REWARD-GUIDED DIFFUSION MODEL FOR DATA-DRIVEN BLACK-BOX DESIGN OPTIMIZATION

Hadi Keramati *
University of British Columbia

Rajeev Jaiman
University of British Columbia

ABSTRACT

Black-box optimization (BBO) is an important method for design space exploration in high-dimensional domains, including fields such as materials science and robotic design. The diffusion models used for BBO either require a differentiable proxy or lack direct guidance. In this paper, we propose a reward-guided approach for training the Markov decision process (MDP) to increase the likelihood that the posterior generates higher-reward samples. We use the Metropolis–Hastings (MH) algorithm for Markov Chain Monte Carlo (MCMC) sampling to guide the reverse process. We first pre-train the diffusion model to match the distribution of the initial data, then fine-tune it so that the model acts as a policy that adapts its parameters to generate high-reward samples. This is a policy gradient method in which the policy is sampled from a pre-trained model to reduce the variance in training. Our experiments demonstrate that the reward-guided diffusion model achieves state-of-the-art performance across a variety of design problems, particularly in problems where the oracle is non-differentiable or an exact function.

1 INTRODUCTION

Design optimization in high-dimensional space is crucial in engineering and scientific problems. In real-world scenarios (e.g. biology, materials science, robotics, and hardware design), a well-defined function for the objective value is not always present Kumar & Levine (2020); Baque et al. (2018). In some cases, an objective function exists, but its evaluation is computationally expensive. Keramati et al. (2022). Therefore, utilizing precomputed objective function values for a given dataset is of significant interest in optimization. As a result, data-driven, black-box optimization is gaining increasing attention in both practical applications and scientific research. One way of optimizing a function using a dataset is to train a proxy to predict the performance function (f(x)) and use that proxy for online optimization. This forward approach requires the proxy to generalize outside the offline dataset, which can cause out-of-distribution problems Trabucco et al. (2021).

To address this challenge, inversion networks are used for data-driven optimization to learn a map from high-performance designs to the input and use this inverse mapping to generate the optimized design Kumar & Levine (2020). This mapping lacks direct guidance from the objective function and suffers from reaching the proximity of the optimized points Chen et al. (2024); Kumar & Levine (2020); Krishnamoorthy et al. (2023). A proxy refinement method has recently been suggested Chen et al. (2024) to provide robust guidance that optimizes sampling parameters as a trade-off between exploration and conditional generation for BBO. Many existing diffusion-based generative methods require a pre-trained, differentiable surrogate model (e.g., a neural network) to guide the training process Krishnamoorthy et al. (2023). As a result, common ensemble approaches, such as XGBoost or other tree-based methods, cannot be integrated as guiding proxies for diffusion models. This restriction narrows the selection of proxy models, thus limiting accuracy and generalization and ultimately increasing the likelihood of out-of-distribution errors. Consequently, it becomes challenging to address practical optimization problems that might benefit from the strengths of non-differentiable ensemble methods.

Fine-tuning denoising diffusion probabilistic models (DDPMs) using reinforcement learning (RL) has been proposed to improve sampling Fan & Lee (2023). Instead of following the backward

*E-mail: hadi.keramati@ubc.ca

process, minimizing the integral probability metric (IPM) with a policy gradient has shown effective fine-tuning results. Recently, a framework is suggested that uses RL to optimize text-to-image diffusion models Black et al. (2023). They consider the pre-trained diffusion model as a policy, similar to our method for conditional image generation.

In this paper, we formulate the fine-tuning as a policy learning of the Markov chain that draws intermediate high-reward samples from a pretrained denoising diffusion model. We do not perform any derivation from the objective function with respect to the design vector. Therefore, our method does not require any proxy training and can be used for any real-world performance evaluation and gradient-free simulation. We use Metropolis-Hastings (MH) for intermediate sampling along the Markov chain to sample high-rewarded design vectors. We show that Markov Chain Monte Carlo (MCMC) sampling from pre-trained diffusion models can serve as a low-variance policy training mechanism, and we also show its effectiveness as a technique for black-box optimization tasks.

2 BACKGROUND

This section briefly explains the DDPM to build a foundation to explain the reward-guided fine-tuning in the next section. DDPM is a latent variable generative model in which a forward diffusion process gradually adds noise to the data with the same dimensionality Ho et al. (2020). A learned reverse process, modeled as a Markov chain, iteratively removes the noise to reconstruct high-quality samples.

2.1 FORWARD DIFFUSION PROCESS

In the DDPM forward process, at each timestep $t \in \{1, 2, \dots, T\}$, where T is the final timestep in the forward process, Gaussian noise with variance schedule β_t is added to the input data $\mathbf{x}_0$ at each step of the fixed Markov chain. The transition distribution q from step $t-1$ to step t is formulated as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{1}$$

Where $\alpha_t = 1 - \beta_t$. The distribution at timestep t is defined as follows.

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}) \tag{2}$$

Where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, and $\mathbf{I}$ is the identity matrix Ho et al. (2020).

2.2 REVERSE DENOISING PROCESS

At timestep T, the latent $\mathbf{x}_T$ is close to the isotropic Gaussian distribution. The reverse process chain is designed to recover $\mathbf{x}_0$ from $\mathbf{x}_T$ by estimating the noise added at each timestep. This estimation is performed by a neural network, p_θ, on the mean and variance assuming the noise is Gaussian similar to the forward process.

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \beta_t \mathbf{I}), \tag{3}$$

Where $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ is a learnable estimator for the mean of $\mathbf{x}_{t-1}$ that only depends on $\mathbf{x}_t$ at timestep t.

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t\, \mathbf{z} \tag{4}$$

Here, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ are the neural network estimation of the noise, and σ_t is the variance of the added noise during sampling. By sampling from $\mathbf{x}_0$, new sets of data similar to the initial data are generated Ho et al. (2020).

2.3 DDPM TRAINING OBJECTIVE

The DDPM model in simple terms is trained by minimizing the loss term which is defined by the distance between the predicted noise and the true noise Ho et al. (2020):

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \|^2 \right] \tag{5}$$

2.4 MARKOV DECISION PROCESSES

A Markov decision process (MDP) is a stochastic decision-making process, $\mathcal{M} = (S, A, P, R)$, in which S is the state space, A is the action space, P is a transition kernel ($P : S \times A \to \mathcal{P}(S)$), and R is the reward function. At each time step t, the agent takes an action $a_t \in A$, based on the observation of a state $s_t \in S$, receives a reward $R(s_t, a_t)$, and transitions to a new state $s_{t+1} \sim P\big(s_{t+1} \mid s_t, a_t\big)$. Each action is drawn from a policy $\pi(a_t \mid s_t)$. A trajectory, $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$, is defined as the sequence of states and actions in MDP, where s_0 is drawn from an initial state distribution ρ_0. The objective is for the agent to maximize the expected cumulative rewards $J(\pi)$:

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T} R\big(\tau\big) \right].$$

3 METHODOLOGY

We first pre-train the DDPM using the data without any condition or guidance. Then we redefine the fine-tuning as a policy optimization but with sampling from the pre-trained distribution instead of direct policy optimization.

POLICY GRADIENT FOR REWARD GUIDANCE

We attempt to maximize the expected reward using a policy $p_\theta(\mathbf{x})$ from which we sample $\mathbf{x}$:

$$J(\theta) \;=\; \mathbb{E}_{\mathbf{x} \sim p_\theta}\big[r(\mathbf{x})\big] \tag{6}$$

We also rely on vanilla policy gradient to conceptualize the objective function and update in the Markov chain.

$$\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\mathbf{x} \sim p_\theta}\Big[r(\mathbf{x})\, \nabla_\theta \log p_\theta(\mathbf{x})\Big] \tag{7}$$

In diffusion models, we define reverse transitions $p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t)$

$$\log p_\theta(\mathbf{x}_0) \;=\; \sum_{t=1}^{T} \log p_\theta\big(\mathbf{x}_{t-1} \mid \mathbf{x}_t\big) \;+\; \log p(\mathbf{x}_T) \tag{8}$$

Where $\mathbf{x}_T$ typically comes from timestep T (e.g. a Gaussian). Therefore:

$$\nabla_\theta \log p_\theta(\mathbf{x}_0) \;=\; \sum_{t=1}^{T} \nabla_\theta \log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \tag{9}$$

We can express $\log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ in a differentiable form similar to DDPM.

$$\nabla_\theta \big[-\log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \big] \;\propto\; \nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right\|^2 \tag{10}$$

Thus, minimizing the MSE between the true noise $\boldsymbol{\epsilon}$ and the predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ corresponds to maximizing $\log p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.

$$\nabla_\theta \, \mathrm{Loss}_{\mathrm{RLdiff}} \;\propto\; -\,\mathbb{E}_{\mathbf{x}_0 \sim p_\theta}\big[r(\mathbf{x}_0)\, \nabla_\theta \log p_\theta(\mathbf{x}_0)\big] \tag{11}$$

Equation 11 matches the standard policy gradient formula with the reverse process distribution p_θ in the diffusion model acting as the policy $\pi(\theta)$ in the standard RL problem. This is basically an RL problem in which we consider the pre-trained p_θ as a policy.

3.1 METROPOLIS-HASTINGS UPDATE

Inspired by neural simulated annealing Correia et al. (2023), we use MH Metropolis et al. (1953), a popular choice for MCMC sampling. For each time step in the reverse process, $t \to t-1$, a distribution determined by the diffusion model's predicted noise is proposed. Figure 1 shows the

graphical model of the proposal and transition from pre-trained to reward-guided model. We can write the proposal density as:

$$\tilde{q}_\theta\big(\tilde{\mathbf{x}}_{t-1} \mid \mathbf{x}_t\big) \;=\; \mathcal{N}\Big(\tilde{\mathbf{x}}_{t-1} \;\Big|\; \mu_\theta(\mathbf{x}_t),\; \tilde{\beta}_t\,\mathbf{I}\Big), \tag{12}$$

where

$$\mu_\theta(\mathbf{x}_t) \;=\; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \tfrac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\Big) \tag{13}$$

The proposed new distribution acts as an action in multi-step MDP.

$$\tilde{\mathbf{x}}_{t-1} \sim \tilde{q}_\theta(\tilde{\mathbf{x}}_{t-1} \mid \mathbf{x}_t) \tag{14}$$

We then accept or reject the proposed $\tilde{\mathbf{x}}_{t-1}$ using Equation 15. We consider the MH step as a stochastic transition kernel, governed by the temperature of the system in Equation 15.

$$_{\text{PMH}}(\tilde{\mathbf{x}}_{t-1}, \mathbf{x}_t) \;=\; \min\Big\{1,\; \exp\big[\big(\,r(\tilde{\mathbf{x}}_{t-1}) \;-\; r(\mathbf{x}_t)\,\big)\,/\,T_t\big]\Big\} \tag{15}$$

where T_{t} is a temperature parameter. To balance exploration in early timesteps with exploitation in later timesteps, we implement annealed diffusion with a temperature parameter T_t that decreases over time t. Higher temperatures in early steps promote exploration, while lower temperatures in later steps encourage greedy exploitation of high-reward generations. We draw $u \sim \text{Uniform}[0, 1]$ and perform a slightly modified version of MH. Instead of setting the distribution back to $\mathbf{x}_t$ in case of rejection, we set it to the pre-trained version at the same timestep $\mathbf{x}_{t-1}$. If $u \;\leq\; p_{\text{MH}}(\tilde{\mathbf{x}}_{t-1}, \mathbf{x}_t)$, the new step is set to $\tilde{\mathbf{x}}_{t-1}$. Otherwise, the new step remains the same as previously trained version $\mathbf{x}_{t-1}^{pre}$.

$$\mathbf{x}_{t-1} \;=\; \begin{cases} \tilde{\mathbf{x}}_{t-1}, & \text{if } u \;\leq\; p_{\text{MH}}(\tilde{\mathbf{x}}_{t-1}, \mathbf{x}_t) \\ \mathbf{x}_{t-1}^{pre}, & \text{otherwise} \end{cases} \tag{16}$$

By transitioning from $t = T$ to $t = 1$ on the trajectory τ, we reach the final state, $\mathbf{x}_0$, where the reward $r(\mathbf{x}_0)$ is evaluated. When the temperature parameter T_{t} is high, the probability of accepting new states increases, promoting the exploration of the state space. As T_{t} decreases, the chain increasingly favors states with higher rewards, based on the principles of simulated annealing Kirkpatrick et al. (1983). The distribution of accepted final states by training phase sampling using MH, $p_\theta^{\text{SA}}(\mathbf{x}_0)$, depends on the pre-trained model parameters, the current model parameters θ, the reward function $r(\cdot)$, the schedule temperature T_{t}.

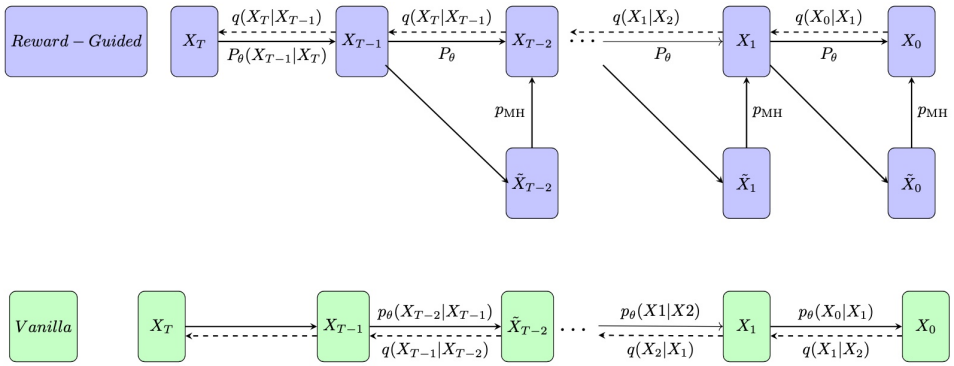

Figure 1: Graphical model of the vanilla DDPM and reward-guided diffusion model

4 EXPERIMENT

First, we perform low-dimensional optimizations for demonstration purposes on the Branin and Rosenbrock task. The global maxima for $\mathbf{x}_1 \in [-5, 10]$ and $\mathbf{x}_2 \in [0, 15]$ are $(-\pi, 12.275)$, $(\pi, 2.275)$, and $(9.42478, 2.475)$, with a maximum value of –0.397887. More details about this

task are available in the Appendix A. We use a dataset of 5000 randomly generated samples for training. Sample reverse process of the model after training as well as the function value curve are shown in Figure 2. The process of increase in reward in single episode is shown across the timesteps. Because of the nature of the RL, only one of the optimal points from Branin functions are highly rewarded. This process might be different from the denoising diffusion optimization models (DDOM) Krishnamoorthy et al. (2023). The reverse sample process for Rosenbrock is shown in Figure 3. Note how the distribution is different between Figure 3 and Figure 2 due to the difference in the position of the optimal points.

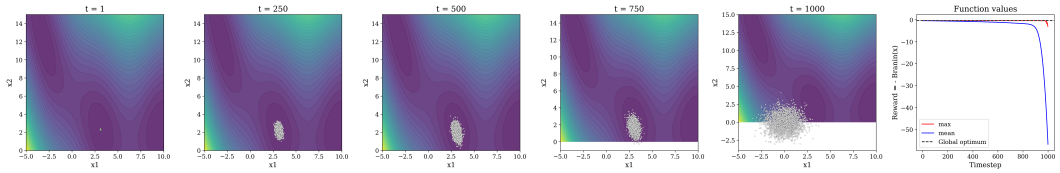

Figure 2: Sample reverse diffusion trajectory for Branin task

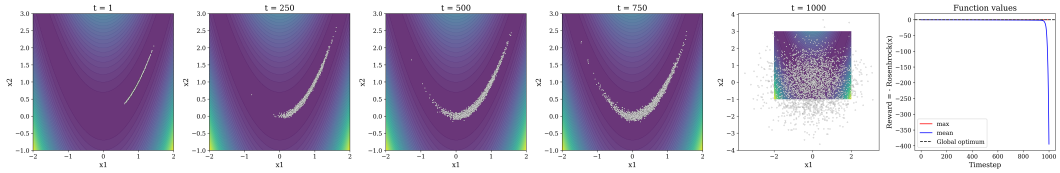

Figure 3: Sample reverse diffusion trajectory for Rosenbrock task

We perform further studies on design-bench problems. The problems are 1) Superconductor Hamidieh (2018): the dataset consists of 21,263 rows and 82 columns: 81 columns corresponding to the features extracted and 1 column of the observed critical temperature values. The goal is to optimize and find a design with maximum critical temperature. 2) D'Kitty Morphology (D'Kitty) : 25,008 samples of the design of a quadrupedal D'Kitty robot with 56 dimensions related to the morphological structure. The goal is to maximize the crawling speed. 3) Rosenbrock (Rosen): The goal is to maximize the Rosenbrock function. We used this function in 2 dimensions for demonstration purposes and here in 60 dimensions for evaluation purposes. For this study, we focus on 50,000 vectors that produce lower function values Rosenbrock (1960). 4) Ant Morphology (Ant): Similarly to D'Kitty Morphology, the Ant Morphology task consisted of 25008 samples of a quadrupedal ant robot, with 60 continuous parts, and we want to optimize crawling velocity. We also study 4 problems in the discrete domain: 1)TF Bind 8 (TF8) Barrera et al. (2016): This dataset contains 32,898 8-unit DNA sequences, each evaluated for binding activity. The goal is to discover a sequence that maximizes this activity. 2) TF Bind 10 (TF10) Barrera et al. (2016): Similar to TF8, but here each design is a 10-unit DNA sequence, resulting in a larger pool of 50,000 samples. As before, the objective is to maximize the binding activity. 3) Neural Architecture Search (NAS) Zoph (2016): This task involves 1,771 candidate neural network architectures designed for the CIFAR-10 dataset. The objective is to identify the architecture that achieves the highest test accuracy. 4) ChEMBL Gaulton et al. (2012): a collection of 1093 drug data with 31 dimensions that the goal is to optimize for particular chemical properties.

Table 1 and Table 2 show the results, which are reported based on $y_{\text{normalized}}(y) = \frac{y - y_{\min}}{y_{\max} - y_{\min}}$ to be consistent with previous work(Trabucco et al.). We used $T = 1000$ in all experiments. All experiments were conducted on an NVIDIA GeForce RTX 3090.

The benchmark methods used in this table are CMA-ES: the CMA evolution strategy Hansen (2006), REINFORCE: policy gradient method Sutton et al. (1999), GRAD: simple gradient ascent, COMs Trabucco et al. (2021), MINs Kumar & Levine (2020), DDOM Krishnamoorthy et al. (2023).

Table 1: Performance comparison of methods across four continuous tasks (maximum normalized score, $mean \pm std$)

Method	Superconductor	Ant Morphology	D'Kitty Morphology	Rosenbrock
D(best)	0.399	0.565	0.884	0.483
CMA-ES	0.465 ± 0.024	$\mathbf{1.11 \pm 0.554}$	0.724 ± 0.001	0.443 ± 0.058
REINFORCE	0.481 ± 0.013	0.266 ± 0.032	0.562 ± 0.196	0.536 ± 0.033
Grad	0.490 ± 0.009	0.932 ± 0.015	0.932 ± 0.005	0.684 ± 0.070
COMs	$\mathbf{0.504 \pm 0.022}$	0.818 ± 0.017	0.905 ± 0.017	0.581 ± 0.075
MIN	0.499 ± 0.017	0.845 ± 0.080	0.892 ± 0.001	0.590 ± 0.010
DDOM	0.495 ± 0.012	0.959 ± 0.014	$\mathbf{0.935 \pm 0.001}$	$\mathbf{0.709 \pm 0.00}$
OURS	$\mathbf{0.569 \pm 0.003}$	$\mathbf{0.977 \pm 0.007}$	$\mathbf{0.965 \pm 0.004}$	$\mathbf{0.783 \pm 0.013}$

Table 2: Performance comparison of methods across four discrete tasks (maximum normalized score, $mean \pm std$)

Method	TF Bind 8	TF Bind 10	NAS	ChEMBL
D(best)	0.439	0.467	0.436	0.605
CMA-ES	0.941 ± 0.014	$\mathbf{0.681 \pm 0.016}6$	$\mathbf{0.982 \pm 0.036}$	0.630 ± 0.009
REINFORCE	$\mathbf{0.960 \pm 0.02}$	0.649 ± 0.021	0.112 ± 0.006	0.635 ± 0.056
Grad	$0..868 \pm 0.011$	0.662 ± 0.019	$\mathbf{0.9654 \pm 0.065}$	$\mathbf{0.642 \pm 0.099}$
COMs	0.951 ± 0.016	0.623 ± 0.009	0.765 ± 0.005	0.638 ± 0.083
MIN	0.913 ± 0.214	0.637 ± 0.014	0.732 ± 0.009	$\mathbf{0.659 \pm 0.014}$
DDOM	$\mathbf{0.9655 \pm 0.012}$	0.652 ± 0.006	0.735 ± 0.005	0.627 ± 0.008
OURS	$\mathbf{0.960 \pm 0.013}$	$\mathbf{0.686 \pm 0.020}$	0.798 ± 0.003	0.618 ± 0.003

The result in Table 1 and Table 2 highlights the superior performance of the reward-guided diffusion model. The best performing models are bold for clear demonstration.

The results indicate that by pre-training the diffusion model, we can benefit from the distribution of the reverse process which is close to an isotropic Gaussian. This enables us to use MH sampling to accept or reject transitions based on this distribution, rather than relying on arbitrary policies that often introduce high variance and complicate the training process.

5 Summary

We proposed a reward-guided multi-step MDP method for fine-tuning the denoising diffusion model for black-box optimization. Our method uses Metropolis-Hastings (MH), as a Markov Chain Monte Carlo (MCMC) sampling to guide the policy towards higher rewards and optimal solution. This sampling using MH along the multi-step MDP is stationary for policy update and is effective in reward maximization. We showed the effectiveness of the method on different tasks and dataset dimensions.

Limitation and Future Work. Our model has no termination point during multi-step MDP other than reaching the final timestep, T, which can cause divergence from the pre-trained model particularly in the early stage when temperature and exploration rate are higher.

References

Pierre Baque, Edoardo Remelli, Francois Fleuret, and Pascal Fua. Geodesic convolutional shape optimization. In *International Conference on Machine Learning*, pp. 472–481. PMLR, 2018.

Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351 (6280):1450–1454, 2016.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Can Chen, Christopher Beckham, Zixuan Liu, Xue Liu, and Christopher Pal. Robust guided diffusion for offline black-box optimization. 2024.

Alvaro HC Correia, Daniel E Worrall, and Roberto Bondesan. Neural simulated annealing. In *International Conference on Artificial Intelligence and Statistics*, pp. 4946–4962. PMLR, 2023.

Ying Fan and Kangwook Lee. Optimizing ddpm sampling with shortcut fine-tuning. *arXiv preprint arXiv:2301.13362*, 2023.

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.

Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

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

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

Hadi Keramati, Feridun Hamdullahpur, and Mojtaba Barzegari. Deep reinforcement learning for heat exchanger shape optimization. *International Journal of Heat and Mass Transfer*, 194: 123112, 2022.

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

Siddarth Krishnamoorthy, Satvik Mehul Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *International Conference on Machine Learning*, pp. 17842–17857. PMLR, 2023.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *Advances in neural information processing systems*, 33:5126–5137, 2020.

P. Langley. Crafting papers on machine learning. In Pat Langley (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

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.

HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The computer journal*, 3(3):175–184, 1960.

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*, 12, 1999.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization, 2021. *URL https://github.com/brandontrabucco/design-bench*, 30:31–32.

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

B Zoph. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Optimization Functions

A.1 Branin Function

Branin is a two-dimensional function with three global minima within a commonly used rectangular domain. A common form of the Branin function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by $f(x_1, x_2) = a\big(x_2 - b\,x_1^2 + c\,x_1 - r\big)^2 + s\,(1-t)\cos(x_1) + s$, where the recommended parameter values are: $a = 1, \quad b = \frac{5.1}{4\pi^2}, \quad c = \frac{5}{\pi}, \quad r = 6, \quad s = 10, \quad t = \frac{1}{8\pi}$. A standard domain for benchmarking is the rectangular region, $x_1 \in [-5, 10], \quad x_2 \in [0, 15]$. The Branin function has three global minima, $(-\pi,\, 12.275), \quad (\pi,\, 2.275), \quad (3\pi,\, 2.475)$, within the rectangular domain. At each of these points, the function value is approximately $f(x_1^*, x_2^*) \approx 0.397887$.

A.2 Rosenbrock Function

The Rosenbrock function in two dimensions that we used for the demonstration is defined as $f(x_1, x_2) \;=\; 100\big(x_2 - x_1^2\big)^2 \;+\; (1 - x_1)^2$, and has a single global minimum at $(1,\, 1)$, where the value of the function is $f(1, 1) \;=\; 0$. We consider the domain to be $x_1 \in [-2,\, 2], \quad x_2 \in [-1,\, 3]$.

The generalized form of the Rosenbrock function in higher dimensions (we used a 60 dimension vector) is written as:

$$f(\mathbf{x}) \;=\; \sum_{i=1}^{59} \Big[100\big(x_{i+1} - x_i^2\big)^2 \;+\; (1 - x_i)^2\Big],$$

where each coordinate x_i is restricted to $[-2,\, 2]$ and the global minimum occurs at $\mathbf{x}_{opt} \;=\; (1, 1, 1...1)$ where $f(\mathbf{x}_{opt}) = 0$.

B Metropolis Acceptance

B.1 Boltzmann Distribution

The probability distribution of each state x in statistical mechanics is expressed by the exponential term:

$$p(x) \;\propto\; \exp\Big(-\frac{E(x)}{k_B\,T}\Big),$$

where $E(x)$ is the energy of each state x, k_B is the Boltzmann constant, and T is the temperature. If we consider the energy of the system as the negative reward value, $E(x) \;=\; -\,r(x)$, the probability distribution is proportional to:

$$\exp\Big(-\tfrac{E(x)}{k_B\,T}\Big) \;=\; \exp\Big(\tfrac{r(x)}{k_B\,T}\Big).$$

B.2 Standard Metropolis–Hastings

The Metropolis–Hastings (MH) algorithm constructs a Markov chain that samples from $p(\mathbf{x})$. The acceptance probability for a proposed state $\tilde{\mathbf{x}}$ for the transition from the state $\mathbf{x}$ is:

$$p_{\mathrm{MH}}(\tilde{\mathbf{x}} \mid \mathbf{x}) \;=\; \min\Big\{1,\, \exp\Big(-\tfrac{E(\tilde{\mathbf{x}}) - E(\mathbf{x})}{k_B\,T}\Big)\Big\}.$$

Substitute $E(\mathbf{x}) = -\,r(\mathbf{x})$ and absorb k_B (considered to be 1) into the temperature:

$$\exp\Big(-\tfrac{E(\tilde{\mathbf{x}}) - E(\mathbf{x})}{T}\Big) \;=\; \exp\Big(\tfrac{r(\tilde{\mathbf{x}}) - r(\mathbf{x})}{T}\Big).$$

Hence, with reward notation, the acceptance rule becomes:

$$p_{\mathrm{MH}}(\tilde{\mathbf{x}} \mid \mathbf{x}) \;=\; \min\Big\{1,\, \exp\Big(\tfrac{r(\tilde{\mathbf{x}}) - r(\mathbf{x})}{T}\Big)\Big\}.$$

B.3 TIME-DEPENDENT COOLING

In order to induce early exploration and greedy exploitation at higher timesteps, we adopt the annealed diffusion, and let the temperature decrease across steps labeled by t. The early steps have higher T_t to allow more exploration, and later steps have smaller T_t, to greedily exploit high reward generation.

B.4 MODIFIED ACCEPTANCE FORMULA

We simply replace the constant T with a time-varying T_t. Then, the acceptance probability for going from $\mathbf{x}_t$ to a proposed $\tilde{\mathbf{x}}_{t-1}$ is:

$$p_{\mathrm{MH}}(\tilde{\mathbf{x}}_{t-1}, \mathbf{x}_t) \;=\; \min\Big\{1,\, \exp\Big(\tfrac{r(\tilde{\mathbf{x}}_{t-1}) - r(\mathbf{x}_t)}{T_t}\Big)\Big\}.$$

We draw a uniform random $u \sim U(0,1)$. If $u \le p_{\mathrm{MH}}$, we accept $\tilde{\mathbf{x}}_{t-1}$. Otherwise, we reject $\tilde{\mathbf{x}}_{t-1}$.

In standard MH, upon rejection, the chain remains at the old state $\mathbf{x}_t$.

$$\mathbf{x}_{t-1} \;=\; \begin{cases} \tilde{\mathbf{x}}_{t-1}, & \text{if } u \le p_{\mathrm{MH}}, \\ \mathbf{x}_t, & \text{otherwise.} \end{cases}$$

In modified version:

$$\mathbf{x}_{t-1} \;=\; \begin{cases} \tilde{\mathbf{x}}_{t-1}, & \text{if } u \le p_{\mathrm{MH}}, \\ \mathbf{x}_{t-1}^{\mathrm{pre}}, & \text{otherwise.} \end{cases}$$

In other words, instead of reverting to $\mathbf{x}_t$, we return to a pre-trained version of $\mathbf{x}_{t-1}$.

C PARAMETERS OF THE REWARD GUIDED DIFFUSION

We used $T = 1000$ for all high-dimensional experiments in this study. We report the effect of the number of timesteps on the three datasets. The score ratio is the performance of the model compared to the performance at $T = 1000$.

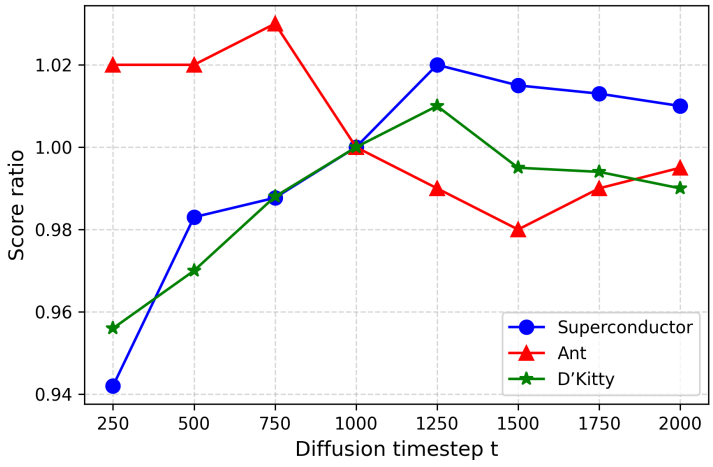

Figure 4: Performance of the reward-guided model at different number of timesteps T compared to $T = 1000$

D VISUALIZATION OF OPTIMIZED DESIGN IN THE DATASET

This section shows the t-SNE plots of the datasets and the optimized design points.

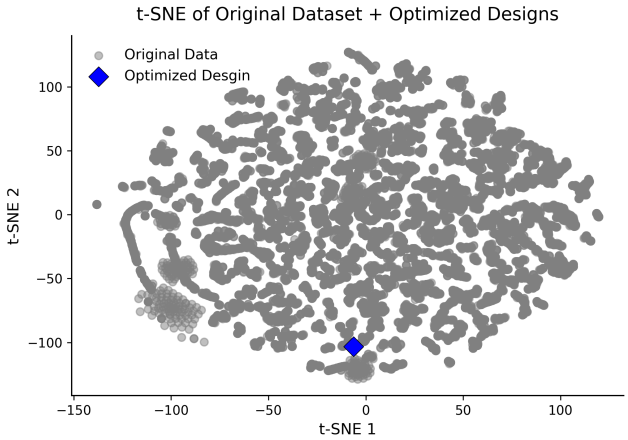

Figure 5: t-SNE plots of the Superconductor dataset and optimized design

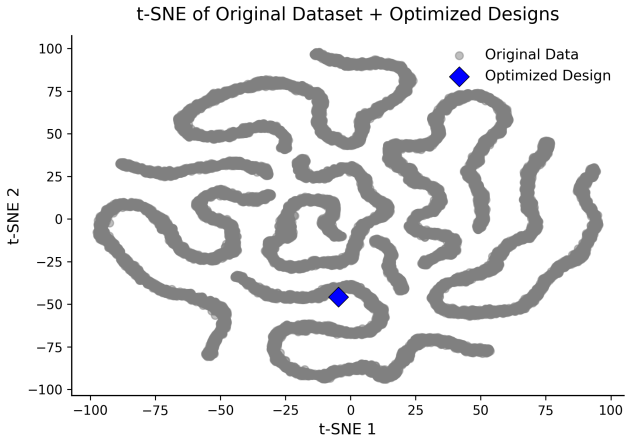

Figure 6: t-SNE plots of the D'Kitty Morphology dataset and optimized design