\mathbf{x})] = f(\mathbf{x}^*)$, and conversely any optimal p^* of equation 20 satisfies $\text{supp}(p^*) \subseteq \arg\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x})$.

We consider the relaxation to replace the pointwise constraints with the relaxed constraint $\mathbb{E}_{\mathbf{x} \sim p}\left[[g_i(\mathbf{x})]_+\right] \leqslant \delta$ and $\mathbb{E}_{\mathbf{x} \sim p}\left[|h_i(\mathbf{x})|\right] \leqslant \delta$. When $\delta = 0$, the feasible set is strictly equivalent. In practice, one can choose a small δ or set a schedule of $\delta \to 0$. With the relaxed constraints, the Lagrangian is

$$\min_{p \in \mathcal{P}(\mathcal{X})} \max_{\lambda^g \geqslant 0, \lambda^h} \mathbb{E}_{\mathbf{x} \sim p}\left[f_{\mathbf{s}}(\mathbf{x}) + \sum_{i=1}^{m} \lambda^g_{\mathbf{s},i} \left([g_{\mathbf{s},i}(\mathbf{x})]_+ - \delta \right) + \sum_{i=1}^{n} \lambda^h_{\mathbf{s},i} \left(|h_{\mathbf{s},i}(\mathbf{x})| - \delta \right) \right], \tag{21}$$

which is bilinear w.r.t p and λ.

Gradient Descent with Diffusion Model Taking the forward Euler discretization of the gradient flow, and using the relationship between the ϵ and score parametrization in diffusion models $s(\mathbf{x}_t, t) = -\frac{\epsilon(\mathbf{x}_t, t)}{\sigma_t}$, we arrive at the following diffusion model training loss

$$\mathcal{L}_{\text{primal}}(\theta) = \mathbb{E}_{\mathbf{s},\mathbf{x}}\left[\left\| -\frac{\epsilon_\theta(\mathbf{s}, \mathbf{x}_t, t)}{\sigma_t} - \text{sg}\left(-\frac{\epsilon_{\bar{\theta}}(\mathbf{s}, \mathbf{x}_t, t)}{\sigma_t} + \rho_s \hat{g}_{\tau,t} \right) \right\|_2^2, \right] \tag{22}$$

where sg denotes the stop gradient operator, ρ_s is the stepsize and $\hat{g}_{\tau,t}$ is the time derivative of perturbed score estimated from Equation 18.

Gradient Ascent with Neural Network We model $\lambda(\mathbf{s})$ with a neural network $\lambda^\phi(\mathbf{s})$ and train it with the following loss:

$$\mathcal{L}_{\text{dual}}(\phi) = \mathbb{E}_{\mathbf{s},\mathbf{x}}\left[\sum_{i=1}^{m} \left(\lambda_i^{g,\phi}(\mathbf{s}) - \text{sg}\left(\lambda_i^{g,\text{tgt}} \right) \right)^2 + \sum_{i=1}^{n} \left(\lambda_i^{h,\phi}(\mathbf{s}) - \text{sg}\left(\lambda_i^{h,\text{tgt}} \right) \right)^2 \right], \tag{23}$$

with

$$\lambda_i^{g,\text{tgt}} = \lambda_i^{g,\phi}(\mathbf{s}) + \eta\left([g_i(\mathbf{s}, \mathbf{x})]_+ - \delta \right), \quad \lambda_i^{h,\text{tgt}} = \lambda_i^{h,\phi}(\mathbf{s}) + \eta\left(|h_i(\mathbf{s}, \mathbf{x})| - \delta \right) \tag{24}$$

followed by projecting $\lambda^{g,\phi}(\mathbf{s})$ onto $\mathbb{R}^m_+$.

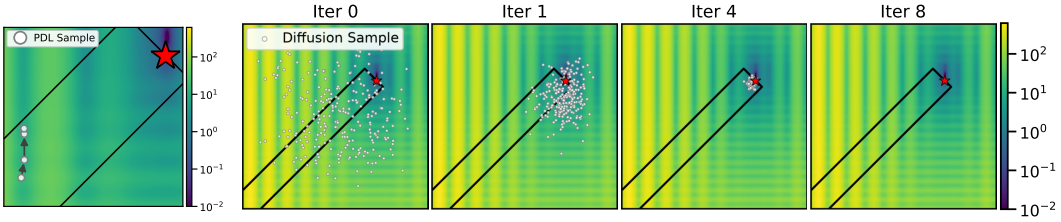
Self-Supervised Primal-Dual Training Building upon these components, we present the algorithm of self-supervised primal-dual training for amortized optimization (Algorithm 1). Each iteration, the algorithm performs the primal update and the dual update alternatively. A self-distillation step is performed at the end of each iteration to make sure the learned diffusion model remain a valid standard diffusion model, which is important to ensure that the $\hat{g}_t$ estimation is accurate.

5 EXPERIMENTS

We evaluate DIFLOWOPT on a variety of nonconvex constrained optimization problems, including standard benchmarks and real-world problems from control and materials science. We compare against (i) classical local nonlinear programming solvers, including SLSQP (Kraft, 1988), the primal-dual interior point method IPOPT (Wächter & Biegler, 2006) and its multi-start version (MS-IPOPT); (ii) the population-based global optimization method Differential Evolution (DE) (Storn & Price, 1997); (iii) PDL (Park & Van Hentenryck, 2023), a self-supervised amortized optimization method that represents the solution with a single deterministic point and learns it with primal-dual training; and (iv) DiOpt (Ding et al., 2025), a diffusion-based self-supervised method for learning amortized solvers for constrained optimization. For a fair comparison, all self-supervised methods are trained from scratch without access to a pre-collected dataset. Ablation studies on the choice of gradient flow and constraint handling, as well as problem formulations and implementation details, are provided in Section C.

5.1 TOY EXAMPLE

To illustrate why our diffusion-based approach achieves better convergence to global optimality on nonconvex problems, we present a 2D toy example based on the Levy function, whose rugged landscape with multiple basins is challenging for purely gradient-based methods. As shown in Figure 2, DIFLOWOPT explores multiple basins simultaneously and converges to the global optimum, while PDL gets trapped in the basin it is initialized in.



(a) PDL. (b) DiFlowOpt (ours).

Figure 2: Trajectories of the predicted solutions during training on the 2D Levy problem. PDL descends within a single basin and stalls at the local optimum near its initialization. DIFLOWOPT explores multiple basins in parallel and converges to the global optimum (red star).

5.2 STANDARD OPTIMIZATION PROBLEMS

We evaluate DIFLOWOPT on three standard benchmark problems: 100D Levy, 100D Schwefel, and 20D Hartmann, all highly nonconvex and high-dimensional. The landscapes of Levy and Schwefel are rugged, with numerous local minima whose number grows rapidly with the problem dimension, while Hartmann is multimodal with modes of largely different basin sizes and depths. We apply randomly sampled linear inequality constraints whose number equals the problem dimension.

As shown in Table 1, DIFLOWOPT achieves the best overall objective value with negligible constraint violation. We also report the average inference time per test instance, evaluated with batch size=1 to reflect real latency. Even so, the inference time of DIFLOWOPT is orders of magnitude smaller than that of the classical solvers, and its throughput can be further increased by hundreds of times through batching. The gradient-based solvers SLSQP and IPOPT tend to converge to local optima and fail to find the globally optimal basin, and multistart does not scale to high dimensions, as the number of optima can grow exponentially with the problem dimension. DE is effective when the dimension is relatively small, as in Hartmann, but does not scale since it relies on zeroth-order information only. PDL shows a similar failure mode to SLSQP and IPOPT, easily getting trapped in a poor local basin that gradient-based local updates cannot escape, whereas DIFLOWOPT finds better solutions through its parallel exploration of multiple basins. DiOpt represents the solution with a diffusion model and enables global exploration, but its update is purely zeroth-order and thus hard to scale to high dimensions.

Table 1: Benchmarking on standard high-dimensional nonconvex optimization problems. Our method achieves the best overall solution quality with orders of magnitude lower inference time than classical solvers.

	Levy			Schwefel			Hartmann		
Method	**Obj.↓**	**Viol.↓**	**Time(s)↓**	**Obj.↓**	**Viol.↓**	**Time(s)↓**	**Obj.↓**	**Viol.↓**	**Time(s)↓**
Random	3865.141	87.413	–	415.891	6.276	–	−0.018	0.016	–
Classical Solvers									
SLSQP	3001.929	0.000	0.653	200.658	0.000	0.095	−1.522	0.000	0.009
IPOPT	2089.560	0.000	7.259	225.369	0.000	0.328	−1.211	0.000	0.427
MS-IPOPT	483.337	0.000	66.130	192.816	0.000	2.225	**−1.903**	0.000	4.506
DE	2460.519	0.036	7.432	324.817	0.000	7.223	−1.531	0.000	0.489
Learning-based Methods									
PDL	516.255	0.000	0.000*	394.617	0.000	0.000	−1.213	0.000	0.000
DiOpt	4084.261	0.949	0.002	417.735	0.517	0.002	−1.151	0.000	0.002
DiFlowOpt (ours)	**10.533**	0.000	0.002	**150.558**	0.008	0.002	**−1.877**	0.000	0.002

* Inference time of PDL is ~ 0.05 ms.

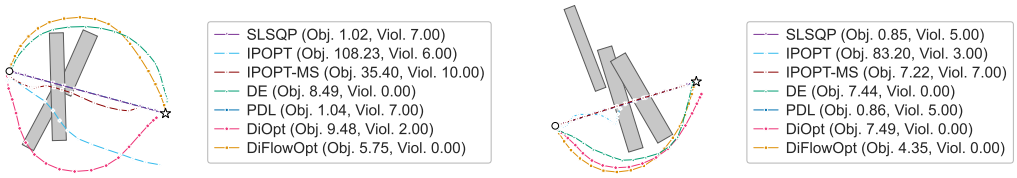


Figure 3: Predicted trajectories. DIFLOWOPT obtains the lowest cost among methods that are collision-free in most cases.

5.3 REAL-WORLD OPTIMIZATION PROBLEMS

We further evaluate on two real-world problems. Morse cluster optimization seeks the atom configuration minimizing the potential energy of an interacting system. Its landscape is rugged with numerous local optima and gradients spanning a wide range of magnitudes. As shown in Table 2, DIFLOWOPT achieves the best objective with small violation, while the wide gradient range makes gradient-based methods unstable or ineffective. Trajectory optimization requires a vehicle to reach its target while avoiding obstacles, with multiple potential paths making the problem multimodal. DIFLOWOPT achieves the best combination of objective and violation (Table 2, Figure 3). DiOpt has close performance, while PDL attains a much lower objective but substantially violates the constraints, since the indicator-function obstacle constraints have uninformative gradients.

Table 2: Results on real-world optimization problems. DIFLOWOPT shows strong capability in optimizing nonconvex problems with rugged landscapes and complex constraints.

		Classical Solvers				*Learning-based Methods*		
Problem	**Metric**	SLSQP	IPOPT	MS-IPOPT	DE	PDL	DiOpt	**DiFlowOpt (ours)**
Morse	Obj.↓	2357412.249	1010646.063	72.383	2409.038	1072.869	12288505.667	**−5.241**
	Viol.↓	4.486	0.036	0.002	2.155	0.502	0.266	0.031
	Time(s)↓	1.133	2.832	27.642	7.126	0.000	0.002	0.002
Trajectory	Obj.↓	1.125	68.249	9.080	9.107	1.264	6.699	5.907
	Viol.↓	3.571	5.633	5.067	0.000	5.241	0.840	0.805
	Time(s)↓	1.946	1.471	12.261	1.956	0.000	0.002	0.002

6 CONCLUSION

We introduce DIFLOWOPT, a diffusion-based self-supervised amortized optimization method for nonconvex constrained problems. Lifting the optimization to the probability measure space, we use gradient flows to derive a principled way of updating the diffusion model. DIFLOWOPT trains along the Wasserstein-Fisher-Rao gradient flow and naturally combines both first- and zeroth- order information. Constraints are handled by a primal-dual method, with instance-specific dual variables learned by a separate network. On high-dimensional, highly nonconvex benchmarks and real-world problems, DIFLOWOPT achieves improved solution quality over classical solvers and existing amortized methods, while requiring orders of magnitude less computation time than classical solvers.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing methods, identifying relevant literature, and polishing the writing. We have not used generative AI tools for formulating the methodology, algorithm or theoretical framework in this paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper focuses on self-supervised amortized optimization, which is at the intersection of machine learning and optimization. We develop a diffusion-based self-supervised amortized optimization method from the perspective of optimization over the probability measure space. The work does not involve human subjects, personal data, or sensitive information and raises no immediate concerns related to privacy, security, or fairness. We have not identified any potential conflicts of interest, legal compliance issues, or potential harmful applications from this research.

REPRODUCIBILITY STATEMENT

We provide detailed discussion of our experimental setup, including benchmark optimization problems, baselines and evaluation protocols in Section 5 and Section C. The code for reproducing our experiments will be released upon acceptance. The proofs for all the propositions in our work can be found in Section B.

REFERENCES

Brandon Amos. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning*, 16(5):592–732, 2023.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 51304–51323, 2024.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Tzuu-Shuh Chiang, Chii-Ruey Hwang, and Shuenn Jyi Sheu. Diffusion for global optimization in r^n. *SIAM Journal on Control and Optimization*, 25(3):737–753, 1987.

Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. An interpolating distance between optimal transport and fisher–rao metrics. *Foundations of Computational Mathematics*, 18(1):1–44, 2018.

Jacob K Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models. *Advances in Neural Information Processing Systems*, 37:89307–89333, 2024.

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*, 2022.

Kevin Clark, Paul Vicol, Kevin Swersky, and David Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, volume 2024, pp. 4793–4822, 2024.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Shutong Ding, Yimiao Zhou, Ke Hu, Xi Yao, Junchi Yan, Xiaoying Tang, and Ye Shi. Diffusion-based learning framework for constrained nonconvex optimization with weighted bootstrapped refinement. *arXiv preprint arXiv:2502.10330*, 2025.

Carles Domingo i Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, volume 2025, pp. 53791–53846, 2025.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

Jonathan PK Doye, Robert H Leary, Marco Locatelli, and Fabio Schoen. Global optimization of morse clusters by potential energy transformations. *INFORMS Journal on Computing*, 16(4): 371–379, 2004.

Alex Dytso, H Vincent Poor, and Shlomo Shamai Shitz. Conditional mean estimation in gaussian noise: A meta derivative identity with applications. *IEEE Transactions on Information Theory*, 69(3):1883–1898, 2022.

Alex Dytso, Martina Cardone, and Ian Zieder. Meta derivative identity for the conditional expectation. *IEEE Transactions on Information Theory*, 69(7):4284–4302, 2023.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in neural information processing systems*, 36:79858–79885, 2023.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, et al. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

Stuart Geman and Chii-Ruey Hwang. Diffusions for global optimization. *SIAM Journal on Control and Optimization*, 24(5):1031–1043, 1986.

Samar Hadou, Yigit Berkay Uslu, and Alejandro Ribeiro. Constrained diffusion models with primal-dual inference. *arXiv preprint arXiv:2606.17192*, 2026.

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

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

Ya-Ping Hsieh, Chen Liu, and Volkan Cevher. Finding mixed nash equilibria of generative adversarial networks. In *International Conference on Machine Learning*, pp. 2810–2819. PMLR, 2019.

William Huang, Yifeng Jiang, Tom Van Wouwe, and C Karen Liu. Constrained diffusion with trust sampling. *Advances in Neural Information Processing Systems*, 37:93849–93873, 2024.

Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International conference on machine learning*, pp. 4880–4889. PMLR, 2020.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

Shervin Khalafi, Dongsheng Ding, and Alejandro Ribeiro. Constrained diffusion models via dual training. *Advances in Neural Information Processing Systems*, 37:26543–26576, 2024.

Lingkai Kong, Yuanqi Du, Wenhao Mu, Kirill Neklyudov, Valentin De Bortoli, Dongxia Wu, Haorui Wang, Aaron Ferber, Yi-An Ma, Carla P Gomes, et al. Diffusion models as constrained samplers for optimization with unknown constraints. *arXiv preprint arXiv:2402.18012*, 2024.

Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

Pedro Larranaga and Concha Bielza. Estimation of distribution algorithms. In *Handbook of Heuristics*, pp. 583–598. Springer, 2025.

Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

Anjian Li, Zihan Ding, Adji Bousso Dieng, and Ryne Beeson. Diffusolve: Diffusion-based solver for non-convex trajectory optimization. *arXiv preprint arXiv:2403.05571*, 2024.

Anjian Li, Bartolomeo Stellato, and Ryne Beeson. Glens: Global search via learning from solver iterates with diffusion models. *arXiv preprint arXiv:2606.00366*, 2026a.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. *Advances in Neural Information Processing Systems*, 38:127168–127196, 2026b.

Matthias Liero, Alexander Mielke, and Giuseppe Savaré. Optimal transport in competition with reaction: The hellinger–kantorovich distance and geodesic curves. *SIAM Journal on Mathematical Analysis*, 48(4):2869–2911, 2016.

Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International conference on machine learning*, pp. 6083–6093. PMLR, 2020.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Guan-Horng Liu, Tianrong Chen, Evangelos Theodorou, and Molei Tao. Mirror diffusion models for constrained and watermarked generation. *Advances in Neural Information Processing Systems*, 36:42898–42917, 2023a.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026.

Linshan Liu, Mateusz B Majka, and Łukasz Szpruch. Polyak–łojasiewicz inequality on the space of measures and convergence of mean-field birth-death processes. *Applied Mathematics & Optimization*, 87(3):48, 2023b.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Aaron Lou and Stefano Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pp. 22675–22701. PMLR, 2023.

Lukas Lüken and Sergio Lucia. Self-supervised learning of iterative solvers for constrained optimization. *Results in Control and Optimization*, pp. 100751, 2026.

Rahul Nellikkath and Spyros Chatzivasileiadis. Physics-informed neural networks for ac optimal power flow. *Electric Power Systems Research*, 212:108412, 2022.

Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Model-based diffusion for trajectory optimization. *Advances in Neural Information Processing Systems*, 37:57914–57943, 2024.

Seonho Park and Pascal Van Hentenryck. Self-supervised primal-dual learning for constrained optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4052–4060, 2023.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Yinuo Ren, Tesi Xiao, Tanmay Gangwani, Anshuka Rangi, Holakou Rahmanian, Lexing Ying, and Subhajit Sanyal. Multi-objective optimization via wasserstein-fisher-rao gradient flow. In *International Conference on Artificial Intelligence and Statistics*, pp. 3862–3870. PMLR, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. Learning to warm-start fixed-point optimization algorithms. *Journal of Machine Learning Research*, 25(166):1–46, 2024.

Augustinos D Saravanos, Oswin So, HM Sabbir Ahmad, and Chuchu Fan. Learning-to-optimize via deep unfolded flows. In *Forty-third International Conference on Machine Learning*.

Louis Sharrock and Christopher Nemeth. Tuning-free sampling via optimization on the space of probability measures. *arXiv preprint arXiv:2510.25315*, 2025.

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*, pp. 32483–32498. PMLR, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1): 25–57, 2006.

Yu Wang, Yang Li, Jiale Ma, Junchi Yan, and Yi Chang. Nexco: Native solution expansion for diffusion-based combinatorial optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Ahmed S Zamzam and Kyri Baker. Learning optimal solutions for extremely fast ac optimal power flow. In *2020 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)*, pp. 1–6. IEEE, 2020.

Hengtong Zhang and Tingyang Xu. Towards controllable diffusion models via reward-guided exploration. *arXiv preprint arXiv:2304.07132*, 2023.

Runyu Zhang, Jiawei Zhang, Gioele Zardini, Saurabh Amin, and Asuman Ozdaglar. Denoising as projection: Constrained optimization with gradient-guided diffusion. *arXiv preprint arXiv:2608.29507*, 2026.

A Related Work

Amortized Optimization Amortized optimization approaches use learning to leverage the shared structure across related problem instances, so as to accelerate the solving process and improve solution quality (Amos, 2023). Methods in amortized optimization can be broadly categorized into two classes. *Fully-amortized* methods learn a solution mapping from the problem parameters to the solutions and assume no access to the objective function during inference. This mapping can be trained via supervised learning on a pre-collected dataset of problem-solution pairs (Zamzam & Baker, 2020; Nellikkath & Chatzivasileiadis, 2022). While conceptually simple, this approach requires collecting a large amount of data for generalization, which can be computationally expensive. Self-supervised learning instead learns from the objective function, eliminating the need for pre-collected solutions (Park & Van Hentenryck, 2023; Donti et al., 2021; Lüken & Lucia, 2026). *Semi-amortized* optimization methods, on the other hand, access the objective function during inference. Representative examples in this class include learning the update rule for an optimizer (Li & Malik, 2016; Andrychowicz et al., 2016) and learning warm-start initial guesses for a refinement solver (Sambharya et al., 2024). Our work falls primarily within the self-supervised, fully-amortized optimization class, and can be naturally extended to the semi-amortized setting for learning warm-start initializations.

Diffusion Fine-Tuning and Guidance A rich body of work studies fine-tuning diffusion models or guiding diffusion sampling at inference time to generate samples that optimize target metrics. Existing fine-tuning methods use techniques such as direct reward backpropagation (Xu et al., 2023; Clark et al., 2024), policy gradient (Fan et al., 2023; Black et al., 2024; Ren et al., 2025; Liu et al., 2026), weighted regression (Fan et al., 2023; Zhang & Xu, 2023; Kang et al., 2023; Ding et al., 2024), and stochastic optimal control (Domingo i Enrich et al., 2025). To prevent the learned distribution from diverging too far from the base distribution, these methods often introduce a KL-regularization term that yields the tilted target distribution $p(\mathbf{x}) \propto p^{\text{base}}(\mathbf{x}) \exp\left(r\left(\mathbf{x}\right)\right)$, where p^{base} is the base distribution and $r(\mathbf{x})$ is the reward function to be maximized. Instead of modifying model weights, diffusion guidance methods such as DPS (Chung et al., 2022), LGD-MC (Song et al., 2023) and UGD (Bansal et al., 2024) steer samples at inference time. A related line of work enforces constraints at inference time (Christopher et al., 2024; Huang et al., 2024) or trains diffusion models to sample from constrained distributions (Lou & Ermon, 2023; Liu et al., 2023a).

Diffusion Models for Optimization Diffusion models have been utilized to learn amortized solvers for optimization, with most efforts focusing on supervised training from classical solver solutions and a growing number of recent works exploring self-supervised training. Sun & Yang (2023) trains a diffusion model to generate high-quality solutions for combinatorial optimization problems by fitting a dataset of problem-solution pairs. Li et al. (2024) learns initial guesses for warm-starting numerical solvers in convex trajectory optimization problems from a pre-collected dataset. Subsequent works extend the supervised training approach to other optimization domains and propose new training strategies (Wang et al., 2026; Li et al., 2026b;a). Collecting solutions from classical solvers can be computationally expensive, especially for high-dimensional nonconvex problems, and several works train diffusion-based solvers without relying on such pre-collected solutions. However, these methods are typically limited to specific optimization objectives and constraint formulations, such as KL-divergence or KL-regularized objectives (Khalafi et al., 2024; Hadou et al., 2026). DiOpt (Ding et al., 2025) proposes a self-supervised weighted regression loss that supports general objective and constraint formulations, but requires additional supervised pretraining on a pre-collected dataset to perform well. Saravanos et al. learns a flow model as an iterative optimizer but applies only to unconstrained settings. Inference-time techniques for constrained optimization have also been studied in prior works (Kong et al., 2024; Pan et al., 2024; Zhang et al., 2026), which are orthogonal to our contribution.

B PROOFS

B.1 PROOF OF PROPOSITION 1

Proof By the definition of the perturbed score and Equation 3, we have

$$g_t(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \partial_\tau \log p_{\tau,t}(\mathbf{x}_t) \tag{25}$$

$$= \nabla_{\mathbf{x}_t} \frac{\partial_\tau p_{\tau,t}(\mathbf{x}_t)}{p_{\tau,t}(\mathbf{x}_t)} \tag{26}$$

$$= \nabla_{\mathbf{x}_t} \frac{\partial_\tau \int p_\tau(\mathbf{x}_0) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{27}$$

$$= \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p_\tau(\mathbf{x}_0) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)}, \tag{28}$$

where the first equality exchanges the order of differentiation in τ and $\mathbf{x}_t$, and the last one exchanges ∂_τ with the integral, since the forward kernel $q_t\left(\mathbf{x}_t|\mathbf{x}_0\right)$ does not depend on τ. ■

B.2 PROOF OF PROPOSITION 2

Proof With p_τ transported according to the Wasserstein gradient flow, we substitute equation 10 into the general form in Proposition 1 and obtain,

$$g_t^W(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p_\tau(\mathbf{x}_0) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{29}$$

$$= \nabla_{\mathbf{x}_t} \frac{\int \operatorname{div}\left(p_\tau(\mathbf{x}_0) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)\right) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \quad \text{(by equation 10)} \tag{30}$$

$$= -\nabla_{\mathbf{x}_t} \frac{\int p_\tau(\mathbf{x}_0) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \nabla_{\mathbf{x}_0} q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \quad \text{(integration by parts)} \tag{31}$$

$$= -\nabla_{\mathbf{x}_t} \frac{\int p_\tau(\mathbf{x}_0) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \frac{\alpha_t}{\sigma_t^2}\left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right) q_t\left(\mathbf{x}_t|\mathbf{x}_0\right) d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{32}$$

$$= -\frac{\alpha_t}{\sigma_t^2} \nabla_{\mathbf{x}_t} \int p_{\tau,t}\left(\mathbf{x}_0|\mathbf{x}_t\right) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right) d\mathbf{x}_0 \quad \text{(Bayes' theorem)} \tag{33}$$

$$= -\frac{\alpha_t}{\sigma_t^2} \nabla_{\mathbf{x}_t} \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)} \left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)\right]. \tag{34}$$

Using Theorem 1 in Dytso et al. (2023),

$$\nabla_{\mathbf{x}_t} \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)} \left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)\right] \tag{35}$$

$$= \operatorname{Cov}\left(\nabla_{\mathbf{x}_t} \log q_t\left(\mathbf{x}_t|\mathbf{x}_0\right), \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right) \mid \mathbf{x}_t\right) \tag{36}$$

$$+ \mathbb{E}\left[\nabla_{\mathbf{x}_t} \left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)\right] \mid \mathbf{x}_t\right] \tag{37}$$

$$= \operatorname{Cov}\left(-\frac{1}{\sigma_t^2}\left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right), \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right) \mid \mathbf{x}_t\right) + \mathbb{E}\left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right) \mid \mathbf{x}_t\right] \tag{38}$$

$$= \frac{\alpha_t}{\sigma_t^2} \operatorname{Cov}\left(\mathbf{x}_0, \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right) \mid \mathbf{x}_t\right) + \mathbb{E}\left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right) \mid \mathbf{x}_t\right] \tag{39}$$

$$= \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)} \left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right) + \frac{\alpha_t}{\sigma_t^2}\left(\mathbf{x}_0 - m_{\tau,t}\left(\mathbf{x}_t\right)\right) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)^\top \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)\right] \tag{40}$$

$$= \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)} \left[\nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right) + \frac{\alpha_t}{\sigma_t^2}\left(\mathbf{x}_0 - m_{\tau,t}\left(\mathbf{x}_t\right)\right) \left(\mathbf{x}_t - \alpha_t \mathbf{x}_0\right)^\top \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)\right]. \tag{41}$$

Therefore,

$$g_t^W(\mathbf{x}_t) = -\frac{\alpha_t}{\sigma_t^2} \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)} \left[W_{\tau,t}\left(\mathbf{x}_0, \mathbf{x}_t\right) \nabla_{\mathbf{x}_0} \mathcal{L}_\lambda\left(\mathbf{x}_0\right)\right]. \tag{42}$$

Here, $\mathcal{L}_\lambda$ plays the role of f, as in the primal variable optimization setting. ■

B.3 Proof of Proposition 3

Proof Substituting the Fisher-Rao gradient flow formulation equation 13 into the numerator of Proposition 1,

$$g^{\mathrm{FR}}_{\tau,t}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{43}$$

$$= \nabla_{\mathbf{x}_t} \frac{\int p_\tau(\mathbf{x}_0)\left(\bar{f}_\tau - f(\mathbf{x}_0)\right) q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{44}$$

$$= \nabla_{\mathbf{x}_t} \int \frac{p_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)}{p_{\tau,t}(\mathbf{x}_t)} \left(\bar{f}_\tau - f(\mathbf{x}_0)\right)\, d\mathbf{x}_0 \tag{45}$$

$$= \nabla_{\mathbf{x}_t} \int p_{\tau,t}\left(\mathbf{x}_0 \mid \mathbf{x}_t\right) \left(\bar{f}_\tau - f(\mathbf{x}_0)\right)\, d\mathbf{x}_0 \tag{46}$$

$$= \nabla_{\mathbf{x}_t} \left(\bar{f}_\tau - \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)}\left[f(\mathbf{x}_0)\right]\right) \tag{47}$$

$$= -\nabla_{\mathbf{x}_t} \mathbb{E}_{p_{\tau,t}(\mathbf{x}_0|\mathbf{x}_t)}\left[f(\mathbf{x}_0)\right]. \tag{48}$$

From Theorem 1 in Dytso et al. (2022), we obtain

$$g^{\mathrm{FR}}_{\tau,t}(\mathbf{x}_t) = -\frac{\alpha_t}{\sigma_t^2}\mathrm{Cov}\left(\mathbf{x}_0, f\left(\mathbf{x}_0\right) \mid \mathbf{x}_t\right) = -\frac{\alpha_t}{\sigma_t^2}\mathbb{E}\left[\left(\mathbf{x}_0 - m_{\tau,t}\left(\mathbf{x}_t\right)\right) f\left(\mathbf{x}_0\right)\right]. \tag{49}$$

■

B.4 Proof of Proposition 5

Proof Starting from the general form in Proposition 1 and substituting the linear decomposition of $\partial_\tau p_\tau$, we have

$$g_{\tau,t}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{50}$$

$$= \nabla_{\mathbf{x}_t} \frac{\int \left(a\, \partial_\tau p^1_\tau(\mathbf{x}_0) + b\, \partial_\tau p^2_\tau(\mathbf{x}_0)\right) q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{51}$$

$$= \nabla_{\mathbf{x}_t} \left[a \frac{\int \partial_\tau p^1_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} + b \frac{\int \partial_\tau p^2_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)}\right] \tag{52}$$

$$= a\, \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p^1_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} + b\, \nabla_{\mathbf{x}_t} \frac{\int \partial_\tau p^2_\tau(\mathbf{x}_0)\, q_t\left(\mathbf{x}_t \mid \mathbf{x}_0\right)\, d\mathbf{x}_0}{p_{\tau,t}(\mathbf{x}_t)} \tag{53}$$

$$= a\, g^1_{\tau,t}(\mathbf{x}_t) + b\, g^2_{\tau,t}(\mathbf{x}_t), \tag{54}$$

where the third and fourth equalities use the linearity of integration and of the gradient operator. ■

C Experiment Details and Additional Results

C.1 Ablation Studies

We conduct ablation studies on the choice of gradient flow and the constraint enforcement approach to analyze the individual components of our algorithm. In Table 3a, we compare our algorithm under the Wasserstein gradient flow (W), the Fisher-Rao gradient flow (FR), and the Wasserstein-Fisher-Rao gradient flow (WFR) on the Levy problem. WFR outperforms both variants by leveraging both zeroth-order information for global mode selection and first-order information for local refinement.

We further present ablations on constraint handling in Table 3b and Table 3c. In the unconstrained version, we remove all constraint handling from our algorithm. In the penalty version, we replace the learned dual variable with a fixed penalty term. The primal-dual update achieves better constraint satisfaction while preserving the objective landscape, particularly on the trajectory optimization problem with its more complex constraints.

Gradient Flow	Obj.↓	Viol.↓
W	29.342	0.000
FR	16.843	0.000
WFR	**10.533**	0.000

(a) Choice of gradient flow (Levy).

Method	Obj.↓	Viol.↓
Unconstrained	4.311	2.435
Penalty	11.336	0.000
Primal-dual	10.533	0.000

(b) Constraint handling (Levy).

Method	Obj.↓	Viol.↓
Unconstrained	1.524	5.156
Penalty	9.967	1.936
Primal-dual	5.907	0.805

(c) Constraint handling (TO).

Table 3: Ablation studies on the choice of gradient flow and constraint handling method. Experiments are conducted on the Levy problem. To further evaluate constraint handling under more complex constraints, we additionally test on the trajectory optimization (TO) problem.

C.2 Standard Optimization Problems

Levy2D In the toy example, we consider the optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sin^2(\pi w_1) + (w_1-1)^2\left(1+10\sin^2(\pi w_1+1)\right) + (w_2-1)^2\left(1+\sin^2(2\pi w_2)\right) \\ \text{where} \quad & w_i = 1+\frac{x_i-a-1}{4}, \qquad i=1,2, \\ \text{s.t.} \quad & x_1+x_2 \leqslant p, \\ & x_1-x_2 \leqslant b, \\ & x_2-x_1 \leqslant b. \end{aligned} \tag{55}$$

We set a and p as problem parameters that vary across instances, and the coefficient b is fixed at 4.0.

Levy This problem is an extension of the Levy2D problem,

$$\min_{x\in\mathbb{R}^d} \quad \sin^2(\pi w_1) + \sum_{i=1}^{d-1}\left[(w_i-1)^2\left(1+10\sin^2(\pi w_i+1)\right)\right] + (w_d-1)^2\left(1+\sin^2(2\pi w_d)\right) \tag{56}$$

$$\text{where} \quad w_i = 1+\frac{x_i-a-1}{4}, \qquad i=1,\dots,d. \tag{57}$$

$$\text{s.t.} \quad x_1+x_2 \leqslant p, \tag{58}$$

$$x_i - x_{i+1} \leqslant b, \quad i=1,\dots,d-1, \tag{59}$$

$$x_{i+1} - x_i \leqslant b, \quad i=1,\dots,d-1. \tag{60}$$

The problem parameters are the 2D vector $[a,p]$, with $a\sim\mathcal{U}[6.0,9.0]$ and $p\sim\mathcal{U}[10.0,20.0]$. We set $d=100$ and $b=10.0$.

Schwefel The Schwefel optimization problem is given as follows,

$$\min_{x\in\mathbb{R}^d} \quad c - \frac{1}{d}\sum_{i=1}^{d} z_i \sin\left(\sqrt{|z_i|}\right), \tag{61}$$

$$\text{where} \quad z_i = w(x_i-a_i), \qquad i=1,\dots,d, \tag{62}$$

$$\text{s.t.} \quad Ax-b \leqslant 0, \tag{63}$$

$$-10 \leqslant x_i \leqslant 10, \qquad i=1,\dots,d. \tag{64}$$

The problem parameters are the d-dimensional vector a, with $a\sim\mathcal{U}[-1.5,1.5]$. We set $d=100$, $c=418.9829$ and $w=50.0$. A is a randomly sampled $d\times d$ matrix and b is a randomly sampled d-dimensional vector.

Hartmann We also include the Hartmann optimization problem in the evaluation,

$$\min_{x\in\mathbb{R}^d} \quad -\sum_{i=1}^{M}\alpha_i \exp\left(-\sum_{j=1}^{d} W_{ij}\left(x_j - P_{ij}\right)^2\right), \tag{65}$$

$$\text{s.t.} \quad Ax - b \leqslant 0. \tag{66}$$

The problem parameters are the $M \times d$ matrix P, with $P_{ij} \sim \mathcal{U}[-1.5, 1.5]$. We set $d = 20$ and $M = 12$. The coefficients α, W, A and b are randomly sampled once and remain fixed.

C.3 Real World Experiments

Morse Cluster Optimization The Morse cluster optimization problem Doye et al. (2004) has the form

$$\min_{x_1,\ldots,x_{d-d^{\text{fixed}}}\in\mathbb{R}^3} \quad \sum_{1\leqslant i<j\leqslant d}\left[\left(\exp\left(\rho\left(1-\|x_i - x_j\|_2\right)\right)-1\right)^2 - 1\right] \tag{67}$$

$$\text{s.t.} \quad \|x_i\|_2 \leqslant R, \qquad i = 1,\ldots,d-d^{\text{fixed}}, \tag{68}$$

where $x_i \in \mathbb{R}^3$ is the coordinate vector of the i-th atom. The last d^{fixed} atoms have fixed positions, while the first $d - d^{\text{fixed}}$ atoms can move within the feasible space. The problem parameter is the randomly sampled positions x^{fixed} of the fixed atoms. We set $d = 40$, $d^{\text{fixed}} = 10$, $\rho = 10.0$ and $R = 3.0$.

Trajectory Optimization The trajectory optimization problem has the form

$$\min_{\bar{U}} J(x_0, \bar{U}) \tag{69}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k) \tag{70}$$

$$g(x_k) \leqslant 0. \tag{71}$$

where x_k is the state, u_k is the control at time k. The goal here is to find the optimal control sequence $\bar{U} = [u_0; \ldots; u_{T-1}]$ over a finite horizon T to minimize the objective $J(x_0, \bar{U})$ subject to the dynamics $x_{k+1} = f(x_k, u_k)$ and additional state constraints $b(x_k) \leqslant 0$.

In our setup, we assume 2D double integrator dynamics with discretization step 20. The cost is of the form

$$J(\bar{X}, \bar{U}) = \sum_{k=0}^{T-1} \ell_k(x_k, u_k) + \phi(x_T) \tag{72}$$

where $\ell_k(\cdot)$ is the running cost and $\phi(\cdot)$ is the terminal cost. Each problem instance contains 3 obstacles with random positions and rotations. The obstacle avoidance constraints are given by indicator functions of whether a trajectory point collides with an obstacle.

C.4 Baselines

We evaluate two classes of baselines: classical solvers, which optimize each problem instance from scratch, and prior self-supervised amortized optimization methods, which are learning-based methods that

Local nonlinear programming solvers. We include two widely used gradient-based local solvers. Sequential Least Squares Programming (SLSQP) (Kraft, 1988) is a sequential quadratic programming method that solves a sequence of least-squares subproblems built from a quasi-Newton approximation of the Lagrangian Hessian. IPOPT (Wächter & Biegler, 2006) is a primal–dual interior point method that handles inequality constraints through a logarithmic barrier and is the de facto standard for large-scale nonlinear programming. Since both methods converge to a local optimum that depends on the starting point, we also report a multi-start variant of IPOPT (MS-IPOPT), which runs the solver from 10 initial points drawn uniformly from the domain and returns the best feasible solution found.

Differential Evolution (DE). DE (Storn & Price, 1997) is a derivative-free global optimization method that maintains a population of candidate solutions. At each generation, a trial vector is formed for every member by adding a scaled difference of two other members to a third (mutation) and mixing it with the current member (crossover). The trial replaces the current member if it obtains a better fitness. Constraint violations are handled through a penalty term added to the objective.

PDL. Primal–Dual Learning (PDL) (Park & Van Hentenryck, 2023) is a self-supervised amortized method that trains a primal network, mapping problem parameters to a solution, jointly with a dual network that predicts the corresponding Lagrange multipliers. The two networks are updated alternately without access to ground-truth solutions. PDL is deterministic and returns a single solution per instance.

DiOpt. DiOpt (Ding et al., 2025) is a diffusion-based self-supervised method that learns a conditional diffusion model over solutions. Rather than regressing to labeled optima, it iteratively improves its own samples using objective and constraint-violation feedback, concentrating the learned distribution on feasible low-objective regions.

C.5 Implementation Details

All learning-based methods are trained on a single NVIDIA L40S (46 GB) with an Intel Xeon Gold 6548Y+ host CPU. The classical solvers run on the CPU through `cyipopt` 1.7 (IPOPT) and SciPy 1.17 (SLSQP, differential evolution). Every learning-based method is trained and evaluated on the same three random seeds $\{0, 1, 2\}$, and we report the mean objective value and violation over seeds. For a fair comparison, we use the standard Lagrangian formulation for all primal-dual based learning methods. We set diffusion step to be 10 for both diffusion-based methods DiOpt and DiFlowOpt and keep the network architecture and size the same. We fix the hidden layer number to be 3 for all the methods. Both diffusion-based methods, DiOpt and DiFlowOpt, use 10 diffusion steps and the same network architecture and size, and all learning-based methods use 3 hidden layers. The two diffusion methods share the same conditional denoiser: an MLP with Mish activations, conditioned on the problem state and a 32-dimensional sinusoidal embedding of the diffusion time. It predicts the noise ϵ under a variance-preserving schedule ($\beta_{\min} = 0.1$, $\beta_{\max} = 10$), and samples are drawn with the standard 10-step ancestral sampler. All classical solvers are run with a maximum of 200 iterations and uniformly drawn initial points. All self-supervised methods are trained from scratch without access to a pre-collected dataset, and no best-of-n sampling is used at inference of a trained network.

C.6 Other Experimental Results

We report the complete random seed results for the learning-based algorithms in the standard optimization problems in Table 4.

Table 4: Experimental results on standard optimization problems.

	Levy		Schwefel		Hartmann	
	Obj.	Viol.	Obj.	Viol.	Obj.	Viol.
PDL	516.255 ± 14.358	0.000 ± 0.000	394.617 ± 0.208	0.000 ± 0.000	-1.213 ± 0.000	0.000 ± 0.000
DiOpt	4084.261 ± 176.642	0.949 ± 0.852	417.735 ± 0.324	0.517 ± 0.020	-1.151 ± 0.006	0.000 ± 0.000
DiFlowOpt	$\mathbf{10.533 \pm 0.885}$	0.000 ± 0.000	$\mathbf{150.558 \pm 14.773}$	0.008 ± 0.004	$\mathbf{-1.877 \pm 0.000}$	0.000 ± 0.000