REWARD-GUIDED DISCRETE DIFFUSION VIA CLEAN-SAMPLE MARKOV CHAIN

Prin Phunyaphibarn & Minhyuk Sung
KAIST
{prin10517,mhsung}@kaist.ac.kr

ABSTRACT

Discrete diffusion models have recently emerged as a powerful class of generative models for chemistry and biology data. In these fields, the goal is to generate various samples with high rewards (e.g., drug-likeness in molecules), making reward-based guidance crucial. Most existing methods are based on guiding the diffusion model using intermediate rewards but tend to underperform since intermediate rewards are noisy due to the non-smooth nature of reward functions used in scientific domains. To address this, we propose Clean-Sample Markov Chain (CSMC) Sampler, a method that performs effective test-time reward-guided sampling for discrete diffusion models, enabling local search without relying on intermediate rewards. CSMC constructs a Markov chain of clean samples using the Metropolis-Hastings algorithm such that its stationary distribution is the target distribution. We design a proposal distribution by sequentially applying the forward and backward diffusion processes, making the acceptance probability tractable. Experiments on molecule and biological sequence generation with various reward functions demonstrate that our method consistently outperforms prior approaches that rely on intermediate rewards.

1 INTRODUCTION

Discrete diffusion models have recently emerged as a powerful generative framework for discrete data, showing particular promise in chemistry and biology for generating complex structures such as molecules and DNA sequences. Unlike autoregressive models that assume a canonical left-to-right ordering on the data, discrete diffusion models are more naturally suited for scientific domains whose data (e.g., molecules or DNA sequences) lack a natural fixed ordering. For instance, the widely used SMILES (Weininger, 1988) representation for molecules is based on heuristic rules such as depth-first search which does not use a unified fixed ordering (Lee et al., 2025).

In many applications in chemistry and biology, there are well-defined notions of *quality* for the data; for example, drug-likeness (Bickerton et al., 2012) for molecular structures or enhancer activity (Taskiran et al., 2024; Wang et al., 2025) for DNA sequences. Thus, generative models must not only produce *natural* samples that resemble the training data but also achieve high-quality scores according to such domain-specific criteria.

In the emerging regime of test-time scaling, quality considerations are incorporated by defining a reward function and optimizing it during inference through reward-guided sampling. The simplest strategy is Best-of-N sampling, where N samples are generated, and the one with the highest reward is selected. However, this brute-force method is inefficient since it does not perform any structured guidance. Recent works (Kim et al., 2025b; Wu et al., 2023; Yu et al., 2023; Bansal et al., 2023; Chung et al., 2023) have instead proposed guidance using *process rewards* or *intermediate rewards*, which are computable at intermediate steps of generation.

While methods involving intermediate rewards are technically applicable to discrete diffusion models, they pose particular challenges for many types of chemistry and biology data as reward functions in these domains are often non-smooth, meaning that small perturbations in the data can cause large changes in the reward. For example, in molecular structures, modifying even a single element in the string representation can render the entire molecule invalid, collapsing the reward to zero, as shown in Fig. 1 (left). As a result, relying on intermediate rewards does not provide an effective local search strategy in these cases.

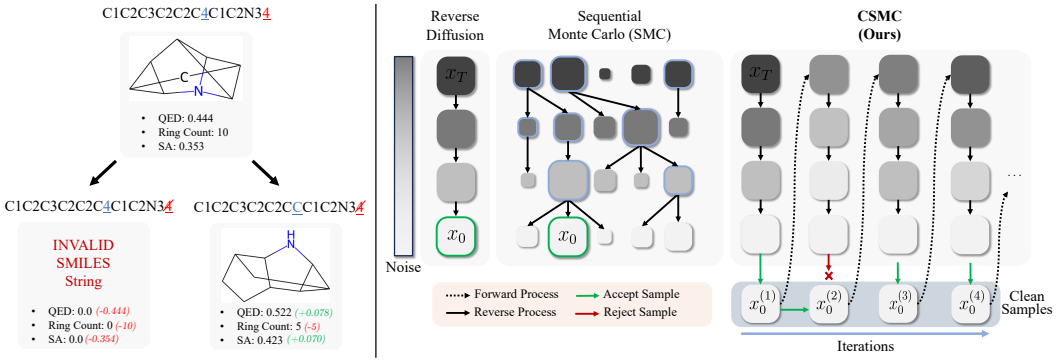


Figure 1: **Left:** In scientific applications, the rewards defined on discrete spaces are highly sensitive to small perturbations. A one or two-character change to a SMILE string can result in drastically different rewards (QED, ring count, SA). **Right:** Reverse diffusion and typical inference-time scaling methods (Kim et al., 2025b; Li et al., 2025) such as SMC (Kim et al., 2025b) guide samples through noise levels by constructing a Markov chain beginning with pure noise and ending with clean samples. Our CSMC constructs a Markov chain clean samples by successively applying the forward and reverse processes sequentially at each step, bypassing the need for intermediate rewards by evaluating the reward directly on clean samples while leveraging information from past samples for guidance.

The key question that arises here is how to enable local search in reward-guided generation without relying on intermediate rewards. We introduce **Clean-Sample Markov Chain (CSMC) Sampler** which performs iterative search over clean data samples using the Metropolis–Hastings (MH) algorithm. To propose a new clean sample from an existing one, we use forward–backward combinations—applying the forward process to corrupt clean data followed by running the reverse process to obtain a new sample. While the acceptance probability for the MH algorithm is intractable due to intractable clean sample probabilities, we show that the forward-backward proposal distribution makes the acceptance probability tractable, enabling efficient sampling.

We validate CSMC on molecule and biological sequence generation across four different reward functions. CSMC achieves the highest reward in all settings, even for SMILES string generation where other methods fail due to inaccurate intermediate rewards.

2 Related Work

In continuous diffusion models, reward-guided sampling is conventionally performed using gradient-based methods (Dhariwal & Nichol, 2021; Ho & Salimans, 2022; Chung et al., 2023; Song et al., 2023; Rozet et al., 2024; Bansal et al., 2023; Yoon et al., 2025; Kim et al., 2025b; Wu et al., 2023) which offer strong guidance towards high-reward regions. However, gradient-based approaches cannot be applied to discrete diffusion models as gradients are ill-defined in discrete spaces and it is not theoretically valid to add a continuous gradient to a discrete objective.

Training-Free Reward-Guided Sampling for Discrete Diffusion. Recently, inference-time scaling methods for discrete diffusion models have been proposed to tackle reward-guided sampling. A comparison of these methods and our method is shown in Tab. 1.

The simplest method is Best-of-N (BoN) sampling (Stiennon et al., 2020), which generates N samples independently and selects the one with the highest reward. Due to its simplicity, BoN is applicable to all types of discrete diffusion models. However, BoN does not guide the denoising trajectory using the reward, resulting in inefficient search, especially when high-reward samples lie in low-density regions unlikely to be sampled by the model.

On the other hand, particle-based methods (Ma et al., 2025; Kim et al., 2025a; Singhal et al., 2025; Li et al., 2025), such as SMC (Doucet et al., 2001; Naesseth et al., 2019; Moral, 2004) and SVDD (Li et al., 2025), can be applied to discrete diffusion and incorporate reward signals *during* the denoising process through a reward-based resampling step. SMC (Doucet et al., 2001; Naesseth et al., 2019; Moral, 2004), takes N particles (samples) at each step and samples a subset of the particles to keep while throwing the rest away. By adjusting the probability of keeping each particle based

	Uniform	Masked	Clean Reward	Trajectory Guidance
BoN	✓	✓	✓	✗
SMC	✓	✓	✗	✓
SVDD	✓	✓	✗	✓
SGDD	✓	✗	✗	✓
CSMC	✓	✓	✓	✓

Table 1: Comparison of discrete diffusion inference-time scaling methods for reward-guided sampling. Our CSMC applies to all discrete diffusion frameworks and leverages the clean reward while guiding the sampling trajectory.

on its expected reward, SMC encourages local exploration around potentially promising samples. SVDD (Li et al., 2025) exploits local search around high-reward samples by generating multiple candidate samples at each timestep and retaining only one sample by sampling the candidates with probability proportional to the reward. A key challenge with these methods is that they require computing **intermediate rewards** on noisy samples rather than on fully denoised samples. To address this, these approaches exploit the diffusion model's ability to predict an approximation of the clean sample x_0 given a noisy sample x_t, and compute the rewards on this x_0 prediction instead. This enables reward-guidance *during* the denoising process, but also inherently assumes that the predicted x_0 is a good approximation of the true clean sample since exploration will be performed locally around samples with high intermediate rewards, an assumption which does not hold in many scientific applications.

Recently, Chu et al. (2025) proposed SGDD which uses the split Gibbs sampler (Vono et al., 2019) by alternating between denoising steps and running MCMC to optimize intermediate noisy samples. However, SGDD applies *only* to uniform diffusion models where intermediate noisy samples can be treated as clean samples. The characteristics of uniform discrete diffusion is discussed in more detail in Sec. 4. Furthermore, SGDD still relies on computing the reward on the intermediate noisy samples during the MCMC optimization step.

In this work, we bypass the need for intermediate rewards by using the Metropolis-Hastings algorithm (Robert & Casella, 2009) to construct a Markov chain of *clean* samples that converges to the desired target distribution. Our method is applicable to both uniform and masked discrete diffusion models, and only requires the computation of **clean rewards** on clean samples, which can then be used to efficiently guide the Markov chain (Tab. 1).

Training-Based Reward-Guided Sampling for Discrete Diffusion. Unlike continuous diffusion, discrete diffusion models cannot leverage gradient signals due to the non-differentiable nature of discrete spaces. As such, only gradient-free approaches can be applied to discrete diffusion models. Guidance methods (Nisonoff et al., 2025; Schiff et al., 2025) have also been developed for diffusion models but require training a classifier on noisy data. Due to its non-differentiability, RLFT approaches are often used in discrete diffusion (Rector-Brooks et al., 2025; Wang et al., 2025; Zekri & Boullé, 2025) but are difficult to train since non-differentiability forces fine-tuning to be done using policy gradients based on highly complex reward landscapes (Uehara et al., 2025). Crucially, modifying the model weights also runs the risk of deviating from the pretrained prior, affecting the naturalness of the generated samples and further complicating training.

3 Background

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) learn to reverse a forward Markov process. Although originally designed for continuous spaces, diffusion models have also been successfully applied to discrete state spaces (Austin et al., 2021; Campbell et al., 2022; Lou et al., 2024; Sahoo et al., 2024; Schiff et al., 2025; Shi et al., 2024).

3.1 Discrete and Continuous-Time Discrete Diffusion

Discrete-Time Discrete Diffusion. These discrete diffusion models are characterized by forward transition matrices $\boldsymbol{Q}_t$. Let $\overline{\boldsymbol{Q}}_t = \boldsymbol{Q}_1 \boldsymbol{Q}_2 \cdots \boldsymbol{Q}_t$. These transition matrices define the forward process:

$$p_{t|t-1}(x_t|x_{t-1}) = \text{Cat}\left(x_t; p = x_{t-1}\boldsymbol{Q}_t\right),$$
$$p_t(x_t|x_0) = \text{Cat}\left(x_t; p = x_0\overline{\boldsymbol{Q}}_t\right),$$

where $\text{Cat}(\cdot; p)$ denotes the categorical distribution with probabilities given by p.

While the reverse process probability $p(x_{t-1}|x_t)$ is not directly tractable, by additionally conditioning on x_0, the reverse process can be derived in closed form as

$$p(x_{t-1}|x_t, x_0) = \text{Cat}\left(x_{t-1}; p = \frac{x_t \boldsymbol{Q}_t^\top \odot x_0 \overline{\boldsymbol{Q}}_{t-1}}{x_0 \overline{\boldsymbol{Q}}_t x_t^\top}\right).$$

A neural network $p_\theta(x_0|x_t) \approx p(x_0|x_t)$ is learned and denoising is performed by the following parameterization:

$$p_\theta(x_{t-1}|x_t) \propto \sum_{x_0} q(x_{t-1}, x_t|x_0) p_\theta(x_0|x_t).$$

Notably, the learned neural network $p_\theta(x_0|x_t)$ predicts a distribution over x_0 given x_t which enables sampling x_0-predictions $\hat{x}_0(x_t) \sim p_\theta(x_0|x_t)$.

Continuous-Time Discrete Diffusion. Campbell et al. (2022) proposed a continuous-time framework based on Continuous-Time Markov Chains (CTMC) where state transitions can occur at any time. Instead of defining transition matrices, CTMC-based discrete diffusion defines a forward transition *rate* matrix $\boldsymbol{R}_t$ which defines the infinitesimal transition probability between two timesteps:

$$p(x_t|\tilde{x}_{t-\Delta t}) = \delta_{x_t, \tilde{x}_{t-\Delta t}} + \boldsymbol{R}_t(\tilde{x}_{t-\Delta t}, x_t)\Delta t + o(\Delta t).$$

By using a continuous-time framework, advanced sampling strategies such as predictor-corrector methods (Campbell et al., 2022; Zhao et al., 2025) and planning (Liu et al., 2025; Peng et al., 2025) and score-based approaches (Meng et al., 2022; Sun et al., 2023; Lou et al., 2024) have been proposed.

3.2 Masked and Uniform Discrete Diffusion

Discrete diffusion models allow the user to choose the transition matrix. The two most common choices of transition matrices result in masked diffusion models (MDMs) (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) and uniform state models (USMs) Austin et al. (2021); Schiff et al. (2025).

Masked Diffusion Models (MDMs). MDMs define the forward process by progressively replacing tokens with a special mask token. The denoising model learns to recover the original sequence from these masked inputs. Notably, samples at intermediate time steps are not valid samples because of the mask tokens, which are not present in the dataset.

Uniform State Models (USMs). USMs replace each token with a randomly chosen token from the vocabulary as noise increases. This creates a uniform corruption process in which every other possible token substitution is equally likely. Unlike MDMs, samples at intermediate time steps may constitute valid samples.

4 Clean-Sample Markov Chain Sampler

Let the reward function be denoted by $r(\cdot) : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ is the domain, and $\Delta(\mathcal{X})$ denote the set of all probability distributions defined on $\mathcal{X}$. Given a reward function $r(x)$, our objective is to generate samples with high rewards while maintaining naturalness by leveraging a pretrained generative model $p^{\text{pre}}(\cdot)$. To accomplish this, we sample from the reward-weighted distribution $p_\beta(x)$:

$$\begin{aligned} p_\beta(x) &:= \arg\max_{p \in \Delta(\mathcal{X})} \mathbb{E}_{x \sim p(\cdot)}\left[r(x)\right] - \beta D_{\text{KL}}(p(\cdot) \| p^{\text{pre}}(\cdot)) \\ &\propto \exp(r(x)/\beta) p^{\text{pre}}(x), \end{aligned}$$

where β is a hyperparameter controlling the "naturalness" of the samples through KL-regularization with the pretrained model.

Most previous methods such as particle-based methods perform guidance by utilizing intermediate rewards $r(\hat{x}_0(x_t))$ computed from the x_0-prediction $\hat{x}_0(x_t)$ at x_t as an approximation to the clean sample. However, intermediate rewards are often inaccurate and noisy in many scientific applications due to the non-smooth reward functions. In scientific applications, even a slight perturbation can significantly impact the reward. For instance, when generating molecules based on

the SMILES (Weininger, 1988) representation, modifying a single token on a high-reward molecule can result in an invalid molecule with zero reward or a molecule with very different properties, as shown in Fig. 1 (left). This is in stark contrast to rewards defined on continuous space where a slight perturbation smoothly affects the reward (small perturbations to some pixel values of an image leave the semantics of the image unchanged). Due to the sensitivity of the reward, the x_0 prediction must be perfectly accurate, otherwise the intermediate rewards will be uninformative. A sample whose intermediate reward is zero may be prematurely removed by a particle-based method even though changing a single token may yield a high-reward sample (Fig. 1, left).

In these cases, it is more beneficial to leverage only clean reward $r(x_0)$ computed on a clean sample x_0 rather than on an approximation $\hat{x}_0(x_t)$. One such method leveraging clean rewards is Best-of-N (BoN) sampling (Stiennon et al., 2020). However, BoN does not provide guidance during the denoising process, resulting in inefficient exploration. Its effectiveness is therefore limited to what the pretrained model can already generate: if high-reward samples lie in low-density regions of the model's distribution, they are unlikely to ever be produced, even when many samples are drawn. A natural question arises: **How can we leverage clean rewards while using information from past samples for guidance?**

Our proposed method, **C**lean-**S**ample **M**arkov **C**hain **(CSMC)** Sampler, answers this question by using the Metropolis-Hastings (MH) algorithm to construct a Markov chain of clean samples that converges to the reward-weighted distribution $p_\beta(x_0)$. By using only clean samples, CSMC leverages accurate clean rewards. By using the MH algorithm to iteratively refine the samples, the clean rewards of past samples can be used for guidance.

4.1 Bypassing the Intermediate Rewards

Previous methods require intermediate rewards because the denoising trajectory of diffusion models output clean samples only at the very end. A natural solution is to instead explore the clean data space by constructing a chain of *clean* samples $\{x_0^{(i)}\}_i$, which converges to the target distribution $p_\beta(x_0)$. At each iteration, rewards can then be computed directly on clean samples, yielding clean rewards $r(x_0)$ which can then be used to guide the chain towards the target distribution. To accomplish this, we propose to use the Metropolis-Hastings (MH) algorithm.

Suppose we want to sample from the (potentially unnormalized) target distribution $p_\beta(x_0)$. We first define a proposal distribution $q(x_0'|x_0)$ to generate the next candidate sample $x_0' \sim q(x_0'|x_0)$ given the current sample x_0. After generating the proposal candidate, we decide whether or not to accept this proposal with probability $A(x_0'|x_0)$ defined as

$$A(x_0'|x_0) = \min(1, \alpha), \quad \alpha = \frac{p_\beta(x_0')q(x_0|x_0')}{p_\beta(x_0)q(x_0'|x_0)}.$$

With a slight abuse of notation, we use $A(x_0'|x_0)$ interchangeably with A when the choice of x_0' and x_0 are not important. The MH algorithm can be shown to construct a Markov chain whose stationary distribution is the target distribution. For more details and proof of convergence on the MH algorithm, please refer to App. A. To generate samples from the target reward-weighted distribution $p_\beta(x_0) \propto \exp(r(x_0)/\beta)p(x_0)$, the acceptance probability is as follows:

$$A(x_0'|x_0) = \min(1, \alpha), \tag{1}$$

$$\alpha = \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right)\frac{p(x_0')q(x_0|x_0')}{p(x_0)q(x_0'|x_0)}. \tag{2}$$

The difficulty in applying the MH algorithm to diffusion models is the intractibility of the acceptance probability. Since $p(x_0)$ is intractable, the acceptance probability cannot be efficiently computed.

4.2 Forward-Backward Proposal Distribution

To bypass the computation of $p(x_0)$, we define the proposal distribution in such a way that the proposal probabilities $q(x_0|x_0')$ and $q(x_0'|x_0)$ cancel out $p(x_0)$ and $p(x_0')$. We define the proposal distribution $q(x_0'|x_0)$ using a forward-backward diffusion process as follows. Starting from the current clean sample x_0, we choose a random time $t \sim \mathcal{U}(t_{\text{lo}}, t_{\text{hi}})$ (where t_{lo} and t_{hi} are user-defined parameters) and apply the forward process to obtain a noisy auxiliary sample $x_t \sim p_t(\cdot|x_0)$. Then,

Algorithm 1 CSMC (Clean-Sample Markov Chain Sampler)

1: $x_0^{(1)} \sim p(x_T) \prod_{t=1}^{T} p(x_{t-1}|x_t)$
2: **for** $k = 1, \ldots, K$ **do**
3: $\quad t_0 \sim \mathcal{U}(t_{\text{lo}}, t_{\text{hi}})$
4: $\quad x_{t_0}^{(k)} \sim p_{t_0}(\cdot|x_0^{(k)})$
5: $\quad \tilde{x}_0^{(k+1)} \sim \prod_{i=1}^{M} p(x_{t_{i-1}}^{(k)}|x_{t_i}^{(k)})$
6: $\quad \alpha \leftarrow \exp((r(\tilde{x}_0^{(k+1)}) - r(x_0^{k}))/\beta)$
7: $\quad A \leftarrow \min(1, \alpha)$
8: $\quad \rho \sim \mathcal{U}(0, 1)$
9: $\quad x_0^{(k+1)} \leftarrow \tilde{x}_0^{(k+1)}$ if $\rho < A$ else $x_0^{(k)}$
10: **end for**
11:
12: $\{x_0^{(K/2)}, \ldots, x_0^{(K)}\}$

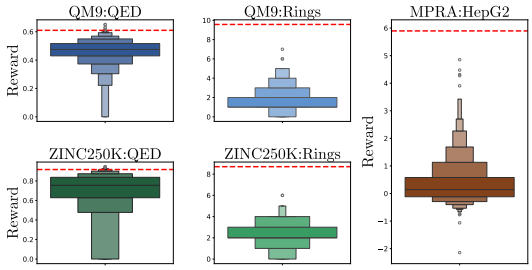


Figure 2: Reward distributions of the pretrained USM. The red dotted line represents the average reward achieved by CSMC. For ring count and HepG2, the pretrained model reward distribution has low density at higher rewards, resulting in degraded performance for BoN sampling.

we run the reverse process to obtain a new clean sample $x_0' \sim p(\cdot|x_t)$ This proposal distribution is defined by

$$q(x_0'|x_0) := p_t(x_t|x_0)p(x_0'|x_t), \tag{3}$$

where $t \sim \mathcal{U}(t_{\text{lo}}, t_{\text{hi}})$ and $x_t \sim p(\cdot|x_0)$ are resampled at each step.

However, we find that running a one-step reverse process results in an inaccurate x_0' in practice. Instead, we run an M-step reverse process on x_t to obtain $(x_{t_M}, x_{t_{M-1}}, \ldots, x_1, x_0') \sim \prod_{i=1}^{M} p(x_{t_{i-1}}|x_{t_i})$ where $t = t_M > \cdots > t_0 = 0$. We then discard the x_{t_i} and retain only x_0'. This proposal distribution can be interpreted as exploring locally around x_0.

Assuming that the marginal distributions of the learned reverse process matches that of the forward process, we have

$$\begin{aligned}
\alpha &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0')p(x_t|x_0')p(x_0|x_t)}{p(x_0)p(x_t|x_0)p(x_0'|x_t)} \\
&= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0')p(x_t|x_0')p(x_0|x_t, x_0')}{p(x_0)p(x_t|x_0)p(x_0'|x_t, x_0)} \\
&= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right).
\end{aligned}$$

Thus, the acceptance probability simplifies to

$$A(x_0'|x_0) = \min(1, \alpha), \quad \alpha = \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right), \tag{4}$$

which can now be efficiently computed. The assumption that the marginals of the reverse and forward process match is mild and is enforced by the training objective of discrete diffusion models (Austin et al., 2021). The detailed derivation of the acceptance probability can be found in App. B.

CSMC begins by running the full reverse process to obtain a clean initial sample $x_0^{(1)}$. At each step, we generate a candidate sample using our forward-backward diffusion process and choose to accept or reject the candidate based on the MH acceptance probability given in Eq. 4, as shown in Fig. 1 (right). This results in a chain of clean samples whose stationary distribution is the reward-weighted distribution. While MH is guaranteed to construct a Markov chain whose stationary distribution is $p_\beta(x_0)$, initial samples may not come from the stationary distribution. Thus, we discard the first half of the chain and take as many samples as desired from the latter half at equal intervals. The practice of "throwing away" the initial samples of a Markov chain is known as **burn-in** (Robert & Casella, 2009; Murphy, 2023). The full algorithm for CSMC is shown in Alg. 1.

5 Experiments

We conduct experiments on molecule generation with QM9 (Ramakrishnan et al., 2014) and ZINC250K (Irwin et al., 2012), and biological sequence design using the MPRA (Gosai et al.,

2023) dataset. We pretrain four discrete diffusion models on each of the dataset: **MDM (Sahoo et al., 2024)**, **USM (Schiff et al., 2025)**, **SEDD-M (Lou et al., 2024)**, **SEDD-U (Lou et al., 2024)**.

We compare the following training-free reward-guided sampling methods for discrete diffusion:

- **Pretrained models:** Samples are generated using the pretrained model.
- **Best-of-N (BoN):** N samples are generated and the one with the highest reward is selected.
- **SMC (Doucet et al., 2001):** The representative derivative-free particle-based sampling method which approximates the target distribution by updating and resampling a set of N particles at each step using the intermediate reward.
- **SVDD (Li et al., 2025):** N candidate samples are generated at each step and a sample is selected by randomly choosing a sample with probability proportional to its intermediate reward.
- **SGDD (Chu et al., 2025):** A posterior sampling method specifically tailored for uniform CTMC discrete diffusion models which uses split Gibbs sampling (Vono et al., 2019). Since this method is based on the forward process of uniform CTMC discrete diffusion models, we only evaluate SGDD on SEDD-U and not on the discrete-time model USM.
- **CSMC(Ours):** We run Alg. 1 to construct a Markov chain converging to $p_\beta(x_0)$ and draw multiple samples from the chain. **CSMC-B (Ours)** denotes CSMC with batching.

We match the total diffusion model NFE for each method. More experiment details can be found in App. C.

Molecule Generation. We test our method on two molecule datasets: QM9 (Ramakrishnan et al., 2014) and ZINC250K (Irwin et al., 2012). QM9 is a dataset consisting of ∼133,000 small organic molecules, and ZINC250K is a dataset of 250,000 commercially available compounds. The molecules in both datasets are represented as SMILES strings (Weininger, 1988). For our reward functions, we use **QED** (Bickerton et al., 2012), ring count **(Rings)**, and synthetic accessibility **(SA)** (Ertl & Schuffenhauer, 2009). QED measures the drug-likeness of a compound based on eight widely used molecular properties (number of hydrogen bond donors/acceptors, molecular polar surface area, number of aromatic rings, etc.). Higher QED values indicate higher drug-likeness. Ring count measures the number of rings in the symmetrized smallest set of smallest rings (SSSR). Finally, SA measures the ease of synthesis of drug-like molecules through a combination of known common structural features in known synthesized molecules, and a penalty based on complex structural features of the molecule. SA takes on a value bewteen 1 and 10 where higher values indicate that the molecule is harder to synthesize. In this work, SA is converted to a reward function by applying the renormalization $(10 - SA)/9$ so that higher values indicate better performance. For all rewards, higher is better.

Biological Sequence Generation. We pretrain the discrete diffusion models on the DNA dataset provided by Gosai et al. (2023) (which we refer to as MPRA) which measures the enhancer acitivity of ∼700,000 DNA sequences using massively parallel reporter assays (MPRA). An Enformer model (Avsec et al., 2021) is trained to predict the enhancer activity level in the HepG2 cell line which is used as the reward function. We use the model trained on two different subsets of data and use one exclusively to provide the guidance signal during sampling and the other for evaluation. Higher predicted HepG2 activity indicates better performance.

Results. The quantitative results of reward-guided generation are shown in Tab. 2. As shown, CSMC or CSMC-B achieves the best reward in all cases. We also compute the diversity in Tab. 5 in App. D.1, showing that the samples remain sufficienctly diverse despite achieving high reward.

SMC and SVDD require intermediate rewards which are often inaccurate. As such, these methods sometimes only yield slightly improved results over the pretrained model and, in some cases, slightly worse results due to inaccurate exploration. The performance of these methods is also highly reliant on both the smoothness of the reward function and the diffusion model's accuracy in x_0 prediction. CSMC, on the other hand, only use accurate clean rewards for guidance, resulting in accurate and efficient guidance.

While BoN performs well on QED and SA for QM9 and ZINC250K, its performance is not as good on the HepG2 activity reward for MPRA and the ring count reward for QM9 and ZINC250K. This is due to the lack of trajectory guidance: BoN performs well only for settings where high-reward samples are

		QM9			ZINC250K			MPRA
		QED	Rings	SA	QED	Rings	SA	HepG2
MDM	Pretrained	0.461±0.138	2.755±3.432	0.558±.0227	0.663±0.323	2.020±2.488	0.742±0.323	0.442±1.767
	BoN	0.580±0.073	5.479±1.286	0.818±0.129	0.854±0.121	3.854±1.923	0.862±0.121	1.842±2.093
	SMC	0.512±0.144	2.803±1.713	0.759±0.271	0.637±0.310	2.128±2.594	0.769±0.310	3.087±2.975
	SVDD	0.567±0.117	2.849±1.462	0.836±0.174	0.776±0.255	2.490±1.848	0.754±0.255	2.319±2.654
	CSMC	**0.610**±0.085	**10.00**±0.933	**0.913**±0.077	0.910±0.060	**8.091**±2.328	**0.905**±0.060	**5.259**±0.849
	CSMC-B	**0.610**±0.083	8.678±2.531	0.911±0.095	**0.914**±0.049	6.032±1.918	0.903±0.073	5.127±1.428
USM	Pretrained	0.461±0.152	2.032±1.341	0.620±0.223	0.739±0.270	2.598±1.883	0.768±0.189	0.351±1.541
	BoN	0.600±0.054	5.044±1.267	0.839±0.097	0.901±0.065	4.184±1.307	0.889±0.048	1.753±1.965
	SMC	0.485±0.170	1.957±1.212	0.660±0.203	0.750±0.270	2.633±1.828	0.785±0.184	0.743±2.048
	SVDD	0.454±0.157	2.135±1.460	0.624±0.217	0.750±0.260	2.562±1.861	0.773±0.195	0.695±1.920
	CSMC	**0.610**±0.085	**9.570**±0.791	0.908±0.063	**0.917**±0.050	**8.703**±3.454	**0.898**±0.060	**5.897**±1.499
	CSMC-B	0.609±0.081	9.240±2.401	**0.915**±0.088	0.910±0.063	6.310±2.494	0.895±0.073	5.292±1.876
SEDD-M	Pretrained	0.460±0.143	2.305±3.318	0.588±0.240	0.669±0.328	2.350±2.648	0.731±0.243	0.373±1.631
	BoN	0.582±0.069	5.241±2.441	0.831±0.123	0.852±0.119	4.057±2.022	0.857±0.088	1.874±2.243
	SMC	0.461±0.162	2.425±3.365	0.599±0.263	0.669±0.338	2.251±2.856	0.745±0.224	0.386±1.639
	SVDD	0.450±0.159	2.378±3.243	0.575±0.244	0.666±0.326	2.362±2.707	0.739±0.245	0.456±1.824
	CSMC	**0.619**±0.096	**9.792**±3.402	**0.894**±0.113	**0.875**±0.150	**10.026**±4.305	**0.885**±0.088	**7.153**±1.439
	CSMC-B	0.594±0.098	8.977±3.041	0.821±0.262	0.846±0.189	5.677±5.179	0.863±0.162	5.991±1.643
SEDD-U	Pretrained	0.458±0.154	1.654±2.355	0.637±0.230	0.741±0.266	2.524±1.753	0.771±0.189	0.455±1.755
	BoN	0.583±0.066	3.862±1.996	0.843±0.112	0.904±0.045	4.056±1.241	0.889±0.049	1.729±2.216
	SMC	0.546±0.113	2.718±2.790	0.811±0.196	0.850±0.174	2.610±1.684	0.847±0.136	3.334±3.337
	SVDD	0.518±0.120	2.482±2.624	0.779±0.234	0.809±0.217	2.759±1.784	0.817±0.148	3.189±3.330
	SGDD	0.536±0.121	2.644±2.791	0.684±0.211	0.844±0.152	2.535±1.627	0.847±115	9.240±2.050
	CSMC	**0.619**±0.080	**7.488**±5.376	0.886±0.075	**0.922**±0.073	**5.499**±2.614	**0.898**±0.074	**10.09**±2.637
	CSMC-B	0.572±0.068	7.213±4.325	**0.908**±0.120	0.894±0.112	3.625±0.870	0.883±0.088	9.350±1.762

Table 2: **Average reward (with 95% confidence intervals) for each method and pretrained discrete diffusion model.** Higher is better. **Bold** indicates the best method, and underline denotes the second best. CSMC achieves the best average reward across all datasets and discrete diffusion models.

likely to be sampled by the pretrained model and performs worse in cases where high-reward samples lie in low-density regions of the pretrained model. As shown in Fig. 2, for ring count and HepG2 rewards, the high-reward samples generated by CSMC (red dashed line) lie in low-density regions unlikely to be sampled from by the pretrained model. CSMC, which uses the forward-backward diffusion process to guide the samples toward high-reward regions, is able to achieve high rewards across all datasets and reward functions even when high-reward samples lie in low-density regions of the pretrained model.

Although SGDD achieves the second-best result on the MPRA dataset, it is only applicable to uniform CTMC discrete diffusion models whereas our CSMC is applicable to all discrete diffusion models. Furthermore, SGDD still relies on computing intermediate rewards, limiting its performance on datasets such as QM9 and ZINC250K where the intermediate rewards are inaccurate.

Wall-Clock Time. We include comparisons of wall-clock time and performance in Tab. 8. Under matched wall-clock time, CSMC-B significantly outperforms all other baselines in ring count and HepG2. Additional wall-clock time comparisons are included in App. D.2.

6 Conclusion

We propose **CSMC**, a novel training-free reward-guided sampler for discrete diffusion which avoids reliance on noisy intermediate rewards based on constructing a Markov Chain of clean samples via the Metropolis-Hastings algorithm. Our proposal distribution, modeled through a forward–backward combination, makes the acceptance probability tractable. Experiments on molecular and biological sequence generation with various reward functions demonstrate superior performance of our method compared to previous methods that rely on intermediate rewards.

Limitations and Future Work. While our method is effective in reward-guided sampling, using burn-in means many samples are simply discarded. Optimizing the initial sample so that it is closer to the target distribution may help reduce the burn-in time and is left for future work. Lastly, designing or optimizing the proposal distribution that is more aligned with the target distribution is also an exciting area for future research on more efficient sampling.

ACKNOWLEDGEMENTS

This work was supported by IITP grants (RS-2024-00399817, RS-2025-25441313, RS-2025-25443318), funded by the Korean government (MSIT); the Industrial Technology Innovation Program (RS-2025-02317326), funded by the Korean government (MOTIE); the National Supercomputing Center (KSC-2025-CRE-0475); and the DRB-KAIST SketchTheFuture Research Center.

REFERENCES

Kingma DP Ba J Adam et al. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1412(6), 2014.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *NeurIPS*, 2021.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18 (10):1196–1203, 2021.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *CVPR*, 2023.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *NeurIPS*, 2022.

Wenda Chu, Zihui Wu, Yifan Chen, Yang Song, and Yisong Yue. Split gibbs discrete diffusion posterior sampling. *NeurIPS*, 2025.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *ICLR*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

Sager J Gosai, Rodrigo I Castro, Natalia Fuentes, John C Butts, Susan Kales, Ramil R Noche, Kousuke Mouri, Pardis C Sabeti, Steven K Reilly, and Ryan Tewhey. Machine-guided design of synthetic cell type-specific cis-regulatory elements. *bioRxiv*, 2023.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. ZINC: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7): 1757–1768, 2012.

Jaihoon Kim, Taehoon Yoon, Jisung Hwang, and Minhyuk Sung. Inference-time scaling for flow models via stochastic generation and rollover budget forcing. *NeurIPS*, 2025a.

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization. In *ICLR*, 2025b.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Gopal Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. In *ICML*, 2025.

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. *NeurIPS*, 2025.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stark, Yilun Xu, Tommi Jaakkola, and Rafael Gomez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising. In *ICLR*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, 2024.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, et al. Scaling inference time compute for diffusion models. In *CVPR*, 2025.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *NeurIPS*, 2022.

Pierre Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer, 2004.

Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

Christian A Naesseth, Fredrik Lindsten, Thomas B Schön, et al. Elements of sequential monte carlo. *Foundations and Trends in Machine Learning*, 12(3):307–392, 2019.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *ICLR*, 2025.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*, 2025.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Cheng-Hao Liu, Sarthak Mittal, Nouha Dziri, Michael M Bronstein, Pranam Chatterjee, Alexander Tong, and Joey Bose. Steering masked discrete diffusion models via discrete denoising posterior prediction. In *ICLR*, 2025.

Christian P Robert and George Casella. Metropolis–hastings algorithms. In *Introducing Monte Carlo Methods with R*, pp. 167–197. Springer, 2009.

François Rozet, Gérôme Andry, François Lanusse, and Gilles Louppe. Learning diffusion priors from observations by expectation maximization. *NeurIPS*, 2024.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *NeurIPS*, 2024.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. *ICLR*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *NeurIPS*, 2024.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *ICML*, 2025.

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

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *ICLR*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. In *ICML*, 2024.

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. *NeurIPS*, 2020.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *ICLR*, 2023.

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.

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.

Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. Split-and-augmented gibbs sampler—application to large-scale inference problems. *IEEE Transactions on Signal Processing*, 67 (6):1648–1661, 2019.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Tommaso Biancalani, et al. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. In *ICLR*, 2025.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *NeurIPS*, 2023.

Taehoon Yoon, Yunhong Min, Kyeongmin Yeo, and Minhyuk Sung. Ψ-sampler: Initial particle sampling for smc-based inference-time reward alignment in score models. *NeurIPS*, 2025.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. Freedom: Training-free energy-guided conditional diffusion model. In *ICCV*, 2023.

Oussama Zekri and Nicolas Boullé. Fine-tuning discrete diffusion models with policy gradient methods. *NeurIPS*, 2025.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models. *NeurIPS*, 2025.

APPENDIX

A METROPOLIS-HASTINGS ALGORITHM

In this section, we present a brief overview of the Metropolis-Hastings (MH) algorithm. For more details, we refer the reader to chapter 12 section 2 of Murphy (2023).

The MH algorithm constructs a Markov chain which converges to a target distribution $p^\star(x)$. In order to do so, a **proposal distribution** $q(\cdot|x)$ proposes to move from the current state x to a new state $x' \sim q(\cdot|x)$. After proposing the new state x', MH decides whether to accept or reject the new state with the acceptance probability

$$A(x'|x) = \min(1, \alpha), \quad \alpha = \frac{p^\star(x')q(x|x')}{p^\star(x)q(x'|x)}.$$

Intuitively, the accept-reject step is necessary as the proposal distribution may not match the target distribution. Samples closer to the target distribution are accepted with higher probability whereas samples further from the target distribution are rejected. The MH algorithm is summarized in Alg. 2. The Markov chain constructed by the MH algorithm has the following transition matrix:

$$p(x'|x) = \begin{cases} q(x'|x)A(x'|x) & \text{if } x' \neq x \\ q(x|x) + \sum_{x' \neq x} q(x'|x)(1 - A(x'|x)) & \text{otherwise} \end{cases} \tag{5}$$

Algorithm 2 Metropolis-Hastings Algorithm (MH)

1: **for** $k = 1, \ldots, K$ **do**
2: $\quad \tilde{x}^{(k+1)} \sim q(\cdot|x^{(k)})$
3: $\quad \alpha \leftarrow \frac{p^\star(x')q(x|x')}{p^\star(x)q(x'|x)}$
4: $\quad A \leftarrow \min(1, \alpha)$
5: $\quad \rho \sim \mathcal{U}(0, 1)$
6: $\quad x^{(k+1)} \leftarrow \tilde{x}^{(k+1)}$ if $\rho < A$ else $x^{(k)}$
7: **end for**
8:
9: $\{x^{(1)}, \ldots, x^{(K)}\}$

Theorem A.1 (Theorem 12.2.1 from Murphy (2023))**.** *If the transition matrix defined by Eq. 5 defined by the MH algorithm is ergodic and irreducible, $p^\star$ is its unique limiting distribution.*

Proof. Consider two states x' and x. A Markov chain is said to satisfy the detailed balance equation if the following holds:

$$p^\star(x)p(x'|x) = p^\star(x')p(x|x'). \tag{6}$$

It is known that if a Markov chain satisfies the detailed balance equation, then $p^\star$ is its stationary distribution (Theorem 2.6.3 from Murphy (2023)).

To show that $p^\star$ is the unique limiting distribution of the Markov chain defined by Eq. 5, it suffices to show that it satisfies the detailed balance condition in Eq. 6.

Without loss of generality, assume $p^\star(x)q(x'|x) \geq p^\star(x')q(x|x')$. Then, $\alpha = \frac{p^\star(x')q(x|x')}{p^\star(x)q(x'|x)} < 1$ and thus $A(x'|x) = \alpha$. Similarly, by switching the arguments, $A(x|x') = 1$.

To move from x to x', x' must be proposed and accepted. Hence,

$$\begin{aligned} p(x'|x) &= q(x'|x)A(x'|x) && \text{from Eq. 5} \\ &= \frac{p^\star(x')q(x|x')}{p^\star(x)} \end{aligned}$$

It suffices to show that $q(x|x') = p(x|x')$:

$$\begin{aligned} q(x|x') &= q(x|x')A(x|x') && \because A(x|x') = 1 \\ &= p(x|x') && \text{from Eq. 5} \end{aligned}$$

Since the MH Markov chain satisfies the detailed balance equation, $p^\star$ is its stationary distribution. □

B Derivation of the Acceptance Probability

Recall from Sec. 4 that the proposal distribution $q(\cdot|x_0)$ is defined in Eq. 3 as:

$$q(x_0'|x_0) := p_t(x_t|x_0)p(x_0'|x_t) \tag{7}$$

We calculate α which is used to compute the acceptance probability A. First, we draw and fix a noisy sample $x_t \sim p_t(\cdot|x_0)$. Once x_t has been drawn, α simplifies to

$$\begin{aligned} \alpha &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0')q(x_0|x_0')}{p(x_0)q(x_0'|x_0)} \\ &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0')p(x_t|x_0')p(x_0|x_t)}{p(x_0)p(x_t|x_0)p(x_0'|x_t)} \\ &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0')p(x_t|x_0')p(x_0|x_t, x_0')}{p(x_0)p(x_t|x_0)p(x_0'|x_t, x_0)} \\ &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right) \frac{p(x_0, x_t, x_0')}{p(x_0', x_t, x_0)} \\ &= \exp\left(\frac{r(x_0') - r(x_0)}{\beta}\right), \end{aligned}$$

where the third line follows from the conditional independence of x_0 and x_0' given x_t due to the Markov property: $p(x_0|x_t) = p(x_0|x_t, x_0')$.

Although it may appear that this Markov chain is time-inhomogeneous since $t \sim \mathcal{U}(t_{\text{lo}}, t_{\text{hi}})$ and $x_t \sim p(\cdot|x_0)$ are resampled at each step, it is secretly *time-homogeneous*. To see this, we simplify the proposal distribution to an equivalent time-homogeneous proposal distribution. The probability of proposing x_0' given that we are currently at state x_0 is $p_t(x_t|x_0)p(x_0'|x_t)$ with probability $\frac{1}{t_{\text{hi}} - t_{\text{lo}}}$. To obtain the probability $q(x_0'|x_0)$, we simply condition the proposal probability on t and x_t and integrate over all possible t and x_t:

$$\begin{aligned} q(x_0'|x_0) &= \int_{t_{\text{lo}}}^{t_{\text{hi}}} \int_{\mathbb{X}_t} p_t(x_t|x_0)p(x_0'|x_t)f(t)dx_t dt \\ &= \mathbb{E}_{t\sim\mathcal{U}(t_{\text{lo}}, t_{\text{hi}})}\mathbb{E}_{x_t\sim p_t(\cdot|x_0)}[p(x_0'|x_t)], \end{aligned}$$

which is time-homogeneous. Thus, we still have an equivalent time-homogeneous Markov chain and standard convergence proofs in App. A apply.

C Experiment Details

C.1 Pretrained Models

We provide details on the training setup and hyperparameters of each diffusion model on each dataset in Tab. 3. For QM9 and ZINC250K, we use the transformer architecture for the diffusion, whereas for MPRA we use the CNN architecture, following Stark et al. (2024) and Wang et al. (2025). Note that the SEDD-U model pretrained on MPRA is taken from the publicly available checkpoint provided by Chu et al. (2025). All models except for SEDD-U was trained using Adam (Adam et al., 2014) while SEDD-U was trained using AdamW (Loshchilov & Hutter, 2019).

	MDM, USM, and SEDD-M			SEDD-U	
	QM9	ZINC250K	MPRA	QM9	ZINC250K
Train steps	25,000	50,000	131,500	40,000	100,000
Context size	32	74	200	32	74
Batch size	1024	384	512	512	512
LR	$3e^{-4}$	$3e^{-4}$	$2e^{-3}$	$3e^{-4}$	$3e^{-4}$
Optim.	ADAM	ADAM	ADAM	ADAMW	ADAMW
	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
LR sched.	Constant Warmup	Constant Warmup	Cosine Decay	Constant Warmup	Constant Warmup
	-	-	$3e^{-6}$ min.	-	-
LR warmup steps	2,500	2,500	3,000	2,500	2,500
GPU count	2	2	2	8	4
GPU type	RTX3090	RTX3090	RTX3090	RTX3090	RTX3090

Table 3: **Training setup and Hyperparameters.** Training setup and hyperparameters on QM9, ZINC250K, and MPRA using MDM (Sahoo et al., 2024), USM (Schiff et al., 2025), SEDD-U (Lou et al., 2024), and SEDD-M (Lou et al., 2024).

C.2 REWARD-GUIDED SAMPLING

We provide experimental details on each evaluation setup. We sample 1024 molecules for QM9 and ZINC250K, and 640 DNA sequences for MPRA. We fix 32 denoising steps for QM9, 74 denoising steps for ZINC250K, and 128 denoising steps for MPRA. For CSMC-B, we use a batch size of 8 for QM9 and ZINC250K, and a batch size of 4 for MPRA.

For experiments on the QM9 (Ramakrishnan et al., 2014) and ZINC250K (Irwin et al., 2012) datasets, we fix the total diffusion model NFE per sample to be 1024. For MPRA (Gosai et al., 2023), we fix the NFE as 1000 as done by Chu et al. (2025). Since one run of CSMC generates multiple samples, we draw S samples from the resulting chain by discarding the first half of the chain (burn-in) and taking S equally-spaced samples from the latter half. Since S samples are generated, we scale the NFE by S for CSMC. This scaling highlights one of the key benefits of CSMC: after the initial burn-in period, samples can be generated quickly using a few steps. For all experiments, we fix $\beta = 0.02$. To obtain an x_0 prediction from SEDD, we run sampling using one step to jump to time $t = 1$ (clean sample).

Hyperparameters used by our method is shown in Tab. 4. Initial denoising steps refer to the number of steps used to obtain the initial clean sample $x_0^{(1)}$ whereas denoising steps M refer to the number of steps used during the M-step reverse process in every MH iteration. Note that M is much smaller than the initial denoising steps.

	QM9	ZINC250K	MPRA
Samples per Iteration S	128	128	64
MH Iterations K	26208	26199	6390
Initial Denoising Steps	32	74	100
Denoising Steps M	5	5	10
t_l	0.2	0.2	0.2
t_h	0.5	0.5	0.7

Table 4: **Hyperparameters for CSMC.**

D Additional Experimental Results

D.1 Diversity

We compute the diversity as follows:

- For molecule tasks (QM9 and ZINC250K), we compute the mean pairwise Tanimoto similarity based on the Morgan2 fingerprint, and subtract it from 1.
- For the DNA task (MPRA), compute the mean pairwise cosine similarity of the one-hot encodings, and subtract it from 1.

The results are shown in Tab. 5. BoN and CSMC have a slight decrease in diversity while SMC suffers from significant degradation in diversity. This can be attributed to samples clustering around the high-reward modes. CSMC consistently achieves a diversity score of over 0.8 on molecule tasks which suggests that the samples are indeed diverse, with an average Tanimoto similarity of less than 0.2. On DNA tasks, the cosine similarity between the sequences is less than 0.3, indicating that the generated sequences are diverse.

		QM9			**ZINC250K**			**MPRA**
		QED	Rings	SA	QED	Rings	SA	HepG2
MDM	Pretrained	0.922	0.922	0.922	0.910	0.910	0.910	0.749
	BoN	0.904	0.884	0.894	0.876	0.874	0.876	0.749
	SMC	0.834	0.916	0.802	0.929	0.918	0.876	0.747
	SVDD	0.920	0.920	0.920	0.900	0.896	0.900	0.747
	CSMC	0.885	0.804	0.862	0.874	0.797	0.874	0.742
	CSMC-B	0.891	0.854	0.868	0.872	0.877	0.839	0.729
USM	Pretrained	0.921	0.921	0.921	0.879	0.879	0.879	0.748
	BoN	0.895	0.890	0.912	0.862	0.866	0.841	0.749
	SMC	0.923	0.920	0.927	0.874	0.875	0.874	0.748
	SVDD	0.922	0.918	0.921	0.875	0.875	0.875	0.748
	CSMC	0.880	0.827	0.868	0.856	0.837	0.841	0.734
	CSMC-B	0.881	0.856	0.881	0.853	0.861	0.838	0.732
SEDD-M	Pretrained	0.926	0.926	0.926	0.905	0.905	0.905	0.748
	BoN	0.903	0.889	0.926	0.874	0.874	0.877	0.749
	SMC	0.925	0.914	0.927	0.902	0.902	0.902	0.748
	SVDD	0.928	0.928	0.928	0.907	0.907	0.907	0.748
	CSMC	0.885	0.847	0.890	0.865	0.880	0.837	0.749
	CSMC-B	0.898	0.858	0.898	0.887	0.888	0.870	0.732
SEDD-U	Pretrained	0.922	0.922	0.922	0.879	0.879	0.879	0.631
	BoN	0.900	0.900	0.915	0.866	0.867	0.851	0.661
	SMC	0.901	0.905	0.910	0.872	0.873	0.866	0.661
	SVDD	0.906	0.907	0.914	0.873	0.877	0.870	0.666
	SGDD	0.906	0.913	0.908	0.868	0.866	0.859	0.650
	CSMC	0.880	0.864	0.863	0.855	0.868	0.825	0.709
	CSMC-B	0.869	0.865	0.848	0.859	0.863	0.843	0.662

Table 5: **Diversity metrics for each method and reward.** Higher is better.

D.2 Wall-Clock Time Comparisons

We provide additional wall-clock time comparisons in Tab. 7. We also provide a comparison of the wall-clock times for CSMC and CSMC-B in Tab. 6, showing that CSMC-B provides significantly accelerated sampling compared to CSMC.

	CSMC	CSMC-B
Time (s)	3029	334
Batch Size	1	8
NFE	1024	1024
Reward	0.916	0.917

Table 6: **Wall-clock time comparisons.** Wall-clock time measured CSMC applied to USM guided by ZINC250K QED reward. Batching significantly improves the speed of CSMC.

	ZINC250K														
	QED					Rings					SA				
	CSMC	BoN	BoN	SMC	SVDD	CSMC	BoN	BoN	SMC	SVDD	CSMC	BoN	BoN	SMC	SVDD
Time (s)	334	334	359	359	264	321	331	359	320	276	337	328	300	341	279
Batch Size	8	78	8	14	886	8	78	8	14	886	8	78	8	14	886
NFE	1024	5750	1024	1024	65536	1024	5750	1024	1024	65536	1024	5750	1024	1024	65536
Reward	0.917	0.934	0.9133	0.766	0.719	5.315	4.945	4.312	2.630	2.607	0.912	0.919	0.892	0.792	0.755

Table 7: **USM ZINC250K rewards under matched wall-clock time.** Rewards and time are computed from 128 drawn samples. CSMC-B significantly outperforms all other baselines in ring count while being comparable with BoN in QED and SA.

D.3 Comparison with Training-based Methods

We provide comparison with standard training-based guidance methods. In Tab. 9, we compare CSMC against discrete classifier-free guidance (D-CFG) (Nisonoff et al., 2025) on the QM9 dataset. CSMC outperforms D-CFG across all rewards when using MDM as the base model. When using USM as the base model, CSMC outperforms D-CFG on QED and ring count while achieving comparable SA. Furthermore, CSMC can be applied on top of D-CFG, resulting in further improvements except in ring count for ZINC250K MDM. This is due to the MDM CFG model collapsing to narrow modes when trained on ZINC250K ring count, resulting in insufficient exploration or the space when applying CSMC.

	ZINC250K: Rings				
	CSMC-B	BoN	BoN	SMC	SVDD
Batch Size	8	78	8	14	886
NFE	1024	5750	1024	1024	65536
Reward	5.315	4.945	4.312	2.630	2.607
Time (s)	321	331	359	320	276

Table 8: **ZINC250K rewards under matched wall-clock time.** Rewards and time are computed from 128 drawn samples. CSMC-B significantly outperforms all other baselines in ring count while being comparable with BoN in QED and SA.

		QM9		
		QED	Rings	SA
MDM	D-CFG	0.820	5.886	0.888
	CSMC	0.910	**8.091**	0.905
	D-CFG + CSMC	**0.927**	7.214	**0.928**
USM	D-CFG	0.903	5.613	0.908
	CSMC	0.917	8.703	0.898
	D-CFG + CSMC	**0.927**	**8.943**	**0.920**

Table 9: **Comparison with training-based guidance on ZINC250K.** CSMC outperforms D-CFG. CSMC can also be combined with D-CFG by adopting D-CFG-based proposals, resulting in further improvements.

D.4 Analysis and Hyperparameter Studies

In this section, we analyze the effects of the various components of CSMC.

Convergence. Since CSMC requires running sufficiently many iterations until the Markov chain converges to its stationary distribution, we empirically analyze the speed of convergence as well as the impact of the chain length. The autocorrelation function shown in Fig. 3 quickly decays to zero within the first 2000 iterations and remains small until the end, indicating that the chain converges quickly. We also report additional autocorrelation plots as well as the acceptance rate in App. D.5.

Initialization. Since CSMC constructs a Markov chain of clean samples, the choice of initialization could impact the convergence rate of the chain. We have already shown in Fig. 3 that the Markov chain shows fast mixing. We now provide further evidence that the initialization $x_0^{(1)}$ of the chain does not significantly affect the convergence. In Fig. 4, we plot the reward trajectories for CSMC

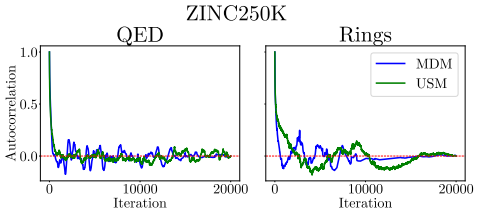


Figure 3: **Autocorrelation plots for ZINC250K MDM and USM.** The autocorrelation function quickly vanishes to zero within the first 2000 iterations, indicating fast mixing.

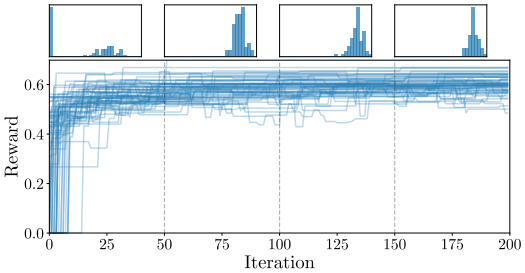


Figure 4: **Reward trajectories with different initializations.** We plot the reward trajectories of CSMC for 64 different initializations using MDM on QM9 QED (bottom), along with the reward distribution of the 64 different chains (top). Trajectories with lower reward also quickly converge to high reward regions of the distribution, demonstrating robustness to the initialization.

with 64 different initializations. The reward distribution across the 64 chains are shown as histograms on top (corresponding to the iterations marked by the gray dashed lines). Although each initial point starts with a different reward, many of which are zero, as shown in the histogram, the rewards of each chain quickly converge to the same high-reward region.

Results with Varying Number of Reverse Steps M. CSMC enables fast sampling by leveraging an M-step reverse sampler in the forward-reverse proposal to produce a new candidate sample. As shown in Tab. 10, varying M does not significantly affect the rewards achieved by CSMC, except when using large M for the ring count reward. We hypothesize that this is due to limited compute since using for ZINC250K tasks results in only 88 iterations of CSMC, which may not be sufficient for harder tasks such as ring count where high reward samples lie in low density regions of the pretrained model's distribution. This hypothesis is also supported by the autocorrelation plot in Fig. 3 which shows that the autocorrelation function for USM (green) requires more iterations to converge for ring count.

	ZINC250K		
M	QED	Ring	SA
5	0.917	8.703	0.898
10	0.910	8.477	0.901
15	0.917	8.256	0.895
20	0.917	6.885	0.897

Table 10: **Results with varying number of reverse steps M for ZINC250K USM.** We vary the number of reverse steps M in the forward-reverse proposal. Varying does not significantly affect the rewards achieved by CSMC.

D.5 Autocorrelation Plots and Acceptance Rate

We provide autocorrelation plots for ZINC250K MDLM and USM in Fig. 5 to further investigate the burn-in time. The autocorrelation function quickly vanishes to zero within the first 2000 iterations and remains small until the end, indicating that the chain converges quickly. Since these metrics are only heuristics for measuring convergence, we conservatively discard the first half of the chain. However, the autocorrelation plots suggest that it may be possible to burn fewer samples.

We provide acceptance rates for CSMC with MDLM and USM in Tab. 11. Although the acceptance rate for MDLM is quite low due to MDLM often generating invalid molecules, the acceptance rate is still acceptable as we draw 128 samples from 13k samples after the burn-in period. On the other hand, USM has much higher acceptance rates which can be attributed to its higher validity rate due to the ability to correct errors from previous diffusion steps

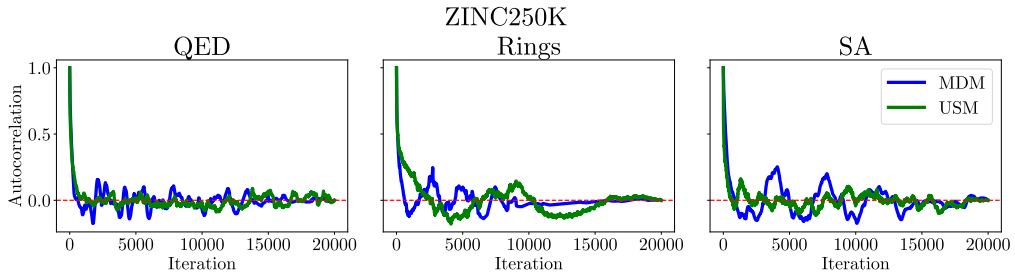


Figure 5: **Autocorrelation plots for ZINC250K MDM and USM.**

	QM9			ZINC250K			MPRA
	QED	Rings	SA	QED	Rings	SA	HepG2
MDM	0.032	0.002	0.009	0.009	0.003	0.011	0.029
USM	0.510	0.330	0.346	0.725	0.597	0.711	0.038

Table 11: **Acceptance rates for MDM and USM.**

NFE Comparison We compare the performance of each method across various NFEs for QM9 and ZINC250K. We use NFEs $\in \{512, 1024, 2048, 4096\}$ for molecule tasks and NFEs $\in \{500, 1000, 2000, 4000\}$ for MPRA. The results are shown in Fig. 6. As shown in the figure, CSMC consistently and significantly outperforms all other methods in ring count and HepG2 activity across all diffusion models and datasets. CSMC also outperforms all other methods in SA except for BoN when using USM on ZINC250K, and outperforms all other methods in QED except for BoN when using USM. We note that for the hardest reward ring count where the high-reward samples lie in extremely low density regions, CSMC consistently outperforms all other methods by a large margin.

Time Parameters t_{lo} and t_{hi} We provide additional analysis on the time parameters t_{lo} and t_{hi}. Larger t_{lo} and t_{hi} lead to increased exploration as the samples from the proposal distribution become more uncorrelated due to the larger noise scale. Smaller values lead to increased exploitation as samples are more correlated due to the smaller noise scale.

To test the sensitivity of CSMC with respect to these time parameters, we fix $t_{\text{lo}} = 0.2$ and vary $t_{\text{hi}} \in [0.2, 0.7]$. We plot the reward and diversity for ZINC250K USM samples with respect to these parameters. We also include a plot where we fix $t_{\text{hi}} = 0.8$ and vary $t_{\text{lo}} \in [0.3, 0.8]$. The results are shown in Fig. 7. As shown in the figure, our method is not sensitive to the time parameters in general. However, For ring count, the reward decreases as t_{lo} increases. Intuitively, as t_{lo} increases, each CSMC step changes the current x_0 sample significantly, resulting in greater exploration but less exploitation. For simpler rewards such as QED and SA, this does not significantly impact the average reward. However, for ring count where high-reward samples lie in extremely low density regions of the pretrained model's learned distribution (refer to Fig. 2), it is necessary to choose smaller times to exploit and explore around the current x_0.

E Qualitative Results

Qualitative results are shown in Fig. 8.

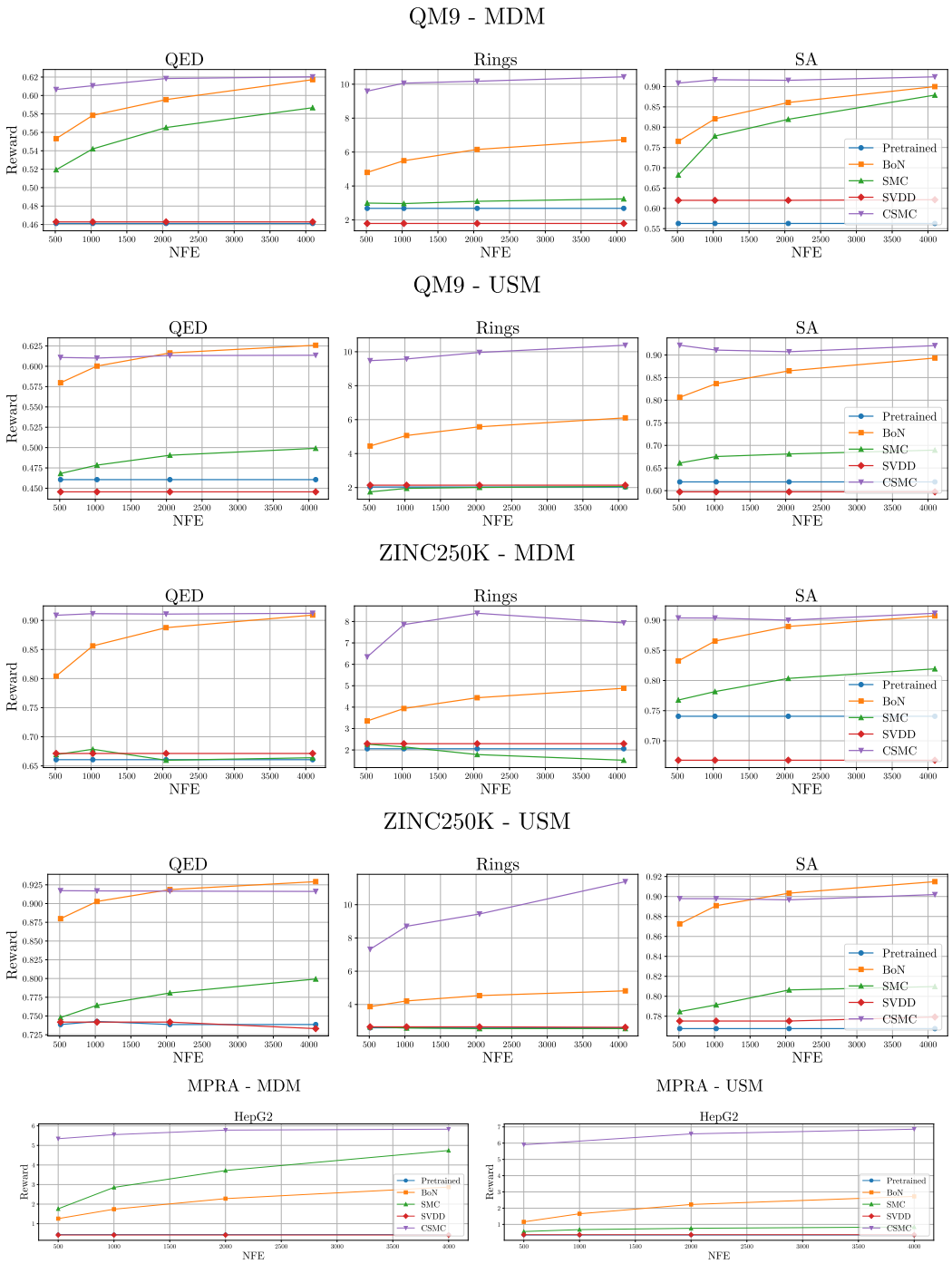


Figure 6: **Performance across various NFEs.** We run each method on molecule generation using NFEs $\in \{512, 1024, 2048, 4096\}$ for molecule tasks and NFE $\in \{500, 1000, 2000, 4000\}$ for MPRA.

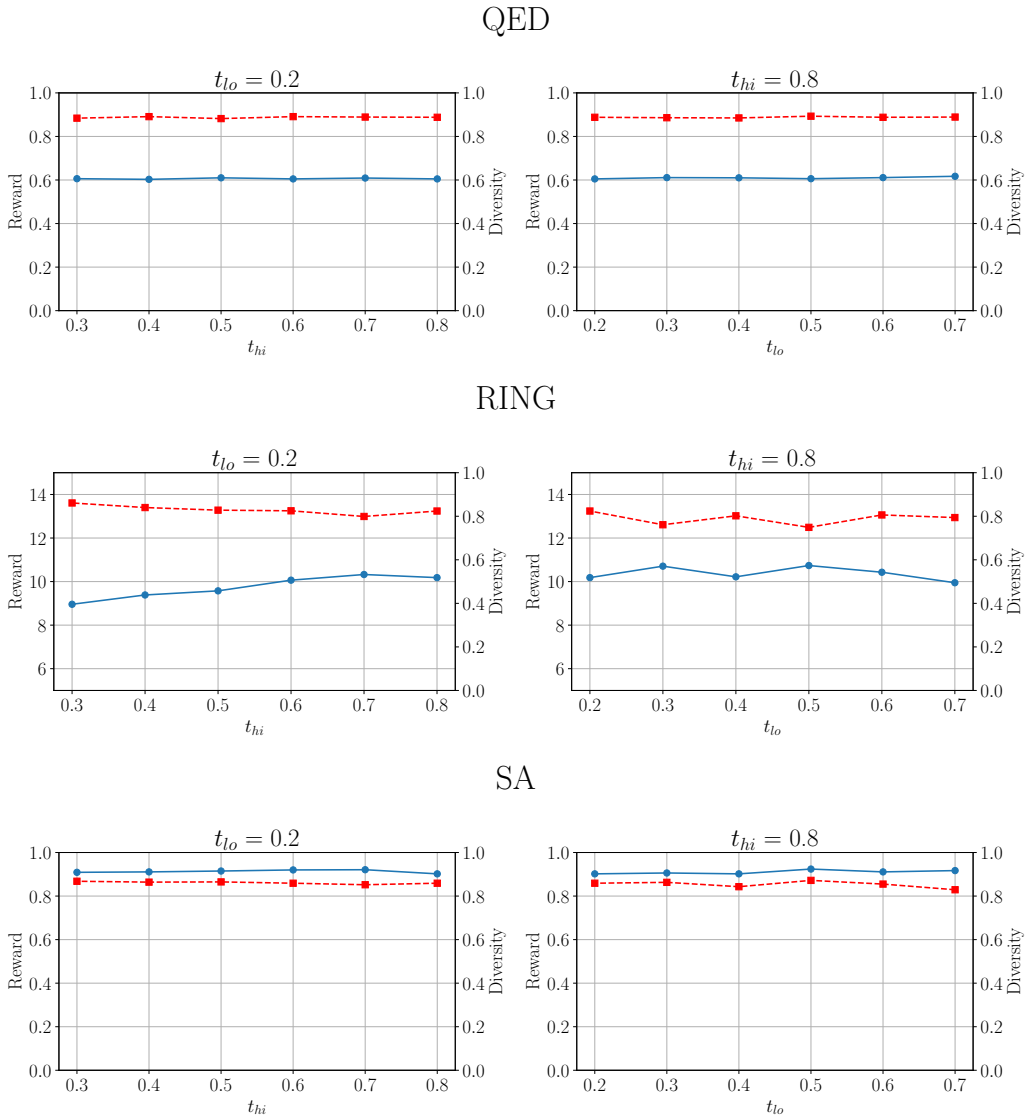


Figure 7: **Analysis of time parameters.** We plot the reward and diversity for ZINC250K USM samples with respect to (1) fixing $t_{\text{lo}} = 0.2$ and vary $t_{\text{hi}} \in [0.3, 0.8]$, and (2) fixing $t_{\text{hi}} = 0.8$ and vary $t_{\text{lo}} \in [0.2, 0.7]$.

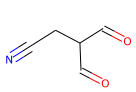
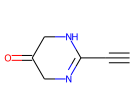
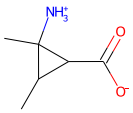
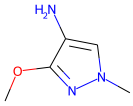
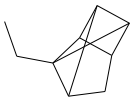
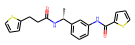
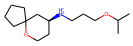
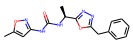
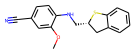
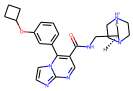
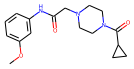
	Pretrained			CSMC		
QM9	QED: 0.378	Rings: 1	SA: 0.501	QED: 0.578	Rings: 9	SA: 0.920
						
ZINC250K	QED: 0.610	Rings: 2	SA: 0.802	QED: 0.934	Rings: 7	SA: 0.915
						

Figure 8: **Qualitative results.** Molecules randomly sampled from discrete diffusion models pretrained on QM9 (Ramakrishnan et al., 2014) and ZINC250K (Irwin et al., 2012). CSMC generates molecules with high rewards as shown on the right.