BIAS MITIGATION IN GRAPH DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing graph generative diffusion models suffer from significant exposure bias during graph sampling. We observe that the forward diffusion's maximum perturbation distribution in most models deviates from the standard normal distribution, while reverse sampling consistently starts from a standard normal distribution. This mismatch results in a reverse-starting bias, which, together with the exposure bias, degrades generation quality. The exposure bias typically accumulates and propagates throughout the sampling process. In this paper, we effectively address both biases. To mitigate the reverse-starting bias, we employ a newly designed Langevin sampling algorithm to align with the forward maximum perturbation distribution, establishing a new reverse-starting point. To address the exposure bias, we introduce a fraction correction mechanism based on a newly defined score difference. Our approach, which requires no network modifications, is validated across multiple models, datasets, and tasks, achieving state-of-the-art results.

1 INTRODUCTION

In recent years, graph generative models have made significant progress in the field of generation by learning the underlying distribution of graphs, which is crucial for areas such as molecular structures, recommendation systems, and social networks. For the graph generative process, autoregressive models (You et al., 2018; Liao et al., 2019) rely on sequential decision-making to gradually generate nodes and edges of the graph, while one-shot models (De Cao & Kipf, 2018; Liu et al., 2019) generate all components of the graph at once. Currently, GDSS (Song et al., 2021) introduces score-based diffusion models to the one-shot graph generation task, demonstrating remarkable results and proving superior to baselines, sparking widespread follow-up and discussion among scholars. However, we believe that directly introducing diffusion models to graph generative tasks may exacerbate some inherent issues of diffusion models.

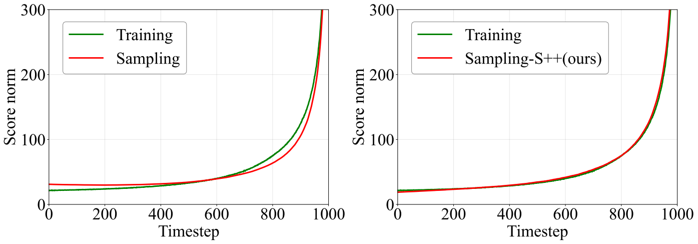
Reverse-Starting Bias. The core mechanism of graph generative diffusion models (Ho et al., 2020; Song et al., 2021) is the process of forward noise addition and reverse denoising. Ideally, the forward process gradually perturbs the data distribution to a standard normal distribution, while the reverse diffusion process starts from the standard normal distribution and gradually recover the original data distribution. However, in the field of graph learning, due to limitations in data scale and network's learning ability, it is difficult to accurately predict scores or noise from high-noise data. This forces the forward perturbation to adopt a conservative strategy, where the maximum perturbation distribution falls far short of the standard normal distribution (Jo et al., 2022; Luo et al., 2023; Wen et al., 2024; Lee et al., 2023). Yet, the sampling starting point remains the standard normal distribution, resulting in a severe reverse-starting bias, as shown in Figs. 1a and 1b, which significantly affects the generation quality.

Exposure Bias. Diffusion models heavily rely on score networks to guide the generation process, but score networks (Song & Ermon, 2019) cannot predict the ground-truth scores. This leads to deviations from the ideal path during the generation process. Coupled with the limitations of data scale and network's learning capability, this exposure bias exists and needs to mitigate. Ning et al. (2024) have already shown that exposure bias accumulate during the sampling process, further exacerbating the exposure bias (Ning et al., 2023) (described as the input mismatch between training and sampling) of diffusion models and ultimately affecting generative quality.

This paper aims to analyze and mitigate bias issues in graph diffusion models from a unified perspective. We first pose two interesting questions:

Model	GDSS	GSDM	HGDM	MOOD
Dataset	Comm.	Enz	QM9	ZINC250k
Type	Edge	Eigen	Edge	Node
SDE	VPSDE	VPSDE	VESDE	VPSDE
$\beta_{\min}$	0.1	0.1	0.1	0.1
$\beta_{\max}$	1.0	1.0	1.0	1.0
μ_T	0.7596	0.7596	1.0	0.7596
σ_T^2	0.4231	0.4231	1.0	0.4231

(a) reverse-starting Bias



(b) Baseline model-GDSS (c) Our methods-GDSS-S++

Figure 1: (a) The maximum perturbation distribution of these graph diffusion models in the training phase is $\mathcal{N}\left(\mu_T \boldsymbol{X}_0, \sigma_T \mathbf{I}\right)$, but the reverse-starting step of the sampling phase always obeys $\mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$, resulting in significant inconsistencies between training and sampling. We provide more details in appendix A. (b) and (c) Expectation of $\|\boldsymbol{s}_{\theta,t}(\cdot)\|_2$ during sampling (without corrector) and training on Community-small. Due to the initial deviation, there is a significant difference between training and inference in the early stages of sampling in (b). However, after improvement through our method, not only is the initial deviation eliminated, but the exposure bias during the sampling process is also further alleviated.

Q1: Is it possible to address the reverse-starting bias while mitigating exposure bias?

We find that: Graph diffusion models are highly sensitive to perturbations of scores or data in high-noise states, while their ability to correct biases significantly increases in low-noise states. Many models' maximum forward perturbation distributions are already in low-noise states. Considering that Langevin sampling can guide samples from a prior distribution to the target distribution, we use Langevin sampling guided by the score $\boldsymbol{s}_{\theta,T}(\cdot)$ to obtain samples aligned with the maximum forward perturbation distribution $q(\boldsymbol{x}_T|\boldsymbol{x}_0)$. The model uses this as a new starting point, and leveraging its ability to correct bias in low-noise states, successfully generates high-quality data.

However, the prediction error of $\boldsymbol{s}_{\theta,T}(\cdot)$ severely affects the stable distribution of Langevin sampling, forcing us to directly confront exposure bias. In particular, we also focus on the cost of achieving this goal of **Q1**.

Q2: How to solve Q1 without modifying the network or introducing other components?

Current graph diffusion models design different networks based on various standards (spatial domain, spectral domain and hyperbolic domain, etc.). Our bias correction method is seamlessly integrated into these approaches without modifying the models, and we do not introduce any additional components, such as GAN (Goodfellow et al., 2014), Flows, or an discriminator (Kim et al., 2023). We aim to fully utilize existing components of graph diffusion models to solve their own bias problems. Thus, we propose a correction mechanism based on the score difference. Specifically, we use the generated data from the diffusion model to train another score network $\boldsymbol{s}_{\psi,t}(\cdot)$, and then use a newly defined score difference between the two score $\boldsymbol{s}_{\theta,t}(\cdot) - \boldsymbol{s}_{\psi,t}(\cdot)$ to correct the bias of the score. We emphasize that the correction task of $\boldsymbol{s}_{\psi,T}(\cdot)$ is far more important than its real-time step score correction.

In summary, our contributions are:

- To the best of our knowledge, we are the first to systematically address bias issues in graph diffusion models, effectively employing Langevin sampling to resolve the reverse-starting bias while significantly mitigating the exposure bias in the graph sampling.

- We propose a score correction mechanism based on the score difference, and prove both theoretically and practically that the corrected scores are closer to the true scores, further mitigating the reverse-starting bias and the exposure bias.

- Our method does not require modifying the network or introducing new components. It has been validated on multiple graph diffusion models, multiple datasets, and multiple tasks, achieving state-of-the-art metrics.

2 RELATED WORK

Diffusion models were first introduced by Sohl-Dickstein et al. (2015) and later improved by Ho et al. (2020). Notably, Song et al. (2021) proposed a unified framework for diffusion models based on stochastic differential equations, greatly advancing their development. GDSS (Jo et al., 2022) was the first to introduce diffusion models to both nodes and edges of graphs. GSDM (Luo et al., 2023) extended GDSS by introducing the diffusion process of adjacency matrices into the spectral domain. HGDM (Wen et al., 2024) introduced node diffusion into hyperbolic space based on degree distribution characteristics. Huang et al. (2023) proposed a conditional diffusion model based on discrete graph structures. Additionally, Vignac et al. (2023) defined a discrete denoising diffusion model through the process of adding or removing edges and changing categories. Furthermore, Xu et al. (2022) proposed a diffusion model for predicting molecular conformations.

The reverse-starting bias of diffusion models was first discovered by Lin et al. (2024), which proposed modifying the diffusion noise schedule to force the last time step of forward diffusion to have zero Signal-to-Noise Ratio. Shortly after, Everaert et al. (2024) estimated the actual maximum perturbation distribution of forward noise addition as the starting point for inference to match the endpoint of forward noise addition. The exposure bias of diffusion models was first discovered by ADM-IP (Ning et al., 2023), which proposed re-perturbing the perturbation distribution to simulate exposure bias during inference. EB-DDPM (Li & van der Schaar, 2023) estimated the upper bound of cumulative errors and used it as a regularization term to retrain the model. MDSS (Ren et al., 2024) proposed a multi-step timed sampling strategy to mitigate exposure bias. It's worth noting that ADM-IP, EB-DDPM, and MDSS all require model retraining. In contrast, ADM-ES (Ning et al., 2024) proposed a noise scaling mechanism to mitigate exposure bias without retraining, while TS-DPM (Li et al., 2024) only needs to find the optimal time steps during inference to match the forward process as closely as possible.

We emphasize that our work focuses more on the reverse-starting bias, hoping to address it by utilizing components of the diffusion model itself while also mitigating exposure bias to some extent. This is a novel and interesting perspective.

3 PRELIMINARIES

3.1 GRAPH DIFFUSION MODELS

First, we define a graph with N nodes as $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ represents node features, with F indicating that each node has F features; $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ represents the weighted adjacency matrix. Then, we formally represent the graph diffusion process as the trajectory of the random variable $\boldsymbol{G}$ over time $[0, T]$, as shown below:

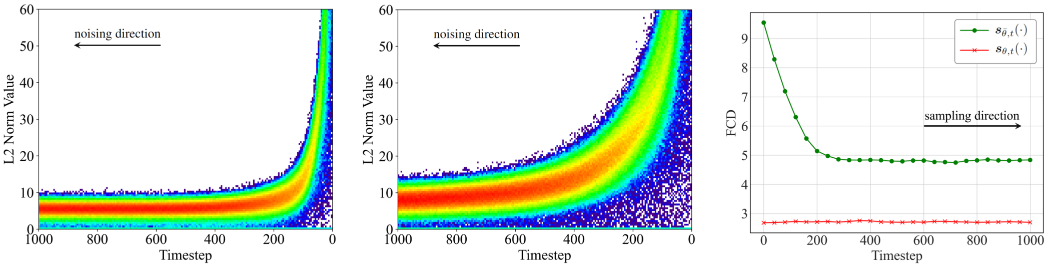
$$\mathrm{d}\boldsymbol{G}_t = \mathbf{f}_t(\boldsymbol{G}_t)\mathrm{d}t + g_t(\boldsymbol{G}_t)\mathrm{d}\boldsymbol{w}, \quad \boldsymbol{G}_0 \sim p_{\mathrm{data}}\,. \tag{1}$$

We view this diffusion process as an SDE, where $\mathbf{f}_t(\boldsymbol{G}_t)$ is the linear drift coefficient, $g_t(\boldsymbol{G}_t)$ is the diffusion coefficient, $\boldsymbol{w}$ is a standard Wiener process, and $\boldsymbol{G}_0$ is a graph from the original distribution p_{data}. Specifically, we replace $\boldsymbol{G}$ in Eq. (1) with node $\boldsymbol{X}$ or edge $\boldsymbol{A}$, representing the forward diffusion process of node $\boldsymbol{X}$ or edge $\boldsymbol{A}$, separately.

Following GDSS Jo et al. (2022), we separate $\boldsymbol{X}$ and $\boldsymbol{A}$ in the reverse diffusion:

$$\begin{aligned} \mathrm{d}\boldsymbol{X}_t &= \left[\mathbf{f}_{1,t}(\boldsymbol{X}_t) - g_{1,t}^2 \nabla_{\boldsymbol{X}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)\right] \mathrm{d}\bar{t} + g_{1,t}\mathrm{d}\bar{\boldsymbol{w}}_1\,, \\ \mathrm{d}\boldsymbol{A}_t &= \left[\mathbf{f}_{2,t}(\boldsymbol{A}_t) - g_{2,t}^2 \nabla_{\boldsymbol{A}_t} \log p_t(\boldsymbol{X}_t, \boldsymbol{A}_t)\right] \mathrm{d}\bar{t} + g_{2,t}\mathrm{d}\bar{\boldsymbol{w}}_2 \end{aligned} \tag{2}$$

where $\mathbf{f}_{1,t}$ and $\mathbf{f}_{2,t}$ satisfy $\mathbf{f}_t(\boldsymbol{X}, \boldsymbol{A}) = (\mathbf{f}_{1,t}(\boldsymbol{X}), \mathbf{f}_{2,t}(\boldsymbol{A}))$, representing the drift coefficients of the reverse diffusion process for nodes and edges respectively. $g_{1,t}$ and $g_{2,t}$ are the corresponding scalar diffusion coefficients, $\bar{\boldsymbol{w}}_1$ and $\bar{\boldsymbol{w}}_2$ are standard Wiener processes in reverse time, and $\nabla_{\boldsymbol{X}_t} \log p\left(\boldsymbol{X}_t, \boldsymbol{A}_t\right)$ and $\nabla_{\boldsymbol{A}_t} \log p\left(\boldsymbol{X}_t, \boldsymbol{A}_t\right)$ represent the partial scores of nodes and edges respectively. It's worth noting that each SDE in Eq. (2) corresponds to the diffusion process of $\boldsymbol{X}$ and $\boldsymbol{A}$ respectively. We choose different types of SDEs for $\boldsymbol{X}$ and $\boldsymbol{A}$ based on actual conditions. For example, for VPSDE (Song et al., 2021), $\mathbf{f}_{1,t}(\boldsymbol{X}_t) = -\frac{1}{2}\beta(t)\boldsymbol{X}_t$, $\mathbf{f}_{2,t}(\boldsymbol{A}_t) = -\frac{1}{2}\beta(t)\boldsymbol{A}_t$, $g_{1,t} = g_{2,t} = \sqrt{\beta(t)}$, $\beta(t) = \bar{\beta}_{\min} + t(\bar{\beta}_{\max} - \bar{\beta}_{\min})$, where $\bar{\beta}_{\max}$ and $\bar{\beta}_{\min}$ are parameters we set in advance.



(a) $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$ (b) $\boldsymbol{s}_{\theta,t}(\cdot)$ (c) FCD metric

Figure 2: (a) and (b) The ℓ_2 norm distribution of the predicted outputs of the two score networks at different time steps. (c) The response of the predicted outputs of the two score networks to perturbations at different time steps during the sampling phase.

Next, we use $\boldsymbol{s}_{\theta,t}(\boldsymbol{G}_t)$ and $\boldsymbol{s}_{\phi,t}(\boldsymbol{G}_t)$ to estimate the partial scores $\nabla_{\boldsymbol{X}_t} \log p\left(\boldsymbol{X}_t, \boldsymbol{A}_t\right)$ and $\nabla_{\boldsymbol{A}_t} \log p\left(\boldsymbol{X}_t, \boldsymbol{A}_t\right)$ respectively. Based on the idea of reverse denoising score matching, we derive $\boldsymbol{s}_{\theta,t}(\boldsymbol{G}_t) \approx \nabla \boldsymbol{X}_t \log p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0)$ and $\boldsymbol{s}_{\phi,t}(\boldsymbol{G}_t) \approx \nabla \boldsymbol{A}_t \log p_{0t}(\boldsymbol{A}_t|\boldsymbol{A}_0)$, and then give the loss function of the model:

$$\begin{aligned} &\min_{\theta} \mathbb{E}_t \left\{ \lambda_1(t) \mathbb{E}_{\boldsymbol{G}_0} \mathbb{E}_{\boldsymbol{G}_t|\boldsymbol{G}_0} \left\| \boldsymbol{s}_{\theta,t}(\boldsymbol{G}_t) - \nabla_{\boldsymbol{X}_t} \log p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0) \right\|_2^2 \right\}. \\ &\min_{\phi} \mathbb{E}_t \left\{ \lambda_2(t) \mathbb{E}_{\boldsymbol{G}_0} \mathbb{E}_{\boldsymbol{G}_t|\boldsymbol{G}_0} \left\| \boldsymbol{s}_{\phi,t}(\boldsymbol{G}_t) - \nabla_{\boldsymbol{A}_t} \log p_{0t}(\boldsymbol{A}_t|\boldsymbol{A}_0) \right\|_2^2 \right\} \end{aligned} \tag{3}$$

where $\lambda_1(t)$ and $\lambda_2(t)$ are positive weight functions, t is uniformly sampled from [0,1]. For nodes, we have $\boldsymbol{X}_0 \sim p_0(\boldsymbol{X})$, $\boldsymbol{X}_t \sim p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0)$, and similarly for edges, we have $\boldsymbol{A}_0 \sim p_0(\boldsymbol{A})$, $\boldsymbol{A}_t \sim p_{0t}(\boldsymbol{A}_t|\boldsymbol{A}_0)$. Since $\mathbf{f}_{1,t}$ and $\mathbf{f}_{2,t}$ are affine, the transition kernels $p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0)$ and $p_{0t}(\boldsymbol{A}_t|\boldsymbol{A}_0)$ are always Gaussian distributions, and closed-form means and variances are obtained based on standard techniques. For example, the node transition kernel in VPSDE (Song et al., 2021) form is shown as follows:

$$p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0) = \mathcal{N}\left(\boldsymbol{X}_t; e^{-\frac{1}{4}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-\frac{1}{2}t\bar{\beta}_{\min}} \boldsymbol{X}_0, \mathbf{I} - \mathbf{I}e^{-\frac{1}{2}t^2(\bar{\beta}_{\max}-\bar{\beta}_{\min})-t\bar{\beta}_{\min}}\right). \tag{4}$$

For simplicity, the subsequent derivations only focus on $\boldsymbol{X}$, as the derivations for $\boldsymbol{A}$ are the same as those for $\boldsymbol{X}$.

3.2 REVERSE-STARTING BIAS

In this section, we use GDSS as the basic model and QM9 as the dataset to demonstrate the phenomenon of the reverse-starting bias and cleverly corroborate our motivation. We have two score networks: the first is a pretrained network $\boldsymbol{s}_{\theta,t}(\cdot)$ whose forward maximum perturbation is far from reaching standard normal; the second is a network $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$ whose forward maximum perturbation distribution is forced to be standard normal.

Figs. 2a and 2b show the ℓ_2 norm distribution of the predicted outputs of the two score networks at different time steps. Taking Fig. 2a as an example, at each step, we obtain perturbed samples through forward noising, then use $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$ to obtain the predicted score and calculate the corresponding ℓ_2 norm value. We present the details of the figure in Appendix B. At time step 0, the score ℓ_2 norm of the ground truth $\boldsymbol{X}_0$ spans approximately $(0, 2500)$, demonstrating the diversity of the original data and its scores. As the noise intensity increases, the range of the score ℓ_2 norm narrows, eventually stabilizing within $(0, 10)$. The evolution of the score ℓ_2 norm of perturbed samples at different time steps indicates that as the distribution approaches standard normal, the model becomes highly sensitive to score changes. The tightened score ℓ_2 norm also implies that slight perturbations in scores during the early sampling stages significantly affect generation performance. For Fig. 2b, the evolution pattern of $\boldsymbol{s}_{\theta,t}(\cdot)$ is consistent with that of $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$, but since the forward perturbation of $\boldsymbol{s}_{\theta,t}(\cdot)$ is far from reaching standard normal, its ℓ_2 norm range is wider, indicating a higher tolerance for score deviations.

Fig. 2c illustrates the response of the predicted outputs of the two score networks to perturbations at different time steps during the sampling phase. Each point in Fig. 2c represents a perturbation experiment.

The x-axis represents the addition of a standard normal noise perturbation to the score prediction output at the current time step, while the y-axis represents the final generation metric for this perturbation experiment (details in Appendix B). For $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$, at time step 0, we start from standard normal and perturb the predicted score at the current step. Subsequent sampling is not perturbed, ultimately resulting in a rather poor generation metric. As sampling progresses, the destructive effect of score perturbation on generation quality rapidly weakens and stabilizes after 200 steps. The evolution pattern of the score perturbation experiment indicates that diffusion models heavily rely on accurate scores in the early sampling stages, where score deviations severely impact generation quality. For $\boldsymbol{s}_{\theta,t}(\cdot)$, instead of using standard normal as the sampling starting point, we artificially use samples from the forward maximum perturbation distribution as the actual starting point to eliminate the reverse-starting bias.

The above two experiments demonstrate that diffusion models are highly sensitive to score deviations in high-noise states, while in low-noise states, their resistance to score deviations significantly increases. Notably, these experiments also provide us with two directions for addressing the reverse-starting bias: $\boldsymbol{s}_{\bar{\theta},t}(\cdot)$ suggests that we need to retrain and force the forward maximum perturbation distribution to be standard normal, while $\boldsymbol{s}_{\theta,t}(\cdot)$ implies that we need to explore a starting distribution aligned with the forward maximum perturbation distribution during the sampling phase. We find that the latter not only resolves the reverse-starting bias but also provides stronger tolerance to subsequent deviations.

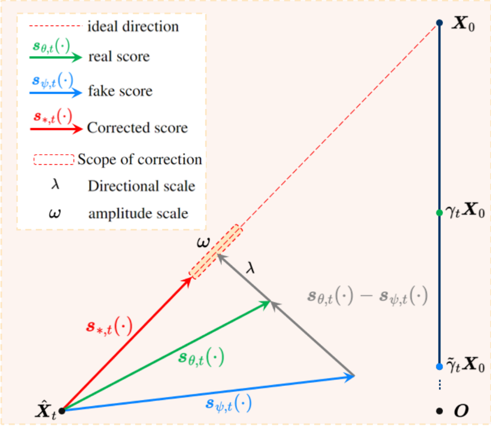

ideal direction
$\boldsymbol{s}_{\theta,t}(\cdot)$ real score
$\boldsymbol{s}_{\psi,t}(\cdot)$ fake score
$\boldsymbol{s}_{*,t}(\cdot)$ Corrected score
Scope of correction
λ Directional scale
ω amplitude scale
$\boldsymbol{X}_0$
$\gamma_t \boldsymbol{X}_0$
$\tilde{\gamma}_t \boldsymbol{X}_0$
$\boldsymbol{O}$
ω
λ
$\boldsymbol{s}_{\theta,t}(\cdot) - \boldsymbol{s}_{\psi,t}(\cdot)$
$\boldsymbol{s}_{*,t}(\cdot)$
$\boldsymbol{s}_{\theta,t}(\cdot)$
$\boldsymbol{s}_{\psi,t}(\cdot)$
$\hat{\boldsymbol{X}}_t$

Figure 3: Fractional correction based on the score difference. At the reverse sampling time step t, the ideal score always points to $\boldsymbol{X}_0$. $\boldsymbol{s}_{\theta,t}(\cdot)$ points to $\gamma_t \boldsymbol{X}_0$ with some deviation (partially containing $\boldsymbol{X}_0$), while $\boldsymbol{s}_{\psi,t}(\cdot)$ points to $\tilde{\gamma}_t \boldsymbol{X}_0$ with larger deviation (containing little $\boldsymbol{X}_0$). The difference between real and fake scores guides the real score towards the ideal score. We use λ to control this extent and β to adjust the magnitude of $\hat{s}_{\theta,t}(\cdot)$. The final corrected score flexibly approaches the real score within the dashed box.

4 METHODOLOGY

4.1 STABLE DISTRIBUTION

Langevin sampling is a key component of SDE-based diffusion models. Given sufficiently small step sizes and a large number of steps, Langevin sampling can utilize the score function to obtain samples from a probability distribution. Importantly, the prior distribution of Langevin sampling can be consistent with that of the diffusion model, typically a standard normal distribution. Moreover, we already have a pretrained score network $\boldsymbol{s}_{\theta,T}(\cdot) \approx \nabla \log q(\boldsymbol{X}_T|\boldsymbol{X}_0)$. This score guides Langevin sampling to obtain samples from the distribution $p(\hat{\boldsymbol{X}}_T) \approx q(\boldsymbol{X}_T|\boldsymbol{X}_0)$:

$$\hat{\boldsymbol{X}}_T^{i+1} \leftarrow \hat{\boldsymbol{X}}_T^i + \epsilon_T^i \boldsymbol{s}_{\boldsymbol{\theta}}(\hat{\boldsymbol{X}}_T^i, T) + \sqrt{2\epsilon_T^i}\boldsymbol{z}_T^i \tag{5}$$

where the subscript T represents the time step parameter of the diffusion model. In the presampling stage, we only use the score $\boldsymbol{s}_{\theta,T}(\cdot)$ at time T. The superscript i denotes the time step parameter of Langevin sampling, ϵ_T^i represents the step size at the current sampling step, and $\boldsymbol{z}_T^i$ is standard normal noise. After obtaining a batch of samples $\hat{\boldsymbol{X}}_T$ based on Eq. (5), we use $\hat{\boldsymbol{X}}_T$ as the new starting point for the reverse sampling process. We refer to this stage as the presampling stage.

4.2 BIAS CORRECTION METHOD

In theory, the presampling stage based on Eq. (5) can obtain samples from the distribution $q(\boldsymbol{X}_T|\boldsymbol{X}_0)$. However, the converged score network $\boldsymbol{s}_{\theta,T}(\cdot)$ can never access the true score $\nabla_{\boldsymbol{X}_T} \log q(\boldsymbol{X}_T|\boldsymbol{X}_0) = \frac{\boldsymbol{X}_T - \sqrt{\bar{\alpha}_T}\boldsymbol{X}_0}{1-\bar{\alpha}_T}$. We have to consider the exposure bias of the score network. Without loss of generality, we consider the predicted value of the score at any time step:

$$s_{\theta,t}(\hat{X}_t) = -\frac{\hat{X}_t - \sqrt{\bar{\alpha}_t}\hat{X}_0}{1-\bar{\alpha}_t}\,. \tag{6}$$

Clearly, it is challenging for the score network to analytically predict the original data X_0. We can rewrite this as:

$$\hat{X}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\big(\hat{X}_t + (1-\bar{\alpha}_t)s_{\theta,t}(\hat{X}_t)\big)\,. \tag{7}$$

Following Zhang et al. (2023), we model the estimate of $\hat{X}_0$ as:

$$\hat{X}_\theta = \gamma_t X_0 + \eta_t \epsilon_a \tag{8}$$

where $\eta_t < M$, $\epsilon_a \sim \mathcal{N}(\mathbf{0}, I)$, and for $0 \le j < k \le N$, we have $1 > \gamma_j > \gamma_k \ge 0, \quad 0 \le \eta_j < \eta_k$. Now, we use the diffusion model to generate a batch of samples $\tilde{X}_0$ and train a new score network using these as the original data. We know that $\tilde{X}_0$ always has exposure bias compared to X_0, so the score $s_{\phi,t}(\cdot)$ trained on $\tilde{X}_0$ naturally learns these exposure bias. Similarly, we define the estimation of the original data by the new network during the reverse sampling, $\hat{X}_\psi(\hat{X}_t, t) = \tilde{\gamma}_t X_0 + \tilde{\eta}_t \epsilon_b$, and we can easily see that $\tilde{\gamma}_t < \gamma_t$. Now we consider the score difference between the two scores at the same time step and for the same sample:

$$s_{\theta,t}(\hat{X}_t) - s_{\psi,t}(\hat{X}_t) = \sqrt{\bar{\alpha}_t}\frac{(\gamma_t - \tilde{\gamma}_t)X_0 + (\eta_t\epsilon_a - \tilde{\eta}_t\epsilon_b)}{1-\bar{\alpha}_t}\,. \tag{9}$$

We find that the score difference contains information about the original data. We aim to utilize this information. Inspired by classifier-free guidance (Ho & Salimans, 2021) and extrapolation operations (Zhang et al., 2023), we define a new score correction method based on Eq. (9),

$$\begin{aligned}\hat{s}_{\theta,t}(\hat{X}_t) &= s_{\theta,t}(\hat{X}_t) + \lambda\left(s_{\theta,t}(\hat{X}_t) - s_{\psi,t}(\hat{X}_t)\right)\\ &= -\frac{\hat{X}_t - \sqrt{\bar{\alpha}_t}\left((\gamma_t + \lambda(\gamma_t - \tilde{\gamma}_t))X_0 + \eta_t\epsilon_a + \lambda(\eta_t\epsilon_a - \tilde{\eta}_t\epsilon_b)\right)}{1-\bar{\alpha}_t}\end{aligned} \tag{10}$$

where $\lambda \ge 0$ represents the step size for correcting the score using the score difference, Eq. (9). When $\lambda = 0$, no correction is applied. Conceptually, the correction operation pulls the biased direction towards the unbiased direction. Although there is some noise in this correction direction, choosing appropriate parameters λ improve the accuracy of the score. Then, we divide the $\hat{s}_{\theta,t}(\hat{X}_t)$ by a scalar to adjust the magnitude of the score, further driving $\hat{s}_{\theta,t}(\hat{X}_t)$ closer to the true score:

$$\hat{s}_{\theta,t}(\hat{X}_t) = \hat{s}_{\theta,t}(\hat{X}_t)/\omega\,. \tag{11}$$

In particular, we emphasize that the score correction at time step T is far more important than at other time steps, as the score is directly related to the steady-state distribution of Langevin sampling, which is crucial for addressing initialization bias. Therefore, we recommend decoupling the correction parameter at time step T from those at other time steps during the actual score correction process.. Specifically, since we use Langevin sampling to obtain a distribution aligned with the forward maximum perturbation distribution, and this distribution is in a low-noise state, retaining some data information from X_0, we can shorten the sampling chain, significantly reducing the sampling time. Experimental validation is provided in §5.3.

We emphasize that utilizing Langevin sampling to obtain aligned samples and using the difference signal to correct scores are indispensable components for addressing the reverse-starting bias and the exposure bias. The effect is shown in Figure 1c. Additionally, we conduct extensive ablation experiments in §5.4 to demonstrate this point.We provide a detailed geometric illustration in Fig. 3 and provide detailed derivations and proofs of the formulas from §4 in Appendix C.

5 EXPERIMENTS

In this section, we select three generic graph datasets and two molecular datasets to evaluate the performance of our method. In order to demonstrate the broad applicability of this method in addressing the reverse-starting bias and mitigating exposure bias, we tested it on a variety of mainstream graph diffusion models, namely GDSS (Jo et al., 2022), GSDM (Luo et al., 2023), HGDM (Wen et al., 2024), and MOOD (Lee et al., 2023). Our improved model is prefixed with the basic diffusion model and denoted by S++ at its suffix. At the same time, we perform extensive downstream task testing and ablation study to further illustrate the effectiveness and necessity of S++.

5.1 GENERIC GRAPH GENERATION

Experimental Setup We selected three generic graph datasets to test our approach: (1) Community-small: 100 artificially generated graphs with community structure; (2) Enzymes: 600 protein maps representing the enzyme structure in the BRENDA database (Schomburg et al., 2004); (3) Grid: 100 standard 2D grid diagrams. To evaluate the quality of the generated graphs, we followed the practice of Jo, Lee, and Hwang (2022) and we used the Maximum Mean Difference (MMD) to compare the statistical distribution of the graphs between the same number of generated plots and the test plots, including the distribution of measured degrees, clustering coefficients, and the number of occurrences of the 4-node track.

Dataset Info.	Community-small Synthetic, $12 \leq \lvert V\rvert \leq 20$				Enzymes Real, $10 \leq \lvert V\rvert \leq 125$				Grid Synthetic, $100 \leq \lvert V\rvert \leq 400$			
Method	Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓
GDSS-OC	0.050	0.132	0.011	0.064	0.052	0.627	0.249	0.309	0.270	0.009	0.034	0.070
GDSS-OC-S++	**0.021**	**0.061**	**0.005**	**0.029**	**0.067**	**0.099**	**0.007**	**0.058**	**0.105**	**0.004**	**0.061**	**0.066**
GDSS-WC	0.045	0.088	0.007	0.045	0.044	0.069	0.002	0.038	0.111	0.005	0.070	0.070
GDSS-WC-S++	**0.019**	**0.062**	**0.004**	**0.028**	**0.031**	**0.050**	**0.003**	**0.028**	**0.105**	**0.004**	**0.061**	**0.057**
HGDM-OC	0.065	0.119	0.024	0.069	0.125	0.625	0.371	0.374	0.181	0.019	0.112	0.104
HGDM-OC-S++	**0.021**	**0.034**	**0.005**	**0.020**	**0.080**	**0.500**	**0.225**	**0.268**	**0.023**	**0.034**	**0.004**	**0.020**
HGDM-WC	0.017	0.050	0.005	0.024	0.045	0.049	0.003	0.035	0.137	0.004	0.048	0.069
HGDM-WC-S++	**0.021**	**0.024**	**0.004**	**0.016**	**0.040**	**0.041**	**0.005**	**0.029**	**0.123**	**0.003**	**0.047**	**0.058**
GSDM-OC	0.142	0.230	0.043	0.138	0.930	0.867	0.168	0.655	1.996	0.0	1.013	1.003
GSDM-OC-S++	**0.011**	**0.016**	**0.001**	**0.009**	**0.012**	**0.087**	**0.011**	**0.037**	**1.2e-4**	**0.0**	**1.2e-4**	**0.066**
GSDM-WC	0.011	0.016	0.001	0.009	0.013	0.088	0.013	0.038	0.002	0.0	0	7.2e-5
GSDM-WC-S++	**0.011**	**0.016**	**0.001**	**0.009**	**0.011**	**0.086**	**0.010**	**0.036**	**5.0e-5**	**0.0**	**1.1e-5**	**0.066**

Table 1: Generation results on the generic graph datasets (Lower is better). The results of the Enzymes dataset of GDSS are reproduced by ourselves, and the results of other baselines are all from published papers, and we give detailed settings and instructions in the appendix D.

Results Table 1 shows that S++ significantly outperforms all baseline models. For the uncorrected sampling method, the performance indicators of the baseline model are particularly poor due to the existence of the reverse-starting bias and score exposure bias, while S++ can significantly improve the performance of all baseline models and reach or even exceed the level of the baseline model with correctors. Because the method without correctors can significantly reduce the computational consumption, we believe that S++ can really release the ability of the graph diffusion model, which is enlightening for large-scale datasets. For the sampling method with aligners, S++ is still significantly better than all baseline models. At the same time, we also give experimental comparisons of other advanced models in the appendix F, and the results show that S++ can achieve the SOTA indicators of the corresponding tasks.

5.2 MOLECULAR GRAPH GENERATION

Experimental Setup We selected two widely recognized molecular datasets to evaluate our methods: QM9 (Ramakrishnan et al., 2014) and ZINC250k (Irwin et al., 2012). We generated 10,000 molecules and selected the following widely used evaluation metrics: Frechet ChemNet Distance (FCD) (Preuer et al., 2018), Neighborhood subgraph pairwise distance kernel (NSPDK) MMD (Costa & Grave, 2010), validity w/o correction, and the generation time. (1) FCD uses the activation of the penultimate layer of ChemNet to calculate the distance between the benchmark molecular dataset and the generated dataset to characterize the similarity between the two, and the lower the FCD value, the higher the similarity between the two distributions. (2) (NSPDK) MMD considered the characteristics of nodes and edges at the same time, and calculated the MMD between the benchmark molecular dataset and the generated dataset; (3) Sampling time is used to evaluate the rapidity of the model in generating large-scale molecular datasets, and we only count the time spent on sampling, regardless of the time spent on preprocessing and evaluation.

Results Table 2 shows that both in terms of sampling time and generation quality, S++ is significantly better than the baseline model. For the sampling method without correctors, due to the existence of the reverse-starting bias and score exposure bias, the quality of generation from the baseline model is particularly poor, while S++ can significantly improve the performance of all baselines and approximate the sampling methods with correctors of the baseline model. For the sampling method with aligners, S++ is still significantly better than all baseline models and greatly reduces the sampling time. At the same time, we provide more comparative experimental results in appendix F.

Method	QM9			ZINC250k		
	Sampling time ↓	NSPDK MMD ↓	FCD ↓	Sampling time ↓	NSPDK MMD ↓	FCD ↓
GDSS-OC	$0.73e^2$	0.016	4.584	$0.73e^3$	0.047	20.53
GDSS-OC-S++	**5.10**	**0.001**	**1.661**	**0.70**e^3	**0.050**	**16.79**
GDSS-WC	$1.61e^2$	0.004	2.550	$1.41e^3$	0.019	14.66
GDSS-WC-S++	**9.25**	**0.001**	**1.661**	**0.98e**e^3	**0.012**	**12.70**
HGDM-OC	$0.62e^2$	0.005	3.164	$0.76e^3$	0.033	21.38
HGDM-OC-S++	**0.62e**^{2}	**0.003**	**2.512**	**0.77**e^3	**0.034**	**20.79**
HGDM-WC	$1.16e^2$	0.002	2.147	$1.52e^3$	0.016	17.69
HGDM-WC-S++	**0.98e**^{2}	**0.001**	**2.001**	**1.17e**^{3}	**0.016**	**16.24**

Table 2: Comparison of different methods on QM9 and ZINC250k datasets.

5.3 DIVERSITY GENERATION

Characteristic molecule generation To evaluate the performance of S++ in generating novel, drug-like, and synthesizable molecules, we follow (Lee et al., 2023) and assess S++ in the five docking score (DS) optimization tasks under the quantitative estimate of synthetic accessibility (SA), drug-likeness (QED) and novelty constraints. We define the property Y by

$$Y(\boldsymbol{G}) = \widehat{\mathrm{DS}}(\boldsymbol{G}) \times \mathrm{QED}(\boldsymbol{G}) \times \widehat{\mathrm{SA}}(\boldsymbol{G}) \in [0, 1] \tag{12}$$

where $\widehat{\mathrm{DS}}$ refers to the normalized docking score, $\widehat{\mathrm{SA}}$ denotes the normalized synthetic accessibility, and QED represents drug-likeness. We used MOOD-S++ to generate 3000 molecules and evaluate performance using the following metrics. **Novel hit ratio (%)** is the fraction of unique hit molecules whose maximum Tanimoto similarity with the training molecules is less than 0.4. In particular, hit molecules are defined as the molecules that satisfy the following conditions: DS $<$ (the median DS of the known active molecules), QED > 0.5, and SA < 5. **Novel top 5% docking score** refers to the average DS of the top 5% unique molecules that satisfy the constraints QED > 0.5 and SA < 5 and their maximum similarity with the training molecules is below 0.4. To avoid bias in target selection, we utilize five protein targets: parp1, fa7, 5ht1b, braf, and jak2.

Results Tables 3 and 4 show that MOOD-S++ is significantly better than baseline in all target proteins. This indicates that S++ still has advantages in the discovery of drug-like, synthesizable, and novel molecular tasks with high binding affinity, and it can be seen that the reverse-starting bias and exposure bias pose a significant threat to various generation tasks.

Accelerate generation To demonstrate that S++ can generate good samples faster by using fewer steps of reverse diffusion, We chose GDSS-OC as the benchmark model, and QM9 and Comm datasets were selected to test the performance of our method and benchmark model at different sampling total time steps.

Results Table 5 shows that S++ is significantly better than the baseline model at different sampling total time steps. S++ was not only able to generate samples with fewer reverse diffusion steps, but also achieved consistent improvements across generation metrics, especially on the QM9 dataset, where S++ remained close to optimal performance even with a significant reduction in the sampling time step ($T = 100$), while the performance of the benchmark model decreased significantly.

In conclusion, S++ shows higher efficiency, better quality, and stronger robustness in graph generative tasks, which provides a powerful improvement scheme for the application of diffusion model.

Method	Target protein				
	parp1	fa7	5ht1b	braf	jak2
MOOD	7.017 (± 0.428)	0.733 (± 0.141)	18.673 (± 0.423)	5.240 (± 0.285)	9.200 (± 0.524)
MOOD-S++	**8.286 (± 0.214)**	**0.900 (± 0.068)**	**20.354 (± 0.672)**	**5.653 (± 0.073)**	**9.167 (± 0.067)**

Table 3: Novel hit ratio (%) results (↑).

Method	Target protein				
	parp1	fa7	5ht1b	braf	jak2
MOOD	-10.865 (± 0.113)	-8.160 (± 0.071)	-11.145 (± 0.042)	-11.063 (± 0.034)	-10.147 (± 0.060)
MOOD-S++	**-10.961 (± 0.027)**	**-8.182 (± 0.028)**	**-11.231 (± 0.036)**	**-11.143 (± 0.025)**	**-10.163 (± 0.015)**

Table 4: Novel top 5% docking score (kcal/mol) results (↓).

T	Method	QM9			Community-small			
		Val. w/o corr. ↑	NSPDK MMD ↓	FCD ↓	Deg.↓	Clus. ↓	Orbit ↓	Avg. ↓
1000	GDSS-OC	73.5	0.015	4.584	0.050	0.132	0.011	0.064
	GDSS-OCS++	**94.0**	**0.001**	**1.671**	**0.021**	**0.061**	**0.005**	**0.029**
500	GDSS-OC	46.2	0.045	7.960	0.136	0.456	0.151	0.248
	GDSS-OC-S++	**93.9**	**0.001**	**1.665**	**0.029**	**0.142**	**0.008**	**0.060**
100	GDSS-OC	37.8	0.069	9.951	0.092	0.666	0.394	0.384
	GDSS-OC-S++	**93.9**	**0.001**	**1.663**	**0.061**	**0.414**	**0.140**	**0.205**

Table 5: Comparison of different methods on QM9 and ZINC250k datasets under different total sampling time steps.

Method	QM9		
	Val. w/o corr. ↑	NSPDK MMD ↓	FCD ↓
GDSS-OC	73.5	0.0157	4.58
GDSS-w/o Score Correction	94.8	0.0037	2.65
GDSS-w/o Langevin Alignment	89.8	0.0031	2.01
GDSS-OC-S++	**94.0**	**0.0014**	**1.67**

Table 6: Ablation experiments on the OM9 dataset.

5.4 ABLATION STUDY

Table 6 clearly demonstrates the effectiveness and necessity of S++. Experimental results show that GDSS-w/o Langevin Alignment or Langevin Alignment alone can improve the performance of the baseline model to varying degrees, however, when we combine these two methods, the model performance is significantly improved, which strongly proves the effectiveness and necessity of the combination of the two methods, and their synergistic effect. The quality of graph generation has been greatly improved tasks.

6 CONCLUSION

In this paper, we use Langevin sampling to obtain samples aligned with the forward maximum perturbation distribution, which solves the reverse-starting bias and greatly alleviates the exposure bias of the fraction network, and we propose a fraction correction mechanism based on score difference to further promote the stable-state distribution of Langevin sampling to the real forward maximum perturbation distribution, and further alleviate the exposure bias of the fraction network. Our approach does not require network modifications or the introduction of new components, and can be naturally integrated into existing graph diffusion models to achieve state-of-the-art metrics on multiple datasets and multiple tasks.

REFERENCES

Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *ICML*, pp. 255–262, 2010.

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. In *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Martin Nicolas Everaert, Athanasios Fitsios, Marco Bocchio, Sami Arpa, Sabine Süsstrunk, and Radhakrishna Achanta. Exploiting the signal-leak bias in diffusion models. In *WACV*, pp. 4025–4034, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

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

Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Conditional diffusion based on discrete graph structures for molecular graph generation. In *AAAI*, 2023.

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*, pp. 1757–1768, 2012.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *ICML*, pp. 10362–10383, 2022.

Dongjun Kim, Yeongmin Kim, Se Jung Kwon, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. In *ICML*, 2023.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *ICML*, 2023.

Mingxiao Li, Tingyu Qu, Ruicong Yao, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. In *ICLR*, 2024.

Yangming Li and Mihaela van der Schaar. On error propagation of diffusion models. In *ICLR*, 2023.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *WACV*, 2024.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In *NeurIPS*, 2019.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. In *Energy Based Models Workshop-ICLR 2021*, 2021.

Tianze Luo, Zhanfeng Mo, and Sinno Jialin Pan. Fast graph generation via spectral diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. In *ICML*, 2023.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *ICLR*, 2024.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, 2020.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, pp. 1736–1741, 2018.

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

Zhiyao Ren, Yibing Zhan, Liang Ding, Gaoang Wang, Chaoyue Wang, Zhongyi Fan, and Dacheng Tao. Multi-step denoising scheduled sampling: Towards alleviating exposure bias for diffusion models. In *AAAI*, pp. 4667–4675, 2024.

Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, pp. D431–D433, 2004.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a flow-based autoregressive model for molecular graph generation. In *ICLR*, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*, pp. 412–422, 2018.

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

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

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *ICLR*, 2023.

Lingfeng Wen, Xuan Tang, Mingjie Ouyang, Xiangxiang Shen, Jian Yang, Daxin Zhu, Mingsong Chen, and Xian Wei. Hyperbolic graph diffusion model. In *AAAI*, pp. 15823–15831, 2024.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: a geometric diffusion model for molecular conformation generation. In *ICLR*, 2022.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, 2018.

Guoqiang Zhang, Kenta Niwa, and W Bastiaan Kleijn. Lookahead diffusion probabilistic models for refining mean estimation. In *CVPR*, 2023.

A REVERSE-STARTING BIAS

In this section, we provide a detailed discussion of the initialization bias in diffusion models. It is worth noting that these diffusion models are based on diffusion models defined by SDE (Song et al., 2021). For VPSDE, the diffusion model obtains perturbed samples through $p_{0i}(\boldsymbol{X}_i|\boldsymbol{X}_0) = \mathcal{N}(\boldsymbol{X}_i; \sqrt{\alpha_i}\boldsymbol{X}_0, (1-\alpha_i)\mathbf{I})$, where $\alpha_i := \prod_{j=1}^{i}(1-\beta_j)$. When this expression is extended continuously, it leads to Eq. (4), which corresponds to Equation (33) in the SDE. At $t = 1$, Eq. (4) gives the maximum perturbation distribution, which is $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Similarly, for VESDE, the diffusion model obtains perturbed samples through $p_{0i}(\boldsymbol{X}_i|\boldsymbol{X}_0) = \mathcal{N}(\boldsymbol{X}_i; \boldsymbol{X}_0, \sigma_i\mathbf{I})$, where $\sigma_{\min} = \sigma_1 < \sigma_2 < ... < \sigma_N = \sigma_{\max}$. When this expression is extended continuously,

$$p_{0t}(\boldsymbol{X}_t|\boldsymbol{X}_0) = \mathcal{N}\Big(\boldsymbol{X}_t; \boldsymbol{X}_0, \sigma_{\min}^2\Big(\frac{\sigma_{\max}}{\sigma_{\min}}\Big)^{2t}\mathbf{I}\Big), \tag{13}$$

it corresponds to Eq. (31) of the paper of SDE (Song et al., 2021). At $t = 1$ Eq. (13) achieves the maximum perturbation distribution, which is $\mathcal{N}(\boldsymbol{X}_i; \boldsymbol{X}_0, \sigma_{\max}\mathbf{I})$. In particular, we need to make sure that $\sigma_{\max}$ is large enough that $\mathcal{N}(\boldsymbol{X}_i; \boldsymbol{X}_0, \sigma_{\max}\mathbf{I}) \approx \mathcal{N}(\boldsymbol{X}_i; \mathbf{0}, \sigma_{\max}\mathbf{I})$.

However, in practice, Lots of diffusion models (Jo et al., 2022; Luo et al., 2023; Wen et al., 2024) adopted a rather conservative strategy when training the network. For VPSDE, this results in the maximum forward perturbation distribution being $\mathcal{N}(\mu_T\boldsymbol{x}_0, \sigma_T\mathbf{I})$, which is far from reaching $\mathcal{N}(\mathbf{0}, \mathbf{I})$. For VESDE, due to $\sigma_{\max}$ not being large enough, the maximum forward perturbation distribution is $\mathcal{N}(\boldsymbol{X}_i; \boldsymbol{X}_0, \sigma_{\max}\mathbf{I})$, which cannot be approximated by $\mathcal{N}(\boldsymbol{X}_i; \mathbf{0}, \sigma_{\max}\mathbf{I})$. However, GDSS et al. always start reverse sampling from the standard normal distribution, which leads to significant initialization bias. A detailed comparison of the parameters is shown in Tables 7, 8, and 9.

Model	GDSS									
Dataset	Community-small		Enzymes		Grid		QM9		ZINC250k	
Type	Node	Edge	Node	Edge	Node	Edge	Node	Edge	Node	Edge
SDE	VPSDE	VPSDE	VPSDE	VESDE	VPSDE	VPSDE	VESDE	VESDE	VPSDE	VESDE
$\beta_{\min}$	0.1	0.1	0.1	0.2	0.1	0.2	0.1	0.1	0.1	0.2
$\beta_{\max}$	1.0	1.0	1.0	1.0	1.0	0.8	1.0	1.0	1.0	1.0
μ_T	0.7596	0.7596	0.7596	1.0	0.7596	0.7788	1.0	1.0	0.7596	1.0
σ_T^2	0.4231	0.4231	0.4231	1.0	0.4231	0.3935	1.0	1.0	0.4231	1.0

Table 7: The actual parameters of the forward perturbation of the GDSS.

Model	HGDM									
Dataset	Community-small		Enzymes		Grid		QM9		ZINC250k	
Type	Node	Edge	Node	Edge	Node	Edge	Node	Edge	Node	Edge
SDE	VPSDE	VPSDE	VPSDE	VESDE	VPSDE	VESDE	VPSDE	VESDE	VPSDE	VESDE
$\beta_{\min}$	0.1	0.1	0.1	0.2	0.1	0.2	0.1	0.1	0.1	0.2
$\beta_{\max}$	1.0	1.0	1.0	1.0	7.0	0.8	2.0	1.0	1.0	1.0
μ_T	0.7596	0.7596	0.7596	1.0	0.1695	1.0	0.5916	1.0	0.7596	1.0
σ_T^2	0.4231	0.4231	0.4231	1.0	0.9713	0.64	0.6501	1.0	0.4231	1.0

Table 8: The actual parameters of the forward perturbation of the HGDM.

Model	GSDM						MOOD			
Dataset	Community-small		Enzymes		Grid		QM9		ZINC250k	
Type	Node	Eigen	Node	Eigen	Node	Eigen	Node	Edge	Node	Edge
SDE	VPSDE	VPSDE	VPSDE	VPSDE	VPSDE	VPSDE	VPSDE	VESDE	VPSDE	VESDE
$\beta_{\min}$	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.2
$\beta_{\max}$	1.0	1.0	1.0	1.0	1.0	0.8	1.0	1.0	1.0	1.0
μ_T	0.7596	0.7596	0.7596	0.7596	0.7596	0.7788	1.0	1.0	0.7596	1.0
σ_T^2	0.4231	0.4231	0.4231	0.4231	0.4231	0.3935	1.0	1.0	0.4231	1.0

Table 9: The actual parameters of the forward perturbation of the GSDM and MOOD.

B FIGURE DETAILS

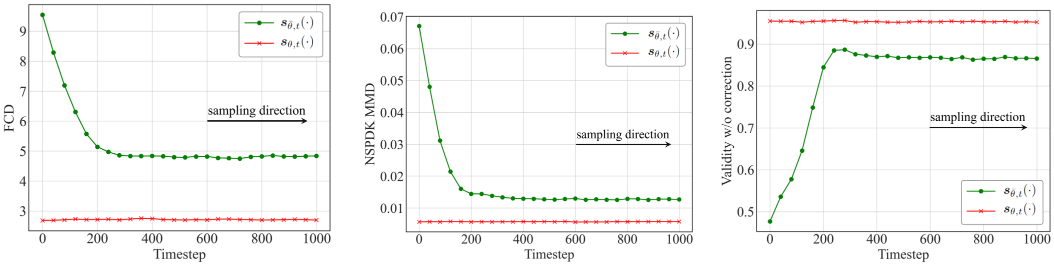
In this section, we present the detailed procedures to plot Fig. 2. Let $\boldsymbol{s}_{\theta,t}(\cdot)$ represent the GDSS pretrained score network. Due to the conservative strategy of GDSS, with $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 1$, the maximum perturbation distribution is $\mathcal{N}\left(0.7596\boldsymbol{X}_0, 0.4231\mathbf{I}\right)$ at $t = 1$. On the other hand, $\boldsymbol{s}_{\psi,t}(\cdot)$ is defined with the forced constraints of $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 20$. At $t = 1$, the maximum perturbation distribution is $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, $\boldsymbol{s}_{\theta,t}(\cdot)$ and $\boldsymbol{s}_{\psi,t}(\cdot)$ not only represent two different networks, but also indicate that their maximum perturbation distributions during the training phases are completely different.

To plot Figs. 2a and 2b, we freeze the converged $\boldsymbol{s}_{\theta,t}(\cdot)$ and $\boldsymbol{s}_{\psi,t}(\cdot)$, then replace $\boldsymbol{X}$ in Eq. (3) with $\boldsymbol{A}$ to obtain perturbation samples of 1024 edges at different timesteps. We then compute $\|\boldsymbol{s}_{\theta,t}(\cdot)\|_2$ and $\|\boldsymbol{s}_{\psi,t}(\cdot)\|_2$ and plot them on the figure.

To plot Fig. 2c, we introduce perturbations to $\boldsymbol{s}_{\theta,t}(\cdot)$ at different timesteps during the sampling phase. We employ a sampling method without a corrector and perturb the score at the selected timestep (horizontal axis) using Gaussian noise:

$$\boldsymbol{s}_{\theta,t}(\cdot) = \boldsymbol{s}_{\theta,t}(\cdot) + \boldsymbol{z}_t \tag{14}$$

where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For the other timesteps, we do not introduce any perturbations, allowing the diffusion model to perform sampling and record the generation metrics. We conduct the perturbation experiment on $\boldsymbol{s}_{\theta,t}(\cdot)$ using the same method, and ultimately compare the results of the two perturbation experiments based on the timesteps to evaluate how different score networks in the diffusion model resist bias at various timesteps. We present a detailed comparison of the generation metrics from the perturbation experiments, as shown in Fig. 4.



(a) FCD (b) NSPDK (c) Validity w/o correction

Figure 4: Generation metric responses to perturbations at different timesteps for two-score networks.

C DERIVATIONS FOR §4.2

For a diffusion model, let the original data be $\boldsymbol{X}_0$, and the pretrained score network be $\boldsymbol{s}_{\boldsymbol{\theta},t}(\cdot)$. Based on the set noise addition method, we have:

$$\nabla_{\boldsymbol{X}_t} \log q(\boldsymbol{X}_t|\boldsymbol{X}_0) = -\frac{\boldsymbol{X}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{X}_0}{1 - \bar{\alpha}_t} \tag{15}$$

However, $\boldsymbol{s}_{\boldsymbol{\theta},t}(\cdot)$ often deviates from the ideal logarithmic gradient. In the reverse process, assuming the current time step t has a data state $\hat{\boldsymbol{X}}_t$, the score network's predicted output is:

$$\boldsymbol{s}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) = -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{X}}_0}{1 - \bar{\alpha}_t} \tag{16}$$

Since it's difficult for the trained score to analytically predict $\boldsymbol{X}_0$. We model $\hat{\boldsymbol{X}}_0$ by:

$$\hat{\boldsymbol{X}}_0 = \gamma_t \boldsymbol{X}_0 + \eta_t \boldsymbol{\epsilon}_a \tag{17}$$

Eq. (16) becomes:

$$\boldsymbol{s}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) = -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}(\gamma_t \boldsymbol{X}_0 + \eta_t \boldsymbol{\epsilon}_a)}{1 - \bar{\alpha}_t} \tag{18}$$

Now, we train a new score network $\boldsymbol{s}_{\boldsymbol{\psi},t}(\cdot)$ based on the generated data $\bar{\boldsymbol{X}}$ from the score network. Following the above derivation, we can write the predicted score at the current time step t as:

$$\boldsymbol{s}_{\boldsymbol{\psi},t}(\hat{\boldsymbol{X}}_t) = -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}\tilde{\mathbf{x}}_0}{1-\bar{\alpha}_t} \tag{19}$$

Due to the bias of $\boldsymbol{s}_{\boldsymbol{\theta},t}(\cdot)$, the generated data $\boldsymbol{X}_1$ always deviates from $\boldsymbol{X}_0$, and considering the prediction error of the network, we can easily obtain:

$$\tilde{\boldsymbol{X}}_0 = \tilde{\gamma}_t \boldsymbol{X}_0 + \tilde{\eta}_t \boldsymbol{\epsilon}_b \tag{20}$$

where $\tilde{\gamma}_t < \gamma_t$, $\tilde{\eta}_t > \eta_t$, meaning that at the same reverse time step t, $\boldsymbol{s}_{\boldsymbol{\psi},t}(\cdot)$ trained on generated data has a larger bias in predicting the target distribution than $\boldsymbol{s}_{\boldsymbol{\theta},t}(\cdot)$ trained on original data. We can rewrite $\boldsymbol{s}_{\boldsymbol{\psi},t}(\cdot)$ as:

$$\boldsymbol{s}_{\boldsymbol{\psi},t}(\hat{\boldsymbol{X}}_t) = -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}(\tilde{\gamma}_t \boldsymbol{X}_0 + \tilde{\eta}_t \boldsymbol{\epsilon}_b)}{1-\bar{\alpha}_t} \tag{21}$$

Next, we derive the meaning of the score difference, defined as Eq. (18) minus Eq. (21):

$$\begin{aligned}
\boldsymbol{s}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) - \boldsymbol{s}_{\boldsymbol{\psi},t}(\hat{\boldsymbol{X}}_t) &= -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}(\gamma_t \boldsymbol{X}_0 + \eta_t \boldsymbol{\epsilon}_a)}{1-\bar{\alpha}_t} - \left(-\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}(\tilde{\gamma}_t \boldsymbol{X}_0 + \tilde{\eta}_t \boldsymbol{\epsilon}_b)}{1-\bar{\alpha}_t}\right) \\
&= \sqrt{\bar{\alpha}_t}\frac{(\gamma_t - \tilde{\gamma}_t)\boldsymbol{X}_0 + (\eta_t \boldsymbol{\epsilon}_a - \tilde{\eta}_t \boldsymbol{\epsilon}_b)}{1-\bar{\alpha}_t}
\end{aligned} \tag{22}$$

We add this score difference as a correction term to the original predicted score and introduce a hyperparameter to control the influence of the original and the noise scores:

$$\begin{aligned}
\hat{\boldsymbol{s}}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) &= \boldsymbol{s}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) + \lambda\left(\boldsymbol{s}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) - \boldsymbol{s}_{\boldsymbol{\psi},t}(\hat{\boldsymbol{X}}_t)\right) \\
&= -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}\left(\gamma_t \boldsymbol{X}_0 + \eta_t \boldsymbol{\epsilon}_a + \lambda(\gamma_t - \tilde{\gamma}_t)\boldsymbol{X}_0 + \lambda(\eta_t \boldsymbol{\epsilon}_a - \tilde{\eta}_t \boldsymbol{\epsilon}_b)\right)}{1-\bar{\alpha}_t} \\
&= -\frac{\hat{\boldsymbol{X}}_t - \sqrt{\bar{\alpha}_t}\left((\gamma_t + \lambda(\gamma_t - \tilde{\gamma}_t))\boldsymbol{X}_0 + \eta_t \boldsymbol{\epsilon}_a + \lambda(\eta_t \boldsymbol{\epsilon}_a - \tilde{\eta}_t \boldsymbol{\epsilon}_b)\right)}{1-\bar{\alpha}_t}
\end{aligned} \tag{23}$$

Because $\tilde{\gamma}_t < \gamma_t$, this score difference helps the original score add more information from the original data $\boldsymbol{X}_0$. By setting an appropriate hyperparameter λ, we can always use the information from the original score to guide the correction of the score. Finally, we add a coefficient of adjustment amplitude to the score corrected based on the score difference to further promote the prediction error to approximate the true score.

$$\hat{\boldsymbol{s}}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t) = \hat{\boldsymbol{s}}_{\boldsymbol{\theta},t}(\hat{\boldsymbol{X}}_t)/\omega \tag{24}$$

We theoretically prove that the score difference helps to correct the score.

D DETAILS FOR EXPERIMENT

We provide detailed parameters for experiments related to §5, as shown in Tables 10 and 11. In particular, we differentiate the relevant parameters for sampling methods with correctors and those without correctors.

E SAMPLING ALGORITHM

In this subsection, we present the sampling algorithm procedure for S++, as shown in algorithm1. Additionally, our method can be naturally integrated into the reverse sampling of various diffusion models, greatly improving the generation quality of sampling methods without corrector. For methods with corrector, we can significantly reduce the correction time interval by introducing a truncation time step. Specifically, we only apply the corrector when the time step exceeds t_c further reducing computational cost.t

Algorithm 1 The S++ sampling algorithm.

```
Input: pretrained real diffusion model $\boldsymbol{s}_{\theta,t}(\cdot)$; Trained fake diffusion model $\boldsymbol{s}_{\psi,t}(\cdot)$; Correction step size $\lambda$; Cut-off time $t_c$
Initialize: $\boldsymbol{X}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $j = 1$ to $M$ do
    $\boldsymbol{s}_{\theta,N}(\boldsymbol{X}_N) \leftarrow \Big(\boldsymbol{s}_{\boldsymbol{\theta},N}(\boldsymbol{X}_N) + \lambda_1(\boldsymbol{s}_{\boldsymbol{\theta},N}(\boldsymbol{X}_N) - \boldsymbol{s}_{\boldsymbol{\psi},N}(\boldsymbol{X}_N))\Big)/\omega_1$
    $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{X}_N \leftarrow \boldsymbol{X}_N + \boldsymbol{\epsilon}_N \boldsymbol{s}_{\theta,N}(\boldsymbol{X}_N) + \sqrt{2\boldsymbol{\epsilon}_N}\boldsymbol{z}_N$
end for
for $i = N - 1$ to $0$ do
    $\boldsymbol{s}_{\theta,i}(\boldsymbol{X}_i) \leftarrow \Big(\boldsymbol{s}_{\boldsymbol{\theta},i}(\boldsymbol{X}_i) + \lambda_2(\boldsymbol{s}_{\boldsymbol{\theta},i}(\boldsymbol{X}_i) - \boldsymbol{s}_{\boldsymbol{\psi},i}(\boldsymbol{X}_i))\Big)/\omega_2$
    $\boldsymbol{X}'_{i-1} \leftarrow (2 - \sqrt{1-\beta_i})\boldsymbol{X}_i + \beta_i \boldsymbol{s}_{\theta,i}(\boldsymbol{X}_i)$
    $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{X}_i \leftarrow \boldsymbol{X}'_i + \sqrt{\beta_{i+1}}\boldsymbol{z}$
    if $i \leq t_c$ then
        for $j = 1$ to $M'$ do
            $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
            $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \beta_i \boldsymbol{\epsilon}_i \boldsymbol{s}_{\theta,i+1}(\boldsymbol{X}_{i+1}) + \sqrt{2\boldsymbol{\epsilon}_i}\boldsymbol{z}$
        end for
    end if
end for
return $\boldsymbol{X}_0$
```

Model	Hyper.	Comm.	Enzymes	Grid	QM9	ZINC250k
GDSS-OC-S++	M	400	420	350	400	400
	λ_1	0.2	0.0008	0.06	1.19	2.5
	ω_1	0.998	1.0	1.0	1.09	1.0
	λ_2	0	0	0	0	0
	ω_2	1.0	1.0	1.0	1.0	1.0
HGDM-OC-S++	M	280	310	280	240	220
	λ_1	0.02	0.0	0.02	0.1	0.025
	ω_1	1.0	1.0	1.0	1.0	1.07
	λ_2	0.36	0.0	0.0	0.36	0.0
	ω_2	1.0	1.0	1.0	0.78	1.0
GSDM-OC-S++	M	200	400	400	-	-
	λ_1	0.0	0.0	0.0	-	-
	ω_1	1.0	1.0	1.0	-	-
	λ_2	0.0	0.0	0.0	-	-
	ω_2	1.0	1.0	1.0	-	-

Table 10: Experimental parameters for sampling methods with and without a corrector (OC).

F ADDITIONAL EXPERIMENTS

To demonstrate the superiority of S++, we selected generative models other than diffusion models as baseline models for comparison. GraphVAE (Simonovsky & Komodakis, 2018) is a graph generation model based on variational autoencoders; DeepGMG (Li et al., 2018) is a deep generative model that generates graphs in a sequential, pnode-by-node manner; GraphAF (Shi et al., 2020) is an autoregressive flow-based model. GraphRNN (You et al., 2018) is an autoregressive model using recurrent neural networks to generate graphs; EDP-GNN (Niu et al., 2020) is a score-based generative model using energy-based dynamics. GraphEBM (Liu et al., 2021) is an energy-based generative model that generates molecules by minimizing energy through Langevin dynamics, which is categorized as a one-shot generative method. We provide detailed comparative experiments in Tables 12 and 13, and the results show that our method significantly outperforms the baseline models and other generative models.

Model	Hyper.	Comm.	Enzymes	Grid	QM9	ZINC250k
GDSS-WC-S++	M	400	420	350	400	400
	λ_1	0.2	0.0008	0.06	1.19	2.5
	ω_1	0.998	1.0	1.0	1.09	1.0
	t_c	0.2	0.45	0.055	0.1	0.4
	λ_2	0	0	0	0	0
	ω_2	1.0	1.0	1.0	1.0	1.0
HGDM-WC-S++	M	280	200	360	240	220
	λ_1	0.02	0.0	0.18	0.1	0.25
	ω_1	1.0	1.0	1.0	1.0	1.07
	t_c	0.2	0.5	0.1	0.65	0.6
	λ_2	0.36	0.0	0.0	0.0	0.0
	ω_2	1.0	1.0	1.0	1.44	0.87
GSDM-WC-S++	M	200	400	400	-	-
	λ_1	0.0	0.0	0.0	-	-
	ω_1	1.0	1.0	1.0	-	-
	t_c	0.05	0.70	0.45	-	-
	λ_2	0.0	0.0	0.0	-	-
	ω_2	1.0	1.0	1.0	-	-

Table 11: Experimental parameters for sampling methods with and without a corrector (WC).

Dataset	Community-small				Enzymes				Grid			
Info.	Synthetic, $12 \le \lvert V\rvert \le 20$				Real, $10 \le \lvert V\rvert \le 125$				Synthetic, $100 \le \lvert V\rvert \le 400$			
Method	Deg.	Clus.	Orbit	Avg.	Deg.	Clus.	Orbit	Avg.	Deg.	Clus.	Orbit	Avg.
DeepGMG	0.220	0.950	0.400	0.523	-	-	-	-	-	-	-	-
GraphRNN	0.080	0.120	0.040	0.080	0.017	0.062	0.046	0.042	0.064	0.043	0.021	0.043
GraphAF	0.18	0.20	0.02	0.133	1.669	1.283	0.266	1.073	-	-	-	-
GraphDF	0.06	0.12	0.03	0.070	1.503	1.061	0.202	0.922	-	-	-	-
GraphVAE	0.350	0.980	0.540	0.623	1.369	0.629	0.191	0.730	1.619	0.0	0.919	0.846
EDP-GNN	0.053	0.144	0.026	0.074	0.023	0.268	0.082	0.124	0.455	0.238	0.328	0.340
GDSS-OC	0.050	0.132	0.011	0.064	0.052	0.627	0.249	0.309	0.270	0.009	0.034	0.070
GDSS-OC-S++	**0.021**	**0.061**	**0.005**	**0.029**	**0.067**	**0.099**	**0.007**	**0.058**	**0.105**	**0.004**	**0.061**	**0.066**
GDSS-WC	0.045	0.088	0.007	0.045	0.044	0.069	0.002	0.038	0.111	0.005	0.070	0.070
GDSS-WC-S++	**0.019**	**0.062**	**0.004**	**0.028**	**0.031**	**0.050**	**0.003**	**0.028**	**0.105**	**0.004**	**0.061**	**0.057**
HGDM-OC	0.065	0.119	0.024	0.069	0.125	0.625	0.371	0.374	0.181	0.019	0.112	0.104
HGDM-OC-S++	**0.021**	**0.034**	**0.005**	**0.020**	**0.080**	**0.500**	**0.225**	**0.268**	**0.023**	**0.034**	**0.004**	**0.020**
HGDM-WC	0.017	0.050	0.005	0.024	0.045	0.049	0.003	0.035	0.137	0.004	0.048	0.069
HGDM-WC-S++	**0.021**	**0.024**	**0.004**	**0.016**	**0.040**	**0.041**	**0.005**	**0.029**	**0.123**	**0.003**	**0.047**	**0.058**
GSDM-OC	0.142	0.230	0.043	0.138	0.930	0.867	0.168	0.655	1.996	0.0	1.013	1.003
GSDM-OC-S++	**0.011**	**0.016**	**0.001**	**0.009**	**0.012**	**0.087**	**0.011**	**0.037**	**1.2e-4**	**0.0**	**1.2e-4**	**0.066**
GSDM-WC	0.011	0.016	0.001	0.009	0.013	0.088	0.013	0.038	0.002	0.0	0	7.2e-5
GSDM-WC-S++	**0.011**	**0.016**	**0.001**	**0.009**	**0.011**	**0.086**	**0.010**	**0.036**	**5.0e-5**	**0.0**	**1.1e-5**	**0.066**

Table 12: Additional experiments on generic graph datasets.

Method	QM9			ZINC250k		
	Val. w/o corr. (%)↑	NSPDK MMD ↓	FCD ↓	Val. w/o corr. (%)↑	NSPDK MMD ↓	FCD ↓
GraphAF	67.00	0.020	5.268	68.00	0.044	16.289
GraphDF	82.67	0.063	10.816	89.03	0.176	34.202
MoFlow	91.36	0.017	4.467	63.11	0.046	20.931
EDP-GNN	47.52	0.005	2.680	82.97	0.049	16.737
GraphEBM	8.22	0.030	6.143	5.29	0.212	35.471
GDSS-OC	73.49	0.005	3.164	41.84	0.047	20.53
GDSS-OC-S++	**93.87**	**0.001**	**1.666**	**59.50**	**0.050**	**16.79**
GDSS-WC	94.91	0.004	2.550	95.83	0.019	14.66
GDSS-WC-S++	**93.87**	**0.001**	**1.666**	**93.15**	**0.012**	**12.70**
HGDM-OC	92.22	0.005	3.164	66.47	0.033	21.38
HGDM-OC-S++	**94.95**	**0.003**	**2.512**	**67.12**	**0.034**	**20.79**
HGDM-WC	98.02	0.002	2.147	93.26	0.016	17.69
HGDM-WC-S++	**97.03**	**0.001**	**2.001**	**91.03**	**0.016**	**16.24**

Table 13: Additional experiments on QM9 and ZINC250k datasets.