Trust-Region Noise Search for Black-Box Alignment of Diffusion and Flow Models

Niklas Schweiger[1,2] **Karnik Ram**[1,2,†] **Daniel Cremers**[1,2,†]

[1]TU Munich
[2]Munich Center for Machine Learning
niklas.schweiger@tum.de

Abstract

Optimizing the noise samples of diffusion and flow models is an increasingly popular approach to align these models to target rewards at inference time. However, we observe that these approaches are usually restricted to differentiable or cheap reward functions, the formulation of the underlying pre-trained generative model, or are memory/compute inefficient. We instead propose a simple trust-region based search algorithm (TRS) which treats the pre-trained generative and reward models as a black-box and only optimizes the source noise. Our approach achieves a good balance between global exploration and local exploitation, and is versatile and easily adaptable to various generative settings and reward models with minimal hyperparameter tuning. We evaluate TRS across text-to-image, molecule and protein design tasks, and obtain significantly improved output samples over the base generative models and other inference-time alignment approaches which optimize the source noise sample, or even the entire reverse-time sampling noise trajectories in the case of diffusion models. Our source code is publicly available*.

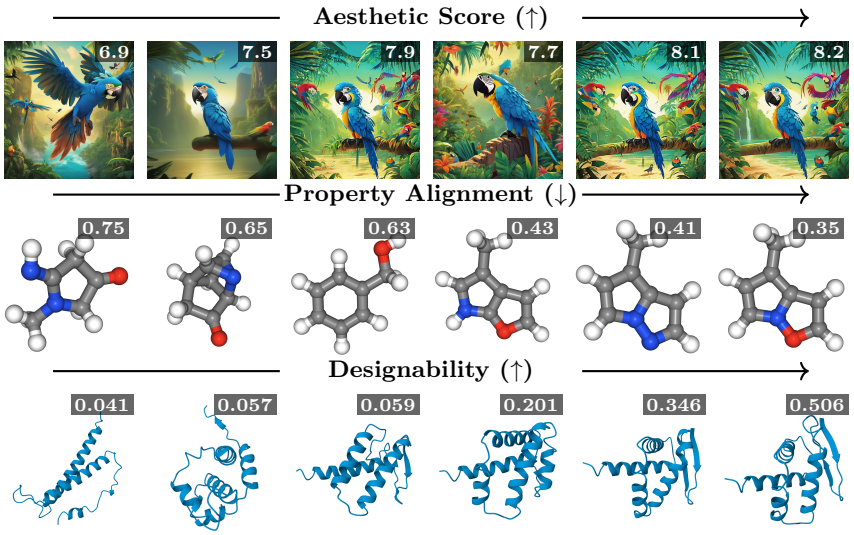


Figure 1: Progression of TRS output samples for text-to-image, molecules, and proteins across $i \in \{0, 3, \ldots, 15\}$. *Top*: "Animated movie poster..." (aesthetic reward). *Middle*: Molecular properties. *Bottom*: Protein designability. Arrows (↑ / ↓) show optimization direction. TRS explores diverse samples before converging to refined, high-reward outputs.

[†]Equal supervision.
*Project page: https://niklasschweiger.github.io/trust-region-noise-search/

1 Introduction

Inference-time alignment is an exciting new paradigm in which the quality of generated samples is optimized post-training using feedback from target reward functions (Bansal et al., 2024; Eyring et al., 2024). Several approaches exist, out of which *gradient-based methods* back-propagate through the entire (ODE-based) iterative process of the model to adjust the initial noise sample (Ben-Hamu et al., 2024; Wang et al., 2025). But these are prone to incur high GPU-memory costs and may drift off the training-data manifold. *Sequence-based methods*, which are developed for SDE samplers, range from filtering approaches that iteratively reweigh and resample (Singhal et al., 2025; Kim et al., 2025) to tree-search based methods (Jain et al., 2025; Li et al., 2025a). These however require a large number of expensive reward calls or accurate value estimates for the terminal rewards which are not always available. *Black-box search methods* on the other hand, which are relative underexplored, treat the entire generator and reward function as a black box and apply local or global search heuristics (Ma et al., 2025; Tan et al., 2025) for the source noise sample. Though these are versatile to the choice of the generative model and reward function, we notice that existing approaches either search too globally or locally, unable to find a good balance.

Inspired by Bayesian optimization algorithms (Eriksson et al., 2019), which also optimize expensive black-box functions, we propose a simple and effective trust region search (TRS) approach that balances between global exploration and local exploitation in a structured manner. And by controlling only the source noise our approach is readily applicable to a wide range of generative models and reward functions. Our approach begins by exploring multiple seed noise samples which are then pruned and iteratively refined with local perturbations. These perturbations are adaptively controlled based on observed reward values, in both magnitude and direction, which we find to be crucial to stay within the data manifold and generate stable and high-quality samples.

The contributions of our work can be summarized as follows:

1. A simple *trust region search* approach for inference-time reward alignment of black-box diffusion and flow models by adaptively controlling the source noise.
2. An extensive evaluation on text-to-image generation, where we obtain significantly more aligned and high-quality image samples over other search and even full noise sequence search baselines under the same compute budget.
3. An extended evaluation on small molecule and protein design tasks with expensive reward functions, where we show that our approach is also readily applicable and effective with almost no hyperparameter turning.

2 Methodology

2.1 Problem Statement and Notation

Let $\mathcal{F}$ be a black-box generative process mapping noise $\mathbf{x}_0 \sim p_0$ to a sample $\mathbf{x}_1$ via a continuous-time process $\{\mathbf{x}_t\}_{t\in[0,1]}$. $\mathcal{F}$ can represent a diffusion model (Song et al., 2021) reversing an SDE via the score function $\epsilon_\theta(\mathbf{x}_t, t)$, or a flow matching model (Lipman et al., 2023) following a velocity field $v_\theta(\mathbf{x}_t, t)$.

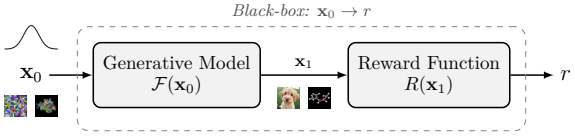


Figure 2: The noise-to-reward pipeline. We treat the dashed box as an expensive black-box process.

Given a reward function R, we seek the noise initialization that maximizes alignment:

$$\mathbf{x}_0^\star = \arg\max_{\mathbf{x}_0 \in \mathbb{R}^M} R(\mathcal{F}(\mathbf{x}_0)). \tag{1}$$

We treat the entire mapping $R(\mathcal{F}(\mathbf{x}_0))$ as a computationally expensive black-box. Consequently, we focus on methods that can effectively steer generation under a strict evaluation budget.

2.2 Trust-Region Search (TRS)

TRS is a trust-region search algorithm with structured noise sampling designed for inference-time alignment with expensive black-box evaluations. Our approach is inspired by TuRBO (Eriksson et al., 2019) for Bayesian optimization, but introduces several important modifications for inference-time steering of large generative models. In particular, unlike surrogate-based Bayesian optimization, TRS relies purely on structured sampling, since we find that surrogates do not contribute meaningfully to performance, due to the highly non-linear noise space, which is difficult to predict under constrained budgets. Additionally, we use a different center selection scheme, by always choosing the top-k observed noises, which is crucial for the performance. An overview of the method is given in Algorithm 1.

Warm-up. We begin with a short warm-up phase for bootstrapping the search. We sample N_{warm} initial noise samples from the model prior p_0,

$$\mathbf{x}_i \sim p_0, \qquad \mathbf{x}_{1,i} = \mathcal{F}(\mathbf{x}_i), \qquad r_i = R(\mathbf{x}_{1,i}),$$

with $i = 1, \ldots, N_{\text{warm}}$, and select the top-performing k points as initial trust-region centers $\{\mathbf{x}^{\text{c}}_{0,j}\}_{j=1}^{k}$. All trust regions are initialized with the same side length $\ell_j = \ell_{\text{init}}$. In practice, this warm-up is implemented using standard iterations with batch size B. We find allocating approximately 20% of the total evaluation budget N_{total} to this phase to be a robust heuristic across tasks and budgets; an ablation is provided in Appendix F.3.

Trust-region iterations. After warm-up, the algorithm maintains k hypercubic trust regions $\mathcal{T}^j \subset \mathbb{R}^M$, each defined by a center $\mathbf{x}^{\text{c}}_{0,j}$ and side length ℓ_j. In each iteration, we collect a global batch set $\mathcal{S}_{\text{batch}}$ of size B across all regions:

1. **Propose:** For each region j, generate B/k candidate noise vectors $\mathbf{x}_{0,j,b}$ by perturbing the center $\mathbf{x}^{\text{c}}_{0,j}$. Following Eriksson et al. (2019), perturbations $\tilde{\mathbf{x}}_{0,j,b}$ are generated within an axis-aligned hypercube with side length ℓ_j and then combined with a stochastic coordinate mask. This mask is built by drawing a perturbation probability $p_{j,b} \sim \text{Uniform}(p_{\min}, p_{\max})$ and applying $\mathbf{m}_{j,b} \sim \text{Bernoulli}(p_{j,b})^M$. The resulting candidates are:
$$\mathbf{x}_{0,j,b} = \mathbf{x}^{\text{c}}_{0,j} + \left(\tilde{\mathbf{x}}_{0,j,b}\right) \odot \mathbf{m}_{j,b}. \tag{2}$$
2. **Evaluate:** All B candidates across regions are aggregated into $\mathcal{S}_{\text{batch}}$ and evaluated in parallel to obtain the rewards $\{r_{j,b}\} = R(\mathcal{F}(\mathcal{S}_{\text{batch}}))$.
3. **Update:** After batch evaluation, we adapt trust-region side lengths $\{\ell_j\}$ using success-based rules per region. A key distinction from vanilla TuRBO (Eriksson et al., 2019) is that trust regions are not treated independently. While exploring multiple regions is beneficial early on, we observe that only a subset typically remains promising. Accordingly, after each batch iteration we re-center all trust regions $\{\mathbf{x}^{\text{c}}_{0,j}\}$ at the globally best k points observed so far. This mechanism naturally shifts computation from exploration toward exploitation, reallocating evaluation budget to promising regions over time. An ablation study on different re-centering strategies is provided in Appendix F.1.

3 Text-to-Image Experiment

Experimental Setup. We evaluate on DrawBench (Saharia et al., 2022) (200 prompts) using SD1.5 (Rombach et al., 2022) and SDXL-Lightning (Lin et al., 2024) with ImageReward (Xu et al., 2023) and HPSv2 (Wu et al., 2023). TRS is compared against OC-Flow (Wang et al., 2025), DTS* (Jain et al., 2025), Fast Direct (Tan et al., 2025), and black-box methods (Ma et al., 2025) under fixed evaluation budgets (NFE). We report the optimized reward averaged over all prompts (mean best reward) in Table 1 along with the number of reward evaluations. The mean best reward across NFE budgets, for showing scaling performance, is shown in Fig. 3). Further runtime analysis and baseline implementations are provided in Appendices H and J.

Results. Table 1 demonstrates that TRS is consistently better than all baselines across generative models and reward functions. On both models, TRS exceeds the state-of-the-art

Table 1: Comparison of noise search algorithms on DrawBench. We report the mean best rewards (IR, HPSv2) for SD1.5 and SDXL alongside the computational cost in terms of number of reward calls. ↑ / ↓ indicate the direction of improvement. Best values are in **bold**, second best are underlined.

	SD1.5		SDXL		
Algorithm	IR ↑	HPS ↑	IR ↑	HPS ↑	**Reward calls**
Base	-0.16	0.246	0.50	0.262	1
Gradient-based guidance					
OC-Flow (Wang et al., 2025)	0.42	0.277	0.85	0.287	–
Noise sequence search					
Fast Direct (Tan et al., 2025)	1.35	0.296	1.50	0.314	420
DTS* (Jain et al., 2025)	1.59	0.303	1.62	0.327	503
Black-box search					
Random (Ma et al., 2025)	1.44	0.302	1.54	0.324	400
Zero-Order (Ma et al., 2025)	1.50	0.313	1.59	0.332	400
TRS (Ours)	**1.62**	**0.322**	**1.66**	**0.340**	400

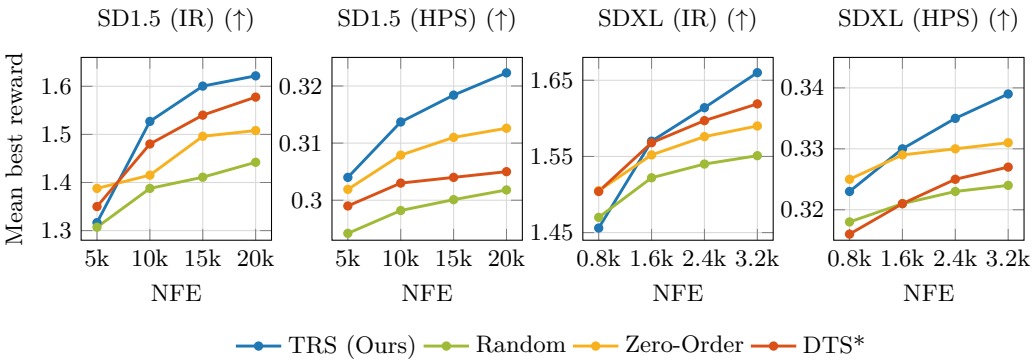


Figure 3: Here we plot the mean best rewards for SD1.5/SDXL optimizing for HPSv2 and ImageReward across different NFE budgets. TRS shows the best scaling performance among all methods.

DTS* performance with up to 4× reduction in wall-clock time and fewer reward evaluations. OC-Flow and Fast Direct underperform relative to random search, highlighting the difficulty of surrogate modeling and gradient-based optimization in high-dimensional noise spaces, while TRS scales efficiently. Full hyperparameter configurations and extended scaling plots are provided in Appendix C.

4 Conclusion

In this work we investigate inference-time scaling and preference alignment, where our simple trust-region source noise search achieves state-of-the-art performance across text-to-image, molecule, and protein design tasks. Our approach is model and reward agnostic, making it particularly suited for real-world settings where reward functions are often expensive or unknown. It also offers good balance between exploration and exploitation by searching multiple noise regions early and refining promising ones, and noticeably does not drift off the data manifold and remains stable as we observe in Appendix D.1. While all methods are limited by the accuracy of the reward models (Appendix K), scaling improvements Wu et al. (2025) suggest this limitation will diminish and our efficient source noise optimization is particulary well-suited for this development.

Acknowledgements

We thank Weirong Chen for his valuable feedback on this work, and the Munich Center for Machine Learning (MCML) for providing computational resources.

References

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Roni Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal Guidance for Diffusion Models. In *The Twelfth International Conference on Learning Representations*, 2024.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-Flow: Differentiating through Flows for Controlled Generation. In *Forty-first International Conference on Machine Learning*, 2024.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian FATRAS, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-Stochastic Flow Matching for Protein Backbone Generation. In *The Twelfth International Conference on Learning Representations*, 2024.

J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion Models Beat GANs on Image Synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*, pp. 5496–5507, 2019.

Luca Eyring, Shyamgopal Karthik, Karsten Roth, Alexey Dosovitskiy, and Zeynep Akata. ReNO: Enhancing One-step Text-to-Image Models through Reward-based Noise Optimization. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling Flow-based Protein Structure Generative Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. Initno: Boosting Text-to-Image Diffusion Models via Initial Noise Optimization. In *CVPR*, pp. 9380–9389, 2024.

Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887. PMLR, 17–23 Jul 2022.

Vineet Jain, Kusha Sareen, Mohammad Pedramfar, and Siamak Ravanbakhsh. Diffusion Tree Sampling: Scalable inference-time alignment of diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Purvish Jajal, Nick John Eliopoulos, Benjamin Shiue-Hal Chou, George K Thiruvathukal, James C Davis, and Yung-Hsiang Lu. Inference-Time Alignment of Diffusion Models with Evolutionary Algorithms. *arXiv preprint arXiv:2506.00299*, 2025.

Stephen Joe and Frances Y. Kuo. Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008. doi: 10.1137/070709359.

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time Alignment of Diffusion Models without Reward Over-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, sriniker, Peter Gedeck, Gareth Jones, Eisuke Kawashima, NadineSchneider, Dan Nealschneider, tadhurst cdd, Andrew Dalke, Matt Swain, Brian Cole, Samo Turk, Aleksandr Savelev, Niels Maeder, Alain Vaucher, Maciej Wójcikowski, Hussein Faara, Ichiru Take, Rachel Walker, Vincent F. Scalfani, Daniel Probst, Kazuya Ujihara, Axel Pahl, guillaume godin, and Juuso Lehtivarjo. rdkit/rdkit: 2025_09_5 (Q3 2025) Release, January 2026.

Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scalia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. Dynamic search for inference-time alignment in diffusion models. *arXiv preprint arXiv:2503.02039*, 2025a.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gökcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-Free Guidance in Continuous and Discrete Diffusion Models with Soft Value-based Decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Scaling Inference Time Compute for Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2523–2534, June 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1): 140022, 2014. doi: 10.1038/sdata.2014.22.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A General Framework for Inference-time Scaling and Steering of Diffusion Models. In *Forty-second International Conference on Machine Learning*, 2025.

I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(67)90144-9.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2021.

Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant Flow Matching with Hybrid Probability Transport for 3D Molecule Generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kim Yong Tan, Yueming Lyu, Ivor Tsang, and Yew-Soon Ong. Fast Direct: Query-Efficient Online Black-box Guidance for Diffusion-model Target Generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-Time Alignment of Diffusion Models with Direct Noise Optimization, 2025.

Michel van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with Foldseek. *Nature Biotechnology*, 42(2):243–246, 2024. doi: 10.1038/s41587-023-01773-0.

Luran Wang, Chaoran Cheng, Yizhen Liao, Yanru Qu, and Ge Liu. Training Free Guided Flow-Matching with Optimal Control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jie Wu, Yu Gao, Zilyu Ye, Ming Li, Liang Li, Hanzhong Guo, Jie Liu, Zeyue Xue, Xiaoxia Hou, Wei Liu, Yan Zeng, and Weilin Huang. RewardDance: Reward Scaling in Visual Generation, 2025.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human Preference Score v2: A Solid Benchmark for Evaluating Human Preferences of Text-to-Image Synthesis, 2023.

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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

A Appendix

This Appendix provides additional implementation information and results to support the main text. It is structured as follows:

B Extended Background and Related Work

B.1 Diffusion and Flow-based Models

Diffusion and flow-based generative models aim to transport a simple noise distribution $p_{\text{noise}} = p_0$ to a complex data distribution $p_{\text{data}} = p_1$. A sample $x_0 \sim p_0$ is gradually transformed into a data sample $x_1 \sim p_1$ through a continuous-time process $\{\mathbf{x}_t\}_{t\in[0,1]}$, typically discretized into T steps with $t_k = k/T$. In continuous data regimes, p_0 is usually white Gaussian noise $\mathcal{N}(0, I)$.

Diffusion models are characterized by a forward noising process that gradually transforms data into noise. This process is governed by a stochastic differential equation (SDE):

$$dx_t = f(t)x_t dt + g(t)dw_t,$$

where $f(t)$ and $g(t)$ are the drift and diffusion coefficients that define the noise schedule (Song et al., 2021). To generate new samples, this process must be reversed. The generative

(reverse) process relies on the score function $\epsilon(x_t, t) \triangleq \nabla_x \log p_t(x)$, which represents the gradient of the log-density of the noisy data. Once the score is estimated via a neural network, sampling can be performed using either the deterministic or stochastic formulations shown in Table 2.

Flow matching models (Lipman et al., 2023) instead learn a continuous vector field $v_\theta(x_t, t) = dx_t/dt$ that defines the velocity of the samples. Unlike diffusion, the paths are not necessarily tied to a corruption process; a common choice is the Optimal Transport (OT) path, defined by the linear interpolation:

$$x_t = (1-t)x_0 + tx_1,$$

which yields a constant target velocity $v_t = x_1 - x_0$. Standard flow matching integrates the learned ODE for sampling, while stochastic variants (Bose et al., 2024) can introduce Brownian motion and optional score corrections as detailed in Table 2.

Unified view. Both paradigms can be expressed in a general form $dx_t = f(x_t, t)dt + g(t)dw_t$. As summarized in Table 2, the two frameworks differ primarily in how the drift $f(x_t, t)$ is parameterized, either through the score function $\epsilon(x_t, t)$ to reverse a corruption process, or through a velocity field $v_\theta(x_t, t)$ to match a probability flow.

Table 2: Generative sampling formulations ($t = 0$ as noise, $t = 1$ as data). Note that for diffusion, $\epsilon(x_t, t)$ denotes the score function $\nabla_x \log p_t(x)$.

Formulation	**ODE (deterministic)**	**SDE (stochastic)**
Diffusion	$dx_t = \left[f(t)x_t - \frac{1}{2}g^2(t)\epsilon(x_t, t)\right] dt$	$dx_t = \left[f(t)x_t - g^2(t)\epsilon(x_t, t)\right] dt + g(t)dw_t$
Flow Matching	$dx_t = v_\theta(x_t, t)dt$	$dx_t = v_\theta(x_t, t)dt + g(t)dw_t$

B.2 Alignment of diffusion and flow models.

A pretrained diffusion model defines a generative distribution $p_\theta(x)$, but in many applications not all samples are equally useful. For instance, in molecular generation one may prefer molecules with specific chemical properties, or in text-to-image synthesis samples that better match a given prompt. To formalize this, access to a reward function $R(x)$ is assumed that evaluates the desirability of a generated sample.

Reward alignment can then be viewed as modifying the sampling distribution by weighting it with the exponential of the reward:

$$\pi^*(x) \propto p_\theta(x)\ \exp(R(x)).$$

Here, $p_\theta(x)$ captures the pretrained generative model, while the reweighting biases sampling toward high-reward regions. The alignment task is therefore either to approximate sampling from $\pi^*(x)$ or to directly search for noise initializations that generate high-reward samples.

Gradient-based guidance. One approach to noise optimization involves back-propagating gradients from a reward function through the entire iterative sampling process to the initial noise (Ben-Hamu et al., 2024; Wang et al., 2025; Guo et al., 2024; Tang et al., 2025). While effective, these methods require differentiable reward functions and incur substantial GPU memory and computational overhead, as they necessitate storing or recomputing the full diffusion trajectory during inference. This limitation becomes particularly severe in high-dimensional settings, such as image or 3D generation, or when many solver steps are required. Furthermore, pure gradient-based refinement often shifts the generation off the natural data manifold, necessitating additional regularization or alignment terms to preserve sample quality (Wang et al., 2025). An exception is direct noise optimization (DNO) (Tang et al., 2025), which additionally supports gradient approximations for non-differentiable rewards and an SDE-based mode that can refine intermediate noises along the trajectory. Despite these extensions, the same disadvantages of high runtime and GPU memory consumption remain.

Table 3: **Feature comparison across optimization frameworks**. We compare methods by their support for gradient-free objectives, black-box pipeline compatibility, batch efficiency, and their ability to perform global vs. local optimization.

Method	Grad-free	Black-box	Batch Eff.	Global	Local
OC-Flow	✘	✘	✘	✘	✔
GNSO	✔	✔	✔	✔	✘
DTS*	✔	✘	✘	✔	✘
Random Search	✔	✔	✔	✔	✘
Zero-Order	✔	✔	✔	✘	✔
TRS (Ours)	✔	✔	✔	✔	✔

Noise sequence search. Another popular direction is to steer the generation throughout the whole or parts of the sample trajectory $\{x_0, \dots, x_1\}$. This includes sequential Monte Carlo (SMC) approaches using resampling (Singhal et al., 2025; Kim et al., 2025), tree search methods such as DSearch and DTS (Li et al., 2025a; Jain et al., 2025), and Fast Direct which uses Gaussian process-based black-box optimization over the full noise sequence (Tan et al., 2025). While these methods can be effective for stochastic diffusion sampling, they typically rely on intermediate reward approximations (Li et al., 2025b) or are difficult to utilize with batched evaluations. This becomes problematic for ODE-based flow matching or many 3D diffusion models where intermediate noise sequences are not available or cheap reward estimates are inaccurate.

Black-box search. Most versatile are source noise optimization algorithms, which treat the generative model and the reward function as black-box. Consequently, they can be applied to any generative model, that maps from a noise distribution to some data distribution. The reward function can be chosen arbitrarily as well. While generally simpler by concept and implementation, they lead to surprisingly good results. This category includes random search, zero-order search (Ma et al., 2025), and some recent concurrent work including evolutionary and genetic algorithms such as SNES or CoSyne (Jajal et al., 2025). Our work belongs to this category, spending all evaluation budget for finding the best source noise. This approach succeeds in all our experiments, providing great flexibility for choosing models and reward function, while producing state-of-the-art results in relevant applications.

C Algorithm Details

Algorithm 1 Trust-Region Search (TRS)

Require: Total/Warmup Budget $N_{\text{total}}/N_{\text{warm}}$, Batch size B, Regions k, Model $\mathcal{F}$, Reward function R

Warm-up: Sample $\{\mathbf{x}_{0,i}\}_{i=1}^{N_{\text{warm}}} \sim p_0$; evaluate $r_i = R(\mathcal{F}(\mathbf{x}_{0,i}))$ in batches of size B.
Init: Set centers $\{\mathbf{x}^{\text{c}}_{0,j}\}_{j=1}^{k}$ to top k; set lengths $\ell_j \leftarrow \ell_{\text{init}}$; set remaining budget $N \leftarrow N_{\text{total}} - N_{\text{warm}}$
while budget $N \geq B$ **do**
  $\mathcal{S}_{\text{batch}} \leftarrow \emptyset$
  for region $\mathcal{T}_j$ and sample $b \in \{1..B/k\}$ **do**
    $p_{j,b} \sim \text{U}(p_{\min}, p_{\max})$; $\mathbf{m}_{j,b} \sim \text{Ber}(p_{j,b})^M$
    $\tilde{\mathbf{x}}_{0,j,b} \sim \text{Perturb}(\ell_j)$ ▷ Gauss or Sobol
    $\mathbf{x}_{0,j,b} \leftarrow \mathbf{x}^{\text{c}}_{0,j} + (\tilde{\mathbf{x}}_{0,j,b} \odot \mathbf{m}_{j,b})$
    $\mathcal{S}_{\text{batch}} \leftarrow \mathcal{S}_{\text{batch}} \cup \{\mathbf{x}_{0,j,b}\}$
  end for
  Evaluate: $\{r_{j,b}\} \leftarrow R(\mathcal{F}(\mathcal{S}_{\text{batch}}))$
  Adapt: Update lengths $\{\ell_j\}$
  Shift: Re-center $\{\mathbf{x}^{\text{c}}_{0,j}\}$ to the global top k.
  $N \leftarrow N - B$
end while
return best $\{\mathcal{F}(\mathbf{x}_0), r\}$

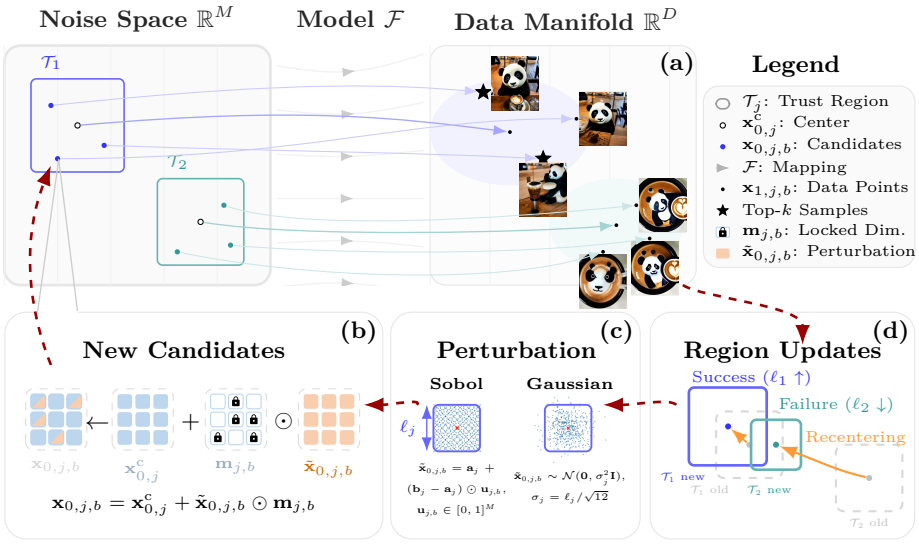


Figure 4: Illustration of the trust region search algorithm with a two-region example for the prompt "A panda making latte art" from DrawBench (Saharia et al., 2022). (a) Samples from the noise space $\mathbb{R}^M$ are mapped to the data manifold in $\mathbb{R}^D$ via the generative model $\mathcal{F}$. Generated samples from the same region ($\mathcal{T}_j$) exhibit visual similarity; here, $\mathcal{T}_1$ shows markedly better prompt alignment than $\mathcal{T}_2$. (b) New candidates $\mathbf{x}_{0,j,b}$ are generated by adding masked ($\mathbf{m}_{j,b}$) relative perturbations ($\tilde{\mathbf{x}}_{0,j,b}$) to the current center $\mathbf{x}^c_{0,j}$. (c) Comparison of Sobol and Gaussian perturbation schemes used to fill the trust-region hypercube. (d) The update and shift logic: $\mathcal{T}_1$ expands upon identifying a top-k sample, while the underperforming $\mathcal{T}_2$ is re-centered (shifted) to a more promising region. 9

This section gives more detail about the perturbation schemes we use and summarizes the trust-region management strategy underlying the TRS algorithm and provides additional implementation details. Further, we show a pseudo algorithm 1 here to summarize the method outlined in section 2.2. The update rules for the trust-region side length largely follow the strategy introduced in TuRBO (Eriksson et al., 2019). For each region, we maintain success and failure counters based on whether newly evaluated candidates improve upon the best objective value observed within that region. If a candidate yields an improvement, the success counter is incremented and the failure counter is reset. Otherwise, the failure counter is incremented and the success counter is reset.

We use different perturbations strategies, since deterministic low-discrepancy samplers are often more efficient to fill spaces Sobol' (1967). However, they are only computed up to 21,201 dimensions Joe & Kuo (2008). In our experiment in section 3, we use SDXL, whose noise dimension is 65,536. In this case we also introduce a gaussian perturbation scheme, that is designed to follow the trust-region hypercube exploration closely.

Sobol perturbations. For Sobol-based proposals, a point $\mathbf{u}_{j,b} \in [0,1]^M$ is sampled and mapped affinely into the trust region. Let $\mathbf{a}_j = -\frac{1}{2}\ell_j \mathbf{1}_M$ and $\mathbf{b}_j = \frac{1}{2}\ell_j \mathbf{1}_M$, where $\mathbf{1}_M$ denotes a M-dimensional vector of ones. The resulting proposal is

$$\tilde{\mathbf{x}}_{0,j,b} = \mathbf{a}_j + (\mathbf{b}_j - \mathbf{a}_j) \odot \mathbf{u}_{j,b}.$$

Gaussian perturbations. For Gaussian perturbations, we sample

$$\tilde{\mathbf{x}}_{0,j,b} \sim \mathcal{N}(\mathbf{0}, \sigma_j^2 \mathbf{I}), \qquad \sigma_j = \ell_j / \sqrt{12},$$

where $\mathbf{I} \in \mathbb{R}^{M \times M}$ is the identity matrix. The standard deviation $\sigma_j = \ell_j/\sqrt{12}$ is chosen to match the variance of a uniform distribution $\mathcal{U}[-\ell_j/2, \ell_j/2]$, ensuring the Gaussian proposals cover the trust-region hypercube with equivalent spread.

Trust-region adaptation. Two thresholds, c_{succ} and c_{fail}, govern the adaptation of the trust-region side length ℓ_j. When the success counter reaches c_{succ}, the trust region is expanded according to

$$\ell_j^{\text{new}} = \min(\ell_j \cdot \alpha_\ell, \ell_j^{\max}),$$

where α_ℓ is an expansion factor for the length (we set it to 1.5 by default). Conversely, when the failure counter reaches c_{fail}, the trust region is contracted as

$$\ell_j^{\text{new}} = \max(\ell_j / \alpha_\ell, \ell_j^{\min}).$$

This mechanism enables adaptive control over the exploration scale based on recent optimization progress.

Restart logic. When a contraction is triggered while the trust-region side length is already at its minimum value $\ell_j^{\min}$, the region is considered to have locally converged. In this case, the side length is reset, and the region is subsequently re-centered at one of the globally best points observed so far. This restart mechanism allows the algorithm to escape from locally saturated regions and to continue allocating evaluations toward more promising areas of the search space.

Length-dependent perturbation constraints. To maintain stable behavior across different trust-region scales, we apply simple constraints that couple the side length ℓ_j with the perturbation probability p_j used for coordinate masking $\mathbf{m}_j$, by rejection sampling. These constraints limit the number of perturbed dimensions when the trust region is large, which is particularly important in high-dimensional noise spaces. Specifically, we enforce the following upper bounds:

- If $\ell_j \geq 2.0$, $p_j \leq 0.2$.
- If $\ell_j \geq 1.6$, $p_j \leq 0.5$.
- If $\ell_j \geq 1.2$, $p_j \leq 0.7$.

These constraints were found to improve robustness without introducing additional tuning complexity. We provide an intuitive visualization for those rules in Figure 8, where we show that under which combinations of mask and region length the perturbations fail.

The default hyperparameters for the trust region state are summarized in Table 4.

Table 4: Hyperparameters for all our experiments in section 3. When entries are separated with a slash / , we refer to the difference between SD1.5 / SDXL. The dagger† means that is the setting the entry refers to the highest NFE setting in the respective experiment in section 3 and Appendix D The asterix * excludes OC-Flow and DTS which use a batch size of 1.

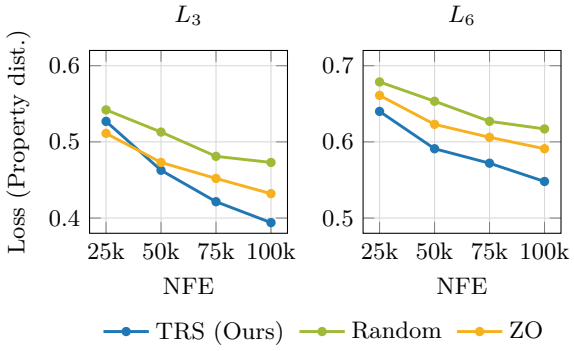
Hyperparameter	**T2I**	**Moleculs**	**Protein**
All source noise and sequence optimizers			
Noise Dimension D	16384 / 65536	291	$3 \times n_{\text{res}}$
Batchsize B*	20	100	8
Inference steps	50/8	50	400
OC-Flow (Wang et al., 2025)			
Batch size B	1	1	–
Step size η	0.25	–	–
Weight decay	0.998	0.995	–
Weight constraint	0.4	0.4	
Number of steps	50	50	–
Optimizer	SGD	L-BFGS	–
Learning rate α	1.0	1.0	–
DTS* (Jain et al., 2025)			
Expansion steps	[50, 40, 30, 20, 10] / [8, 6, 4, 2]	–	–
Exploration Constant λ	0.1 (IR), 0.01 (HPS)	–	–
Progressive Width Constant C	2.0	–	–
Progressive Width α	0.4	–	–
Exploration type	UCB	–	–
Zero-Order Search (Ma et al., 2025)			
Added noise ϵ	0.1	0.1	0.1
Fast Direct (Tan et al., 2025)			
Step size α	80	–	–
Total steps T	6†	–	–
Noise sigma (GP) σ_n	0.1	–	–
TRS (Ours)			
Perturbations	Sobol / Gaussian	Sobol	Sobol
# Trust regions k	15	20	5
Initial Trust Region length l_{init}	0.8	0.8	0.8
Minimal Length l_{min}	0.05	0.05	0.05
Maximal Lenght l_{max}	2.4	2.4	2.4
Length Update factor α_ℓ	1.5	1.5	1.5
Success counter threshold c_{succ}	3	3	3
Failure counter threshold c_{fail}	3	3	3
Fraction warm-up iterations	20%	20%	20%
Min. probability for masks p_{min}	0.1	0.1	0.1
Max. probability for masks p_{max}	0.9	0.9	0.9

D Additional Experiments

To demonstrate the versatility of TRS beyond diffusion based image generation, we evaluate its performance on pure flow-matching models and complex 3D modalities. We first apply the algorithm to 3D molecule generation to steer chemical properties within ODE-based latent spaces. Additionally, we investigate a protein design task featuring and computationally intensive reward evaluations, involving multiple large neural networks.

Algorithm	$L_3(\downarrow)$	$L_6(\downarrow)$	MSP	VUP
Base	1.21	1.15	88.3	88.3
OC-Flow	0.83	0.93	68.0	68.0
Random	0.47	0.62	84.2	84.1
Zero-order	<u>0.43</u>	<u>0.59</u>	84.4	84.3
TRS (Ours)	**0.39**	**0.55**	85.7	85.6

(a)



(b)

Figure 5: **Molecule optimization results.** (a) Distance to the combined targets of 3 and 6 chemical property values, together with MSP stability and validity VUP metrics which we do not explicitly optimize; (b) Scaling trends over the NFE budget. The best values are in **bold**, second best are <u>underlined</u>.

D.1 Molecule Generation

Setup. In this experiment, we move to ODE-based flow matching and consider a different data modality. Our goal is to generate small molecules with specified target values of chemical properties (Ramakrishnan et al., 2014). We use EquiFM (Song et al., 2023) with 50 integration steps, operating on a joint continuous Gaussian coordinate- and encoded feature space (291 total dimensions), with pretrained chemical property prediction models. This setting is a standard benchmark for evaluating classifier-free guidance (Hoogeboom et al., 2022) and was also used by OC-Flow (Wang et al., 2025) for noise optimization toward single-property targets. However, we observe that random search based methods already achieve very strong performance in this regime and therefore extend the benchmark to multi-property target matching. Specifically, we define a distance-based loss as the mean absolute deviation between predicted and target values across multiple chemical properties. Since our framework is formulated as reward maximization, we maximize the negative of this loss. We consider two reward functions, R_3 and R_6, involving 3 and 6 chemical properties, respectively and additional property details are provided in Appendix E.2. As baselines, we exclude noise-sequence search methods, as they are incompatible with ODE-based flow matching, and compare against OC-Flow, random search, and zero-order search.

Metrics. As our primary metric, we report the mean best losses $L_3(x) = -R_3(x)$ and $L_6(x) = -R_6(x)$ (distance to the target properties) over 200 molecules, with the target property values sampled randomly. In addition, we track other relevant metrics such as the molecule stability percentage (MSP), and valid and unique percentage (VUP).

Results. The results are summarized in Figure 5. We observe that TRS consistently achieves the lowest losses, indicating highest alignment with the specified multi-property targets. Importantly, this improvement does not come at the expense of other quality metrics: molecule stability and novelty remain comparable to those of the base model. In contrast, the gradient-based OC-Flow method exhibits degraded stability and novelty, even when regularization is applied, suggesting that gradient-based optimization can drift off the data manifold in this setting. Overall, we find that sampling-based methods are better suited for this task, with even random search outperforming OC-Flow, highlighting the challenges of gradient-based noise optimization for molecular generation.

D.2 Protein Design

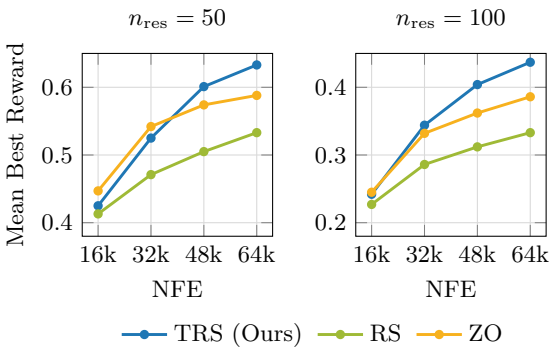
Setup. Protein design is another challenging data modality with expensive reward models and is relatively underexplored by inferenece-time alignment methods. State of the art 3D

protein generation models, like Proteina (Geffner et al., 2025), are flow matching models that are typically trained via ODE integration, but sampling is often performed using an SDE scheduler with noise reduction to improve designability. However, this procedure alters the target distribution and no longer samples from the full flow-matching distribution, which can negatively impact other metrics such as diversity and novelty. In this experiment, we therefore focus on steering ODE-based sampling while preserving the full distribution. For completeness, we provide a comparison to SDE-based sampling in Appendix G. We further note that SDE-based noise-sequence optimization methods are not applicable in this setting: they either rely on value estimation functions that are unavailable for protein design, or are computationally infeasible due to the high cost of reward evaluation. We design the experiment by fixing the number of residues $n_{\text{res}} = 50$ and $n_{\text{res}} = 100$. For the reward function, we use a computationally expensive designability reward based on large structure extraction and protein folding models (see Appendix E for details). We compare against purely black-box baselines, including random search and zero-order search Ma et al. (2025).

Metrics. As alignment metric we report the mean best rewards for each run. Additionally we show the cluster diversity and the pairwise TM-score and a novelty metric, which compares the proteins to the entire PDB dataset (Berman et al., 2000). Further details of these are given in Appendix I.2.

n_{res}	**Method**	Des.↑	Clu.↑	TM↓	Nov.↓
50	Base	0.05	0.71	0.50	0.70
	Random	0.53	0.27	0.68	0.85
	Zero-order	<u>0.59</u>	0.39	0.62	0.85
	TRS(Ours)	**0.65**	0.30	0.67	0.89
100	Base	0.02	0.71	0.56	0.86
	Random	0.33	0.49	0.55	0.84
	Zero-order	<u>0.39</u>	0.59	0.54	0.84
	TRS(Ours)	**0.45**	0.44	0.57	0.86

(a)



(b)

Figure 6: **Protein optimization results.** (a) Optimized mean designability rewards at 64k NFE, together with diversity and novelty metrics which we do not explicity optimize. The best values are in **bold**, second best are <u>underlined</u>. (b) Reward improvement across NFE budgets.

Results. In terms of the designability steering, we see in Figure 6, that we significantly improve over other search algorithms in both settings. We see that generally the rewards are higher when we optimize proteins with 50 residues, which is expected, but also their diversity and novelty metrics decrease compared to the base model. We assume that, since the solution space is rather small, it is likely that well designable proteins share similar features. However, this is not comparable to the mode collaps, which we observe for SDE noise reduction (see Appendix G). Naturally, due to a larger solution space, optimizing proteins with 100 residues leads to better diversity and novelty, while still gaining major improvements over the base model. This counts for all optimization methods, but TRS achieves the best steering results again.

E Experiment Details

In this section, we provide additional details of the experiments in Section 3 with a specific focus on the generative models and the reward functions that are applied. We provide the links to all external resources in Table 5.

Table 5: Summary of generative models and reward functions for the experiments in Section 3.

Domain	Model / Tool	Params	Source / Repository
T2I	SD v1.5	∼860M	HF: SD-v1.5
	SDXL-Lightning	∼2.6B	HF: SDXL-Lightning
	ImageReward	∼446M	GitHub: ImageReward
	HPSv2	∼986M	GitHub: HPSv2
	Aesthetic Pred.	∼428M	GitHub: Aesthetic
Molecules	EquiFM	∼22M	GitHub: EquiFM / OC-Flow
Proteins	Proteina ($\mathcal{M}_{\text{FS}}^{\text{small}}$)	∼60M	GitHub: Proteina
	ProteinMPNN	∼1.7M	GitHub: ProteinMPNN
	ESMFold-v1	∼3B	GitHub: ESMFold

E.1 Text-to-Image

Generative models. We use Stable Diffusion (Rombach et al., 2022) v1.5 (∼859.5M parameters), which generates $3 \times 512 \times 512$ images from a noise space of $4 \times 64 \times 64$, and a distilled version of the larger SDXL (Lin et al., 2024) (∼2.57B parameters), which generates $3 \times 1024 \times 1024$ images from a noise space of $4 \times 128 \times 128$. Both models encode text prompts using CLIP-based text encoders (Radford et al., 2021). Classifier-free guidance (Dhariwal & Nichol, 2021) is applied during sampling by linearly combining conditional and unconditional model predictions with conditioning signal c. For image sampling, we use the DDIM scheduler. We set $\eta = 0$ (deterministic sampling) for black-box source noise optimization methods and $\eta = 1.0$ (stochastic sampling) for noise sequence optimization methods. The number of inference steps is set to 50 for SD1.5 and 8 for SDXL, and all experiments are conducted in `float16` precision.

Reward functions. We use three reward functions for the text-to-image experiments, each defined by the scalar output of a pretrained model. ImageReward (Xu et al., 2023) and HPSv2 (Wu et al., 2023) evaluate image–prompt alignment and are defined as $R(\mathbf{x}_1, c)$, where $\mathbf{x}_1$ denotes the generated image and c the text prompt. In contrast, the Aesthetic predictor evaluates only the generated image and is defined as $R(\mathbf{x}_1)$.

E.2 Molecule Generation

Generative model. We use EquiFM (Song et al., 2023) (∼22M parameters) to generate small QM9 molecules and base our implementation on the code of OC-Flow (Wang et al., 2025). EquiFM is an ODE-based flow matching model that operates on a combined continuous Gaussian noise representation for both atomic coordinates and atom types, with total dimensionality $M = 291$. The number of atoms per molecule ranges from 3 to 29. To accommodate variable-sized molecules while keeping a fixed input dimension, unused atom entries are masked to zero. In the experiments described in Section D.1, the number of atoms is sampled according to the empirical distribution in the QM9 training set, which most frequently yields molecules with 15 to 20 atoms.

Reward functions. The reward is defined as the negative sum of absolute deviations between target property values and the corresponding predictions of pretrained regression models. For the R_6 reward, we consider the properties $\{\alpha, \mu, \Delta\varepsilon, \varepsilon_{\text{HOMO}}, \varepsilon_{\text{LUMO}}, c_v\}$, corresponding to isotropic polarizability, dipole moment, HOMO–LUMO gap, HOMO energy, LUMO energy, and heat capacity. For the R_3 reward, we restrict the objective to the subset $\{\alpha, \mu, \varepsilon_{\text{LUMO}}\}$.

E.3 Protein Design

Generative model. For protein design, we use the $\mathcal{M}_{\text{FS}}^{\text{small}}$ variant of Proteina (Geffner et al., 2025) with 400 integration steps, which consists of approximately 60M transformer parameters and does not include triangle layers. The source noise is drawn from a continuous Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ with dimensionality $n_{\text{res}} \times 3$, corresponding to the 3D coordinates of n_{res} residues.

Once the model has learned the ODE dynamics, sampling can be performed either via deterministic ODE integration, $\mathrm{d}\mathbf{x}_t = \mathbf{v}_t^\theta(\mathbf{x}_t, \tilde{c})\,\mathrm{d}t$, or using stochastic SDE sampling, $\mathrm{d}\mathbf{x}_t = \left[\mathbf{v}_t^\theta(\mathbf{x}_t, \tilde{c}) - \gamma g(t)\mathbf{s}_t^\theta(\mathbf{x}_t, \tilde{c})\right]\mathrm{d}t + \sqrt{2\gamma g(t)}\,\mathrm{d}\mathbf{w}_t$. We discuss the implications of these sampling choices in more detail in Appendix G. In practice, setting $\gamma < 1$ improves designability at the expense of diversity and novelty.

Reward functions. Following (Geffner et al., 2025), we optimize the designability of generated 3D protein backbones. Given a generated structure, a pretrained inverse folding model, ProteinMPNN (Dauparas et al., 2022) (∼1.7M parameters), is used to design a compatible amino acid sequence. This sequence is subsequently processed by a folding model, for which we use the transformer-based ESMFold-v1 (Lin et al., 2023) (∼3B parameters), to predict its 3D structure. We compare the predicted structure to the original generated backbone by computing the self-consistency root mean square deviation (scRMSD). The scalar reward is then defined as $R(\mathbf{x}_1) = \exp(-\text{scRMSD}(\mathbf{x}_1))$, which maps the score to the range $(0, 1]$, with values closer to 1 indicating higher designability. Note that usually 8 sequences are extracted and folded and the best of those is chosen, but for computational efficiency we only extract 1 per protein backbone.

F Ablations

Here we perform further ablation studies to justify the design choices in Section 2.2 and hyperparameters in Appendix C. Most importantly, Section F.1 emphasises the importance of our top-k center selection instead of keeping them strictly apart. Section F.2 provides visual and quantitative insights into the relationship of the trust-region lengths ℓ_j and the mask probabilities p_j, while Section F.3 shows that 20% is a good heuristic for the fraction of warm-up budgets across multiple total budgets. Finally, Section F.4 shows that even the number of regions k is a robust hyperparameter, which is the only one that we change across our experiments in Section 3.

F.1 Center Selection

Setup. We investigate four center selection strategies for text-to-image generation using Stable Diffusion 1.5 (SD1.5) and ImageReward as reward function. *LocalBest* and *LocalLastIter* enforce strict region separation, updating centers using either the historical best or the most recent best sample within each region $\mathcal{T}_j$, respectively. In contrast, *GlobalTopk* and *GlobalLastIter* allow interaction by re-centering regions globally based on the overall top-k samples observed so far or in the latest iteration. All experiments use a batch size $B = 20$ and 20 iterations ($20k$ NFE), consistent with the configuration in Section 3.

Metrics. Performance is evaluated on the full DrawBench benchmark. We use the mean best ImageReward as the primary metric to rank the generated noise samples and determine center updates. Random search is included as a baseline reference to compare the relative improvement of each selection strategy.

Results. As shown in Figure 7, *GlobalTopk* consistently outperforms all other strategies. The results indicate that enforcing strict separation between regions (as in the original TuRBO algorithm (Eriksson et al., 2019)) limits performance in this setting. Conversely, global re-centering enables a more effective allocation of the evaluation budget toward the most promising areas of the search space, significantly improving generation quality.

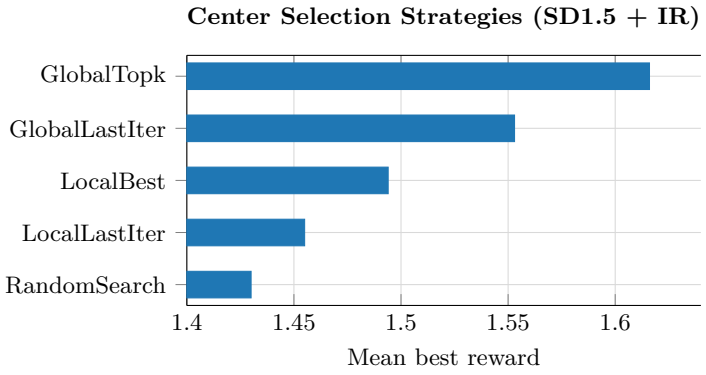


Figure 7: Effect of center selection strategy on DrawBench using SD1.5 and ImageReward. We compare the mean best rewards across the prompts in the benchmark.

F.2 Trust Region Dynamics: Interaction of Length and Masks

Setup. We perform a grid ablation across varying initial trust region lengths ℓ_{init} and masking sparsity ranges defined by $p_{\min}$ and $p_{\max}$. First, we do this quantitatively by investigating different combinations of initial trust-region lengths ℓ_{init} and set the mask probabilities in different ranges, with different min and max probabilities $p_{\min}$, $p_{\max}$. This experiment is conducted using SD1.5 on a representative subset of 55 prompts from DrawBench, covering all categories of the original benchmark to ensure broad coverage of the latent space. Additionally, we visualize the relationship of the trust-region length ℓ_j and sampled mask probability p_j in Figure 8.

Metrics. In Figure 8 we aim to track the point at which the samples begin to exhibit structural degradation and visualize these noising effects, while we show the mean best ImageReward for the quantitative experiment in Table 6.

Results. Figure 8 reveals a clear stability threshold at approximately $\ell_j \approx 1.6$, beyond which images show visible noise. However, we find that low-probability masks (e.g., $p_j = 0.05$) act as a regularizer and allow for significantly higher exploration lengths, up to $\ell_j = 6.4$, while maintaining valid, coherent structures. Table 6 reveals that the exact masking range is not particularly important and works well for different ranges. The initial length also shows good robustness, even when changing it to very high or low values, due to the adaptive nature of the algorithm.

Table 6: Mean best rewards across varying trust-region init lengths (ℓ_{init}) and $[p_{\min}, p_{\max}]$ settings for SD1.5 and ImageReward.

TR init length	**Mask probabilities $[p_{\min}, p_{\max}]$**			
(ℓ_{init})	[0.05, 0.50]	[0.10, 0.90]	[0.30, 0.70]	[0.50, 1.00]
0.2 (Low)	1.6223	1.6430	1.6353	1.6564
0.8 (Medium)	**1.6928**	1.6761	1.6820	1.6543
2.0 (Large)	1.6404	1.6197	1.6306	1.6185

F.3 Warm-Up

Setup. We investigate the impact of the warm-up phase on performance to determine the optimal allocation of the total compute budget between initialization and the main optimization process. This experiment uses Stable Diffusion 1.5 on the subset of 55 DrawBench prompts, equally distributed across all categories. We evaluate three total evaluation budgets ($N_{\text{total}} \in \{120, 360, 720\}$) with a fixed batch size of $B = 24$.

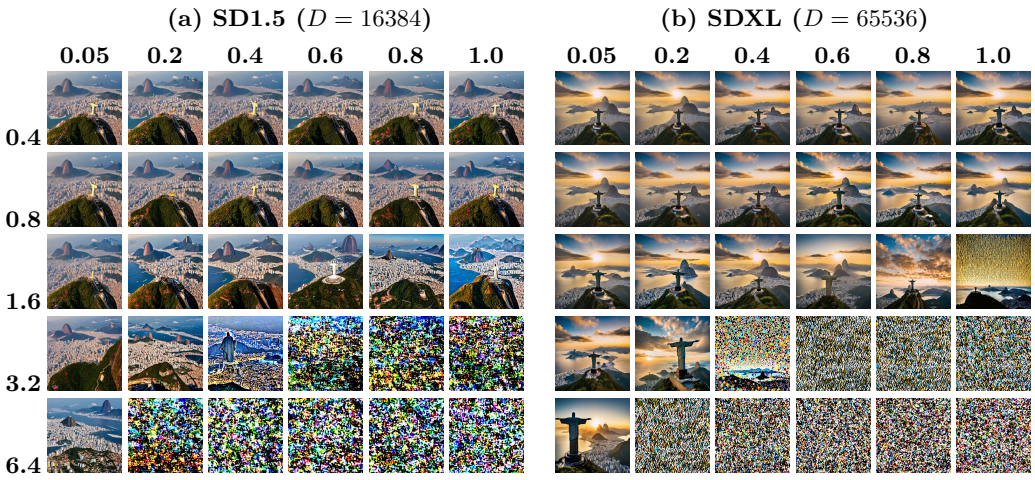


Figure 8: TRS perturbation landscape comparison: Interaction between **perturbation length** (rows) and **masking probability** (columns) for SD1.5 and SDXL. For both we used the prompt: "A breathtaking view from behind the Cristo Redentor (Christ the Redeemer) statue in Rio de Janeiro, Brazil, with layered mountains stretching into the distance and the sparkling Atlantic Ocean clearly visible below with blue color; warm golden-hour light, atmospheric haze, ultra-detailed, cinematic, wide-angle landscape, beautiful and serene and the city below."

Metrics. To assess the effectiveness of different budget allocations, we utilize ImageReward as the reward function. We report the final mean best rewards achieved across all 55 prompts, tracking how the optimization efficiency changes as the ratio of warm-up to main optimization varies.

Results. As shown in Figure 9a, TRS achieves peak performance when the warm-up phase accounts for 10% to 20% of the total compute budget across all tested settings. We note that while performance is sensitive to very small warm-up budgets, it degrades only slowly when the warm-up phase is slightly larger than the optimal range. This suggests that the exact choice is not overly critical, provided the initialization period is sufficient; consequently, we set the warm-up fraction to 20% for all main experiments.

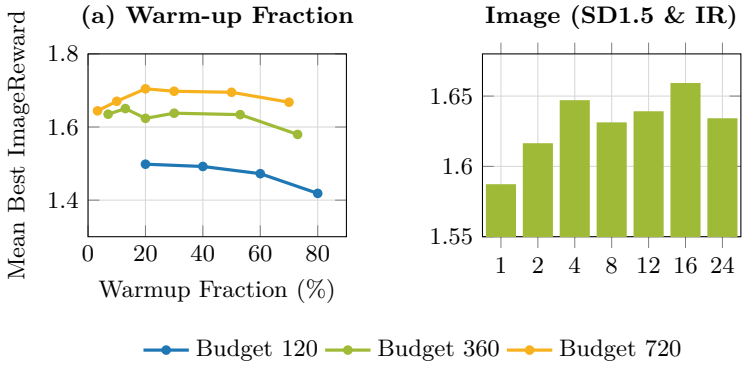


Figure 9: **Ablation Studies.** (a) Impact of the warm-up phase fraction across different compute budgets. (b) Sensitivity of TRS to the number of regions k. Performance remains stable across a wide range of k values.

F.4 Number of Regions

Setup. We ablate the impact of the number of trust regions $k \in \{1, 2, 4, 8, 16, 24\}$ on the optimization performance for text-to-image generation. These experiments use SD1.5 on a

55-prompt subset of DrawBench, utilizing a fixed batch size of $B = 24$. The optimization process is partitioned into 3 warm-up iterations followed by 12 trust-region iterations.

Metrics. Consistent with our previous ablations, we align the optimization to ImageReward. We evaluate the final mean best rewards across the prompt subset to identify the optimal range for k and to assess how the number of regions interacts with the total evaluation budget and batch size.

Results. As illustrated in Figure 9b, using a very small number of regions leads to sub-optimal performance. However, we find that selecting k in a medium range between 1 and the batch size B consistently yields strong results. These findings suggest that k is not a highly sensitive hyperparameter, as performance remains stable across a wide range of values, indicating that it does not require extensive per-task tuning. Since k is the only hyperparameter that was changed between the main experiments in Section 3, we conclude that TRS is a highly robust algorithm for noise optimization of flow and diffusion models.

G ODE vs. SDE Sampling Dynamics

Trust-Region Search (TRS) is a black-box optimization method that relies on local exploration to find better samples. This approach is more effective with ODE sampling because the deterministic paths provide a clear sense of locality. However, since SDE sampling with noise reduction leads to higher designability in proteina Geffner et al. (2025), we want to see how our algorithm handles the trade-off between the designability gains of SDEs and the diversity preserved by ODEs. We also include experiments for Text-to-Image (T2I) generation to see if these sampling dynamics remain consistent across different model types.

Setup We compare TRS against standard random search (RS) in both deterministic and stochastic settings. For protein generation, we generate 100 proteins of length 50, optimizing specifically for designability. For T2I generation, we use the same setup as in Section 3, utilizing the DDIM scheduler with both a stochastic setting ($\eta = 1.0$) and a deterministic setting ($\eta = 0.0$). A key difference between these setups is the inference budget: proteina uses 400 steps, while the T2I model uses 50.

Metrics We evaluate the results based on designability, diversity, and novelty. While designability is the primary goal, diversity and novelty are crucial for detecting mode-collapse, which is a known risk when using SDE-based noise reduction. These distributions are visualized in Figure 10 using rank–size and cumulative mass plots to show how concentrated the generated samples are.

Results and Discussion As shown in Table 7, SDE sampling does lead to higher designability, but it significantly hurts diversity and novelty. When we compare TRS with ODE to a random search and TRS with SDE ($\gamma = 0.6$), we observe, that the designability metrics are on par, while TRS with ODE shows much better diversity and novelty metrics. This becomes especially clear, when observing the Rank-1 metric, where we see, that SDE optimized samples, where 54 or even 68% of the samples belong to the same cluster. We also find that most current SDE-based scaling methods are not practical for proteina, either because they lack necessary value estimations Li et al. (2025b); Singhal et al. (2025) or because they are too slow without efficient batch utilization Jain et al. (2025), especially, when a lot of inference steps are necessary and the reward function is expensive. Interestingly, TRS shows a smaller advantage over random search in proteina when using SDEs compared to T2I. This is likely because the high number of inference steps in proteina allows the stochastic noise to eventually override the local search signal provided by the trust region.

Algorithm	T2I	Protein (SDE with $\gamma = 0.6$)				
	ImageReward↑	Design.↑	Rank-1 ↓	Clust. Div.↑	TM Div.↓	Nov.↓
RS + ODE	1.43	0.53	<u>37</u>	<u>0.27</u>	0.68	**0.85**
RS + SDE	1.45	**0.66**	68	0.19	0.73	0.94
TRS (Ours) + ODE	**1.62**	<u>0.65</u>	**18**	**0.30**	**0.67**	<u>0.89</u>
TRS (Ours) + SDE	<u>1.52</u>	**0.66**	54	<u>0.27</u>	<u>0.70</u>	0.92

Table 7: Comparison of ODE and SDE variants in protein design and T2I.

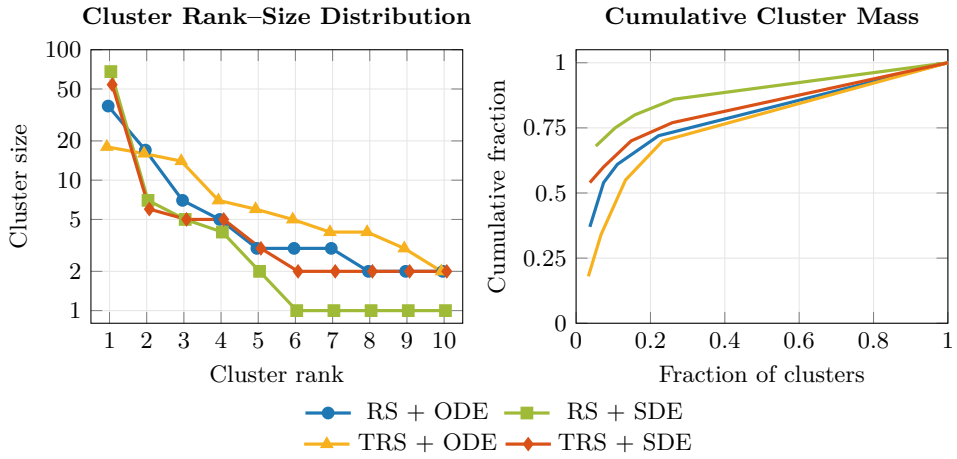


Figure 10: Protein cluster distributions. The rank–size plot (left) shows that ODE methods maintain higher diversity, whereas SDE methods suffer from mode-collapse. The cumulative mass (right) confirms that TRS + ODE provides the most balanced distribution of samples.

H Baselines

H.1 Gradient-based guidance

Optimal Control Flow (OC-Flow). OC-Flow (Wang et al., 2025) provides a theoretically grounded, training-free framework for guided flow matching by framing generation as an optimal control problem. It augments pre-trained flow dynamics with a time-dependent control term $\mathbf{u}_t$: $\dot{\mathbf{x}}_t = \mathbf{v}_t(\mathbf{x}_t) + \mathbf{u}_t$. The framework seeks to minimize a cost functional $J(\mathbf{u}) = R(\mathbf{x}_1) + \int_0^1 \frac{1}{2\lambda}\|\mathbf{u}_t\|^2 dt$, comprising a terminal reward loss $R(\mathbf{x}_1)$ and a quadratic running cost that regulates the trajectory's deviation from the prior distribution. Leveraging Pontryagin's Maximum Principle, the control trajectory is optimized iteratively over a fixed number of steps using either SGD or L-BFGS. In each iteration, gradient information is propagated backward via a co-state flow $\boldsymbol{\mu}_t$ to update the control parameters with a step size η and weight decay. To ensure trajectory regularity and stable convergence, a weight constraint is enforced on the magnitude of the control term throughout the optimization process.

H.2 Noise sequence search

Diffusion Tree Sampling (DTS). DTS (Jain et al., 2025) frames the inference-time alignment of diffusion models as a tree-structured optimization problem over the denoising sequence. The framework employs a recursive value-based search guided by a soft value function $V(\mathbf{x}_t)$, which is estimated via a soft-Bellman backup with an exploration constant λ:

$$V(\mathbf{x}_t) = \frac{1}{\lambda} \log \mathbb{E}_{\mathbf{x}_{t-1} \sim p_\theta(\cdot|\mathbf{x}_t)} \left[\exp(\lambda V(\mathbf{x}_{t-1}))\right]$$

At each node, the search proceeds for a fixed number of expansion steps, where the selection of trajectories is governed by a specific exploration type (e.g., UCT). To effectively navigate the continuous branching space of the diffusion process, the algorithm utilizes progressive widening to determine the number of children k for a node $\mathbf{x}_t$ based on its visit count $N(\mathbf{x}_t)$:

$$k(\mathbf{x}_t) = \lceil C \cdot N(\mathbf{x}_t)^\alpha \rceil$$

where C is the progressive width constant and α is the progressive width exponent. A stochastic rollout parameter ρ serves as a decision gate: with probability ρ, the algorithm performs a full rollout to the terminal state $t = 0$ to obtain an exact reward, while with probability $1 - \rho$, it continues recursive tree expansion.

Fast Direct Fast Direct (Tan et al., 2025) is an efficient trajectory-level optimization method designed for black-box guidance. It bypasses step-wise branching by optimizing the complete noise sequence $\{\boldsymbol{\epsilon}_t\}_{t=1}^T$ simultaneously. The method identifies a pseudo-target $\mathbf{x}^*$ on the manifold via Gaussian Process (GP) regression over previous evaluations and computes a universal direction to update the entire sequence:

$$\boldsymbol{\epsilon}_t^{\text{new}} = \text{Norm}(\boldsymbol{\epsilon}_t^{\text{old}} + \alpha(\mathbf{x}^* - \mathbf{x}_0))$$

where α is the step size. This global refinement allows it to converge to high-reward regions in fewer iterations than per-step filtering methods.

H.3 Source noise search

Random Search. Is the simplest search-based method one can apply, where N source noises $\mathbf{x}_0$ are randomly sampled and evaluated. This technique can be used both with ODE- and SDE-based samplers.

Zero-Order Search. Zero-order Search (Ma et al., 2025) can be seen as a special case of our algorithm. Like ours, it iteratively refines the initial latent noise to maximize a target reward. The algorithm begins by sampling an initial set of B Gaussian noise vectors $\{\mathbf{x}_0^j\}_{j=1}^B$ and selecting the candidate that yields the highest reward as the initial center $\mathbf{x}_0^{\text{c}}$. This is equivalent to using only one warm-up iteration in TRS and setting the number of regions to one. In each subsequent search iteration, B new candidates are generated by perturbing the current center:

$$\mathbf{x}_0^{\text{new}} = \mathbf{x}_0^{\text{c}} + \varepsilon \cdot \boldsymbol{\delta}, \quad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

where ε (scalar) controls the radius of the local neighborhood search. This perturbation scheme is similar to using TRS without probability masks and choosing a fixed region length. Similar to ours, this technique works best for ODE samplers.

I Metrics

In this section, we describe all additional evaluation metrics used in this work that are not employed as reward functions. The reward-based metrics are detailed in Appendix E. We consider evaluation criteria for both molecule generation and protein design.

I.1 Molecule Generation

For the evaluation of 3D molecule generation, we follow the methodology described by Song et al. (2023), which assesses the physical and chemical plausibility of the generated Cartesian coordinates and atom types.

Stability We evaluate the structural integrity of the generated samples using *Atom Stability* and *Molecule Stability*. Following the convention of Hoogeboom et al. (2022), chemical bonds are inferred based on the Euclidean distances between atoms using a threshold-based lookup table of covalent radii. An individual atom is considered stable if its inferred bond count matches its expected chemical valency (e.g., 4 for carbon, 1 for hydrogen). *Molecule Stability* is then defined as the percentage of generated molecules for which all constituent atoms are stable. This metric serves as a strict proxy for the geometric consistency of the generated 3D structures.

Validity and Uniqueness The chemical validity of a molecule is determined by its ability to be successfully parsed into a molecular graph using RDKit Landrum et al. (2026). We report the *Valid Fraction*, which measures the proportion of generated samples that satisfy fundamental chemical connectivity and valency constraints without requiring post-hoc corrections. Additionally, we report the *Unique Fraction*, which indicates the percentage of unique molecules among all generated compounds as determined by RDKit. In addition, we report the *Valid and Stable Fraction*, defined as the percentage of molecules that are both valid according to RDKit and satisfy the Molecule Stability criterion defined above. This combined metric provides a stringent indicator of the model's ability to generate chemically realistic, unique, and geometrically consistent 3D molecules.

I.2 Protein Design

To evaluate the structural quality and variability of the generated protein backbones, we adopt the metric suite introduced by Geffner et al. (2025). These metrics assess two complementary aspects: the uniqueness of the generated structures relative to known proteins (*Novelty*) and the structural variability within the generated set (*Diversity*). Following (Geffner et al., 2025), we compute these scores only for designable samples, which means, that scRMSD $>$ 2.0 Å or equivalently the designability needs to be > 0.1353.

Novelty Novelty measures the extent to which the model generates protein folds that differ from those observed in existing experimental and predicted structure databases. For each generated, designable backbone structure, we compute the maximum TM-score against all entries in a reference set using Foldseek van Kempen et al. (2024). We then report the average of these maximum TM-scores across the entire sample set. Lower average maximum TM-scores indicate higher structural novelty. We evaluate novelty with respect to the PDB dataset Berman et al. (2000).

Diversity We quantify the internal diversity of the generated, designable backbone samples using two complementary metrics:

1. **Average Pairwise TM-score:** We compute the mean pairwise TM-score between all designable backbone samples for each generated protein length. These values are subsequently aggregated to obtain a global average. Since the TM-score measures structural similarity on a scale from 0 to 1, lower values correspond to higher diversity.
2. **Cluster Ratio:** We cluster the generated backbones using Foldseek with a TM-score threshold of 0.5. The diversity score is defined as the ratio of the number of unique

J Runtime Comparison

We analyze the runtime of the T2I experiment in section 3 and the additional experiments of Appendix D for all methods we use on a single NVIDIA A100-SMX4-40GB. We plot this in Figure J, where we can see for all the budgets we have used. We refer here to the settings, where ImageReward is used to T2I, R_6 for molecules and the designability for proteins with 50 residues. We observe that DTS* generally has the highest compute time. This arises as the algorithm is sequential and difficult to realize with a fixed batch size As noted in Appendix H.2, the batch implementation is not used to allow other packages for speed up, as in the code, provided by the authors. In our comparisons we observe that both version are almost 4× slower than TRS. For OC-Flow, which uses a batch size of 1, the computation time is high in SD1.5, since we back-propagate through 50 steps, but for the distilled SDXL lightning it turns out to be faster, while marking very high memory usage. In molecule generation, it is the slowest method, where we use it for the choice of hyperparameters in Table 4. The memory usage is lower here, compared to TRS and the other sampling-based methods, which however use a batch size of 100. In Fast Direct, we use slightly higher NFE budgets since the iterative nature of increasing steps, makes it difficult to set the exact same NFE, while remaining the batch size. Apart from that, the computation times and memory usage are only slightly larger, as their GP is cheap. Random Search, Zero-Order Search

and TRS are generally most efficient, as their compute time consists almost purely of the generative model and reward function evaluations.

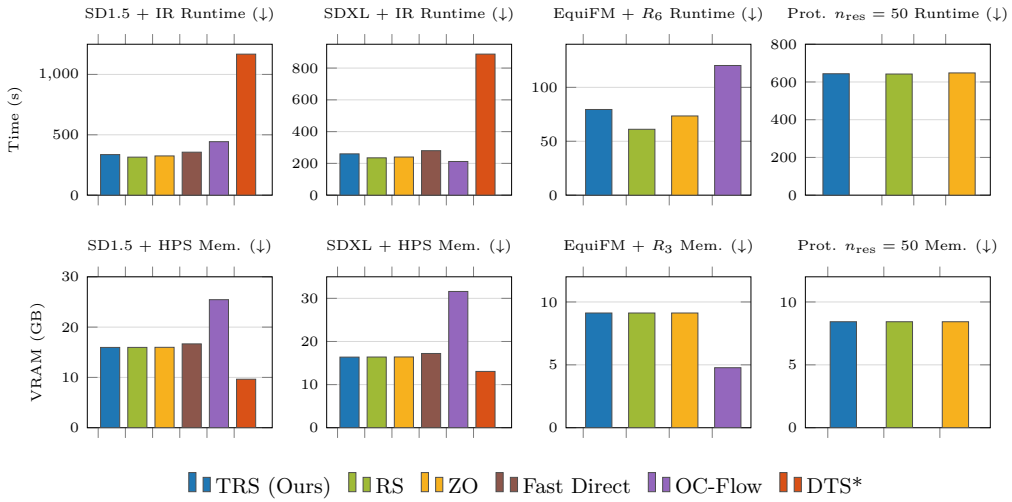


Figure 11: Comparison of runtime and memory consumption across methods on a single NVIDIA A100-SXM4-40GB.

K Limitations

Reward Functions. A primary limitation of these methods lies in the reliability of reward functions; when derived from neural network predictors, performance is inherently constrained by these models. In our experiments, reliance on pretrained networks for feedback makes optimization susceptible to model bias and failure modes. Figure 12 illustrates this via TRS-optimized SDXL images that receive high rewards despite failing to capture the core prompt intent. This highlights a fundamental challenge: reward models may score samples highly that exploit shortcuts in the learned signal rather than satisfying the true objective. To address this, recent work has focused on larger, more accurate reward models (Wu et al., 2025). TRS is well-positioned to integrate with these advances, offering a flexible and scalable framework as the field progresses toward more expressive models.

Diversity. While TRS begins with a global exploration phase across multiple promising regions, it ultimately converges to and exploits the most dominant region. Our experimental results demonstrate that this is an effective strategy for identifying a single optimal sample that maximizes the reward objective. However, TRS may be less suited for tasks requiring a diverse ensemble of samples. A direction for future work would be extending the framework to optimize for a diverse set of high-reward samples rather than a single point estimate. For instance, diversity could be explicitly enforced in TRS by incorporating a cosine similarity constraint on the center noise vectors of the trust regions.

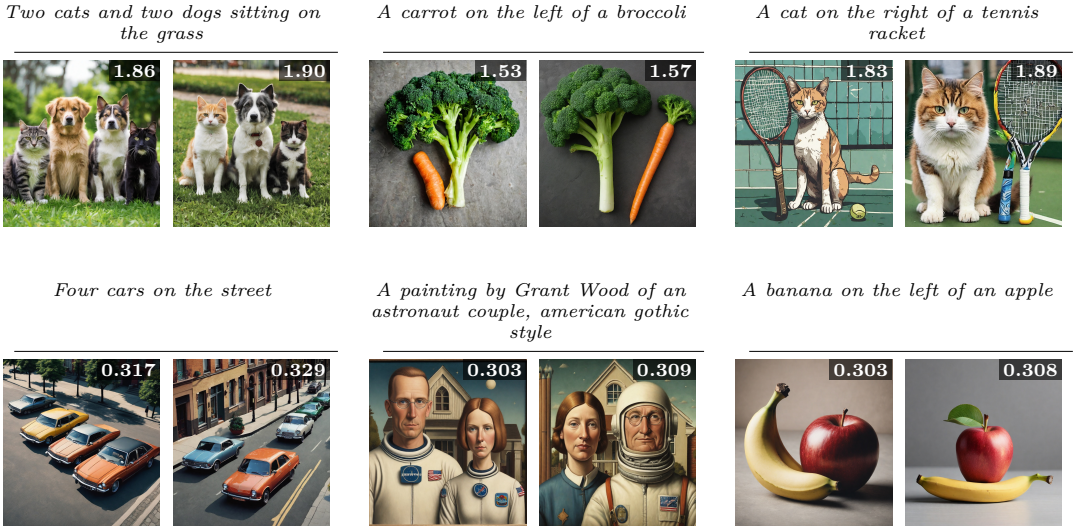


Figure 12: Visualizing reward misalignment: cases where higher reward values are assigned even though the model fails to correctly generate the specific conceptual or spatial requirements of the prompt. All images are generated with SDXL. The upper row shows missalignment to ImageReward and the lower one to HPSv2.

L Optimized samples

We show some examples of the different methods we investigate in this paper in Figure 13 after optimizing SDXL with HPSv2 for 20k NFE. All prompts are contained in DrawBench.

Figure 13: Visualizing some examples of experiment 3 for all methods with SDXL. The first 4 rows are optimized with ImageReward and the lower 5 rows with HPSv2.