Training-Free Guidance Beyond Differentiability: Scalable Path Steering with Tree Search in Diffusion and Flow Models

Yingqing Guo * **Yukang Yang** * **Hui Yuan** * **Mengdi Wang**

Princeton University

Abstract

Training-free guidance enables controlled generation in diffusion and flow models, but most methods rely on gradients and assume differentiable objectives. This work focuses on training-free guidance addressing challenges from non-differentiable objectives and discrete data distributions. We propose **TreeG**: Tree Search-Based Path Steering Guidance, applicable to both continuous and discrete settings in diffusion and flow models. TreeG offers a unified framework for training-free guidance by proposing, evaluating, and selecting candidates at each step, enhanced with tree search over active paths and parallel exploration. We comprehensively investigate the design space of TreeG over the candidate proposal module and the evaluation function, instantiating TreeG into three novel algorithms. Our experiments show that TreeG consistently outperforms top guidance baselines in symbolic music generation, small molecule design, and enhancer DNA design with improvements of 29.01%, 26.38%, and 18.43%. Additionally, we identify an inference-time scaling law showing TreeG's scalability in inference-time computation.[2]

1 Introduction

During the inference process of diffusion and flow models, guidance methods steer generations toward desired objectives, achieving remarkable success in vision [15, 25], audio [35], biology [45, 80], and decision making [1, 12]. In particular, training-free guidance offers high applicability by directly controlling the generation process with off-the-shelf objective functions without extra training [53, 81, 4, 24]. Most training-free guidance methods are gradient-based, leveraging the objective's gradient to steer inference, and thus assume the objective function is differentiable [78, 22].

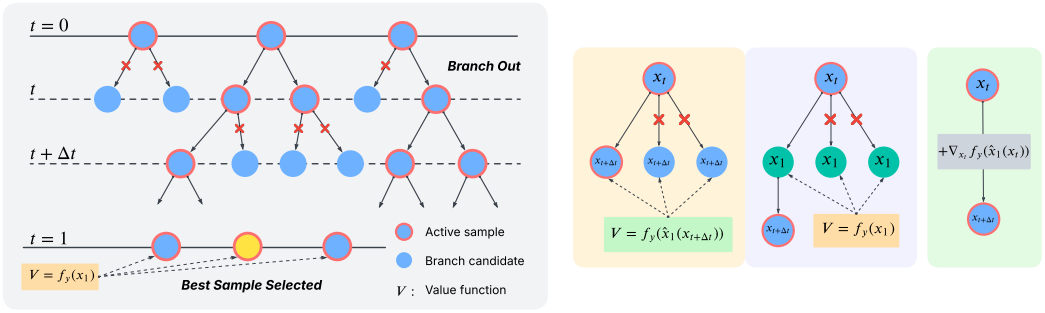
However, recent advances have pushed the boundaries of guided generation beyond differentiability: guidance objectives have expanded to include non-differentiable goals [30, 1]; diffusion and flow models have shown strong performance on discrete data [2, 67], where objectives are inherently non-differentiable unless approximated via differentiable features [36, 77]. In such settings, training-free methods designed for differentiable objectives face fundamental limitations. Yet, the design space beyond differentiability remains under-explored: only a few methods exist [38, 30], they differ significantly and appear disconnected from prior guidance principles [38, 13]. This underscores the need for a unified perspective and comprehensive study of guidance beyond differentiability. To this end, we propose an algorithmic framework **TreeG**: Tree Search-Based Path Steering Guidance, designed for both diffusion and flow models across continuous and discrete data spaces.

TreeG is based on path steering guidance, providing a unified perspective on training-free guidance beyond differentiability. Let $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_t, \boldsymbol{x}_{t+\Delta t}, \cdots, \boldsymbol{x}_1$ denote the inference trajectory of a

*Equal contribution. Department of Electrical and Computer Engineering, Princeton University. Authors' emails are: {yg6736, yy1325, huiyuan, mengdiw}@princeton.edu.

[2]Code available at https://github.com/yukang123/UniTreeG.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) Tree-Search Inference Process

(b) Designs for `BranchOut` and Value Function V, gradient-based guidance

Figure 1: TreeG Overview: (a) An **active set** of size A is maintained, where each sample branches into K candidates. At each step, the top A candidates are retained, and the best sample is selected at the final step. (b) Left: The **current state**-based (`BranchOut`, V) evaluates candidates via a lookahead estimate of the clean sample (Sec. 5.1). Middle: branching and selection occur in the destination state space (Sec. 5.2). Right: Gradient-based guidance can be applied to the current state branch-out module when a differentiable objective predictor is available (Sec. 5.3).

diffusion or flow model. While gradients of the objective, when available, can offer a precise direction to steer inference at each step, an alternative is to search for a favorable path: propose multiple next state candidates $\boldsymbol{x}_{t+\Delta t}$ (via `BranchOut` module), evaluate them using a value function (denoted by V) that reflects the objective, and select the best candidate to proceed. This structured search enables TreeG using only zeroth-order information, making it applicable beyond differentiability.

TreeG adopts a tree-search mechanism to explore multiple trajectories, further enhancing the effectiveness of path steering guidance. An ***active set*** of size A is maintained at each inference step (Fig. 1(a)). Each sample in the active set branches into multiple candidate next states, from which the top A candidates, ranked by the value function V, are selected for the next step. This process iterates until the final step, where the best sample from the active set is chosen as the output. By enlarging A, TreeG achieves higher objective values while flexibly adapting to the computational budget.

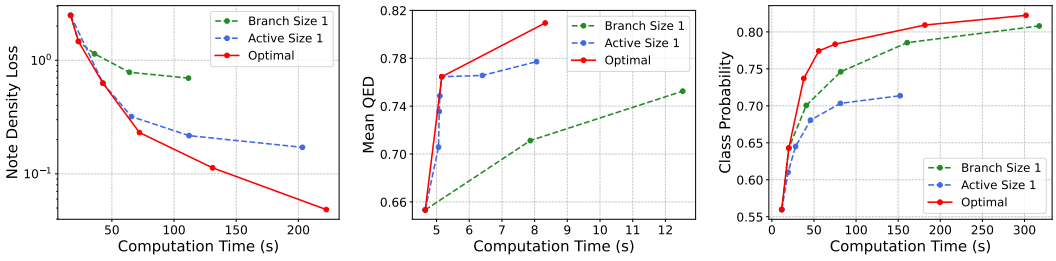
TreeG's design space is over the branching-out module and value function, offering flexible and comprehensive configurations. Effective search requires both efficient exploration and reliable evaluation. As shown in Fig. 1(b), we propose two compatible pairs of (`BranchOut`, V): one based on the current state ($\boldsymbol{x}_t$), the other on the predicted destination ($\hat{\boldsymbol{x}}_1$). The former uses the original diffusion model to generate multiple next states and evaluates them via a lookahead estimate of the clean sample. The latter generates candidate destinations, which indicate the orientation of the next state, and selects the optimal using an off-the-shelf objective function. The top-ranked destination determines the next state. In addition, TreeG introduces a novel gradient-based algorithm for guiding discrete flow models using gradients when a differentiable objective predictor is available.

Contributions. Our methodological contributions are as follows:

- We propose a novel tree search framework TreeG of training-free guidance. It applies to both continuous and discrete, diffusion and flow models (Sec. 4), and supports non-differentiable objectives.
- We instantiate three novel algorithms within this framework, and provide theoretical guarantees showing that they recover the posterior conditional distribution (Sec. 5).
- We show that existing sampling-based methods [30, 37] are special cases of TreeG (Sec. 5.1), limited to exploration at the current state. Our novel exploration strategy at destination states, combined with a comprehensive design space for candidate proposals and value functions, enhances the effectiveness and versatility.

Empirically, we demonstrate that TreeG:

- Outperforms existing guidance methods on diverse tasks: symbolic music generation (continuous diffusion with *non-differentiable* objectives), molecular design, and enhancer DNA design (both on *discrete* flow models). Path steering guidance, the special case of TreeG with the active set size as 1, consistently outperforms the strongest guidance baseline, yielding improvements of $29.01\%, 26.38\%$, and 18.43% respectively (Sec. 6.2).



(a) Symbolic Music (Loss ↓) (b) Small Molecule (QED ↑) (c) Enhancer DNA (Prob ↑)

Figure 2: **Inference-Time Scaling Law of TreeG.** "Optimal" refers to the best-performing combination of active set and branch-out sizes under the same inference time. The results demonstrate the scalability of TreeG, validating the effectiveness of its multi-active-path and branch-out design.[3]

- Exhibits an *inference-time scaling law*, where performance consistently improves as inference-time computation increases by larger active set and branch-out sizes (Fig. 2 and Sec. 6.3).

2 Related Work

We review the most relevant work on *inference-time guidance* here; see App. A for additional work.

Derivative-Free. Huang et al. [30] and Li et al. [37] guide sampling toward non-differentiable objectives using soft values derived from optimal control. These are special cases of our TreeG-SC with an active set size of one. Notably, our TreeG-SD (Sec.5.2) introduces a novel strategy that proposes candidates at the *end* of the sampling trajectory in the clean space, unexplored by existing derivative-free methods. This novel perspective, along with our flexible design for candidate proposals and value functions, underscores the broader versatility and applicability of our framework.

On Discrete Models. Nisonoff et al. [45] extends classifier(-free) guidance [15, 25] to discrete settings, requiring training time-dependent classifiers. Lin et al. [38] reweights the rate matrix using objective values. All three of our algorithms apply to discrete models, including a gradient guidance method we propose for cases with a differentiable objective predictor.

Inference-Time Scaling. The idea of searching across multiple inference paths has also been touched upon by two concurrent works [42, 65]. Our TreeG provides the first systematic study of the design space of tree search, offering a novel methodology for both exploration and evaluation.

3 Preliminaries

Notations. Bold symbols $\boldsymbol{x}$ denote high-dimensional vectors, while x indicates scalars. Superscripts like $x^{(d)}$ refer to the d-th dimension, whereas x^i or $x^{i,j}$ denote independent samples indexed by i and j. p_t is the density of intermediate training distributions. For inference, $\mathcal{T}_t$ represents the sample.

3.1 Diffusion and Flow Models

Diffusion and flow models learn to reverse a transformation from a data distribution p_{data} to a noise distribution p_0. Let $p_1 = p_{\text{data}}$, and define intermediate distributions p_t for $t \in (0,1)$ that progressively corrupt p_1 into p_0 over T uniform timesteps ($\Delta t = 1/T$). Generation starts by sampling $\boldsymbol{x}_0 \sim p_0$, then iteratively samples $\boldsymbol{x}_t \sim p_t$ for $t = i/T$, where $i \in [T]$, ultimately yielding $\boldsymbol{x}_1 \sim p_1$.

Diffusion and flow models are equivalent [39, 16]. We focus on the widely used continuous diffusion models and discrete flow models, denoted u_θ^{diff} and u_θ^{flow}, and refer to both as u_θ or diffusion models when clear from context. We next review the two models in more detail.

Diffusion Models. For diffusion models applied to continuous data, given a data sample $\boldsymbol{x}_1 \sim p_1$, the noisy sample at timestep $t = i/T$ ($i \in [T]$) is constructed as $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_1 + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\{\bar{\alpha}_t\}$ are pre-defined monotonically increasing parameters that control the noise level. The diffusion model $u_\theta^{diff} : \mathcal{X} \times [0,1] \to \mathcal{X}$ parameterized by θ, estimates the noise added to $\boldsymbol{x}_t$, it's equivalent to learning the score function of $p_t(\boldsymbol{x}_t)$ [26, 54]:

$$u_\theta^{diff} = \underset{u_\theta}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}_1 \sim p_1, \epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} ||u_\theta(\boldsymbol{x}_t, t) - \epsilon||^2 = -\sqrt{1-\bar{\alpha}_t}\nabla \log p_t \tag{1}$$

[3] Please refer to App. F.4 for the experimental setup of Fig. 2.

For sampling, we begin with $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and iteratively apply the DDPM sampling step [26]:

$$\boldsymbol{x}_{t+\Delta t} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t + (1-\alpha_t)\nabla \log p_t(\boldsymbol{x}_t)\right) + \sigma_t \epsilon = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} u_\theta^{\textit{diff}}(\boldsymbol{x}_t, t)\right) + \sigma_t \epsilon, \tag{2}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, $\alpha_t = \bar{\alpha}_t / \bar{\alpha}_{t+\Delta t}$ and $\sigma_t = \sqrt{1-\alpha_t}$. This step (2) can be restated as sampling from a distribution centered on a linear interpolation of $\boldsymbol{x}_t$ and $\boldsymbol{x}_{1|t}$:

$$\boldsymbol{x}_{t+\Delta t} = c_{t,1}\boldsymbol{x}_t + c_{t,2}\boldsymbol{x}_{1|t} + \sigma_t \epsilon, \tag{3}$$

where $c_{t,1}$ and $c_{t,2}$ are constants[4], and with $\boldsymbol{x}_{1|t}$ being an estimation of the conditional expectation $\mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{x}_t]$ based on Tweedie's formula [18]:

$$\boldsymbol{x}_{1|t} := \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t} u_\theta^{\textit{diff}}(\boldsymbol{x}_t, t)\right). \tag{4}$$

Flow Models. For flow models applied to discrete data [10], suppose data space is $\mathcal{X} = [S]^D$, where D is the dimension and S is the number of states per dimension. An additional mask state M is introduced as the noise prior distribution. Given a data sample $\boldsymbol{x}_1$, the intermediate distributions are constructed by $p_{t|1}(\boldsymbol{x}_t|\boldsymbol{x}_1) = \Pi_{d=1}^D p_{t|1}(x_t|x_1)$ with $p_{t|1}(x_t|x_1) = t\delta\{x_1, x_t\} + (1-t)\delta\{M, x_t\}$. The flow model $u_\theta^{\textit{flow}}$ estimates the true denoising distribution $p_{1|t}(\boldsymbol{x}_1|\boldsymbol{x}_t)$. Specifically, it's defined as $u_\theta^{\textit{flow}} = (u_\theta^{(1)}, \ldots, u_\theta^{(D)})$, where each component $u_\theta^{(d)}(x_1|\cdot)$ is a function $\mathcal{X} \times [0,1] \to \Delta([S])$. Here, $\Delta([S])$ represents the probability distribution over the set $[S]$ The training objective is:

$$u_\theta^{\textit{flow}} = \operatorname*{argmax}_{u_\theta} \mathbb{E}_{\boldsymbol{x}_1 \sim p_1, \boldsymbol{x}_t \sim p_{t|1}}\left[\log u_\theta^{(d)}(x_1^{(d)}|\boldsymbol{x}_t)\right]. \tag{5}$$

For generation, it requires the rate matrix:

$$R_t^{(d)}(\boldsymbol{x}_t, j) = \mathbb{E}_{x_1^{(d)} \sim p_{1|t}^{(d)}}\left[R_t\left(x_t^{(d)}, j|x_1^{(d)}\right)\right] = \mathbb{E}_{x_1^{(d)} \sim u_\theta^{(d)}(x_1|\boldsymbol{x}_t)}\left[R_t\left(x_t^{(d)}, j|x_1^{(d)}\right)\right], \tag{6}$$

where the pre-defined conditional rate matrix can be chosen as the popular: $R_t(x_t, j|x_1) = \frac{\delta\{j, x_1\}}{1-t}\delta\{x_t, M\}$. The generation process can be simulated via Euler steps [58]:

$$x_{t+\Delta t}^{(d)} \sim \text{Cat}\left(\delta\{x_t^{(d)}, j\} + R_t^{(d)}(\boldsymbol{x}_t, j)\Delta t\right), \tag{7}$$

where $\delta\{k, j\}$ is the Kronecker delta which is 1 when $k = j$ and is otherwise 0.

3.2 Objective

The objective is to sample from the conditional distribution $p(\boldsymbol{x} \mid y)$, where y denotes a desired property, using a pre-trained diffusion model that generates samples from unconditional distribution $p(\boldsymbol{x})$. The extent to which a sample satisfies the property y is quantified by an objective function $f_y : \mathcal{X} \to \mathbb{R}$, where $f_y(\boldsymbol{x}) = \log p(y \mid \boldsymbol{x})$ and $\boldsymbol{x}$ in the clean data space. We aim to sample from:

$$p(\boldsymbol{x} \mid y) \propto p(\boldsymbol{x})\, p(y \mid \boldsymbol{x}) \propto p(\boldsymbol{x})\, \exp(f_y(\boldsymbol{x})).$$

Given the training objectives defined in (1) and (5) for conditional counterparts $\boldsymbol{x}_1 \sim p_1(\boldsymbol{x} \mid y)$, the objective translates to estimating the conditional score or rate matrix:

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid y); \quad R_t^{(d)}(\boldsymbol{x}_t, j \mid y) = \mathbb{E}_{x_1^{(d)} \sim p_{1|t}^{(d)}(\boldsymbol{x}_1|y)}\left[R_t\left(x_t^{(d)}, j|x_1^{(d)}\right)\right], \tag{8}$$

which enables the sampling step (2) or (7) to use the conditional counterparts and generate the next state $\boldsymbol{x}_{t+\Delta t}$ following the conditional transition distribution $\mathcal{T}_t(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t, y)$.

We assume a *perfect* pre-trained diffusion model, where u_θ exactly minimizes the training objectives in (1) or (5). Our focus is on ***training-free*** methods that neither fine-tune u_θ nor train a time-dependent classifier aligned with the diffusion noise schedule.

4 TreeG: Tree Search-Based Path Steering Guidance

While gradients provide precise direction when available [22], search offers an alternative when they are not: proposing multiple candidates, evaluating them via a value function aligned with the objective, and selecting the best to proceed. We demonstrate the tree search framework in this section.

[4] We have $c_{t,1} = \sqrt{\alpha_t}(1-\bar{\alpha}_{t+\Delta t})/(1-\bar{\alpha}_t)$, $c_{t,2} = \sqrt{\bar{\alpha}_{t+\Delta t}}(1-\alpha_t)/(1-\bar{\alpha}_t)$.

4.1 Algorithmic Framework

Let $\boldsymbol{x}_0, \cdots, \boldsymbol{x}_1$ denote the inference path from pure noise to a clean sample. At each step (Alg. 1), the `BranchOut` module proposes K candidate states, among which some candidates are selected into the active set. While the basic approach tracks a single path, the active set size A can be increased to explore multiple paths via tree search. The selection step can be implemented as ranking candidates by their values and selecting the top, or resampling candidates with probabilities proportional to their values. In following sections, we will show that TreeG with resampling enjoys theoretical guarantees, while empirically, experiments (please refer to App. H.3) demonstrate that selection by ranking is superior to resampling.

Algorithm 1 TreeG: Tree Search-Based Path Steering Guidance

1: **Input:** diffusion model u_θ, branch out policy and value function $(\texttt{BranchOut}, V)$, objective function f_y, active set size A, branch out sample size K.
2: **Initialize:** $t = 0$, $\mathcal{A} = \left\{\boldsymbol{x}_0^1, \ldots, \boldsymbol{x}_0^A\right\}$, $\boldsymbol{x}_0^i \sim p_0$.
3: **while** $t < 1$ **do**
4: **Propose candidates for next step:** For $\boldsymbol{x}_t^i \in \mathcal{A}$, $\boldsymbol{x}_{t+\Delta t}^{i,j} \sim \texttt{BranchOut}\left(\boldsymbol{x}_t^i, u_\theta\right)$, $j \in [K]$.
5: **Select:** select A candidates (by ranking or resampling [5]) with respect to the value function $V\left(\boldsymbol{x}_{t+\Delta t}^{i,j}, t, f_y\right)$, $i \in [A], j \in [K]$: $\boldsymbol{x}_{t+\Delta t}^{i_1,j_1}, \ldots, \boldsymbol{x}_{t+\Delta t}^{i_A,j_A}$.
6: **Update the active set:** $\mathcal{A} = \left\{\boldsymbol{x}_{t+\Delta t}^{i_1,j_1}, \ldots, \boldsymbol{x}_{t+\Delta t}^{i_A,j_A}\right\}$
7: $t \leftarrow t + \Delta t$.
8: **end while**
9: **Output**: $\boldsymbol{x}_1^* = \operatorname{argmax}_{\boldsymbol{x}_1 \in \mathcal{A}} f_y\left(\boldsymbol{x}_1\right)$.

In Alg.1, the `BranchOut` module and value function V are two core components that require careful design, for which we propose novel designs in the next section. Those specifications of `BranchOut` and V lead to new algorithms that outperform existing baselines (Sec.6).

5 Design Space of TreeG

This section explores the design space of TreeG, focusing on the $(\texttt{BranchOut}, V)$ pair. We introduce two compatible pairs that operate by sampling and selecting from either the current state or the predicted destination. Additionally, we present a gradient-based discrete guidance method.

5.1 Sample-then-Select on Current States

The target conditional score and rate matrix in (8) relate to their unconditional counterparts as follows:

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid y) = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t), \quad \text{(by Bayes' rule)}$$

$$R_t(\boldsymbol{x}_t, \boldsymbol{x}_t' \mid y) = \frac{p_t(y \mid \boldsymbol{x}_t')}{p_t(y \mid \boldsymbol{x}_t)} \cdot R_t(\boldsymbol{x}_t, \boldsymbol{x}_t'), \quad \text{(by [45])}$$

Both expressions depend on $p_t(y \mid \boldsymbol{x}_t)$, which is key to adapting the unconditional score or rate matrix, estimated by a pre-trained diffusion model, to their conditional counterparts.

The idea is to use the original backward process to generate multiple candidate states at time t, then prioritize samples with higher $p_t(y \mid \boldsymbol{x}_t)$ using it as a value function. We approximate $p_t(y \mid \boldsymbol{x}_t)$ as:

$$p_t(y \mid \boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x}_1 \sim p_{1|t}} p(y \mid \boldsymbol{x}_1) = \mathbb{E}_{\boldsymbol{x}_1 \sim p_{1|t}} \exp\left(f_y(\boldsymbol{x}_1)\right) \simeq \frac{1}{N} \sum_{i=1}^{N} \exp\left(f_y(\hat{\boldsymbol{x}}_1^i)\right). \tag{9}$$

Based on this, we propose the $(\texttt{BranchOut}, V)$ pair operating on current states as follows:

We refer to instantiating Alg. 1 with Module 1 and Value Function 1 as **TreeG-Sampling Current**, abbreviated as **TreeG-SC**. We demonstrate that stochastic control guidance (SCG) [30] and soft value decoding guidance (SVDD) [37] are special cases of TreeG-SC (App. E.1).

The following theorem (proof in App. D.1) shows that TreeG-SC yields a distribution $\hat{\mathcal{T}}$ that closely approximates the target conditional transition distribution $\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)$.

[5] Given a set of candidates $\{x_1, x_2, \ldots, x_n\}$ with associated nonnegative values $V_1, V_2, \ldots, V_n$, ranking involves selecting the top A candidates with the highest V_i values; Resampling defines $P(x_i) = V_i / \left(\sum_{j=1}^n V_j\right)$, and then samples A candidates according to $P(x_i)$.

Module 1 `BranchOut`-Current

1: **Input:** $\boldsymbol{x}_t$, diffusion model u_θ, time step t.
2: Sample the next state by the original generation process: $\boldsymbol{x}_{t+\Delta t} \sim$ (2) or (7).
3: **Output**: $\boldsymbol{x}_{t+\Delta t}$

Value Function 1 V : Current State Evaluator

1: **Input:** $\boldsymbol{x}_t$, diffusion model u_θ, objective function f_y, time step t, (optional) Monte-Carlo sample size N.
2: **Predict the clean sample:**
(continuous) $\hat{\boldsymbol{x}}_1 = \boldsymbol{x}_{1|t}$ in (4).
(discrete) $\hat{\boldsymbol{x}}_1^i \sim \text{Cat}\left(u_\theta(\boldsymbol{x}_t, t)\right), i \in [N]$.
3: **Evaluate:** $V(\boldsymbol{x}_t) = \frac{1}{N}\sum_{i=1}^{N} \exp\left(f_y(\hat{\boldsymbol{x}}_1^i)\right)$.
4: **Output:** $V(\boldsymbol{x}_t)$

Theorem 1. *Consider TreeG-Sampling Current at time t, with an active set of size one and selection performed via resampling. Then, for any $\varepsilon, \delta > 0$, it holds with probability $1-\delta$:*

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\ell < \varepsilon,$$

provided one of the following conditions is satisfied:
***(a, Continuous)** It holds under $\ell = 1$ norm. Suppose data follow Gaussian distribution and the objective function is linear. Branch-Out size $K = \Theta(\frac{\log(1/\delta)}{\varepsilon^2})$ and timestep satisfies $\alpha_t = 1 - O\left(\varepsilon^2\right)$.*
***(b, Discrete)** It holds under $\ell = \infty$ norm. The Branch-Out size is $K = \Theta(\frac{\log(|\mathcal{X}|/\delta)}{\varepsilon^2})$, Monte Carlo size is $N = \Theta(\frac{\log(|\mathcal{X}|/\delta)}{\varepsilon^2})$, and the timestep $\Delta t = O(\varepsilon)$, where $\mathcal{X}$ is the data space.*

5.2 Sample-then-Select on Destination States

During inference, the transition probability at each step is determined by the current state and the *end* state of the path, with the latter estimated by the diffusion model, as stated in the following lemma (proof in App. D.2).

Lemma 1. *In both continuous and discrete cases, the transition probability during inference at timestep t satisfies:*

$$\mathcal{T}(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t) = \mathbb{E}_{\hat{\boldsymbol{x}}_1}\left[\mathcal{T}^\star(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t, \hat{\boldsymbol{x}}_1)\right],$$

where the expectation is taken over a distribution estimated by u_θ, with $\mathcal{T}^\star$ being the true posterior distribution predetermined by the noise schedule.

In diffusion and flow models, $\mathcal{T}^\star$ is centered at a linear interpolation between its inputs: the current state $\boldsymbol{x}_t$, and the predicted destination state $\hat{\boldsymbol{x}}_1$, indicating that the orientation of the next state is partially determined by $\hat{\boldsymbol{x}}_1$, as it serves as one endpoint of the interpolation. If the $\hat{\boldsymbol{x}}_1$ has a high objective value, then its corresponding next state will be more oriented to a high objective. Accordingly, we introduce the $(\texttt{BranchOut}, V)$ pair to operate on destination states as follows.

Module 2 `BranchOut`-Destination

1: **Input:** $\boldsymbol{x}_t$, diffusion model u_θ, time step t, (optional) tuning parameter ρ_t, τ_t.
2: **Sample destination state candidates:**
(continuous) $\hat{\boldsymbol{x}}_1 \sim \mathcal{N}(\boldsymbol{x}_{1|t}, \rho_t \boldsymbol{I})$, $\boldsymbol{x}_{1|t}$ in (4).
(discrete) $\hat{\boldsymbol{x}}_1 \sim \text{Cat}\left(u_\theta(\boldsymbol{x}_t, t)\right)$.
3: **Compute the next state:**
(continuous) $\boldsymbol{x}_{t+\Delta t} \sim \mathcal{N}\left(c_{t,1}\boldsymbol{x}_t + c_{t,2}\hat{\boldsymbol{x}}_1, \tau_t I\right)$.
(discrete) $x_{t+\Delta t}^{(d)} \sim \text{Cat}\left(\delta\{x_t^{(d)}, j\} + R_t\left(x_t^{(d)}, j \mid \hat{x}_1^{(d)}\right)\Delta t\right)$.
4: **Output**: $(\boldsymbol{x}_{t+\Delta t}, \hat{\boldsymbol{x}}_1)$

Value Function 2 V : Destination State Evaluator

1: **Input:** $(\boldsymbol{x}_{t+\Delta t}, \hat{\boldsymbol{x}}_1)$, objective function f_y.
2: Evaluate on the clean sample:
$V\left((\boldsymbol{x}_{t+\Delta t}, \hat{\boldsymbol{x}}_1)\right)$
$= \exp\left(f_y\left(\hat{\boldsymbol{x}}_1\right)\right)$.
3: **Output:**
$V\left((\boldsymbol{x}_{t+\Delta t}, \hat{\boldsymbol{x}}_1)\right)$

We name Alg. 1 with Module 2 and Value Function 2 by **TreeG-Sampling Destination (TreeG-SD)**. TreeG-SD outputs $\hat{\mathcal{T}}$ that closely approximates the conditional transition distribution, as shown below (proof in App. D.3).

Theorem 2. *Consider TreeG-Sampling Destination at time t, with an active set of size one and selection performed via resampling. Then, for any $\varepsilon, \delta > 0$, it holds with probability $1-\delta$:*

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\ell < \varepsilon,$$

provided one of the following conditions is satisfied:
***(a, Continuous)** It holds under the $\ell = 1$ norm. Suppose data follow Gaussian distribution and the*

objective function is linear. The Branch-Out size is $K = \Theta(\frac{\log(1/\delta)}{\varepsilon^2})$.

(b, Discrete) *It holds under the* $\ell = \infty$ *norm. The Branch-Out size is* $K = \Theta(\frac{D^2 \log(|\mathcal{X}|/\delta)}{\varepsilon^2})$ *where* $\mathcal{X}$ *is the data space and* D *is its dimension.*

5.3 Gradient-Based Guidance with Differentiable Objective Predictor

Previously in this section, we derived two algorithms that do not rely on the gradient of the objective. Though we do not assume the true objective is differentiable, leveraging its gradient as guidance is still a feasible option when a differentiable objective predictor is available. Therefore, in what follows, we propose a novel gradient-based training-free guidance for discrete flow models, which also fits into the TreeG framework as a special case with $K = 1$.

For discrete flow models, we aim for the conditional rate matrix [45]:

$$R_t(\boldsymbol{x}_t, j \mid y) = \frac{p_t(y \mid \boldsymbol{x}_t^{\backslash d}(j))}{p_t(y \mid \boldsymbol{x}_t)} \cdot R_t(\boldsymbol{x}_t, j),$$

where $\boldsymbol{x}_t^{\backslash d}$ matches $\boldsymbol{x}_t$ except at dimension d, and $\boldsymbol{x}_t^{\backslash d}(j)$ has its d-dimension set to j. While [45] requires training a time-dependent predictor to estimate $p_t(y \mid \boldsymbol{x})$, we propose to estimate it using (9) in a training-free way. However, computing this estimation over all possible $\boldsymbol{x}_t^{\backslash d}$'s is computationally expensive. As suggested by [45, 67], we can approximate the ratio using Taylor expansion:

$$\begin{aligned} \log \frac{p_t(y \mid \boldsymbol{x}_t^{\backslash d})}{p_t(y \mid \boldsymbol{x}_t)} &= \log p_t(y \mid \boldsymbol{x}_t^{\backslash d}) - \log p_t(y \mid \boldsymbol{x}_t) \\ &\simeq (\boldsymbol{x}_t^{\backslash d} - \boldsymbol{x}_t)^\top \nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t) \end{aligned} \tag{10}$$

We use the Straight-Through Gumbel-Softmax trick [31] to enable gradient backpropagation through the sampling process (details are in App. E.4). We also show that this approximation achieves high accuracy compared to computing (9) for all $\boldsymbol{x}_t^{\backslash d}$, while offering greater efficiency (see App. H.2).

A backward sampling step, utilizing the estimated conditional rate matrix, can be viewed as a `BranchOut` operation, termed `BranchOut`-Gradient (detailed in App. E.2). With $K = 1$, `BranchOut`-Gradient reduces to gradient-based guidance methods. For $K > 1$, it aligns with Value Function 1. We denote this algorithm as **TreeG Gradient (TreeG-G)**.

6 Experiments

This section evaluates TreeG on one continuous and two discrete models across diverse tasks. Sec. 6.1 introduces the comparison methods; Sec. 6.2 details the tasks and results; Sec. 6.3 validates framework scalability; and Sec. 6.4 discusses configuration choices for different scenarios.

6.1 Settings

Below are the methods we would like to compare:
For continuous models: DPS [13], a training-free classifier gradient guidance requiring surrogate neural network predictors for non-differentiable objective functions; **TDS [73]**, a sequential Monte Carlo method based on gradient guidance; **SCG [30]**; **SVDD [37]**; and **TreeG-SD (Sec. 5.2)**.
For discrete models: DG [45], a training-based classifier guidance with a predictor trained on noisy inputs, implemented with Taylor expansion and gradients; **TFG-Flow [38]**, a training-free method estimating the conditional rate matrix; **SVDD [37]**; **TreeG-G (Sec. 5.3)**, which trains a predictor on clean data for non-differentiable objectives; **TreeG-SC (Sec. 5.1)**; and **TreeG-SD (Sec. 5.2)**.

6.2 Guided Generation

To enable a clear comparison with the guidance baselines, we report the active size of TreeG for both $A = 1$ and $A > 1$, while controlling inference time to be comparable across methods and settings.

6.2.1 Symbolic Music Generation

We follow the setup of [30], using a continuous diffusion model pre-trained on several piano midi datasets, detailed in App. F.1. The branch-out size for TreeG-SD is $K = 16$; SCG and SVDD use a sample size of 16, the temperature of SVDD is 0.01, and TDS uses 4 particles.

Guidance Target. Our study focuses on three types of targets: pitch histogram, note density, and chord progression. The objective function is $f_y(\cdot) = -\ell\left(y, \texttt{Rule}(\cdot)\right)$, where ℓ is the loss function. Notably, the rule function $\texttt{Rule}(\cdot)$ is *non-differentiable* for note density and chord progression.

Evaluation Metrics. For each task, we evaluate performance on 200 targets as formulated by [30]. Two metrics are used: (1) Loss, which measures how well the generated samples adhere to the target rules. (2) Average Overlapping Area (OA), which assesses music quality by comparing the similarity between the distributions of the generated and ground-truth music [74].

Table 1: Evaluation on Music Generation. TreeG-SD reduces loss by average 29.01%. Results of no guidance, Classifier, and DPS are copied from [30]. Best results are **bold**, second-best underlined.

Method	Pitch Histogram		Note Density		Chord Progression	
	Loss ↓	OA ↑	Loss ↓	OA ↑	Loss ↓	OA ↑
No Guidance	0.0180 ± 0.0100	0.842 ± 0.012	2.486 ± 3.530	0.830 ± 0.016	0.831 ± 0.142	0.854 ± 0.026
Classifier	0.0050 ± 0.0040	0.855 ± 0.020	0.698 ± 0.587	0.861 ± 0.025	0.723 ± 0.200	0.850 ± 0.033
DPS	0.0010 ± 0.0020	0.849 ± 0.018	1.261 ± 2.340	0.667 ± 0.113	0.414 ± 0.256	0.839 ± 0.039
TDS	0.0027 ± 0.0055	0.845 ± 0.017	0.218 ± 0.241	$\mathbf{0.875 \pm 0.023}$	0.714 ± 0.187	0.857 ± 0.028
SCG	0.0036 ± 0.0057	$\mathbf{0.862 \pm 0.008}$	0.134 ± 0.533	0.842 ± 0.022	0.347 ± 0.212	0.850 ± 0.046
SVDD	0.0085 ± 0.0100	0.846 ± 0.013	0.445 ± 0.437	0.835 ± 0.028	0.528 ± 0.189	$\mathbf{0.865 \pm 0.014}$
TreeG-SD (A=1,K=16)	$\mathbf{0.0002 \pm 0.0003}$	0.860 ± 0.016	0.142 ± 0.423	0.832 ± 0.023	0.301 ± 0.191	0.856 ± 0.032
TreeG-SD (A=4,K=4)	$\mathbf{0.0002 \pm 0.0003}$	0.818 ± 0.013	$\mathbf{0.048 \pm 0.198}$	0.843 ± 0.012	$\mathbf{0.226 \pm 0.144}$	0.862 ± 0.015

Results. As shown in Table 1, our TreeG-SD outperforms baselines while preserving comparable sample quality. For differentiable rules (pitch histogram), TreeG-SD outperforms DPS, which gradient-free methods like SCG and SVDD cannot achieve. For non-differentiable rules (note density and chord progression), TreeG-SD matches or exceeds SCG and significantly outperforms others.

6.2.2 Small Molecule Generation

We validate our methods on the generation of small molecules with discrete flow models. Following [45], small molecules are represented as simplified molecular-input line-entry system (SMILES) strings. These *discrete* sequences are padded to 100 tokens and there are 32 possible token types, including one pad and one mask token M. We adopt the same unconditional flow model and Euler sampling curriculum as [45].

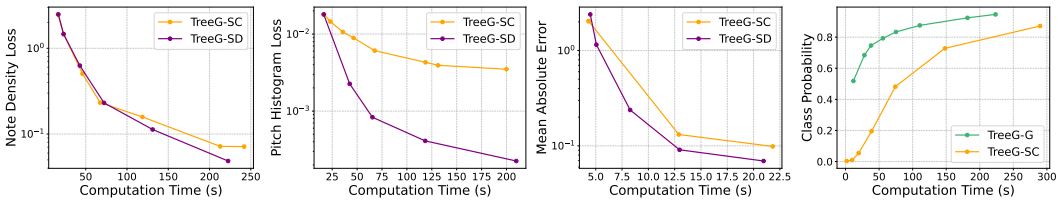
Guidance Target. Following the benchmarks in [20, 37], we deploy guidance to maximize four targets: quantitative estimate of drug-likeness (QED), synthetic accessibility (SA), binding score with dopamine type 2 receptor (DRD2), and binding affinity to protein 5ht1b (Docking). We directly use the target measurement as the objective function $f_y(\boldsymbol{x})$ for guidance in Algorithm 1. The values of QED, SA and DRD2 are estimated using a pretrained predictor model $f(x)$ while Docking score is measured by an oracle tool, QuickVina2-GPU-2.1, and is thus an entirely *non-differentiable* target. In addition, we guide the number of rings N_r toward target values $N_r^* \in [0, 6]$. The objective function is formalized as $f_y(\boldsymbol{x}) = -\frac{(y - f(\boldsymbol{x}))^2}{2\sigma^2}$. Please refer to App. F.2 for details.

Evaluation Metrics. We measure five molecule properties using RDKit, TDC, and QuickVina2-GPU-2.1 [29] for generated valid unique sequences. For N_r, we report the mean absolute error (MAE) against each target value. We evaluate molecular diversity through average Tanimoto similarity (TS) across molecules (Table 2). Further, we also report the Negative Log-Likelihood (NLL) per token (bits/dim), a widely adopted metric in discrete diffusion and flow-based generation [2, 64] (Table 12). The likelihood is estimated by the ELBO of the pretrained discrete flow model.

Table 2: Evaluation on Small Molecule Generation. Complete results for different N_r^* and NLL per token evaluations are deferred to App. G.2.1. Detailed experimental setup are stated in App. F.2.

Method	QED		SA		DRD2		Docking		$N_r^* = 1$	
	VAL ↑	TS ↓	VAL ↑	TS ↓	VAL ↑	TS ↓	VAL ↑	TS ↓	MAE ↓	TS ↓
No Guidance	0.61 ± 0.19	0.12 ± 0.02	0.79 ± 0.10	0.12 ± 0.02	0.06 ± 0.17	0.12 ± 0.02	8.50 ± 1.41	0.12 ± 0.02	2.09 ± 1.16	0.12 ± 0.02
DG	0.62 ± 0.20	0.12 ± 0.02	0.80 ± 0.10	$\mathbf{0.12 \pm 0.02}$	0.17 ± 0.32	$\mathbf{0.12 \pm 0.02}$	—	—	0.11 ± 0.33	0.14 ± 0.03
TFG-Flow	0.61 ± 0.20	0.12 ± 0.02	0.79 ± 0.10	0.12 ± 0.02	0.09 ± 0.22	0.12 ± 0.02	8.74 ± 1.49	0.12 ± 0.02	0.28 ± 0.65	0.13 ± 0.02
SVDD	0.67 ± 0.18	0.13 ± 0.02	0.80 ± 0.09	0.13 ± 0.02	0.43 ± 0.44	0.17 ± 0.04	9.47 ± 1.37	$\mathbf{0.12 \pm 0.01}$	0.35 ± 1.14	0.14 ± 0.03
TreeG-SC (A=1)	0.80 ± 0.13	0.13 ± 0.02	$\mathbf{0.89 \pm 0.06}$	0.21 ± 0.05	$\mathbf{0.77 \pm 0.33}$	0.20 ± 0.04	$\mathbf{9.64 \pm 1.28}$	0.12 ± 0.02	$\mathbf{0.01 \pm 0.07}$	0.14 ± 0.02
TreeG-G (A=1)	0.64 ± 0.19	0.12 ± 0.02	0.79 ± 0.10	0.12 ± 0.02	0.22 ± 0.37	0.13 ± 0.02	—	—	0.44 ± 1.19	0.13 ± 0.02
TreeG-SD (A=1)	0.77 ± 0.14	0.12 ± 0.02	0.86 ± 0.10	0.16 ± 0.04	0.45 ± 0.41	0.14 ± 0.03	8.72 ± 1.27	0.12 ± 0.02	0.11 ± 0.37	$\mathbf{0.12 \pm 0.02}$
TreeG-SD (A>1)	$\mathbf{0.80 \pm 0.11}$	$\mathbf{0.12 \pm 0.02}$	0.89 ± 0.07	0.21 ± 0.05	0.66 ± 0.40	0.19 ± 0.04	9.29 ± 1.30	0.12 ± 0.02	0.02 ± 0.12	0.14 ± 0.02

Results. TreeG consistently outperforms three guidance baselines (Table 2) while maintaining comparable sequence diversity (TS) and sample quality (NLL). Our best guidance method TreeG-SC achieves a 26.38% average improvement across five targets compared to the best baseline methods



(a) ND Music (b) PH Music (c) Molecular ($N_r^* = 5$) (d) Enhancer (Class 4)

Figure 3: **Inference Time Scaling Behavior:** As the active set size and branch-out size increase, the optimization effect of the objective function scales with inference time. This trend is consistently observed across all algorithms and tasks. The inference time is measured with a batch size of 1 for music and 100 for molecule and DNA design. For DNA design, $\gamma = 20$ for TreeG-G.

(Table 11). And increasing the size of the active set A to more than 1 could further improve the performance of TreeG-SD.

6.2.3 Enhancer DNA Design

We follow the experimental setup of [56], using a discrete flow model pre-trained on DNA sequences of length 500, each labeled with one of 81 cell types [32, 61]. For inference, we apply 100 Euler sampling steps. The branch-out size for TreeG-G is set $K = 1$. Setup details are in App. F.3.

Guidance Target. The goal is to generate enhancer DNA sequences that belong to a specific target cell type. The guidance target predictor is provided by an oracle classifier f from [56]. The objective function of given cell class y is $f_y(\cdot) = \log f(y \mid \cdot)$.

Table 3: Evaluation of guidance methods for Enhancer DNA Design at varying guidance strength levels $\gamma_t = \gamma$ in Module 3. TreeG-G consistently achieves significantly higher target class probabilities.

Method (strength γ)		Class 1			Class 2			Class 3		
		Prob ↑	NLL ↓	Div ↑	Prob ↑	NLL ↓	Div ↑	Prob ↑	NLL ↓	Div ↑
No Guidance	—	0.021 ± 0.079	1.311 ± 0.038	373	0.008 ± 0.052	1.311 ± 0.038	373	0.007 ± 0.053	1.311 ± 0.038	373
DG	20	0.359 ± 0.188	1.225 ± 0.036	351	0.627 ± 0.340	1.191 ± 0.052	359	0.693 ± 0.264	1.241 ± 0.044	357
	100	0.372 ± 0.237	1.159 ± 0.074	352	0.571 ± 0.356	1.130 ± 0.073	343	0.173 ± 0.256	1.183 ± 0.081	349
	200	0.251 ± 0.171	1.114 ± 0.098	331	0.350 ± 0.351	1.137 ± 0.084	335	0.064 ± 0.143	1.179 ± 0.074	343
TFG-Flow	200	0.054 ± 0.129	1.398 ± 0.011	**375**	0.012 ± 0.076	1.390 ± 0.016	**375**	0.004 ± 0.029	1.382 ± 0.013	**375**
SVDD	—	0.155 ± 0.173	1.307 ± 0.038	364	0.088 ± 0.193	1.303 ± 0.040	367	0.061 ± 0.168	1.297 ± 0.040	363
TreeG-G (A=1,K=1)	20	0.236 ± 0.178	1.198 ± 0.085	355	0.313 ± 0.343	1.152 ± 0.136	347	0.247 ± 0.280	0.895 ± 0.119	324
	100	0.509 ± 0.242	0.951 ± 0.097	353	0.915 ± 0.154	0.847 ± 0.090	332	0.745 ± 0.217	$\mathbf{0.687 \pm 0.089}$	306
	200	0.560 ± 0.258	0.951 ± 0.104	358	0.951 ± 0.110	$\mathbf{0.801 \pm 0.078}$	331	0.894 ± 0.136	0.711 ± 0.094	321
TreeG-G (A=2,K=2)	200	$\mathbf{0.740 \pm 0.209}$	$\mathbf{0.885 \pm 0.085}$	362	$\mathbf{0.981 \pm 0.042}$	0.823 ± 0.077	336	$\mathbf{0.934 \pm 0.098}$	0.713 ± 0.085	310

Evaluation Metrics. We generate 1000 DNA sequences conditioned on cell type and evaluate performance using three metrics. Target Class Probability, provided by the oracle classifier, where higher probabilities indicate better guidance. We adopt NLL per token to evaluate the fidelity of the generated sequences. Diversity is measured by the average pairwise Hamming distance between sequences.

Results. As shown in Tab. 3, our TreeG-G consistently achieves the highest target probabilities as guidance strength increases, with an average improvement of 18.43% compared to DG over eight test classes [45], and significantly exceeding other training-free baselines. See App. G for details.

6.3 Scalability on Inference-Time Computation

TreeG is scalable to the active size A (i.e., the number of generation paths) and the branch-out size K. It is compatible with all guidance methods. When increasing the active set size and branch-out size, the computational cost of inference rises. We investigate the performance frontier to optimize the objective function concerning inference time. The results reveal an inference-time scaling law, as illustrated in Fig. 3. Our findings indicate consistent scalability across all algorithms and tasks, with App. 3 showcasing four examples. Additional results refer to App. G.

6.4 TreeG Configuration Analysis

This section analyzes the configuration of TreeG, focusing on selecting the instantiated algorithms and balancing A and K under a fixed computational budget.

We analyze the computational complexity of TreeG, summarized in Tab. 4, The cost units are: C_{model} for a forward pass through the diffusion model, C_{pred} for a predictor call, and C_{backprop} for backpropagation through both. N denotes the Monte Carlo sample size used in Value Function 1.

Table 4: Computation Complexity of TreeG

Methods	Computation
TreeG-SC	$AC_{\text{model}} + AK(C_{\text{model}} + NC_{\text{pred}})$
TreeG-SD	$AC_{\text{model}} + AKC_{\text{pred}}$
TreeG-G	$AK\,(C_{\text{model}} + NC_{\text{pred}}) + AC_{\text{backprop}}$

Design Axes Comparison. We evaluate TreeG designs along two axes. **Gradient-free vs. gradient-based:** TreeG-G requires an accurate objective predictor to be effective. If available, predictor latency guides the choice—faster predictors favor TreeG-SC and TreeG-SD, while slower ones make TreeG-G more practical. **TreeG-SC (current state) vs. TreeG-SD (destination state):** As Tab. 4 shows, TreeG-SD is more efficient, costing only A for the diffusion model forward pass, versus AK for alternatives. Experiments show TreeG-SD outperforms TreeG-SC in continuous diffusion, while TreeG-SC is better for discrete flow. Please refer to App. C for detailed discussion.

Trade-off between Active Set Size A and Branch-out Size K. Tab. 4 shows that the computational complexity of TreeG-SC and TreeG-G using `BranchOut`-Current is $O(AK)$. With a fixed product $A \cdot K$ (i.e., fixed inference cost; see Fig. 9 in App. G.3.2), we explore how to best balance A and K. As shown in Fig.4, performance peaks when both values are moderate. In the special case where $K = 1$ inference paths do not interact, often leading to suboptimal performance.

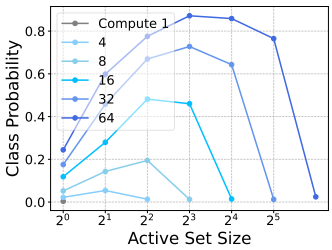


Figure 4: Trade-off between Active Set Size A and Branch-out Size K with Fixed Compute. The results are for TreeG-SC DNA (Class 4).

7 Conclusion

We proposed the framework TreeG based on inference path search, along with three novel instantiated algorithms: TreeG-SC, TreeG-SD, and TreeG-G, which guide diffusion models toward the posterior conditional distribution and address the non-differentiability challenge in training-free guidance. Experimental results demonstrated the improvements of TreeG against existing methods. Furthermore, we identified an inference-time scaling law that highlights TreeG's scalability in inference-time computation.

Acknowledgments

This work received no external funding. The authors declare no competing interests.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[3] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pages 1276–1301. PMLR, 2023.

[4] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 843–852, 2023.

[5] Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. *arXiv preprint arXiv:2401.01879*, 2024.

[6] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

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

[8] Zander W Blasingame and Chen Liu. Adjointdeis: Efficient gradients for diffusion models. *Advances in Neural Information Processing Systems*, 37:2449–2483, 2024.

[9] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[10] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

[11] Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems. *arXiv preprint arXiv:2308.07983*, 2023.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

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

[14] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

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

[16] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky TQ Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

[17] Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

[18] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[19] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS) 2023*. Neural Information Processing Systems Foundation, 2023.

[20] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

[21] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *arXiv preprint arXiv:2407.15595*, 2024.

[22] Yingqing Guo, Hui Yuan, Yukang Yang, Minshuo Chen, and Mengdi Wang. Gradient guidance for diffusion models: An optimization perspective. *arXiv preprint arXiv:2404.14743*, 2024.

[23] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

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

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

[27] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

[28] Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang. Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 178–186, 2021.

[29] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Proceedings of Neural Information Processing Systems, NeurIPS Datasets and Benchmarks*, 2021.

[30] Yujia Huang, Adishree Ghatare, Yuanzhe Liu, Ziniu Hu, Qinsheng Zhang, Chandramouli S Sastry, Siddharth Gururani, Sageev Oore, and Yisong Yue. Symbolic music generation with non-differentiable rule guided diffusion. *arXiv preprint arXiv:2402.14285*, 2024.

[31] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[32] Jasper Janssens, Sara Aibar, Ibrahim Ihsan Taskiran, Joy N Ismail, Alicia Estacio Gomez, Gabriel Aughey, Katina I Spanier, Florian V De Rop, Carmen Bravo Gonzalez-Blas, Marc Dionne, et al. Decoding gene regulation in the fly brain. *Nature*, 601(7894):630–636, 2022.

[33] Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. arxiv, page 2309.09968, 2023. doi: 10.48550. *arXiv preprint ARXIV.2309.09968*.

[34] Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1334–1345, 2024.

[35] Heeseung Kim, Sungwon Kim, and Sungroh Yoon. Guided-tts: A diffusion model for text-to-speech via classifier guidance. In *International Conference on Machine Learning*, pages 11119–11133. PMLR, 2022.

[36] Hongjian Li, Kwong-Sak Leung, Man-Hon Wong, and Pedro J Ballester. Improving autodock vina using random forest: the growing accuracy of binding affinity prediction by the effective exploitation of larger data sets. *Molecular informatics*, 34(2-3):115–126, 2015.

[37] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024.

[38] Haowei Lin, Shanda Li, Haotian Ye, Yiming Yang, Stefano Ermon, Yitao Liang, and Jianzhu Ma. Tfg-flow: Training-free guidance in multimodal generative flow, 2025. Available at http://arxiv.org/abs/2501.14216.

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

[40] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. 2023.

[41] Yueming Lyu, Kim Yong Tan, Yew Soon Ong, and Ivor W Tsang. Covariance-adaptive sequential black-box optimization for diffusion targeted generation. *arXiv preprint arXiv:2406.00812*, 2024.

[42] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Inference-time scaling for diffusion models beyond scaling denoising steps. *arXiv preprint arXiv:2501.09732*, 2025.

[43] Pierre Marion, Anna Korba, Peter Bartlett, Mathieu Blondel, Valentin De Bortoli, Arnaud Doucet, Felipe Llinares-López, Courtney Paquette, and Quentin Berthet. Implicit diffusion: Efficient optimization through stochastic sampling. *arXiv preprint arXiv:2402.05468*, 2024.

[44] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[45] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

[46] Jiachun Pan, Jun Hao Liew, Vincent YF Tan, Jiashi Feng, and Hanshu Yan. Adjointdpm: Adjoint sensitivity method for gradient backpropagation of diffusion probabilistic models. *arXiv preprint arXiv:2307.10711*, 2023.

[47] Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. *arXiv preprint arXiv:2402.06320*, 2024.

[48] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

[49] Yifei Shen, Xinyang Jiang, Yifan Yang, Yezhen Wang, Dongqi Han, and Dongsheng Li. Understanding and improving training-free loss-based diffusion guidance. *Advances in Neural Information Processing Systems*, 37:108974–109002, 2024.

[50] Yuchen Shen, Chenhao Zhang, Chenghui Zhou, Sijie Fu, Newell Washburn, and Barnabas Poczos. Non-differentiable diffusion guidance for improved molecular geometry. In *ICML 2024 AI for Science Workshop*.

[51] 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. *arXiv preprint arXiv:2501.06848*, 2025.

[52] Marta Skreta, Tara Akhound-Sadegh, Viktor Ohanesian, Roberto Bondesan, Alán Aspuru-Guzik, Arnaud Doucet, Rob Brekelmans, Alexander Tong, and Kirill Neklyudov. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. *arXiv preprint arXiv:2503.02819*, 2025.

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

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

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

[56] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

[57] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[58] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

[59] Kim Yong Tan, Yueming Lyu, Ivor Tsang, and Yew-Soon Ong. Fast direct: Query-efficient online black-box guidance for diffusion-model target generation. *arXiv preprint arXiv:2502.01692*, 2025.

[60] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. *arXiv preprint arXiv:2405.18881*, 2024.

[61] Ibrahim I Taskiran, Katina I Spanier, Hannah Dickmänken, Niklas Kempynck, Alexandra Pančíková, Eren Can Ekşi, Gert Hulselmans, Joy N Ismail, Koen Theunis, Roel Vandepoel, et al. Cell-type-directed design of synthetic enhancers. *Nature*, 626(7997):212–220, 2024.

[62] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.

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

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

[65] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

[66] Siddarth Venkatraman, Moksh Jain, Luca Scimeca, Minsu Kim, Marcin Sendera, Mohsin Hasan, Luke Rowe, Sarthak Mittal, Pablo Lemos, Emmanuel Bengio, et al. Amortizing intractable inference in diffusion models for vision, language, and control. *Advances in neural information processing systems*, 37:76080–76114, 2024.

[67] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

[68] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7280–7290, 2023.

[69] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.

[70] Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.

[71] Ziyu Wang, Ke Chen, Junyan Jiang, Yiyi Zhang, Maoran Xu, Shuqi Dai, Xianbin Gu, and Gus Xia. Pop909: A pop-song dataset for music arrangement generation. *arXiv preprint arXiv:2008.07142*, 2020.

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

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

[74] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

[75] Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with spherical gaussian constraint for conditional diffusion. *arXiv preprint arXiv:2402.03201*, 2024.

[76] Soojung Yang, Doyeong Hwang, Seul Lee, Seongok Ryu, and Sung Ju Hwang. Hit and lead discovery with explorative rl and fragment-based molecule generation. *Advances in Neural Information Processing Systems*, 34:7924–7936, 2021.

[77] Chun Wei Yap. Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints. *Journal of computational chemistry*, 32(7):1466–1474, 2011.

[78] Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. Tfg: Unified training-free guidance for diffusion models. *arXiv preprint arXiv:2409.15761*, 2024.

[79] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23174–23184, 2023.

[80] Zaixi Zhang, Marinka Zitnik, and Qi Liu. Generalized protein pocket generation with prior-informed flow matching. *arXiv preprint arXiv:2409.19520*, 2024.

[81] Linxi Zhao, Yihe Deng, Weitong Zhang, and Quanquan Gu. Mitigating object hallucination in large vision-language models via classifier-free guidance. *arXiv preprint arXiv:2402.08680*, 2024.

[82] Yulai Zhao, Masatoshi Uehara, Gabriele Scalia, Sunyuan Kung, Tommaso Biancalani, Sergey Levine, and Ehsan Hajiramezanali. Adding conditional control to diffusion models with reinforcement learning. *arXiv preprint arXiv:2406.12120*, 2024.

A Additional Related Work

In this section, we provide an overview of related work on aligning diffusion models with downstream objectives. The methods in this domain can be broadly categorized into inference-time alignment and post-training/fine-tuning approaches. We begin by reviewing these two main categories, followed by a discussion of other relevant methods, including alternative alignment techniques for diffusion models and discrete diffusion and flow models.

Inference-Time Alignment. Classifier guidance uses a time-dependent classifier to provide directional signals throughout the generation process [55, 15]. A growing body of work explores training-free guidance methods that leverage gradients derived from objective functions [13, 4, 78, 53, 24, 79, 49]. A line of work [73, 47, 17, 11, 52] combined diffusion models with Sequential Monte Carlo methods. [37, 30] use value-based selection and importance sampling When objectives are non-differentiable. A concurrent work [51] proposes an SMC-based framework for inference-time scaling using value resampling. A widely used and straightforward approach is Best-of-N sampling, where the model generates multiple candidates and selects those with the highest objective values [57, 44, 5].

Fine-tuning Diffusion Models. Fine-tuning diffusion models involves directly adjusting model parameters to align with downstream objectives. One common method involves direct fine-tuning through gradient backpropagation across the sampling process [7, 63, 48]. To ensure stable training and prevent divergence from the original distribution, Kullback-Leibler (KL) regularization has been introduced in [63, 16, 66]. Reinforcement learning has also emerged as a powerful tool for fine-tuning [7, 19, 82, 62]. [69] proposes direct preference optimization for aligning diffusion models with human preferences. Additionally, recent work [70] has extended fine-tuning techniques to discrete diffusion models.

Other Diffusion Alignment Methods. Beyond guidance and fine-tuning approaches, an alternative line of training-free methods focuses on optimizing the initial latent state of the reverse diffusion process [68, 6, 34, 60]. These methods typically use an ODE solver to backpropagate the objective gradient directly to the initial latent state, making them a gradient-based version of the Best-of-N strategy. Additionally, adjoint-based methods have been proposed for gradient estimation in diffusion models [43, 46, 8]. [59, 41] focus on improving query efficiency when the objective function is computationally expensive. Other recent work explores diffusion-based alignment techniques and applications of diffusion models in biology [72, 3, 50, 33].

Discrete Diffusion and Flow Models. Austin et al. [2] and Hoogeboom et al. [27] pioneered diffusion in discrete spaces by introducing a corruption process for categorical data. Campbell et al. [9] extended discrete diffusion models to continuous time, while Lou et al. [40] proposed learning probability ratios. Discrete Flow Matching Campbell et al. [10], Gat et al. [21] further advances this field by developing a Flow Matching algorithm for time-continuous Markov processes on discrete state spaces, commonly known as Continuous-Time Markov Chains (CTMCs). Lipman et al. [39] presents a unified perspective on flow and diffusion.

B Limitations and Broader Impacts

Limitations. We do not observe significant limitations in our methods. However, in cases where the objective function is non-differentiable, very expensive to evaluate, and lacks an effective surrogate neural network, our methods can become relatively time-consuming. These represent inherently difficult scenarios, where a trade-off between computational cost and optimization effectiveness is unavoidable.

Broader Impacts. This paper aims to explore inference-time alignment methods for diffusion models, advanced control techniques in generative AI, and contribute to the broader field of artificial intelligence. This work holds promise for improving the accuracy and personalization of AI systems in diverse domains, such as image synthesis and drug discovery. Nonetheless, the same techniques may also be exploited to produce harmful content.

C Discussion on Design Axes

We compare the guidance designs: TreeG-SC, TreeG-SD and TreeG-G based on experimental results, to separate the effect of guidance design from the effect of tree search, we set $A = 1$. A side-to-side comparison on the performance of the three methods are provided in Table 5.

Gradient-based v.s. Gradient-free: depends on the predictor. The choice between gradient-based and gradient-free methods largely depends on the characteristics of the predictor.

The first step is determining whether a reliable, differentiable predictor is available. If not, sampling methods should be chosen over gradient-based approaches. For example, in the chord progression task of music generation, the ground truth reward is obtained from a chord analysis tool in the music21 package [14], which is non-differentiable. Additionally, the surrogate neural network predictor achieves only 33% accuracy [30]. As shown in Table 1, in cases where no effective differentiable predictor exists, the performance of gradient-based methods (e.g., DPS) is significantly inferior to sampling-based methods (e.g., TreeG-SD and SCG).

If a good differentiable predictor is available, the choice depends on the predictor's forward pass time. Our experimental tasks illustrate some typical cases. For molecule generation which uses a pretrained predictor model, forward passes are fast as shown in Table 6. Thus, sampling approaches efficiently expand the candidate set and capture the reward signal, yielding strong results (Table 5). While for enhancer DNA design, where predictors have slow forward passes, increasing the sampling candidate set size to capture the reward signal becomes prohibitively time-consuming, making gradient-based method more effective (Table 5). However, for the optimization of *non-differentiable* targets which employ an oracle as the predictor like small molecules' Docking score, gradient-based methods are not applicable and computationally expensive target measurement becomes an bottleneck (Table 7).

TreeG-SC v.s. TreeG-SD. Experiments on continuous data and discrete data give divergent results along this axis. In the continuous task of music generation (Table 1), TreeG-SD achieves equal or better performance than SCG (equivalent to TreeG-SC) with the same candidate size $K = 16$ and similar time cost (details in Appendix G.1). Thus, TreeG-SD is preferable in this continuous setting. Conversely, for discrete tasks, TreeG-SD requires significantly more samples, while TreeG-SC outperforms it, as shown in Table 5.

Table 5: Comparison of results across TreeG-G, TreeG-SC and TreeG-SD. For molecule generation, the target is specified as the number of rings $N_r = 2$. For enhancer DNA design, the results correspond to Class 3.

		TreeG-G	TreeG-SC	TreeG-SD
Molecule	MAE ↓	0.09 ± 0.54	0.02 ± 0.14	0.10 ± 0.33
	Time ↓	13.5s	12.9s	11.2s
	N	30	30	—
	K	—	2	200
Enhancer	Prob ↑	0.89 ± 0.14	0.13 ± 0.39	0.002 ± 0.000
	FBD ↓	213	384	665
	Div ↑	321	375	376
	Time ↓	10.3s	285.1s	189.7s
	N	20	20	—
	K	—	64	1024

Table 6: Computation time per basic unit (ms)

	C_{model}	C_{pred}	C_{backprop}
Molecule	0.038	2.2e-4	0.036
Enhancer	0.087	0.021	0.11

D Omitted Proofs in Section 5

D.1 Proof of Theorem 1

We begin by restating Theorem 1 with all conditions for completeness, followed by its proof.

Theorem (Restatement of Theorem 1)**.** *Consider TreeG-Sampling Current at time t, with an active set of size one and selection performed via multinomial resampling. Then, for any $\varepsilon, \delta > 0$, it holds with probability $1 - \delta$:*

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\ell < \varepsilon,$$

provided one of the following conditions is satisfied:
(a, Continuous)*It holds under the $\ell = 1$ norm. Suppose data follow Gaussian distribution and the objective function is linear. The Branch-Out size is $K = \Theta(\frac{\log(1/\delta)}{\varepsilon^2})$ and the timestep is sufficiently small such that $\alpha_t = 1 - O\left(\varepsilon^2\right)$.*
(b, Discrete)*It holds under the $\ell = \infty$ norm. The mapping $t \mapsto p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})$ is L-Lipschitz continuous for all $\boldsymbol{x}$ and $\boldsymbol{x}_1$, and the likelihood scores satisfy $0 < B_{\min} \le \exp(f_y(\boldsymbol{x})) \le B_{\max}$ for all $\boldsymbol{x}$. Branch-Out size is $K = \Theta(\frac{\log(|\mathcal{X}|/\delta)}{\varepsilon^2})$, Monte Carlo size is $N = \Theta(\frac{\log(|\mathcal{X}|/\delta)}{\varepsilon^2})$, and time step $\Delta t = O(\varepsilon)$, where $\mathcal{X}$ is the data space.*

Proof. **Proof of (a) continuous case:** Assume the data is drawn from a Gaussian, and the objective function is linear

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}, I), \qquad f_y(\boldsymbol{x}) = \boldsymbol{g}^\top \boldsymbol{x}.$$

Since $\boldsymbol{x}_t \mid \boldsymbol{x}_1 \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_1,\ (1-\bar{\alpha}_t)\,I)$, by Bayes' rule the posterior follows:

$$\boldsymbol{x}_1 \mid \boldsymbol{x}_t \sim \mathcal{N}\Big((1-\bar{\alpha}_t)\,\boldsymbol{\mu} + \sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_t,\ (1-\bar{\alpha}_t)\,I\Big).$$

We have the denoising model's predictive distribution for label y given $\boldsymbol{x}_t$ as

$$p_t(y \mid \boldsymbol{x}_t) = \int p(y \mid \boldsymbol{x}_1)\, p(\boldsymbol{x}_1 \mid \boldsymbol{x}_t)\, \mathrm{d}\boldsymbol{x}_1 = \frac{1}{Z} \int \exp(f_y(\boldsymbol{x}_1))\, p(\boldsymbol{x}_1 \mid \boldsymbol{x}_t)\, \mathrm{d}\boldsymbol{x}_1,$$

where the objective function is linear $f_y(\boldsymbol{x}) = \boldsymbol{g}^\top \boldsymbol{x}$. Since for a Gaussian $\mathcal{N}(m, \Sigma)$,

$$\int \exp(\boldsymbol{g}^\top x)\, \mathcal{N}(x; m, \Sigma)\, \mathrm{d}x = \exp\Big(\boldsymbol{g}^\top m + \frac{1}{2}\,\boldsymbol{g}^\top \Sigma\, \boldsymbol{g}\Big),$$

it follows that

$$p_t(y \mid \boldsymbol{x}_t) \propto \exp\Big(\boldsymbol{g}^\top ((1-\bar{\alpha}_t)\,\boldsymbol{\mu} + \sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_t) \ +\ \frac{1}{2}\,(1-\bar{\alpha}_t)\,\|\boldsymbol{g}\|^2\Big).$$

Therefore, we have

$$\nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t) = \sqrt{\bar{\alpha}_t}\,\boldsymbol{g},$$

which allows us to express the conditional score as:

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t \mid y) = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \sqrt{\bar{\alpha}_t}\,\boldsymbol{g}. \tag{11}$$

Recall the transition step (2). The unconditional transition is given by:

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t) = \mathcal{N}\left(\cdot;\, \bar{\boldsymbol{x}}_{t+\Delta t},\, \sigma_t^2 \boldsymbol{I}\right), \tag{12}$$

where

$$\bar{\boldsymbol{x}}_{t+\Delta t} = \frac{\boldsymbol{x}_t + (1-\alpha_t)\nabla \log p_t(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}.$$

Using the conditional score from (11), the target conditional transition distribution becomes:

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y) = \mathcal{N}\left(\cdot;\, \bar{\boldsymbol{x}}_{t+\Delta t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t+\Delta t}}\,\boldsymbol{g},\, \sigma_t^2 \boldsymbol{I}\right). \tag{13}$$

TreeG-Sampling Current proceeds by generating i.i.d. samples $X_k \sim \mathcal{T}(\cdot \mid \boldsymbol{x}_t)$, followed by multinomial resampling according to the weights

$$V(\boldsymbol{x}) = \exp\left(f_y\left(\boldsymbol{x}_{1|t}(\boldsymbol{x}_t)\right)\right),$$

where, using Tweedie's formula [18], the posterior mean of $\boldsymbol{x}_1$ given $\boldsymbol{x}_t$ is:

$$\boldsymbol{x}_{1|t} = \mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{x}_t] = (1-\bar{\alpha}_t)\,\boldsymbol{\mu} + \sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_t.$$

Therefore, the value function simplifies to:

$$f_y(\boldsymbol{x}_{1|t}(\boldsymbol{x}_t)) = \boldsymbol{g}^\top \left((1-\bar{\alpha}_t)\boldsymbol{\mu} + \sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_t\right).$$

From Lemma 2, with probability at least $1-\delta$, the distribution $\hat{\mathcal{T}}$ produced by this resampling satisfies:

$$\left\|\hat{\mathcal{T}} - \mathcal{T}_1\right\|_1 \le 4(1+C)\sqrt{\frac{\log(4/\delta)}{2K}}, \tag{14}$$

where the adjusted distribution is

$$\mathcal{T}_1(\boldsymbol{x}) \propto \exp\left(\sqrt{\bar{\alpha}_t}\,\boldsymbol{g}^\top \boldsymbol{x}\right) \cdot \mathcal{T}(\boldsymbol{x} \mid \boldsymbol{x}_t),$$

and the constant C is given by

$$C = \frac{\|\boldsymbol{g}\|_2^2}{|\boldsymbol{g}^\top \bar{\boldsymbol{x}}_{t+\Delta t}|}.$$

We have

$$\mathcal{T}_1(\cdot) = \mathcal{N}\left(\cdot;\, \bar{\boldsymbol{x}}_{t+\Delta t} + \sigma_t^2 \sqrt{\bar{\alpha}_t}\,\boldsymbol{g},\, \sigma_t^2 \boldsymbol{I}\right). \tag{15}$$

Now, comparing the means of $\mathcal{T}_1$ and $\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)$, we use the bound via KL divergence:

$$\|\mathcal{T}_1 - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)\|_1 \leq \sqrt{2\mathrm{KL}(\mathcal{T}_1 \| \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y))} = \frac{2\|\mu_1 - \mu_2\|_2}{\sigma_t},$$

where

$$\mu_1 = \bar{\boldsymbol{x}}_{t+\Delta t} + \sigma_t^2 \sqrt{\bar{\alpha}_t}\, \boldsymbol{g}, \quad \mu_2 = \bar{\boldsymbol{x}}_{t+\Delta t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t+\Delta t}}\, \boldsymbol{g},$$

with $\sigma_t = \sqrt{1-\alpha_t}$ and $\alpha_t = \bar{\alpha}_t / \bar{\alpha}_{t+\Delta t}$. Thus,

$$\|\mathcal{T}_1 - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)\|_1 \leq 2\bar{\alpha}_{t+\Delta t}\sqrt{1-\alpha_t}(1-\sqrt{\alpha_t})\, \|\boldsymbol{g}\|_2. \tag{16}$$

Combining this with (14), we obtain:

$$\left\|\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)\right\|_1 \leq 4(1+C)\sqrt{\frac{\log(4/\delta)}{2K}} + 2\bar{\alpha}_{t+\Delta t}\sqrt{1-\alpha_t}(1-\sqrt{\alpha_t})\, \|\boldsymbol{g}\|_2.$$

Thus, if $K = \Theta\left((1+C)^2 \frac{\log(1/\delta)}{\varepsilon^2}\right)$ and $1-\alpha_t = O\left(\varepsilon^2 / \|\boldsymbol{g}\|_2^2\right)$, provided that Δt is sufficiently small, the total variation distance is bounded by ε. Ignoring constant factors, this completes the proof.

Proof of (b) discrete case: We simplify the notation and recall some key definitions for clarity.

At time step t, the current state is given by $\boldsymbol{x}_t$. The unconditional transition probability is defined as:

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t) = \delta\{\cdot, \boldsymbol{x}_t\} + R_t(\boldsymbol{x}_t, \cdot)\, \Delta t,$$

where $\delta\{\cdot, \boldsymbol{x}_t\}$ is the Dirac delta function. The target conditional distribution is

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y) = \delta\{\cdot, \boldsymbol{x}_t\} + R_t(\boldsymbol{x}_t, \cdot \mid y)\, \Delta t,$$

where the conditional rate is defined as [45]:

$$R_t(\boldsymbol{x}_t, \boldsymbol{x} \mid y) = \frac{p_t(y \mid \boldsymbol{x})}{p_t(y \mid \boldsymbol{x}_t)}\, R_t(\boldsymbol{x}_t, \boldsymbol{x}), \quad \text{for } \boldsymbol{x} \neq \boldsymbol{x}_t,$$

and

$$R_t(\boldsymbol{x}_t, \boldsymbol{x}_t \mid y) = -\sum_{\boldsymbol{x} \neq \boldsymbol{x}_t} R_t(\boldsymbol{x}_t, \boldsymbol{x} \mid y).$$

For simplicity, we introduce the shorthand:

$$\mathcal{T}_0 := \mathcal{T}(\cdot \mid \boldsymbol{x}_t), \quad \text{and} \quad \mathcal{T}^* := \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y).$$

We then define the distribution used in importance sampling, where we first sample from $\mathcal{T}_0$ and then reweight according to the likelihood $p_t(y \mid \cdot)$:

$$\mathcal{T}_1(\boldsymbol{x}) = \frac{p_t(y \mid \boldsymbol{x})\, \mathcal{T}_0(\boldsymbol{x})}{\sum_{\boldsymbol{x}} p_t(y \mid \boldsymbol{x})\, \mathcal{T}_0(\boldsymbol{x})}.$$

In our algorithm, we use Monte Carlo estimation to approximate the likelihood:

$$p_t(y \mid \boldsymbol{x}) = \int p(y \mid \boldsymbol{x}_1)\, p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})\, \mathrm{d}\boldsymbol{x}_1 = \frac{1}{Z}\int \exp\left(f_y(\boldsymbol{x}_1)\right) p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})\, \mathrm{d}\boldsymbol{x}_1.$$

This is estimated using the following procedure (as shown in Lines 2 and 3 of Value Function 1):

$$\frac{1}{N}\sum_{i=1}^{N} \exp\left(f_y(\hat{\boldsymbol{x}}_1^i)\right), \quad \hat{\boldsymbol{x}}_1^i \sim \mathrm{Cat}(u_\theta(\boldsymbol{x}, t+\Delta t)), \quad i \in [N], \tag{17}$$

where we assume the pretrained flow model is perfect, i.e., $u_\theta(\boldsymbol{x}, t+\Delta t) = p_{1|t+\Delta t}(\boldsymbol{x}_1 \mid \boldsymbol{x})$.

We denote the Monte Carlo estimate of the likelihood in (17) as $\hat{q}$. Using this, we define the distribution used in importance sampling, where we first sample from $\mathcal{T}_0$ and reweight according to $\hat{q}$:

$$\mathcal{T}_2(\boldsymbol{x}) = \frac{\hat{q}(\boldsymbol{x})\, \mathcal{T}_0(\boldsymbol{x})}{\sum_{\boldsymbol{x}} \hat{q}(\boldsymbol{x})\, \mathcal{T}_0(\boldsymbol{x})}.$$

When resampling over K candidates $\boldsymbol{x}^k \sim \mathcal{T}_0$ and reweighting according to $\hat{q}$, the resulting (empirical) output distribution is:

$$\hat{\mathcal{T}}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k\, \mathbf{1}\left(\boldsymbol{x} = \boldsymbol{x}^k\right),$$

where $w_k = \hat{q}(\boldsymbol{x}^k) / \sum_{i=1}^{K} \hat{q}(\boldsymbol{x}^i)$.

We can therefore decompose the distance between the empirical distribution $\hat{\mathcal{T}}$ and the target distribution $\mathcal{T}^*$ as:

$$\|\mathcal{T}^* - \hat{\mathcal{T}}\|_\infty \leq \|\mathcal{T}^* - \mathcal{T}_1\|_\infty + \|\mathcal{T}_1 - \mathcal{T}_2\|_\infty + \|\mathcal{T}_2 - \hat{\mathcal{T}}\|_\infty. \tag{18}$$

We now proceed to bound the three terms on the right-hand side of (18), starting with the first term involving $\mathcal{T}_1$. We have:

$$\begin{aligned}\sum_{\boldsymbol{x}} p_t(y \mid \boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x}) &= p_t(y \mid \boldsymbol{x}_t)\left(1 - \sum_{\boldsymbol{z}\neq\boldsymbol{x}_t} R_t(\boldsymbol{x}_t, \boldsymbol{z})\,\Delta t\right) + \sum_{\boldsymbol{z}\neq\boldsymbol{x}_t} p_t(y \mid \boldsymbol{z}) R_t(\boldsymbol{x}_t, \boldsymbol{z})\,\Delta t \\ &= p_t(y \mid \boldsymbol{x}_t) - p_t(y \mid \boldsymbol{x}_t) \sum_{\boldsymbol{z}\neq\boldsymbol{x}_t} R_t(\boldsymbol{x}_t, \boldsymbol{z})\,\Delta t + \sum_{\boldsymbol{z}\neq\boldsymbol{x}_t} p_t(y \mid \boldsymbol{z}) R_t(\boldsymbol{x}_t, \boldsymbol{z})\,\Delta t \\ &= p_t(y \mid \boldsymbol{x}_t) + \left(\sum_{\boldsymbol{z}\neq\boldsymbol{x}_t} [p_t(y \mid \boldsymbol{z}) - p_t(y \mid \boldsymbol{x}_t)] R_t(\boldsymbol{x}_t, \boldsymbol{z})\right)\Delta t \\ &= p_t(y \mid \boldsymbol{x}_t) + O(\Delta t).\end{aligned}$$

Therefore, for any $\boldsymbol{x} \neq \boldsymbol{x}_t$, we have:

$$\mathcal{T}_1(\boldsymbol{x}) = \frac{p_t(y \mid \boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})}{\sum_{\boldsymbol{x}} p_t(y \mid \boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})} = \frac{p_t(y \mid \boldsymbol{x})\,R_t(\boldsymbol{x}_t, \boldsymbol{x})\,\Delta t}{p_t(y \mid \boldsymbol{x}_t) + O(\Delta t)} = \frac{p_t(y \mid \boldsymbol{x})}{p_t(y \mid \boldsymbol{x}_t)} R_t(\boldsymbol{x}_t, \boldsymbol{x})\,\Delta t + O(\Delta t^2).$$

As a result, we can bound the distance as:

$$\|\mathcal{T}^* - \mathcal{T}_1\|_\infty = O(\Delta t^2). \tag{19}$$

We now bound the error introduced by the Monte Carlo estimation (17). By Lemma 3, with probability at least $1 - \delta/2$, we have:

$$\max_{\boldsymbol{x}\in\mathcal{X}} |\hat{q}(\boldsymbol{x}) - q_{1|t+\Delta t}(\boldsymbol{x})| \leq \epsilon_1,$$

where the unnormalized quantity is defined as

$$q_{1|t}(\boldsymbol{x}) := \int \exp\left(f_y(\boldsymbol{x}_1)\right) p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})\,\mathrm{d}\boldsymbol{x}_1,$$

and where

$$\epsilon_1 = B_{\max}\sqrt{\frac{\log(4|\mathcal{X}|/\delta)}{2N}}, \quad B_{\max} = \sup_{\boldsymbol{x}\in\mathcal{X}} \exp\left(f_y(\boldsymbol{x})\right).$$

Additionally, due to the Lipschitz continuity of $t \mapsto p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x})$, we have:

$$|p_{1|t}(\boldsymbol{x}_1 \mid \boldsymbol{x}) - p_{1|t+\Delta t}(\boldsymbol{x}_1 \mid \boldsymbol{x})| \leq L\,\Delta t,$$

which implies:

$$|q_{1|t}(\boldsymbol{x}) - q_{1|t+\Delta t}(\boldsymbol{x})| \leq B_{\max} L\,\Delta t.$$

Therefore, combining the Monte Carlo estimation error and the temporal approximation error, we get (with probability at least $1 - \delta/2$):

$$\max_{\boldsymbol{x}\in\mathcal{X}} |\hat{q}(\boldsymbol{x}) - q_{1|t}(\boldsymbol{x})| \leq \epsilon_1 + B_{\max} L\,\Delta t. \tag{20}$$

Recall that:

$$\mathcal{T}_1(\boldsymbol{x}) = \frac{q_{1|t}(\boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})}{\sum_{\boldsymbol{x}} q_{1|t}(\boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})}, \quad \text{where} \quad q_{1|t}(\boldsymbol{x}) = Z\,p_t(y \mid \boldsymbol{x}),$$

and

$$\mathcal{T}_2(\boldsymbol{x}) = \frac{\hat{q}(\boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})}{\sum_{\boldsymbol{x}} \hat{q}(\boldsymbol{x})\,\mathcal{T}_0(\boldsymbol{x})}.$$

By applying (20) and using Lemma 4, we obtain with probability at least $1 - \delta/2$:

$$\|\mathcal{T}_1 - \mathcal{T}_2\|_\infty \leq \frac{4}{B_{\min}}(\epsilon_1 + B_{\max} L\,\Delta t) = \frac{4B_{\max}}{B_{\min}}\left(\sqrt{\frac{\log(4|\mathcal{X}|/\delta)}{2N}} + L\,\Delta t\right), \tag{21}$$

where

$$B_{\min} = \inf_{\boldsymbol{x}\in\mathcal{X}} \exp\left(f_y(\boldsymbol{x})\right).$$

And it requires:

$$\epsilon_1 + B_{\max} L\,\Delta t < B_{\min}.$$

Finally, we bound the error introduced by sampling from the multinomial distribution over the K candidate branches. By Lemma 2, we have that with probability at least $1 - \delta/2$,

$$\|\mathcal{T}_2 - \hat{\mathcal{T}}\|_\infty \leq (1 + B_{\max}/B_{\min})\sqrt{\frac{\log(4|\mathcal{X}|/\delta)}{2K}}. \tag{22}$$

Now, combining the three error terms — from (19), (21), and (22) — into the decomposition (18), we obtain that with overall probability at least $1-\delta$,

$$\|\mathcal{T}^* - \hat{\mathcal{T}}\|_\infty \leq O(\Delta t^2) + \frac{4B_{\max}}{B_{\min}}\left(\sqrt{\frac{\log(4|\mathcal{X}|/\delta)}{2N}} + L\Delta t\right) + \sqrt{\frac{\log(4|\mathcal{X}|/\delta)}{2K}}.$$

To ensure that the total error stays below a prescribed threshold ε. Specifically, we choose the time step $\Delta t = O\left(\sqrt{\varepsilon} + \frac{B_{\min}}{B_{\max}L}\varepsilon\right)$, the Monte Carlo sample size $N = \Theta\left(\frac{B_{\max}^2 \log(4|\mathcal{X}|/\delta)}{B_{\min}^2 \varepsilon^2}\right)$, and the branch candidate size $K = \Theta\left(\frac{B_{\max}^2 \log(4|\mathcal{X}|/\delta)}{B_{\min}^2 \varepsilon^2}\right)$. If we omit constant factors, this completes the proof.

□

D.2 Proof of Lemma 1

Lemma (Recall Lemma 1)**.** *In both continuous and discrete cases, the transition probability during inference at timestep t satisfies:*

$$\mathcal{T}(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t) = \mathbb{E}_{\hat{\boldsymbol{x}}_1}\left[\mathcal{T}^\star(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t, \hat{\boldsymbol{x}}_1)\right],$$

where the expectation is taken over a distribution estimated by u_θ, with $\mathcal{T}^\star$ being the true posterior distribution predetermined by the noise schedule.

Proof. In the continuous case, the forward process is defined as:

$$\boldsymbol{x}_t \sim \mathcal{N}(\bar{\alpha}_t \boldsymbol{x}_1, (1-\bar{\alpha}_t)\boldsymbol{I}), \quad \boldsymbol{x}_{t+\Delta t} \sim \mathcal{N}(\bar{\alpha}_{t+\Delta t}\boldsymbol{x}_1, (1-\bar{\alpha}_{t+\Delta t})\boldsymbol{I}).$$

Given this, the posterior distribution conditioned on both $\boldsymbol{x}_t$ and $\boldsymbol{x}_1$ is:

$$p(\cdot \mid \boldsymbol{x}_t, \boldsymbol{x}_1) = \mathcal{N}(\cdot; c_{t,1}\boldsymbol{x}_t + c_{t,2}\boldsymbol{x}_1, \beta_t \boldsymbol{I}), \tag{23}$$

where the coefficients are defined as:

$$\beta_t = \frac{1-\bar{\alpha}_{t+\Delta t}}{1-\bar{\alpha}_t}(1-\alpha_t), \quad c_{t,1} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t+\Delta t})}{1-\bar{\alpha}_t}, \quad c_{t,2} = \frac{\sqrt{\bar{\alpha}_{t+\Delta t}}(1-\alpha_t)}{1-\bar{\alpha}_t}.$$

We set the transition distribution as:

$$\mathcal{T}^*(\cdot \mid \boldsymbol{x}_t, \boldsymbol{x}_1) = \mathcal{N}(\cdot; c_{t,1}\boldsymbol{x}_t + c_{t,2}\boldsymbol{x}_1, \beta_t \boldsymbol{I}).$$

Let the estimate $\hat{\boldsymbol{x}}_1$ be sampled from the distribution:

$$q_1 = \mathcal{N}(\boldsymbol{x}_{1|t}, (1-\alpha_t)\boldsymbol{I}),$$

where, as given in (4), the posterior mean $\boldsymbol{x}_{1|t}$ is:

$$\boldsymbol{x}_{1|t} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\cdot u_\theta(\boldsymbol{x}_t, t)\right).$$

Then, the expected approximate posterior becomes:

$$\mathbb{E}_{\hat{\boldsymbol{x}}_1 \sim q_1}\left[\mathcal{T}^*(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t, \hat{\boldsymbol{x}}_1)\right] = \mathcal{N}\left(c_{t,1}\boldsymbol{x}_t + c_{t,2}\boldsymbol{x}_{1|t}, \sigma_t^2 \boldsymbol{I}\right), \tag{24}$$

where $\sigma_t = \sqrt{1-\alpha_t}$.

Thus, (24) defines the transition distribution $\mathcal{T}(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t)$ used for sampling during inference.

For the discrete case, we recall the expression for the rate matrix used during inference, as given in (6):

$$R_t^{(d)}(\boldsymbol{x}_t, j) = \mathbb{E}_{x_1^{(d)} \sim u_\theta^{(d)}(x_1 \mid \boldsymbol{x}_t)}\left[R_t(x_t^{(d)}, j \mid x_1^{(d)})\right],$$

where the conditional rate matrix is defined as

$$R_t(x_t, j \mid x_1) = \frac{\delta\{j, x_1\}}{1-t} \cdot \mathbf{1}\{x_t = M\}.$$

Given $\boldsymbol{x}_t$ and $\boldsymbol{x}_1$, the transition probability is

$$\mathcal{T}^{*,(d)}(j \mid \boldsymbol{x}_t, \boldsymbol{x}_1) = \delta\{x_t^{(d)}, j\} + R_t(x_t^{(d)}, j \mid x_1^{(d)}) \cdot \Delta t.$$

Taking the expectation over $\boldsymbol{x}_1 \sim u_\theta(\cdot \mid \boldsymbol{x}_t)$, we obtain

$$\mathbb{E}_{\boldsymbol{x}_1 \sim u_\theta}\left[\mathcal{T}^{*,(d)}(j \mid \boldsymbol{x}_t, \boldsymbol{x}_1)\right] = \delta\{x_t^{(d)}, j\} + \mathbb{E}_{x_1^{(d)} \sim u_\theta^{(d)}(x_1 \mid \boldsymbol{x}_t)}\left[R_t(x_t^{(d)}, j \mid x_1^{(d)})\right] \cdot \Delta t,$$

which corresponds to the transition probability $\mathcal{T}(\boldsymbol{x}_{t+\Delta t} \mid \boldsymbol{x}_t)$. This completes the proof. □

D.3 Proof of Theorem 2

Theorem (Restatement of Theorem 2)**.** *Consider TreeG-Sampling Destination at time t, with an active set of size one and selection performed via multinomial resampling. Then, for any $\varepsilon, \delta > 0$, it holds with probability $1-\delta$:*

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\ell < \varepsilon,$$

provided one of the following conditions is satisfied:

(a, Continuous)*It holds under the $\ell = 1$ norm. Suppose data follow Gaussian distribution and the objective function is linear. The Branch-Out size is $K = \Theta(\frac{\log(1/\delta)}{\varepsilon^2})$ and we set $\rho_t = 1-\alpha_t, \tau_t = \frac{1-\bar{\alpha}_{t+\Delta t}}{1-\bar{\alpha}_t}(1-\alpha_t)$.*

(b, Discrete)*It holds under the $\ell = \infty$ norm. The likelihood scores satisfy $0 < B_{\min} \le \exp(f_y(\boldsymbol{x})) \le B_{\max}$ for all $\boldsymbol{x}$. The Branch-Out size is $K = \Theta(\frac{D^2 \log(|\mathcal{X}|/\delta)}{\varepsilon^2})$ where $\mathcal{X}$ is the data space and D is its dimension.*

Proof. **Proof of (a) continuous case:** We begin by recalling the unconditional transition process given in (3):

$$\boldsymbol{x}_{t+\Delta t} = c_1 \boldsymbol{x}_t + c_2 \boldsymbol{x}_{1|t} + \sigma_t \boldsymbol{\epsilon},$$

where the coefficients are defined as $c_1 = c_{t,1} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t+\Delta t})}{1-\bar{\alpha}_t}, c_2 = c_{t,2} = \frac{\sqrt{\bar{\alpha}_{t+\Delta t}}(1-\alpha_t)}{1-\bar{\alpha}_t}, \sigma_t = \sqrt{1-\alpha_t}$, and $\boldsymbol{x}_{1|t} = \mathbb{E}[\boldsymbol{x}_1 \mid \boldsymbol{x}_t]$.

Assuming that the data distribution is Gaussian and the objective function is linear, we have:

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I}), \qquad f_y(\boldsymbol{x}) = \boldsymbol{g}^\top \boldsymbol{x}.$$

In Line 2 of Module 2 and the associated Value Function 2, a candidate $\hat{\boldsymbol{x}}_1$ is proposed by sampling from $\mathcal{N}(\boldsymbol{x}_{1|t}, \rho_t \boldsymbol{I})$ and reweighting using $\exp(f_y(\boldsymbol{x})) = \exp(\boldsymbol{g}^\top \boldsymbol{x})$. Let $\hat{q}$ denote the distribution density of the output $\hat{\boldsymbol{x}}_1$. According to Lemma 2, with probability at least $1-\delta$, the distance between $\hat{q}$ and the ideal distribution q satisfies

$$\|\hat{q} - q\|_1 < 4(1+C)\sqrt{\frac{\log(4/\delta)}{2K}}, \tag{25}$$

where the ideal distribution is

$$q(\boldsymbol{x}) \propto \exp(\boldsymbol{g}^\top \boldsymbol{x}) \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}_{1|t}, \rho_t \boldsymbol{I}),$$

and the constant C is defined as $C = \frac{\|\boldsymbol{g}\|_2^2}{|\boldsymbol{g}^\top \boldsymbol{x}_{1|t}|}$. It follows from this form that q is a Gaussian with mean $\boldsymbol{x}_{1|t} + \rho_t \boldsymbol{g}$ and covariance $\rho_t I$, i.e.,

$$q = \mathcal{N}(\boldsymbol{x}_{1|t} + \rho_t \boldsymbol{g}, \rho_t \boldsymbol{I}).$$

In Line 3 of Module 2, the next sample $\boldsymbol{x}_{t+\Delta t}$ is generated according to

$$\boldsymbol{x}_{t+\Delta t} \sim \mathcal{N}(c_1 \boldsymbol{x}_t + c_2 \hat{\boldsymbol{x}}_1, \tau_t \boldsymbol{I}), \quad \hat{\boldsymbol{x}}_1 \sim \hat{q}.$$

Consequently, the distribution of $\boldsymbol{x}_{t+\Delta t}$, denoted $\hat{\mathcal{T}}$, is the convolution [6] of $c_2\hat{q}$ and φ, where $\varphi = \mathcal{N}(c_1 \boldsymbol{x}_t, \tau_t \boldsymbol{I})$:

$$\hat{\mathcal{T}} = (c_2 \hat{q}) * \varphi.$$

Next, we will show that the true conditional transition probability satisfies

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y) = (c_2 q) * \varphi,$$

by setting

$$\rho_t = 1 - \alpha_t, \qquad \tau_t = \frac{1-\bar{\alpha}_{t+\Delta t}}{1-\bar{\alpha}_t}(1-\alpha_t).$$

By the property of convolution between Gaussian distributions, we have

$$\begin{aligned}(c_2 q) * \varphi &= \mathcal{N}(c_2 \boldsymbol{x}_{1|t} + c_2 \rho_t \boldsymbol{g},\, c_2^2 \rho_t\, \boldsymbol{I}) * \mathcal{N}(c_1 \boldsymbol{x}_t,\, \tau_t\, \boldsymbol{I}) \\ &= \mathcal{N}(c_1 \boldsymbol{x}_t + c_2 \boldsymbol{x}_{1|t} + c_2 \rho_t \boldsymbol{g},\, (\tau_t + c_2^2 \rho_t)\, \boldsymbol{I}).\end{aligned}$$

[6] The convolution operation models the distribution of the sum of two independent random variables. For probability density functions f and g on $\mathbb{R}^d$, the convolution is defined as $(f * g)(\boldsymbol{x}) = \int_{\mathbb{R}^d} f(\boldsymbol{y}) g(\boldsymbol{x} - \boldsymbol{y})\, d\boldsymbol{y}$. In the case of Gaussian distributions, this results in a new Gaussian with summed means and covariances.

We now verify that this matches the conditional transition distribution by examining the mean and variance. Starting with the variance term, we compute

$$\begin{aligned}\tau_t + c_2^2\rho_t &= \frac{1-\bar{\alpha}_{t+\Delta t}}{1-\bar{\alpha}_t}(1-\alpha_t) + \left(\frac{\sqrt{\bar{\alpha}_{t+\Delta t}}(1-\alpha_t)}{1-\bar{\alpha}_t}\right)^2(1-\bar{\alpha}_t)\\ &= \frac{1-\alpha_t}{1-\bar{\alpha}_t}\left(1-\bar{\alpha}_{t+\Delta t}+\bar{\alpha}_{t+\Delta t}-\bar{\alpha}_{t+\Delta t}\alpha_t\right)\\ &= \frac{1-\alpha_t}{1-\bar{\alpha}_t}(1-\bar{\alpha}_{t+\Delta t}\alpha_t) = 1-\alpha_t.\end{aligned}$$

For the mean, we have

$$c_1\boldsymbol{x}_t + c_2\boldsymbol{x}_{1|t} + c_2\rho_t\boldsymbol{g} = c_1\boldsymbol{x}_t + c_2\boldsymbol{x}_{1|t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t+\Delta t}}\,\boldsymbol{g}.$$

Recalling from (13) in the proof in App. D.1, the target conditional transition distribution is

$$\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y) = \mathcal{N}\left(\cdot;\, \bar{\boldsymbol{x}}_{t+\Delta t} + (1-\alpha_t)\sqrt{\bar{\alpha}_{t+\Delta t}}\,\boldsymbol{g},\, \sigma_t^2\boldsymbol{I}\right),$$

where the mean is $\bar{\boldsymbol{x}}_{t+\Delta t} = c_1\boldsymbol{x}_t + c_2\boldsymbol{x}_{1|t}$.

Hence, the convolution $(c_2 q) * \varphi$ exactly recovers the conditional transition distribution $\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)$. Therefore, we bound the distance between the output distribution and target transition distributions as follows:

$$\begin{aligned}||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_1 &= ||(c_2\hat{q}) * \varphi - (c_2 q) * \varphi||_1\\ &\le ||c_2\hat{q} - c_2 q||_{\mathrm{TV}} = c_2||\hat{q} - q||_1,\end{aligned}$$

where the inequality follows from the data processing inequality under convolution with a fixed distribution.

With (25), we obtain that with probability at least $1-\delta$,

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_1 \le 4c_2(1+C)\sqrt{\frac{\log(4/\delta)}{2K}}.$$

Hence, to ensure the approximation error is at most ε, it suffices to choose the branch-out size $K = \Theta\left(c_2^2(1+C)^2\,\frac{\log(1/\delta)}{\varepsilon^2}\right)$, up to constant factors, which completes the proof.

Proof of (b) discrete case:

In Line 2 of Module 2, and with Value Function 2, a candidate state $\hat{\boldsymbol{x}}_1$ is proposed by sampling from the distribution $p_{1|t}(\cdot \mid \boldsymbol{x}_t)$(since u_θ is optimal estimation), where the reweighting term $\exp(f_y(\boldsymbol{x}_1)) = p(y \mid \boldsymbol{x}_1)$. Let $\hat{q}$ denote the density of the resulting distribution over $\hat{\boldsymbol{x}}_1$.

According to Lemma 2, with probability at least $1-\delta$, the ℓ_∞ distance between $\hat{q}$ and the ideal target distribution q is bounded as:

$$||\hat{q} - q||_\infty \le \left(1 + \frac{B_{\max}}{B_{\min}}\right)\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}, \tag{26}$$

where the ideal distribution is given by

$$q(\cdot) \propto p(y \mid \cdot)\, p_{1|t}(\cdot \mid \boldsymbol{x}_t),$$

and hence

$$q = p_{1|t}(\cdot \mid \boldsymbol{x}_t, y).$$

Here, $B_{\max} = \sup_{\boldsymbol{x}\in\mathcal{X}} \exp(f_y(\boldsymbol{x}))$ and $B_{\min} = \inf_{\boldsymbol{x}\in\mathcal{X}} \exp(f_y(\boldsymbol{x}))$.

Next, in Line 3 of Module 2, the next state $\boldsymbol{x}_{t+\Delta t}$ is sampled according to:

$$P_1\left(x_{t+\Delta t}^{(d)} = j\right) = \mathbb{E}_{\boldsymbol{x}_1\sim\hat{q}}\left[\delta\{x_t^{(d)}, j\} + R_t\left(x_t^{(d)}, j \mid x_1^{(d)}\right)\Delta t\right],$$

where the conditional transition rate matrix satisfies:

$$R_t^{(d)}\left(\boldsymbol{x}_t, j \mid y\right) = \mathbb{E}_{x_1^{(d)}\sim p_{1|t}^{(d)}(\boldsymbol{x}_1|y)}\left[R_t\left(x_t^{(d)}, j \mid x_1^{(d)}\right)\right],$$

as given in (8).

Accordingly, the target conditional transition probability under $\mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)$ is:

$$P_2\left(x_{t+\Delta t}^{(d)}, j\right) = \mathbb{E}_{\boldsymbol{x}_1\sim q}\left[\delta\{x_t^{(d)} = j\} + R_t\left(x_t^{(d)}, j \mid x_1^{(d)}\right)\Delta t\right],$$

with $q(\cdot) = p_{1|t}(\cdot \mid \boldsymbol{x}_t, y)$.

Therefore, for all dimensions $d \in [D]$, the deviation between the approximate and conditional transition probabilities is bounded by:

$$\left|P_1\left(x_{t+\Delta t}^{(d)} = j\right) - P_2\left(x_{t+\Delta t}^{(d)} = j\right)\right| \leq ||\hat{q} - q||_\infty.$$

Taking the product structure across all dimensions, we obtain the total deviation in transition operators:

$$||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\infty \leq D \cdot ||\hat{q} - q||_\infty.$$

By substituting the bound from (26), we conclude that to ensure $||\hat{\mathcal{T}} - \mathcal{T}(\cdot \mid \boldsymbol{x}_t, y)||_\infty \leq \varepsilon$ with probability at least $1 - \delta$, it suffices to choose $K = \Theta\left(\frac{B_{\max}^2}{B_{\min}^2} \cdot \frac{D^2 \log(|\mathcal{X}|/\delta)}{\varepsilon^2}\right)$.

□

D.4 Auxiliary Lemmas

In this section, we provide auxiliary lemmas with proofs.

Lemma 2. *Let $X_1, \ldots, X_K$ be i.i.d. draws from a proposal density p on a measurable space $(\mathcal{X}, \mathcal{A})$. Define*

$$w_i = q(X_i) \geq 0, \qquad Z = \int_{\mathcal{X}} p(x)\, q(x)\, dx,$$

and normalized weights $\tilde{w}_i = \dfrac{w_i}{\sum_{j=1}^K w_j}$. Let $X_1^*, \ldots, X_K^*$ be a multinomial resample of $\{X_i\}$ with probabilities $\{\tilde{w}_i\}$, and define the empirical law

$$\hat{\mu}_K(A) = \frac{1}{K}\sum_{k=1}^K \mathbf{1}_{\{X_k^* \in A\}}, \qquad \mu_q(A) = \int_A \frac{p(x)\, q(x)}{Z}\, dx.$$

Then for any $\delta \in (0,1)$, with probability at least $1 - \delta$, where one may take either

(a) If $0 \leq q(x) \leq M < \infty$ and $\mathcal{X}$ is finite, then

$$\|\hat{\mu}_K - \mu_q\|_\infty = \sup_{A \in \mathcal{A}} |\hat{\mu}_K(A) - \mu_q(A)| \;\leq\; (1 + \frac{M}{Z})\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}.$$

(b) If $p(x) = \mathcal{N}(\mu, \Sigma)$, $q(x) = g^\top x$, then

$$\|\hat{\mu}_K - \mu_q\|_1 \leq\; 4\Big(1 + \frac{\sqrt{g^\top \Sigma\, g}}{|g^\top \mu|}\Big)\sqrt{\frac{\log(4/\delta)}{2K}}.$$

Proof. **For (a):** Define the measure

$$\tilde{\mu}(A) = \sum_{i=1}^K \tilde{w}_i\, \mathbf{1}_{\{X_i \in A\}} = \frac{\sum_{i=1}^K q(X_i)\, \mathbf{1}_{\{X_i \in A\}}}{\sum_{j=1}^K q(X_j)}.$$

Then by the triangle inequality,

$$\|\hat{\mu}_K - \mu_q\|_\infty \leq\; \|\hat{\mu}_K - \tilde{\mu}\|_\infty + \|\tilde{\mu} - \mu_q\|_\infty.$$

To bound $\|\hat{\mu}_K - \tilde{\mu}\|_{\mathrm{TV}}$, note that conditional on $\{X_i\}$, the resampled points $X_1^*, \ldots, X_K^*$ are i.i.d. draws from the finite-support distribution $\tilde{\mu}$. By Hoeffding's inequality, for any measurable A,

$$\Pr(|\hat{\mu}_K(A) - \tilde{\mu}(A)| > \varepsilon \mid \{X_i\}) \;\leq\; 2\exp(-2K\varepsilon^2).$$

A union bound over the two tails and taking the supremum over A implies that, with probability at least $1 - \dfrac{\delta}{2}$,

$$\|\hat{\mu}_K - \tilde{\mu}\|_\infty \leq\; \sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}.$$

To bound $\|\tilde{\mu} - \mu_q\|_\infty$, observe that for any A,

$$\tilde{\mu}(A) - \mu_q(A) = \frac{\frac{1}{K}\sum_{i=1}^K q(X_i)\, \mathbf{1}_A - Z\, \mu_q(A)}{\frac{1}{K}\sum_{j=1}^K q(X_j)}.$$

Since $0 \le q \le M$, Hoeffding's inequality yields simultaneously, with probability at least $1 - \frac{\delta}{2|\mathcal{X}|}$, the two bounds

$$\Big|\frac{1}{K}\sum_{i=1}^{K} q(X_i)\,\mathbf{1}_A - Z\,\mu_q(A)\Big| \;\le\; M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}, \quad \Big|\frac{1}{K}\sum_{j=1}^{K} q(X_j) - Z\Big| \;\le\; M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}.$$

On this event, the denominator satisfies $\frac{1}{K}\sum_j q(X_j) \ge Z - M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}$. Assuming K is large enough that $M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}} < Z/2$, we obtain

$$|\tilde{\mu}(A) - \mu_q(A)| \le \frac{M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}}{Z - M\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}} \;\le\; \frac{M}{Z}\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}},$$

and taking supremum over $A = \{x\}, x \in \mathcal{X}$ gives

$$\|\tilde{\mu} - \mu_q\|_\infty \le \frac{M}{Z}\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}}.$$

By a union bound the two high-probability events both hold with probability at least $1 - \delta$. Adding the two bounds yields

$$\|\hat{\mu}_K - \mu_q\|_\infty \le \sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}} + \frac{M}{Z}\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}} = (1 + M/Z)\sqrt{\frac{\log(2|\mathcal{X}|/\delta)}{2K}},$$

as required.

For (b): Let $\widetilde{Y}_i = \frac{w_i}{\mathbb{E}[w_i]} = \frac{g^\top X_i}{g^\top \mu}$. Define the intermediate measure

$$\nu_K(A) = \frac{1}{K}\sum_{i=1}^{K} \frac{w_i}{\mathbb{E}[w_i]}\,\mathbf{1}_{\{X_i \in A\}} = \frac{1}{K}\sum_{i=1}^{K} \widetilde{Y}_i\,\mathbf{1}_{\{X_i \in A\}}.$$

Since each $\widetilde{Y}_i$ has mean 1 and is sub-Gaussian with parameter $\tau^2 = \frac{g^\top \Sigma g}{(g^\top \mu)^2}$, a Bernstein inequality yields that with probability at least $1 - \frac{\delta}{2}$,

$$\|\nu_K - \mu_q\|_1 \;\le\; 2\Big(1 + \sqrt{2}\,\tau\Big)\sqrt{\frac{\log(4/\delta)}{2K}}.$$

Next observe that

$$\hat{\mu}_K(A) = \frac{Z}{\bar{w}}\,\nu_K(A),$$

so that

$$\|\hat{\mu}_K - \nu_K\|_1 \;\le\; 2|\frac{\bar{w}}{Z} - 1|.$$

Applying the same sub-Gaussian tail bound to $\bar{w}/Z = \frac{1}{K}\sum_i \widetilde{Y}_i$ shows that with probability at least $1 - \frac{\delta}{2}$,

$$|\frac{\bar{w}}{Z} - 1| \;\le\; \Big(1 + \sqrt{2}\,\tau\Big)\,2\sqrt{\frac{\log(4/\delta)}{2K}},$$

provided K is large enough that the deviation does not exceed $1/2$. The union bound then guarantees both concentration events hold with probability at least $1 - \delta$. Finally, the triangle inequality

$$\|\hat{\mu}_K - \mu_q\|_1 \;\le\; \|\hat{\mu}_K - \nu_K\|_1 + \|\nu_K - \mu_q\|_1$$

yields the asserted bound.

□

Lemma 3. *Let $\mathcal{X}$ be a finite set of cardinality M. For each $x \in \mathcal{X}$, let $Y(x)$ be a real-valued random variable with mean $g(x) = \mathbb{E}[Y(x)]$ and almost-sure bounds $0 \le Y(x) \le B$. Draw N independent copies $Y_1(x), \ldots, Y_N(x)$ of each $Y(x)$ and form the empirical average*

$$\hat{g}(x) \;=\; \frac{1}{N}\sum_{i=1}^{N} Y_i(x).$$

Then with probability at least $1 - \delta$,

$$\max_{x \in \mathcal{X}} |\hat{g}(x) - g(x)| \;\le\; B\sqrt{\frac{\log(2M/\delta)}{2\,N}}.$$

Proof. Fix any $x \in \mathcal{X}$. Since $Y_1(x), \dots, Y_N(x)$ are i.i.d. in $[0, B]$ with mean $g(x)$, Hoeffding's inequality yields, for every $\epsilon > 0$,

$$\Pr(|\hat{g}(x) - g(x)| \geq \epsilon) \;\leq\; 2\exp\Big(-\frac{2\,N\,\epsilon^2}{B^2}\Big).$$

Applying the union bound over all M elements of $\mathcal{X}$, we obtain

$$\Pr(\exists\, x \in \mathcal{X} :\; |\hat{g}(x) - g(x)| \geq \epsilon) \;\leq\; 2M\exp\Big(-\frac{2\,N\,\epsilon^2}{B^2}\Big).$$

To ensure this probability is at most δ, set

$$2M\exp\Big(-\frac{2\,N\,\epsilon^2}{B^2}\Big) = \delta.$$

Taking natural logarithms gives

$$-\frac{2\,N\,\epsilon^2}{B^2} + \log(2M) = \log\delta \quad\Longrightarrow\quad \epsilon^2 = \frac{B^2}{2N}\,\log\Big(\frac{2M}{\delta}\Big),$$

hence

$$\epsilon = B\,\sqrt{\frac{\log(2M/\delta)}{2\,N}}.$$

Therefore, with probability at least $1 - \delta$, no empirical average deviates by more than ϵ, i.e.

$$\max_{x\in\mathcal{X}} |\hat{g}(x) - g(x)| \leq \epsilon.$$

□

Lemma 4. *Let $\mathcal{X}$ be a finite state space and let T_0 be a probability distribution on $\mathcal{X}$, so that $\sum_{x\in\mathcal{X}} T_0(x) = 1$. Fix two nonnegative weight functions $q, \hat{q} : \mathcal{X} \to \mathbb{R}_{\geq 0}$ and define*

$$T_1(x) \;=\; \frac{q(x)\,T_0(x)}{\sum_{z\in\mathcal{X}} q(z)\,T_0(z)}, \quad T_2(x) \;=\; \frac{\hat{q}(x)\,T_0(x)}{\sum_{z\in\mathcal{X}} \hat{q}(z)\,T_0(z)}.$$

Suppose there exists $q_{\min} > 0$ and $\Delta \in [0, q_{\min})$ such that

$$\min_{x\in\mathcal{X}} q(x) \;\geq\; q_{\min}, \qquad \max_{x\in\mathcal{X}} |\hat{q}(x) - q(x)| \;\leq\; \Delta.$$

Then

$$\|T_1 - T_2\|_\infty \;\leq\; \|T_1 - T_2\|_1 \;\leq\; \frac{2\Delta}{q_{\min} - \Delta} \;\leq\; \frac{4\,\Delta}{q_{\min}},$$

where the final inequality holds whenever $\Delta \leq q_{\min}/2$.

Proof. Set

$$a_x = q(x)\,T_0(x), \quad b_x = \hat{q}(x)\,T_0(x), \quad A = \sum_x a_x, \quad B = \sum_x b_x.$$

Then $T_1(x) = a_x/A$ and $T_2(x) = b_x/B$. Since $q(x) \geq q_{\min}$ and $\sum_x T_0(x) = 1$,

$$A = \sum_x q(x)T_0(x) \;\geq\; q_{\min},$$

and because $|b_x - a_x| \leq \Delta\,T_0(x)$,

$$|B - A| = \Big|\sum_x (b_x - a_x)\Big| \leq \Delta \sum_x T_0(x) = \Delta,$$

so $B \geq A - \Delta \geq q_{\min} - \Delta > 0$. Now

$$\begin{aligned}
\|T_1 - T_2\|_1 &= \sum_x \Big|\frac{a_x}{A} - \frac{b_x}{B}\Big| = \sum_x \Big|\frac{a_x B - b_x A}{A\,B}\Big| = \frac{1}{A\,B}\sum_x \Big|(a_x - b_x)B + a_x(B - A)\Big| \\
&\leq \frac{B}{A\,B}\sum_x |a_x - b_x| + \frac{|B - A|}{A\,B}\sum_x a_x = \frac{1}{A}\sum_x |a_x - b_x| + \frac{|B - A|}{B} \\
&\leq \frac{\Delta}{A} + \frac{\Delta}{A - \Delta} \;=\; \frac{\Delta\,(A - \Delta + A)}{A\,(A - \Delta)} \;=\; \frac{2\,A\,\Delta - \Delta^2}{A\,(A - \Delta)} \\
&\leq \frac{2\,A\,\Delta}{A\,(A - \Delta)} \;=\; \frac{2\,\Delta}{A - \Delta} \;\leq\; \frac{2\,\Delta}{q_{\min} - \Delta}.
\end{aligned}$$

When $\Delta \leq q_{\min}/2$, $q_{\min} - \Delta \geq q_{\min}/2$, so

$$\|T_1 - T_2\|_1 \leq \frac{2\,\Delta}{q_{\min} - \Delta} \leq \frac{2\,\Delta}{q_{\min}/2} = \frac{4\,\Delta}{q_{\min}},$$

and one may tighten the constants to obtain the stated $\leq 2\Delta/q_{\min}$. □

E Algorithmic Details

E.1 Discussion on Existing Works

In this section, we will show that the recent works, stochastic control guidance (SCG) [30] and soft value decoding guidance (SVDD) [37], can be viewed as special cases of our TreeG-SC.

At timestep $t - \Delta t$, both SCG and SVDD sample multiple candidate next states from the original diffusion model: $\boldsymbol{x}_t^1, \dots \boldsymbol{x}_t^n$. They then select one of these candidates, $\boldsymbol{x}_t^k$, according to the following strategies:

SCG: $k = \arg\max_i \log p(y|\hat{\boldsymbol{x}}_1(\boldsymbol{x}_t^i))$,

SVDD (training-free version): $k \sim \text{Cat}(\frac{w_i}{\sum w_i}), w_i = \exp(r(\hat{\boldsymbol{x}}_1(\boldsymbol{x}_t^i))/\alpha), r(\cdot) = \log p(y|\cdot)$,

where $\hat{\boldsymbol{x}}_1(\boldsymbol{x}_t) = \left(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t} u_\theta\left(\boldsymbol{x}_t, t\right)\right) / \sqrt{\bar{\alpha}_t}$ and $\alpha > 0$ is a temperature parameter.

Within our TreeG-SC algorithm, SCG corresponds to setting the active set size $A = 1$ and selecting the candidate via ranking, while SVDD corresponds to setting the scoring function $f_y(\cdot) = \frac{1}{\alpha} \log p(y \mid \cdot)$, also with $A = 1$, and selecting the candidate via resampling based on soft values.

E.2 TreeG-Gradient Details

We provide the algorithmic details for `BranchOut`-Gradient in Module 3.

Module 3 `BranchOut`-Gradient

1: **Input:** $\boldsymbol{x}_t, t$, diffusion model u_θ, differentiable predictor f_y, guidance strength γ_t, (optional) Monte-Carlo sample size N.
2: **Compute the gradient guidance:**
(continuous) $\boldsymbol{g} = \nabla_{\boldsymbol{x}_t} f_y(\hat{\boldsymbol{x}}_1)$ with $\hat{\boldsymbol{x}}_1 = u_\theta(\boldsymbol{x}_t, t)$.
(discrete) $\boldsymbol{g}^{(d)} = (\boldsymbol{x}_t^{\backslash d} - \boldsymbol{x}_t)^\top \nabla_{\boldsymbol{x}_t} \frac{1}{N} \sum_{i=1}^N f_y(\hat{\boldsymbol{x}}_1^i)$
with $\hat{\boldsymbol{x}}_1^i \sim \text{Cat}\left(u_\theta(\boldsymbol{x}_t, t)\right), i \in [N]$.
3: **Sample the next state:**
(continuous) $\boldsymbol{x}_{t+\Delta t} = \gamma_t \boldsymbol{g} + c_{t,1} \boldsymbol{x}_t + c_{t,2} \hat{\boldsymbol{x}}_1 + \sigma_t \epsilon$ with $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.
(discrete) $x_{t+\Delta t}^{(d)} \sim \text{Cat}\left(\delta\{x_t^{(d)}, j\} + \exp(\gamma_t \boldsymbol{g}^{(d)}) \odot R_{\theta,t}^{(d)}(\boldsymbol{x}_t, j)\, \Delta t\right)$.
4: **Output:** $\boldsymbol{x}_{t+\Delta t}$

E.3 More Implementation Details for Continuous Models

For TreeG-SD on the continuous case, we have two additional designs: the first one is exploring multiple steps when branching out a destination state; the second one involves plugging in Spherical Gaussian constraint(DSG) from [75]. We present the case $A = 1$ for TreeG-SD on the continuous case as follows while $A > 1$ is similar.

Notice that the computation complexity for using N_{iter} step to select is: $AC_{\text{model}} + AKN_{\text{iter}}C_{\text{pred}}$. The setting of N_{iter} will be provided in Appendix F.1.

E.4 More Implementation Details for Discrete Models

Estimate $\nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t)$. Since the sampling process of discrete data is genuinely not differentiable, we adopt the Straight-through Gumbel Softmax trick to estimate the gradient while combining Monte-Carlo Sampling as stated in (9). The whole process is listed in Module 4.

Algorithm 7 TreeG-SD (Continuous, $A = 1$)

Input: diffusion model u_θ, objective function f_y, branch out sample size K, stepsize scale ρ_t, number of iteration N_{iter}
$t = 0$, $\boldsymbol{x}_0 \sim p_0$
while $t < 1$ **do**
  Compute the predicted clean sample $\hat{\boldsymbol{x}}_1 = u_\theta(\boldsymbol{x}_t, t)$
  Set the branch out state $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_1$
  for $n = 1, \ldots, N_{\text{iter}}$ **do**
    Sample $\boldsymbol{x}^i = \boldsymbol{x} + \rho_t \boldsymbol{\xi}^i$, with $\boldsymbol{\xi}^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
    Evaluate and select that maximizes the objective: $k = \text{argmax}_i \, f_y\left(\boldsymbol{x}^i\right)$.
    Update $\boldsymbol{x} \leftarrow \boldsymbol{x}^k$.
  end for
  if DSG [75] **then**
    Compute the selected direction: $\boldsymbol{\xi}^* = \boldsymbol{x} - \hat{\boldsymbol{x}}_1$
    Rescale the direction: $\boldsymbol{\xi}^* \leftarrow \sqrt{D} \cdot \frac{\boldsymbol{\xi}^*}{||\boldsymbol{\xi}^*||}$, with $D = \dim(\boldsymbol{x}_t)$.
    Compute the next state: $\boldsymbol{x}_{t+\Delta t} = c_{t,1}\boldsymbol{x}_t + c_{t,2}\hat{\boldsymbol{x}}_1 + \sigma_t \boldsymbol{\xi}^*$
  else
    Get the selected destination state $\hat{\boldsymbol{x}}_1 \leftarrow \boldsymbol{x}$
    Sample the next state: $\boldsymbol{x}_{t+\Delta t} = c_{t,1}\boldsymbol{x}_t + c_{t,2}\hat{\boldsymbol{x}}_1 + \sigma_t \epsilon$ with $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  end if
  $t \leftarrow t + \Delta t$
end while

Module 4 Gradient Approximation with Straight-Through Gumbel Softmax.

Input: $\boldsymbol{x}_t, t$, diffusion model u_θ, differentiable predictor f_y, Monte-Carlo sample size N, number of possible states S, Gumbel-Softmax temperature τ
Sample $\hat{\boldsymbol{x}}_1^i \sim \text{Cat}\left(u_\theta(\boldsymbol{x}_t, t)\right), i \in [N]$ with Gumbel Max and represent x_1 as an one-hot vector:

$$\hat{\boldsymbol{x}}_1^i = \arg\max_j (\log u_\theta(\boldsymbol{x}_t, t) + g^i), \tag{27}$$

g^i is a S-dimension Gumbel noise where $g_j \sim \text{Gumbel}(0, 1), j \in [S]$.
Since the argmax operation is not differentiable, get the approximation $\hat{\boldsymbol{x}}_1^{i*}$ with softmax:

$$\hat{\boldsymbol{x}}_{1j}^{i*} = \frac{\exp((\log u_\theta(\boldsymbol{x}_t, t) + g_j^i)/\tau)}{\sum_k \exp((\log u_\theta(\boldsymbol{x}_t, t) + g_k^i)/\tau)}, \tag{28}$$

Feed $\hat{\boldsymbol{x}}_1^i$ into the f_y and obtain the gradient through backpropagation: $\nabla_{\hat{\boldsymbol{x}}_1^i} \log f_y(\hat{\boldsymbol{x}}_1^i)$.
Straight-through Estimator: directly copy the gradient $\nabla_{\hat{\boldsymbol{x}}_1^i} \log f_y(\hat{\boldsymbol{x}}_1^i)$ to $\hat{\boldsymbol{x}}_{1j}^{i*}$, i.e., $\nabla_{\hat{\boldsymbol{x}}_1^{i*}} \log f_y(\hat{\boldsymbol{x}}_1^{i*}) \simeq \nabla_{\hat{\boldsymbol{x}}_1^i} \log f_y(\hat{\boldsymbol{x}}_1^i)$.
Since Eq. (28) is differentiable with regard to x_t, get $\nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t) \simeq \nabla_{\boldsymbol{x}_t} \frac{1}{N} \sum_{i=1}^N f_y(\hat{\boldsymbol{x}}_1^{i*})$.
Output: $\nabla_{\boldsymbol{x}_t} \log p_t(y \mid \boldsymbol{x}_t)$

F Experimental Details

All experiments are conducted on one NVIDIA 80G H100 GPU.

F.1 Additional Setup for Symbolic Music Generation

Models. We utilize the diffusion model and Variational Autoencoder (VAE) from [30]. These models were originally trained on MAESTRO [23], Pop1k7 [28], Pop909 [71], and 14k midi files in the classical genre collected from MuseScore. The VAE encodes piano roll segments of dimensions $3 \times 128 \times 128$ into a latent space with dimensions $4 \times 16 \times 16$.

Objective Functions. For the tasks of interest—pitch histogram, note density, and chord progression—the objective function for a given target $\boldsymbol{y}$ is defined as: $f_{\boldsymbol{y}}(\boldsymbol{x}) = -\ell\left(\boldsymbol{y}, \texttt{Rule}(\boldsymbol{x})\right)$, where $\texttt{Rule}(\cdot)$ represents a rule function that extracts the corresponding feature from $\boldsymbol{x}$, and ℓ is the loss function. Below, we elaborate on the differentiability of these objective functions for each task:

For pitch histogram, the rule function $\texttt{Rule-PH}(\cdot)$ computes the pitch histogram, and the loss function ℓ is the L2 loss. Since $\texttt{Rule-PH}(\cdot)$ is differentiable, the resulting objective function $f_{\boldsymbol{y}}^{\text{PH}}$ is also differentiable.

For note density, the rule function for note density is defined as: $\texttt{Rule-ND}(\boldsymbol{x}) = \sum_{i=1}^{n} \mathbf{1}(x_i > \epsilon)$ where ϵ is a small threshold value, and $\mathbf{1}(\cdot)$ is the indicator function which makes $\texttt{Rule-ND}(\cdot)$ non-differentiable. ℓ is L2 loss. $f_{\boldsymbol{y}}^{\text{ND}}$ is overall non-differentiable.

For chord progression, the rule function $\texttt{Rule-CP}(\cdot)$ utilizes a chord analysis tool from the music21 package [14]. This tool operates as a black-box API, and the associated loss function ℓ is a 0-1 loss. Consequently, the objective function $f_{\boldsymbol{y}}^{\text{CP}}$ is highly non-differentiable.

Test Targets. Our workflow follows the methodology outlined by Huang et al. [30]. For each task, target rule labels are derived from 200 samples in the Muscore test dataset. A single sample is then generated for each target rule label, and the loss is calculated between the target label and the rule label of the generated sample. The mean and standard deviation of these losses across all 200 samples are reported in Table 1.

Inference Setup. We use a DDPM scheduler with 1000 inference steps. Guidance is applied only after step 250.

Chord Progression Setup. Since the objective function running by music21 package [14] is very slow, we only conduct guidance during 400-800 inference step.

TreeG-SD Setup. As detailed in Algorithm 7, we use DSG [75], and set $N_{\text{iter}} = 2$ for pitch histogram and note density, $N_{\text{iter}} = 1$ for chord progression. The stepsize $\rho_t = s \cdot \sigma_t / \sqrt{1 + \sigma_t^2}$ [53, 78], with $s = 2$ for pitch histogram, $s = 0.5$ for note density and $s = 1$ for chord progression.

F.2 Additional Setup for Small Molecule Generation

Due to lack of *differentiable* off-the-shelf predictors, we train a suite of regression models $f(\boldsymbol{x})$ on *clean* data x_1 following the same procedure described in Nisonoff et al. [45] to test the efficacy of our gradient-based method TreeG-G on the targets QED, SA, DRD2, and N_r. The same predictor is applied for all compared training-free guidance methods (TreeG, TFG-Flow, and SVDD).

For molecule optimization targets, QED, SA, and DRD2, we directly use the pretrained model $f(\boldsymbol{x})$ as the objective function f_y without setting any target values and generate 500 samples within 1000 denosing steps. As stated in [45], generated SMILES sequences may not yield *valid* molecule structures. Thus, we continue generating samples until the required number of *valid* and unique sequences is reached. For these three targets' results in Table 2, $A = 1, K = 2$ for TreeG-SC; $A = 1, K = 200$ and $A = 2, K = 8$ for TreeG-SD. To ensure comparable runtime between baseline models and TreeG, we set $N = 200$ for TFG-Flow and $K = 2$ for SVDD with a temperature parameter α of 0.1. Monte Carlo sample size for estimating $p_t(y \mid \boldsymbol{x}_t)$ in TreeG-G is $N = 30$ (Eq. (9)) while TreeG-SC and SVDD use $N = 10$.

We use QuickVina2-GPU-2.1 to evaluate the binding affinity of generated molecules to protein 5ht1b following [76]. The original scores DS are normalized as $max(-DS, 0)$ for maximization. Because using oracle docking scores as guidance signals is computationally expensive, we only generate 100 sequences in 200 inference steps for comparison. Since the docking measurement is *non-differentiable*, TreeG-G and DG are not applicable. In Table 2, $A = 1, K = 8$ for TreeG-SC; $A = 1, K = 200$ and $A = 4, K = 8$ for TreeG-SD; $N = 200$ for TFG-Flow; $K = 8, \alpha = 0.1$ for SVDD. Monte Carlo sample size for $p_t(y \mid \boldsymbol{x}_t)$ estimation in TreeG-SC and SVDD is $N = 5$.

While for N_r which is optimized towards specific target values, we adopt $f_y(\boldsymbol{x}) = -\frac{(y - f(\boldsymbol{x}))^2}{2\sigma^2}$ with σ learned during the training of $f(x)$. We report the results on 1000 generated *valid* unique sequences. In Table 2, $A = 1, K = 4$ for TreeG-SC; $A = 1, K = 200$ and $A = 2, K = 8$ for TreeG-SD; $N = 200$ for TFG-Flow; $K = 2, \alpha = 0.1$ for SVDD. Monte Carlo sample size for $p_t(y \mid \boldsymbol{x}_t)$ estimation in TreeG-SC, TreeG-G, and SVDD is $N = 30$.

F.3 Additional Setup for Enhancer DNA Design

We test on eight randomly selected classes with cell type indices 0, 2, 33, 4, 16, 5, 68, and 9. For simplicity, we refer to these as Class 1 through Class 8.

Table 7: Computation time per basic unit during small molecule generation (ms)

	C_{model}	$C_{\text{QED Objective}}$	$C_{\text{Docking Objective}}$
Molecule	0.038	2.0e-4	390

We set the Monte Carlo sample size of TreeG-G and TreeG-SC as $N = 20$, and $N = 200$ for TFG-Flow. For SVDD, we set the sample size to 16 and the temperature parameter $\alpha = 0.01$.

F.4 Experimental Setup in Figure 2

In Figure 2, (a) shows TreeG-SD applied to the note density objective in symbolic music generation; (b) shows TreeG-SD used for QED optimization in small molecule generation; and (c) shows TreeG-G applied to Enhancer DNA design.

G Additional Experiment Results

G.1 Additional Results for Symbolic Music Generation

G.1.1 Additional Infomation for Table 1

For the results in Table 1, Table 8 presents a comparative improvement over the best baseline. We provide the inference time of one generation in Table 8 for results in Table 1. TreeG-SD, SCG, and SVDD use ground-truth objective functions for evaluation, resulting in relatively long inference times for Chord Progression but achieving higher optimization performance. In contrast, TDS relies on gradient-based guidance through a surrogate neural network predictor, which reduces inference time at the cost of optimization effectiveness.

Table 8: Improvement of TreeG-SD in Music Generation.

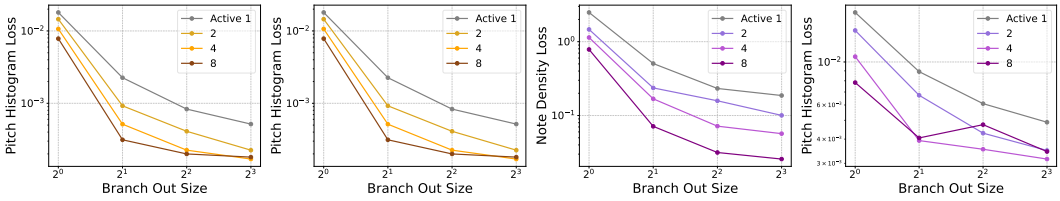
Task	Best Baseline	TreeG-SD	Loss Reduction
PH	0.0010 ± 0.0020 (DPS)	0.0002 ± 0.0003	80%
ND	0.134 ± 0.533 (SCG)	0.142 ± 0.423	-5.97%
CP	0.347 ± 0.212 (SCG)	0.301 ± 0.191	13.26%
Avg	—	—	29.01%

Table 9: Time (s) per sample for results in Table 8.

Task	TDS	SCG	SVDD	TreeG-SD
PH	306	194	194	203
ND	308	194	194	204
CP	305	7267	7267	6660

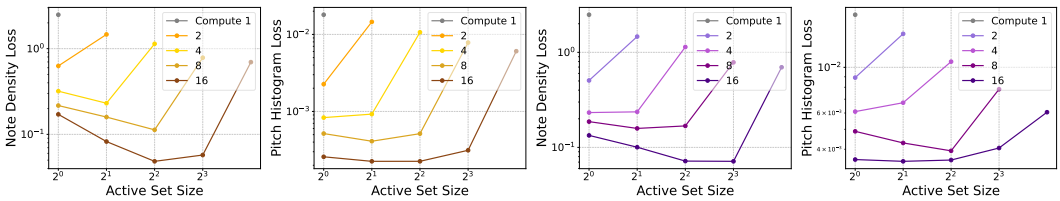
G.1.2 Scalability

We conduct experiments scaling the active set size A and branch-out size K for TreeG-SD and TreeG-SC. Increasing either A or K enhances the performance, as demonstrated in Figure 5. Figure 6 shows the trade-off between A and K with fixed $A * K$.



(a) TreeG-SD for ND (b) TreeG-SD for PH (c) TreeG-SC for ND (d) TreeG-SC for PH

Figure 5: Scaling Behavior with Fixed Active or Branch-Out Size on Music Generation.



(a) TreeG-SD for ND (b) TreeG-SD for PH (c) TreeG-SC for ND (d) TreeG-SC for PH

Figure 6: Trade-off between Active Set Size A and Branch-out Size K on Music Generation.

Scaling Effect on Performance while Maintaining Quality. While the product $A \times K$ gives an indication of total computation, different combinations of A and K can lead to varying computational costs, even when $A \times K$ is held constant, as shown in the computational complexity analysis in Table 4. We report the actual running time for generating one frontier from Figure 6 (a) and Figure 6 (b) in Table 10. The results demonstrate that the optimization exhibits a scaling effect with computation time, while still maintaining high-quality performance.

Table 10: Performance when Scaling Inference Time. Results are for the Optimal (A, K) with Fixed $A * K$ for TreeG-SD.

Note Density			Pitch Histogram		
Time(s)	Loss ↓	OA ↑	Time(s)	Loss ↓	OA ↑
16.7	2.486 ± 3.530	0.830 ± 0.016	16.7	0.0180 ± 0.0100	0.842 ± 0.012
43.1	0.629 ± 0.827	0.826 ± 0.060	42.2	0.0022 ± 0.0021	0.869 ± 0.009
71.8	0.231 ± 0.472	0.823 ± 0.045	65.3	0.0008 ± 0.0008	0.853 ± 0.013
130.7	0.113 ± 0.317	0.834 ± 0.023	118.6	0.0005 ± 0.0008	0.834 ± 0.018
251.9	0.048 ± 0.198	0.843 ± 0.012	209.6	0.0002 ± 0.0003	0.860 ± 0.016

G.2 Additional Results for Small Molecule Generation

G.2.1 Additional Results for Table 2.

As shown in Tables 2 and 11, SVDD achieves the best performance among the baselines for the targets QED, SA, DRD2, and Docking score while DG performs best in guiding toward the N_r target. On average, our method yields a 26.38% relative improvement over the strongest baselines. According to the metrics TS (Table 2) and NLL per token (Table 12), improved target optimization performance may come at the expense of diversity and quality.

Table 11: Relative performance improvement of TreeG-SC compared to DG, TFG-Flow, and SVDD. For N_r, we average MAE over all target values before comparing different methods. The relative improvements over the strongest baselines are bolded.

Method	QED ↑	SA ↑	DRD2 ↑	Docking ↑	N_r MAE ↓
DG	28.6%	12.3%	354%	—	**-21.6%**
TFG-Flow	32.0%	13.7%	803%	10.2%	−69.0%
SVDD	**19.5%**	**11.4%**	**77.7%**	**1.71%**	−75.5%

Table 12: Supplementary results for Table 2: NLL (↓) for small molecule generation.

Method	QED	SA	DRD2	Docking	$N_r^* = 1$
No Guidance	0.70 ± 0.39	0.70 ± 0.39	0.70 ± 0.39	0.68 ± 0.39	0.73 ± 0.41
DG	0.72 ± 0.42	0.73 ± 0.43	0.70 ± 0.40	—	0.86 ± 0.41
TFG-Flow	0.72 ± 0.41	0.74 ± 0.45	0.73 ± 0.46	$\mathbf{1.04 \pm 0.51}$	0.88 ± 0.39
SVDD	$\underline{0.71 \pm 0.39}$	0.70 ± 0.40	$\mathbf{0.67 \pm 0.38}$	1.21 ± 0.54	$\mathbf{0.81 \pm 0.37}$
TreeG-SC (A=1)	0.83 ± 0.46	$\mathbf{0.58 \pm 0.31}$	0.70 ± 0.39	1.13 ± 0.54	1.02 ± 0.40
TreeG-G (A=1)	$\mathbf{0.70 \pm 0.41}$	0.71 ± 0.42	$\underline{0.67 \pm 0.39}$	—	$\underline{0.84 \pm 0.40}$
TreeG-SD (A=1)	1.13 ± 0.51	0.83 ± 0.42	1.09 ± 0.52	1.14 ± 0.47	1.22 ± 0.40
TreeG-SD (A>1)	1.14 ± 0.50	$\underline{0.70 \pm 0.36}$	0.94 ± 0.47	$\underline{1.08 \pm 0.49}$	1.13 ± 0.37

Table 13: Supplementary results for Table 2: $N_r^* \in \{0, 1, 2, 3\}$ for small molecule generation.

Method	$N_r^* = 0$			$N_r^* = 1$			$N_r^* = 2$			$N_r^* = 3$		
	MAE ↓	TS ↓	NLL ↓	MAE ↓	TS ↓	NLL ↓	MAE ↓	TS ↓	NLL ↓	MAE ↓	TS ↓	NLL ↓
No Guidance	3.03 ± 1.26	0.12 ± 0.02	0.73 ± 0.41	2.09 ± 1.16	0.12 ± 0.02	0.73 ± 0.41	1.27 ± 1.02	0.12 ± 0.02	0.73 ± 0.41	0.92 ± 0.86	0.12 ± 0.02	0.73 ± 0.41
DG	0.16 ± 0.44	0.16 ± 0.03	0.83 ± 0.39	0.11 ± 0.33	0.14 ± 0.03	0.86 ± 0.41	0.07 ± 0.27	0.13 ± 0.02	0.80 ± 0.42	0.06 ± 0.25	0.12 ± 0.02	0.78 ± 0.43
TFG-Flow	0.30 ± 0.74	0.16 ± 0.03	0.84 ± 0.34	0.28 ± 0.65	0.13 ± 0.02	0.88 ± 0.39	0.20 ± 0.51	0.12 ± 0.02	0.82 ± 0.42	0.21 ± 0.46	0.12 ± 0.02	0.80 ± 0.44
SVDD	1.28 ± 1.92	$\underline{0.14 \pm 0.05}$	$\underline{0.70 \pm 0.38}$	0.35 ± 1.14	0.14 ± 0.03	$\mathbf{0.81 \pm 0.37}$	0.04 ± 0.37	0.14 ± 0.02	$\mathbf{0.73 \pm 0.36}$	$\mathbf{0.01 \pm 0.14}$	0.13 ± 0.02	$\mathbf{0.68 \pm 0.37}$
TreeG-SC (A=1)	$\underline{0.03 \pm 0.41}$	0.20 ± 0.04	0.86 ± 0.34	$\mathbf{0.01 \pm 0.07}$	0.14 ± 0.02	1.02 ± 0.40	$\mathbf{0.02 \pm 0.13}$	0.13 ± 0.02	0.88 ± 0.42	$\underline{0.02 \pm 0.18}$	0.13 ± 0.02	0.85 ± 0.45
TreeG-G (A=1)	1.45 ± 1.95	$\mathbf{0.13 \pm 0.03}$	$\mathbf{0.70 \pm 0.36}$	0.44 ± 1.19	0.13 ± 0.02	$\underline{0.84 \pm 0.40}$	0.09 ± 0.55	$\underline{0.12 \pm 0.02}$	$\underline{0.79 \pm 0.42}$	0.04 ± 0.30	$\underline{0.12 \pm 0.02}$	$\underline{0.75 \pm 0.42}$
TreeG-SD (A=1)	0.15 ± 0.47	0.17 ± 0.04	1.13 ± 0.41	0.11 ± 0.37	$\underline{0.12 \pm 0.02}$	1.22 ± 0.40	0.10 ± 0.33	0.12 ± 0.02	1.15 ± 0.45	0.15 ± 0.40	0.12 ± 0.02	1.14 ± 0.49
TreeG-SD (A>1)	$\mathbf{0.001 \pm 0.033}$	0.21 ± 0.05	0.95 ± 0.40	$\underline{0.02 \pm 0.12}$	$\mathbf{0.12 \pm 0.02}$	1.13 ± 0.37	$\underline{0.03 \pm 0.18}$	$\mathbf{0.12 \pm 0.02}$	0.99 ± 0.43	0.04 ± 0.22	$\mathbf{0.12 \pm 0.02}$	0.96 ± 0.48

G.2.2 Scalability

We demonstrate the scaling laws for TreeG-SC and TreeG-SD in Figures 7 and 8. Increasing computation time could boost guidance performance, i.e., lower mean absolute errors or higher property values. The computation of the Docking objective function with the oracle tool dominates

Table 14: Supplementary results for Table 2: $N_r^* \in \{4, 5, 6\}$ for small molecule generation.

Method	$N_r^* = 4$ MAE ↓	$N_r^* = 4$ TS ↓	$N_r^* = 4$ NLL ↓	$N_r^* = 5$ MAE ↓	$N_r^* = 5$ TS ↓	$N_r^* = 5$ NLL ↓	$N_r^* = 6$ MAE ↓	$N_r^* = 6$ TS ↓	$N_r^* = 6$ NLL ↓
No Guidance	1.27 ± 0.95	0.12 ± 0.02	0.73 ± 0.41	2.03 ± 1.16	0.12 ± 0.02	0.73 ± 0.41	2.98 ± 1.23	0.12 ± 0.02	0.73 ± 0.41
DG	0.08 ± 0.27	0.12 ± 0.02	0.79 ± 0.49	0.19 ± 0.40	0.13 ± 0.02	0.74 ± 0.52	$\mathbf{0.42 \pm 0.51}$	0.14 ± 0.02	$\mathbf{0.72 \pm 0.57}$
TFG-Flow	0.30 ± 0.53	0.12 ± 0.02	0.78 ± 0.49	0.51 ± 0.64	0.12 ± 0.02	0.78 ± 0.53	0.94 ± 0.86	0.13 ± 0.02	0.76 ± 0.57
SVDD	0.06 ± 0.36	0.13 ± 0.02	$\mathbf{0.70 \pm 0.42}$	0.28 ± 0.97	0.13 ± 0.02	$\mathbf{0.73 \pm 0.52}$	1.42 ± 1.97	0.12 ± 0.02	0.73 ± 0.50
TreeG-SC (A=1)	$\mathbf{0.05 \pm 0.29}$	0.13 ± 0.02	0.92 ± 0.50	$\mathbf{0.13 \pm 0.53}$	0.12 ± 0.02	1.04 ± 0.62	0.60 ± 1.47	0.12 ± 0.03	1.06 ± 0.70
TreeG-G (A=1)	0.08 ± 0.47	0.12 ± 0.02	0.76 ± 0.46	0.26 ± 0.90	0.12 ± 0.02	0.74 ± 0.49	1.25 ± 1.93	0.12 ± 0.02	0.77 ± 0.53
TreeG-SD (A=1)	0.30 ± 0.58	$\mathbf{0.11 \pm 0.02}$	1.26 ± 0.57	0.67 ± 0.84	0.11 ± 0.02	1.39 ± 0.61	1.41 ± 1.17	$\mathbf{0.11 \pm 0.02}$	1.36 ± 0.67
TreeG-SD (A>1)	0.10 ± 0.35	0.12 ± 0.02	1.12 ± 0.56	0.24 ± 0.60	$\mathbf{0.11 \pm 0.02}$	1.30 ± 0.65	0.50 ± 0.94	0.12 ± 0.02	1.48 ± 0.72

the overall runtime (Table 7). Thus, maximizing Docking score may require substantially increased inference compute (Figure 8), highlighting the inherent trade-off between effectiveness and efficiency.

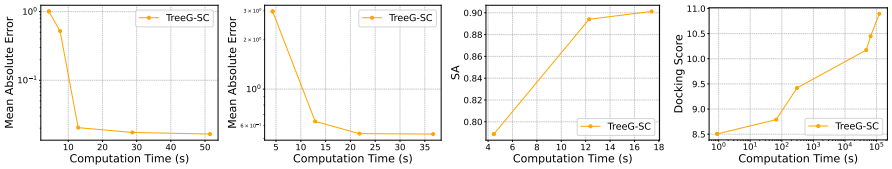


Figure 7: Small Molecule Generation: Scaling Law of TreeG-SC. From left to right, the targets are $N_r^* = 3$, $N_r^* = 6$, SA, and Docking score. The time-axis for Docking score is in logarithmic scale.

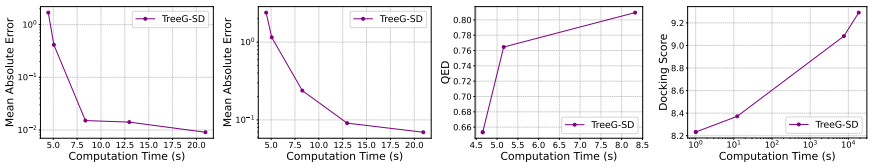


Figure 8: Small Molecule Generation: Scaling Law of TreeG-SD. From left to right, the targets are $N_r^* = 1$, $N_r^* = 5$, QED, and Docking score. The time-axis for Docking score is in logarithmic scale.

G.3 Additional Results for Enhancer DNA Design

G.3.1 Full Results of Table 3

Additional results of guidance methods for Class 4-8 are shown in Table 15 and Table 16. For both DG and our TreeG-G, we experiment with guidance values $\gamma \in [1, 2, 5, 10, 20, 50, 100, 200]$ and compare the highest average conditional probability across the eight classes. On average, TreeG-G outperforms DG by 18.43%.

Table 15: Supplementary results (Part 1): Classes 4–6 for enhancer DNA design.

Method (strength γ)		Class 4 Prob ↑	Class 4 NLL ↓	Class 4 Div ↑	Class 5 Prob ↑	Class 5 NLL ↓	Class 5 Div ↑	Class 6 Prob ↑	Class 6 NLL ↓	Class 6 Div ↑
No Guidance	—	0.007 ± 0.059	1.311 ± 0.038	373	0.035 ± 0.112	1.311 ± 0.038	373	0.037 ± 0.115	1.311 ± 0.038	373
DG	20	0.669 ± 0.377	1.256 ± 0.035	373	0.665 ± 0.332	1.244 ± 0.038	374	0.595 ± 0.334	1.251 ± 0.044	373
	100	0.585 ± 0.380	1.255 ± 0.043	365	0.466 ± 0.376	1.266 ± 0.050	373	0.385 ± 0.334	1.266 ± 0.057	364
	200	0.404 ± 0.384	1.280 ± 0.045	369	0.199 ± 0.284	1.279 ± 0.053	373	0.164 ± 0.235	1.281 ± 0.058	366
TFG-Flow	200	0.015 ± 0.083	1.389 ± 0.012	375	0.008 ± 0.048	1.386 ± 0.012	375	0.033 ± 0.104	1.394 ± 0.012	375
SVDD	—	0.107 ± 0.264	1.311 ± 0.035	374	0.142 ± 0.240	1.283 ± 0.039	374	0.124 ± 0.230	1.292 ± 0.042	373
TreeG-G	20	0.518 ± 0.391	1.151 ± 0.088	365	0.290 ± 0.306	1.078 ± 1.109	372	0.250 ± 0.292	1.052 ± 0.087	366
	100	0.845 ± 0.264	1.100 ± 0.060	365	0.778 ± 0.279	0.874 ± 0.162	359	0.843 ± 0.232	0.890 ± 0.093	368
	200	0.826 ± 0.297	1.099 ± 0.074	370	0.543 ± 0.426	1.015 ± 0.189	374	0.364 ± 0.432	1.104 ± 0.180	374

G.3.2 Scalability

$A * K$ as a Computation Reference. We use $A * K$ as the reference metric for inference time computation in TreeG-SC and TreeG-G, both employing `BranchOut`-Current. The corresponding inference times are shown in Figure 9, measured for a batch size of 100. Combinations of (A, K) that yield the same $A * K$ value exhibit similar inference times. We exclude the case where $K = 1$, as it does not require evaluation and selection, leading to a shorter inference time in practical implementation.

Table 16: Supplementary results (Part 2): Classes 7–8 for enhancer DNA design.

Method (strength γ)		Class 7 Prob ↑	Class 7 NLL ↓	Class 7 Div ↑	Class 8 Prob ↑	Class 8 NLL ↓	Class 8 Div ↑
No Guidance	—	0.010 ± 0.065	1.311 ± 0.038	373	0.013 ± 0.073	1.311 ± 0.038	373
DG	20	0.693 ± 0.346	1.231 ± 0.034	366	0.609 ± 0.350	1.224 ± 0.039	372
	100	0.475 ± 0.398	1.199 ± 0.061	364	0.600 ± 0.342	1.191 ± 0.057	372
	200	0.208 ± 0.313	1.196 ± 0.070	360	0.453 ± 0.370	1.208 ± 0.057	371
TFG-Flow	200	0.001 ± 0.015	1.376 ± 0.015	375	0.006 ± 0.055	1.377 ± 0.014	375
SVDD	—	0.039 ± 0.136	1.294 ± 0.042	369	0.038 ± 0.122	1.296 ± 0.037	373
TreeG-G	20	0.536 ± 0.335	0.720 ± 0.155	307	0.159 ± 0.233	1.011 ± 0.136	366
	100	0.740 ± 0.365	0.723 ± 0.230	343	0.413 ± 0.412	1.104 ± 0.126	373
	200	0.423 ± 0.457	0.936 ± 0.294	367	0.073 ± 0.205	1.204 ± 0.076	373

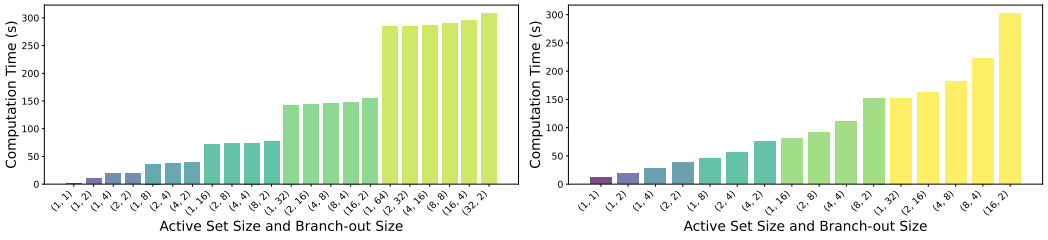


Figure 9: Inference Time for (A, K) Combinations (Left: TreeG-SC; Right: TreeG-G) in DNA Enhancer Design. Combinations with the same product $A \times K$ show similar inference times.

We provide the scaling law of TreeG-G at different guidance strengths in Figure 10. We also provide the scaling law for TreeG-SC in Figure 11.

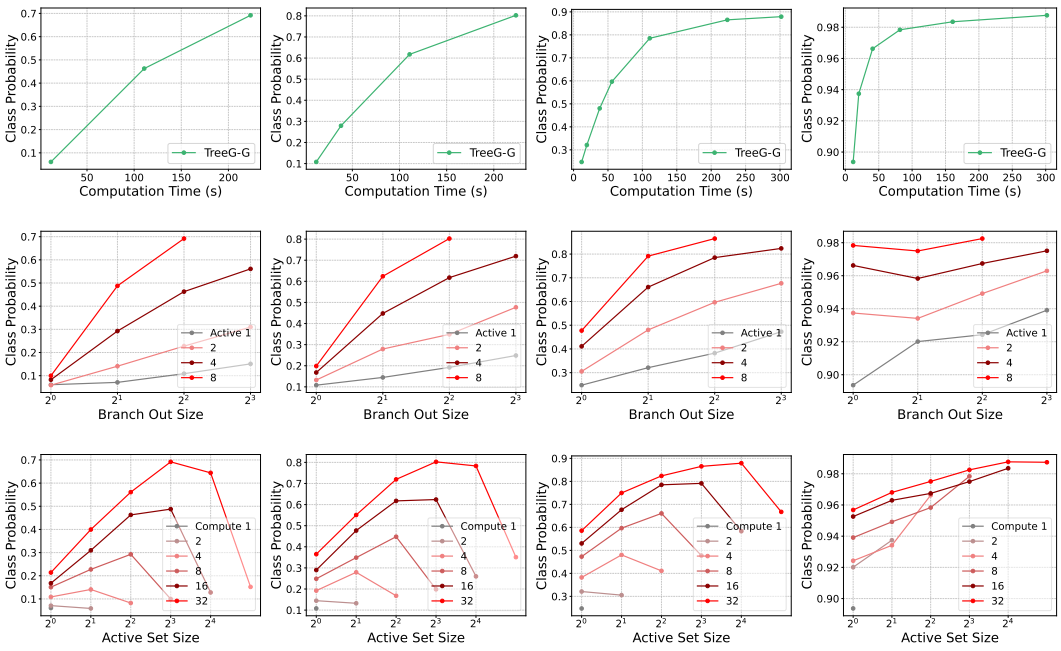


Figure 10: TreeG-G for Enhancer DNA Design. Top row: Effect of scaling on inference time. Middle row: Impact of increasing the active set size A or branch-out size K. Bottom row: Trade-off between A and K with fixed compute $A * K$. Columns from left to right correspond to different guidance strengths: $\gamma = 5,\ 10,\ 20,\ 200$ (results shown for Class 3).

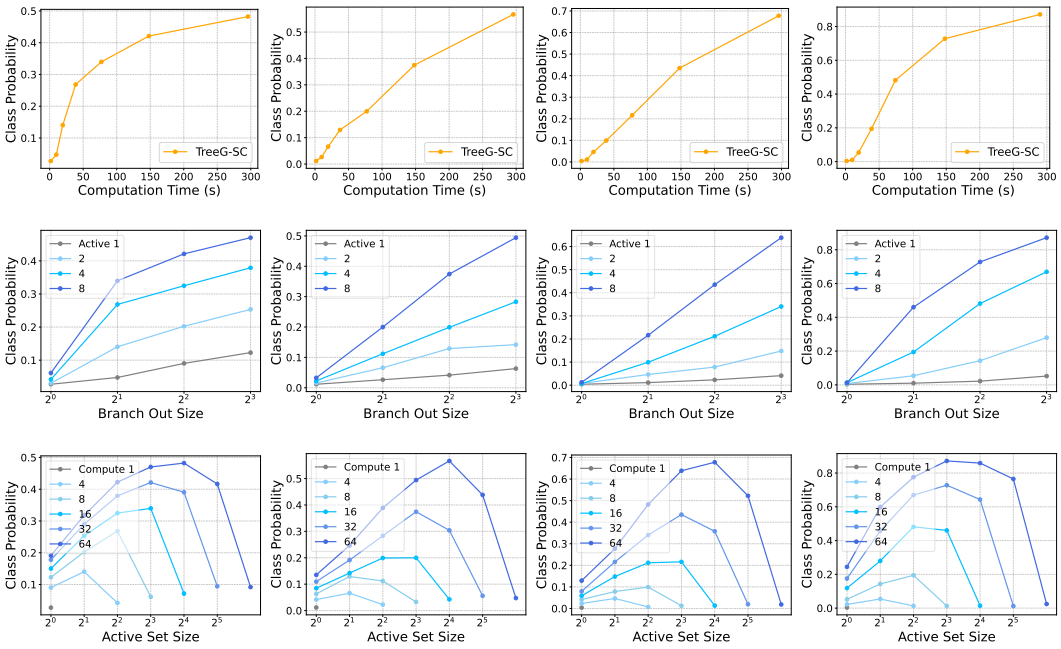


Figure 11: TreeG-SC for Enhancer DNA Design. Top row: Effect of scaling on inference time. Middle row: Impact of increasing the active set size A or branch-out size K. Bottom row: Trade-off between A and K with fixed compute $A * K$. Columns from left to right correspond to Class 1 to 4.

H Ablation Studies

H.1 Symbolic Music Generation

For TreeG-SD, we ablation on N_{iter} (detailed in Algorithm 7). As shown in Table 17, the loss decreases when N_{iter} increases. However, increasing N_{iter} also leads to higher computation costs. Here's a trade-off between controllability and computation cost.

Table 17: Ablation on N_{iter} of TreeG-SD.

N_{iter}	Loss ↓ (PH)	OA ↑ (PH)	Loss ↓ (ND)	OA ↑ (ND)
1	0.0031 ± 0.0037	0.733 ± 0.024	0.207 ± 0.418	0.810 ± 0.049
2	0.0005 ± 0.0008	0.833 ± 0.018	0.217 ± 0.450	0.819 ± 0.050
4	0.0005 ± 0.0006	0.786 ± 0.015	0.139 ± 0.319	0.830 ± 0.029

H.2 Discrete Models

Ablation on Taylor-expansion Approximation. As shown in Table 18, using Taylor-expansion to approximate the ratio (Eq. 10) achieve comparable model performance while dramatically improve the sampling efficiency compared to calculating the ratio by definition, i.e. TreeG-G-Exact.

Ablation on Monte Carlo Sample Size N. As shown in Table 19, increasing the Monte Carlo sample size improves performance, but further increases in N beyond a certain point do not lead to additional gains.

Table 18: Ablation on Taylor-expansion Approximation. The performances are evaluated on 50 generated samples.

Method	$N_r^* = 1$			$N_r^* = 2$		
	MAE ↓	TS ↓	Time	MAE ↓	TS ↓	Time
TreeG-G	0.24 ± 0.76	0.13 ± 0.02	2.4min	0.02 ± 0.14	0.12 ± 0.02	3.5min
TreeG-G-Exact	0.00 ± 0.00	0.14 ± 0.03	356.9min	0.02 ± 0.14	0.13 ± 0.02	345.2min

Table 19: Ablation on Monte Carlo Sample Size N. The performances are evaluated on 200 generated samples.

N	$N_r^* = 2$		$N_r^* = 5$	
	MAE ↓	TS ↓	MAE ↓	TS ↓
1	0.47 ± 1.12	0.12 ± 0.02	0.65 ± 1.37	0.12 ± 0.02
5	0.30 ± 0.97	0.12 ± 0.02	0.41 ± 1.14	0.12 ± 0.02
10	0.17 ± 0.68	0.12 ± 0.02	$\mathbf{0.20 \pm 0.76}$	0.12 ± 0.02
20	$\mathbf{0.09 \pm 0.46}$	0.12 ± 0.02	0.26 ± 0.91	0.12 ± 0.02
40	0.10 ± 0.60	0.12 ± 0.02	0.29 ± 1.01	0.13 ± 0.02

Table 20: Ablation on Selection Choices in TreeG. TreeG-SC and TreeG-SD are evaluated on 500 and 100 generated samples respectively. t means temperature used for resampling weights. lower t leads resampling closer to ranking.

Choices	TreeG-SC ($K = 4$)		TreeG-SD ($K = 16$)	
	QED ↑	TS ↓	QED ↑	TS ↓
Ranking	0.79 ± 0.12	0.12 ± 0.02	0.76 ± 0.12	0.12 ± 0.02
Sampling ($t = 0.1$)	0.71 ± 0.17	0.13 ± 0.02	0.67 ± 0.19	0.12 ± 0.02
Sampling ($t = 0.5$)	0.64 ± 0.19	0.12 ± 0.02	0.67 ± 0.17	0.11 ± 0.02

H.3 Selection Choices in Alg.1

We compare two selection choices, i.e., ranking and resampling, in Alg.1 on small molecule generation with QED as the target. For both TreeG-SC and TreeG-SD, selection by *ranking* produces better guidance performance compared to selection by *resampling* (Tab. 20).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The introduction (Section 1) includes a paragraph "Contributions", together with the abstract, clearly reflecting the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix B.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All the theoretical results Theorem 1, Lemma 1 and Theorem 2 provide the full set of assumptions, and the correspond proofs are in Appendix D.1, Appendix D.2 and Appendix D.3, respectively

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experiments are described in Section 6, and the details are provided in Appendix E, Appendix F and Appendix G.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code available at https://github.com/yukang123/UniTreeG.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experimental details are provided in Appendix E, Appendix F and Appendix G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The experiments that support the main claims of the paper provide error bars, as shown in tables, such as Table 1, Table 2 and Table 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information on the computer resources is included in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research adheres to all guidelines and principles outlined in the NeurIPS Code of Ethics, including considerations for fairness, accountability, and transparency. Ethical practices were maintained throughout the study, ensuring compliance with all applicable standards.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the social impacts of this paper in Appendix B.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators and original owners of assets used in this paper are properly credited. The licenses and terms of use are explicitly mentioned and respected according to their guidelines.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release our code and models along with detailed documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core methodological development in this research does not involve LLMs as significant, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.