GRAPH GENERATION WITH DESTINATION-DRIVEN DIFFUSION MIXTURE

Jaehyeong Jo[*], **Dongki Kim**[*], **Sung Ju Hwang**
KAIST, South Korea
{harryjo97, cleverki, sjhwang82}@kaist.ac.kr

ABSTRACT

Generation of graphs is a major challenge for real-world tasks that require understanding the complex nature of their non-Euclidean structures. Although diffusion models have achieved notable success in graph generation recently, they are ill-suited for modeling the structural information of graphs since learning to denoise the noisy samples does not explicitly capture the graph topology. To tackle this limitation, we propose a novel generative process that models the topology of graphs by predicting the destination of the process. Specifically, we design the generative process as a mixture of diffusion processes conditioned on the endpoint in the data distribution, which drives the process toward the probable destination. Further, we introduce new training objectives for learning to predict the destination, and discuss the advantages of our generative framework that can explicitly model the graph topology and exploit the inductive bias of the data. Through extensive experimental validation on general graph and 2D/3D molecular graph generation tasks, we show that our method outperforms previous generative models, generating graphs with correct topology with both continuous and discrete features.

1 INTRODUCTION

Generation of graph-structured data has emerged as a crucial task for various real-world problems such as drug discovery (Simonovsky & Komodakis, 2018), protein design (Ingraham et al., 2019), and program synthesis (Brockschmidt et al., 2019). To tackle the challenge of learning the underlying distribution of graphs, diverse deep generative models have been proposed, including models based on generative adversarial networks (GANs) (De Cao & Kipf, 2018; Martinkus et al., 2022), recurrent neural networks (RNNs) (You et al., 2018), and variational autoencoders (VAEs) (Jin et al., 2018).

Recently, diffusion models have achieved state-of-the-art performance on the generation of graph-structured data (Niu et al., 2020; Jo et al., 2022; Hoogeboom et al., 2022). These models learn the generation process as the time reversal of the forward process, which corrupts the graphs by gradually adding noise that destroys its topological properties. Since the generative process is derived from the unknown score function (Song et al., 2021) or noise (Ho et al., 2020), existing graph diffusion models aim to estimate them in order to denoise the data from noise, which are commonly referred to as the *denoising diffusion models* (Figure 1, left).

Despite their success, learning the score or noise is fundamentally ill-suited for the generation of graphs. Although the key to generating valid graphs is modeling the topological information such as connectivity, the score or noise does not explicitly model these features. Thereby it is challenging for the diffusion models to recover the topological properties from the corrupted graphs through denoising, which leads to failure cases even for small graphs. Ideally, it would be best to directly learn to predict the resulting graph of the diffusion process for modeling its topology.

However, predicting the destination of the process with denoising diffusion models is not straightforward, as the main objective of the diffusion models is to remove noise from the noisy samples through the reverse process. Few existing works (Hoogeboom et al., 2021; Austin et al., 2021) aim to predict clean data from the corrupted samples by parameterizing the denoising process. However, this parameterization is only possible for categorical data with a finite number of states, and thus cannot generate graphs with continuous features nor utilize such features.

[*]Equal contribution

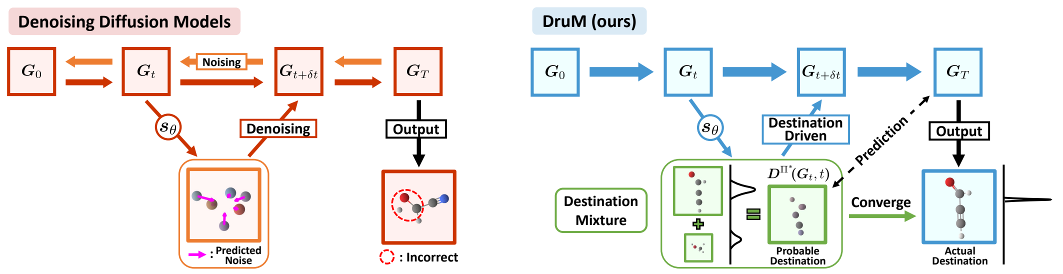


Figure 1: **Illustration of the graph generative process.** Our DruM (**blue**) successfully generates a graph with valid topology by predicting the actual destination, via learning the destination mixture (**green**) as a weighted mean of data (Eq. (1)). To this end, we design the generative process (Eq. (4)) as a mixture of endpoint-conditioned diffusion processes (Eq. (2)), which is driven toward the destination. However, previous diffusion models (**red**) often fail to capture the correct topology as they learn the score function or noise for denoising without the knowledge of the destination. We visualize the generative process of DruM in Appendix E.2.

To address these limitations of existing diffusion models, we propose a novel generative diffusion process that explicitly models the topology by predicting the destination of the generative process (Figure 1, right). Specifically, we design the generative process as a mixture of Ornstein-Uhlenbeck processes conditioned on the endpoint in the data distribution, for which the drift drives the process toward the destination. We establish a theoretical basis for our proposed destination-driven generation process (Section 3.1) and introduce new objectives for learning to predict the destination instead of denoising. Our generative framework can capture the topology of graphs as well as the continuous features, and further allows to exploit the inductive bias of the data which is crucial for the generation of real-world graphs (Section 3.2).

We experimentally validate our method on diverse real-world graph generation tasks. We first validate it on general graph generation benchmarks with synthetic and real-world graphs, on which it outperforms previous deep graph generative models including graph diffusion models, by being able to generate graphs with correct topologies. We further validate our method on 2D and 3D molecule generation tasks to demonstrate its ability to generate graphs with both the continuous and discrete features, on which ours generates a significantly larger number of valid and stable molecules compared to the baselines.

Our main contributions can be summarized as follows:

- We observe that previous graph diffusion models cannot accurately model the graph topology as they learn to denoise each step without the knowledge of the destination of the generative process.
- To fix such a myopic behavior of previous diffusion models, we propose a novel generative process that directly predicts the destination of the process by leveraging a mixture of diffusion processes.
- We derive theoretical groundwork for our Destination-Driven Diffusion Mixture, and discuss the practical advantages of our generative framework.
- Our method significantly outperforms previous graph diffusion models on the generation of diverse real and synthetic graphs, as well as on 2D/3D molecule generation tasks, by being able to generate graphs with accurate topologies, and both the discrete and continuous features.

2 Related Work

Diffusion Models Diffusion models have been shown to successfully generate high-quality samples from diverse data domains such as images (Dhariwal & Nichol, 2021), audios (Chen et al., 2021; Kong et al., 2021; Jeong et al., 2021), and videos (Ho et al., 2022). Despite their success, existing diffusion models for graphs (Niu et al., 2020; Jo et al., 2022) often fail to generate graphs with correct structures since the denoising process they use does not explicitly consider the graph topology. Discrete diffusion model (Vignac et al., 2022) proposed to model the noising process as successive graph edits for graphs with categorical node and edge attributes, but this is not a desirable solution for real-world graph generation tasks since it is not applicable to graphs with continuous features, such as the 3D coordinates of atoms. To address such limitations, we propose a novel graph diffusion framework that learns to directly predict the destination instead of learning to denoise through the process, in which the diffusion process drives the trajectories toward the predicted destination resulting in valid graphs with correct topology.

Diffusion Bridge Process A line of recent works has improved the generative framework of diffusion models by leveraging the diffusion bridge processes, which are processes conditioned on an endpoint. Schrödinger Bridge (Vargas et al., 2021; Wang et al., 2021; Bortoli et al., 2021; Chen et al., 2022) aims to find both the forward and the backward process that transforms arbitrary distributions back and forth, using iterative proportional fittings that require heavy computations. More recent works (Peluchetti, 2021; Wu et al., 2022; Ye et al., 2022) consider learning the generation process as a mixture of diffusion processes instead of modeling it with a reversal of the noising process as in denoising diffusion models. However, approximating the drift as in these works cannot accurately capture the discrete structure of graphs since it does not explicitly consider the topology, and could be problematic as the drift diverges near the terminal time. Instead, we propose to predict the destination of the generative process by leveraging the diffusion mixture, allowing it to model the graph topology.

Graph Generative Models Deep generative models for graphs either generate nodes and edges in an autoregressive manner (You et al., 2018; Liao et al., 2019; Luo et al., 2021) or generate all the nodes and edges at once (De Cao & Kipf, 2018; Zang & Wang, 2020; Martinkus et al., 2022). Recently, diffusion models for graphs (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2022; Hoogeboom et al., 2022) have made large progress in generating synthetic graphs as well as 2D and 3D molecular graphs. However, existing graph diffusion models either fail to capture the graph topology or are not applicable to general tasks due to the structural limitation of the framework. To overcome these limitations, we propose a novel graph diffusion framework that explicitly models the topology by learning to predict the destination. Our graph diffusion framework largely outperforms existing models (Jo et al., 2022; Vignac et al., 2022; Hoogeboom et al., 2022) on general graph generation as well as 2D / 3D molecule generation tasks.

3 Destination-Driven Diffusion Mixture

In this section, we present our novel generative framework, **D**estination-D**r**iven Diff**u**sion **M**ixture (DruM), which learns to predict the destination by designing the generative process as a mixture of diffusion processes.

3.1 Designing the Graph Generative Process

The key to generating graph-structured data is understanding the underlying topology of graphs, which is crucial to determining its validity. However, previous diffusion models fail to do so as their objective is to denoise the noisy graphs, in which the topology is only implicitly captured (Figure 1, left). Few existing works (Hoogeboom et al., 2021; Austin et al., 2021) aim to predict the clean data from a noisy sample by parameterizing the denoising process, but are only applicable to categorical data and thereby not able to generate graphs with continuous features. To overcome the limitation, we propose to design a generative process that yields a direct prediction of the destination.

Destination Mixture Our goal is to directly predict the destination of the generative process, i.e. a diffusion process with a terminal distribution identical to the unknown data distribution Π^*. To be specific, for a diffusion process represented as a trajectory of random variables $\{\boldsymbol{G}_\tau\}_{\tau\in[0,T]}$, we aim to predict the terminus of the process given the current state $\boldsymbol{G}_t$. However, identifying the exact destination at the early stage of the process is problematic, since $\boldsymbol{G}_t$ resembles the initial noise which contains almost no information. Hence predicting a single deterministic data point could lead the generative process in the wrong direction.

To address this problem, we present a new approach to predicting the *probable destination* which we define as a weighted mean of the destinations (Figure 1, right). Assuming that the destination of the process is in the data distribution, the probability of a data point $\boldsymbol{g}$ being the destination is equal to the transition probability of the process, denoted as $p_{T|t}(\boldsymbol{g}|\cdot)$. Thus we define the probable destination for $\boldsymbol{G}_t$ via the expectation of data with respect to the probability of being the destination as follows:

$$\boldsymbol{D}(\boldsymbol{G}_t, t) = \int \boldsymbol{g} \cdot p_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\, \mathrm{d}\boldsymbol{g}, \tag{1}$$

which we refer to as the *destination mixture* of the process, visualized in Figure 1 as green. In order to explicitly model this destination mixture, we construct a generative process that is driven toward the destination which we describe in the following paragraphs.

Ornstein-Uhlenbeck Bridge Process As a building block of our destination-driven generative process, we leverage diffusion processes with fixed endpoints, namely the *diffusion bridge* processes. We propose to use a new family of bridge processes, namely the *Ornstein-Uhlenbeck* (OU) bridge process that generalizes the Brownian bridge used in previous works (Wu et al., 2022; Ye et al., 2022) and provides more flexibility for designing the generative process (see Appendix A.1 for detailed explanation). To derive this bridge process, we start with a reference OU process defined as:

$$\mathbb{Q}:\ \mathrm{d}\boldsymbol{G}_t = \alpha\sigma_t^2\boldsymbol{G}_t\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t,$$

where α is a constant, σ_t is a scalar function, and $\mathbf{W}_t$ is the standard Wiener process. Then the process conditioned on the endpoint $\boldsymbol{g}$, denoted as $\mathbb{Q}^{\boldsymbol{g}} := \mathbb{Q}(\cdot|\boldsymbol{G}_T = \boldsymbol{g})$, can be obtained using the Doob's h-transform (Doob & Doob, 1984) (see Appendix A.1 for the derivation):

$$\mathrm{d}\boldsymbol{G}_t = \Bigg[\underbrace{\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t\right)}_{\eta^{\boldsymbol{g}}(\boldsymbol{G}_t,t)}\Bigg]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t \tag{2}$$

where the scalar function u_t and v_t are defined as follows:

$$u_t = \exp\left(\alpha\int_t^T \sigma_\tau^2\mathrm{d}\tau\right),\ v_t = \frac{1}{2\alpha}\left(1 - u_t^{-2}\right). \tag{3}$$

The destination of this process is fixed to $\boldsymbol{G}_T = \boldsymbol{g}$, since the drift $\eta^{\boldsymbol{g}}(\cdot,t)$ of the process in Eq. (2) forces the trajectory $\boldsymbol{G}_t$ towards the destination $\boldsymbol{g}$. With the OU bridge processes in hand, we present a novel generative process for directly predicting the destination.

Destination-Driven Diffusion Mixture As the destination mixture in Eq. (1) is a weighted mean of endpoints, conceptually, this can be modeled by combining the endpoint-conditioned processes with respect to these weights. Thereby, we propose to design the generation process by mixing the OU bridge processes conditioned on the endpoints from the data distribution. Here, we leverage the concept of the *diffusion mixture representation* (Brigo, 2008; Peluchetti, 2021), which yields the representation of a mixture process that combines a collection of diffusion processes. In a nutshell, the SDE representation of this mixture process is defined to be the weighted mean of the SDE of each diffusion process in the collection (we provide a formal definition in Appendix A.2).

To be more precise, in order to model the generative process, we mix a collection of OU bridge processes $\{\mathbb{Q}^{\boldsymbol{g}} : \boldsymbol{g} \in \Pi^*\}$ which is defined by the following SDE:

$$\mathrm{d}\boldsymbol{G}_t = \left[\int \eta^{\boldsymbol{g}}(\boldsymbol{G}_t,t)\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g})\right]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t, \tag{4}$$

where $p_t^{\boldsymbol{g}}$ is the marginal density of the bridge process $\mathbb{Q}^{\boldsymbol{g}}$ and $p_t(\cdot) := \int p_t^{\boldsymbol{g}}(\cdot)\Pi^*(\mathrm{d}\boldsymbol{g})$ is the marginal density of the mixture process. Notably, the terminal distribution of the mixture process is equal to the data distribution by definition. We refer to this mixture process in Eq. (4) as the *OU bridge mixture*, and denote as $\mathbb{Q}^{\Pi^*}$.

From the SDE representation of OU bridge process in Eq. (2), the drift of $\mathbb{Q}^{\Pi^*}$ can be derived as follows (see Appendix A.3 for the derivation of the drift):

$$\eta(\boldsymbol{G}_t,t) = \alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) - \boldsymbol{G}_t\right),\quad \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) := \int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) \tag{5}$$

Notice that from the definition of the transition distribution, we can derive the following:

$$\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) = \int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) = \int \boldsymbol{g}\ \frac{p(\boldsymbol{G}_t, \boldsymbol{G}_T = \boldsymbol{g})}{p_t(\boldsymbol{G}_t)}\mathrm{d}\boldsymbol{g} = \int \boldsymbol{g}\ p_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\ \mathrm{d}\boldsymbol{g}, \tag{6}$$

implying that $\boldsymbol{D}^{\Pi^*}(\cdot,t)$ coincides with $\boldsymbol{D}(\cdot,t)$ in Eq. (1), corresponding to the destination mixture of $\mathbb{Q}^{\Pi^*}$. Therefore, $\boldsymbol{D}^{\Pi^*}(\cdot,t)$ is the prediction of the final destination, which represents how probable each data is to be the destination as a weighted mean. Moreover, since the drift of Eq. (5) forces the process to the direction of $\boldsymbol{D}^{\Pi^*}(\cdot,t)$, the generation process of the OU bridge mixture is driven toward the destination mixture, and hence to the actual destination in the data distribution (Figure 1, right). Thereby, we name our proposed generative process as Destination-Driven Diffusion Mixture (DruM).

The generative process of DruM is uniquely determined by the constant α and the noise schedule σ_t. Especially, the noise schedule controls the convergence behavior of DruM, since small σ_t causes a large drift in Eq. (5) that forces the trajectory to converge quickly, visualized in Appendix D.2.

3.2 Learning the Destination Mixture

In this section, we introduce the objective for learning the destination mixture and the extension for the attributed graphs. We further discuss the advantages of our method.

Training Objectives In order to estimate $\mathbb{Q}^{\Pi^*}$, we define the generative model $\mathbb{P}^{\theta}$ as follows:

$$\mathbb{P}^{\theta} : \mathrm{d}\boldsymbol{G}_t = \left[\alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t} \left(\frac{1}{u_t} \boldsymbol{s}_{\theta}(\boldsymbol{G}_t, t) - \boldsymbol{G}_t \right) \right] \mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{7}$$

where $\boldsymbol{s}_{\theta}(\cdot, t)$ approximates the destination mixture of Eq. (5). Since we want the generative process to be driven toward the destination in the data distribution, we train $\boldsymbol{s}_{\theta}(\cdot, t)$ so that the trajectories sampled from the generative model fit the trajectories from the OU bridge mixture $\mathbb{Q}^{\Pi^*}$. As introduced in previous works (Wu et al., 2022; Liu et al., 2022), we leverage the Girsanov theorem Øksendal (2003) to upper bound the KL divergence between the data distribution Π^* and the terminal distribution of the generative model $\mathbb{P}^{\theta}$ denoted as p_T^{θ} (see Appendix A.4 for details):

$$D_{KL}(\Pi^* \| p_T^{\theta}) \leq D_{KL}(\mathbb{Q}^{\Pi^*} \| \mathbb{P}^{\theta}) = \mathbb{E}_{\boldsymbol{G} \sim \mathbb{Q}^{\Pi^*}} \left[\frac{1}{2} \int_0^T \gamma_t^2 \left\| \boldsymbol{s}_{\theta}(\boldsymbol{G}_t, t) - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) \right\|^2 \mathrm{d}t \right] + C, \tag{8}$$

where C is a constant independent of θ, $\boldsymbol{G} \sim \mathbb{Q}^{\Pi^*}$ denotes the sampled trajectories from the OU bridge mixture, and $\gamma_t := \sigma_t/(u_t v_t)$. However, since the ground truth destination mixture of $\mathbb{Q}^{\Pi^*}$ is not analytically accessible, Eq. (8) cannot be used directly. Therefore, using the property of conditional expectation, we introduce a new objective for estimating the destination mixture, which is equivalent to minimizing Eq. (8) (see Appendix A.4 for the derivation):

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{G} \sim \mathbb{Q}^{\Pi^*}} \left[\frac{1}{2} \int_0^T \gamma_t^2 \left\| \boldsymbol{s}_{\theta}(\boldsymbol{G}_t, t) - \boldsymbol{G}_T \right\|^2 \mathrm{d}t \right]. \tag{9}$$

During training, the trajectories $\boldsymbol{G} \sim \mathbb{Q}^{\Pi^*}$ can be easily obtained by sampling $\boldsymbol{G}_0 \sim p_0$, $\boldsymbol{G}_T \sim \Pi^*$, $t \sim [0, T]$ and $\boldsymbol{G}_t \sim p_{t|0,T}(\boldsymbol{G}_t | \boldsymbol{G}_0, \boldsymbol{G}_T)$ which is the distribution of $\mathbb{Q}^{\Pi^*}$ at time t given the endpoints $\boldsymbol{G}_0$ and $\boldsymbol{G}_T$. We derive this probability in Appendix A.5 and provide the pseudo-code for the training in Appendix B.1. Note that the objective of the loss in Eq. (9) is for $\boldsymbol{s}_{\theta}$ to estimate the destination mixture $\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t)$, not the exact endpoint $\boldsymbol{G}_T$. Additionally, we introduce a simplified loss using a constant loss coefficient instead of the time-dependent γ_t in Appendix B.2, which shows improved performance for modeling continuous features, such as 3-dimensional positions.

Advantages of the Destination Mixture Learning the destination mixture provides the following advantages: (1) we can explicitly model the graph topology instead of implicitly capturing them via the score or noise, (2) we can model both continuous and discrete features, for example, 3-dimensional molecules with both discrete atom types and continuous coordinates, (3) it is significantly easier to learn the destination mixture compared to learning the drift function $\eta(\cdot, t)$ of the OU bridge mixture, since the destination mixture is supported inside the bounded data space while $\eta(\cdot, t)$ diverges as t approaches the terminal time, and (4) we can exploit the inductive bias of the data for learning the destination mixture, which is critical for training as it dramatically reduces the hypothesis space. To be specific, we can leverage the prior knowledge of the data representation such as one-hot encoding or the categorical type by adding an additional function at the last layer of the model $\boldsymbol{s}_{\theta}$, for instance, softmax function for the one-hot encoded node features and the sigmoid function for the adjacency matrices (see Appendix B.3 for details). This guarantees that the model learns the representation of a weighted mean of data. We experimentally verify these advantages in Section 4.

4 Experiments

4.1 General Graph Generation

Datasets and Metrics We evaluate the quality of generated graphs on three synthetic and real datasets used as benchmarks in previous works (Martinkus et al., 2022; Vignac et al., 2022). (1) **Planar**: 200 synthetic graphs which are planar and connected, (2) **Stochastic Block Model** (SBM): 200 community graphs with up to 5 communities, and (3) **Proteins**: 918 proteins graphs where each node represents the amino acids. We follow the evaluation setting of Martinkus et al. (2022) using

Table 1: **Generation results on the general graph datasets.** Best results are highlighted in bold, where smaller MMD and larger V.U.N. indicates better results. The baseline results are taken from Martinkus et al. (2022) and Vignac et al. (2022), or obtained by running the open-source codes. The hyphen (-) denotes out-of-resources that take more than 2 weeks.

	Planar					SBM					Proteins			
	Synthetic, $\lvert V\rvert = 64$					Synthetic, $44 \leq \lvert V\rvert \leq 187$					Real, $100 \leq \lvert V\rvert \leq 500$			
	Deg.	Clus.	Orbit	Spec.	V.U.N.	Deg.	Clus.	Orbit	Spec.	V.U.N.	Deg.	Clus.	Orbit	Spec.
Training set	0.0002	0.0310	0.0005	0.0052	100.0	0.0008	0.0332	0.0255	0.0063	100.0	0.0003	0.0068	0.0032	0.0009
GraphRNN	0.0049	0.2779	1.2543	0.0459	0.0	0.0055	0.0584	0.0785	0.0065	5.0	0.0040	0.1475	0.5851	0.0152
GRAN	0.0007	0.0426	0.0009	0.0075	0.0	0.0113	0.0553	0.0540	0.0054	25.0	0.0479	0.1234	0.3458	0.0125
SPECTRE	0.0005	0.0785	0.0012	0.0112	25.0	0.0015	0.0521	**0.0412**	0.0056	52.5	0.0056	0.0843	**0.0267**	0.0052
EDP-GNN	0.0044	0.3187	1.4986	0.0813	0.0	0.0011	0.0552	0.0520	0.0070	35.0	-	-	-	-
GDSS	0.0041	0.2676	0.1720	0.0370	0.0	0.0212	0.0646	0.0894	0.0128	5.0	0.0861	0.5111	0.732	0.0748
DiGress	**0.0003**	0.0372	**0.0009**	0.0106	75	0.0013	0.0498	0.0434	0.0400	74	-	-	-	-
DruM (Ours)	0.0005	**0.0353**	**0.0009**	**0.0062**	**90.0**	**0.0007**	**0.0492**	0.0448	**0.0050**	**85.0**	**0.0019**	**0.0660**	0.0345	**0.0030**

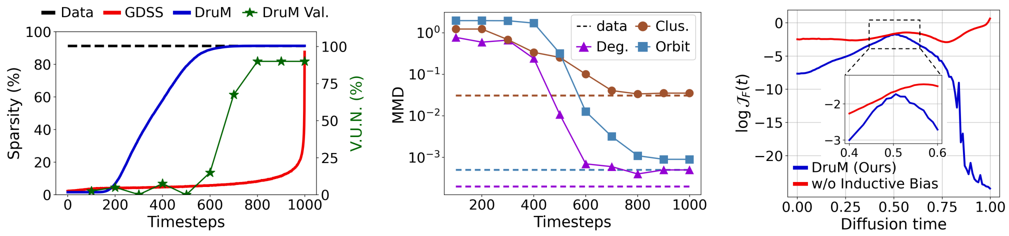


Figure 2: **(Left) Topology analysis** through the generative process. We compare the sparsity of the destination mixture from DruM (blue) against the trajectory of GDSS (red), and visualize the V.U.N results of DruM in green. **(Middle) MMD results of the destination mixture from DruM** through the generative process. The dotted lines indicate the MMD results of the training set. **(Right) Complexity of DruM** with and without using the inductive bias, measured by the Frobenius norm of the Jacobian of the models.

the same data split. We measure the maximum mean discrepancy (MMD) of four graph statistics between the generated graphs and the graphs from the test set: degree (Deg.), clustering coefficient (Clus.), count of orbits with 4 nodes (Orbit), and the eigenvalues of the graph Laplacian (Spec.). Additionally, to verify that the model truly learns the target distribution, we report the percentage of valid, unique, and novel (V.U.N.) graphs among the generated graphs, for which the validness is defined as satisfying the specific property of the dataset: planarity and connectedness for Planar and statistical test for SBM. For more details, please see Appendix C.1.

Baselines We compare DruM against the following graph generative models: **GraphRNN** (You et al., 2018), which is an autoregressive model based on RNN, **GRAN** (Liao et al., 2019), an autoregressive model with attention, **SPECTRE** (Martinkus et al., 2022), a one-shot model based on GAN, **EDP-GNN** (Niu et al., 2020), a score-based model for adjacency matrix, **GDSS** (Jo et al., 2022), a continuous diffusion model, and **DiGress** (Vignac et al., 2022), a discrete diffusion model. We provide the details of the training and the sampling of DruM in Appendix B, and describe further implementation details in Appendix C.1.

Results Table 1 shows that our DruM outperforms all the baselines on all datasets. Especially, DruM achieves the highest validity (V.U.N.) metric, as it accurately learns the underlying topology of the graphs by directly predicting the destination of the process, compared to the previous graph diffusion models. Notably, our method outperforms DiGress by a large margin in V.U.N., even though we do not use specific prior distribution nor structural feature augmentation that are utilized in DiGress. This is because we generate both the adjacency matrices and the eigenvectors of the graph Laplacian as node features, thereby capturing detailed topological information of the final graphs. We provide the visualization of the generated graphs and the diffusion process of DruM in Appendix E, which demonstrate that DruM is able to capture the characteristics of each dataset.

Topology Analysis To show how the destination-driven process of DruM results in graphs with correct topology, we conduct an analysis on the estimated destination mixture of DruM. Figure 2 (Left) shows that the sparsity of the estimated destination mixture from DruM gradually increases and achieves the sparsity of the data at an early stage, while GDSS recovers the sparsity abruptly at the late stage of the reverse diffusion process. Further, we observe that the V.U.N. of the estimated destination mixture increases after achieving the desired sparsity, finally resulting in 90% validity. This shows that predicting the destination allows it to better capture the global topologies. Moreover,

Table 2: **Generation results on the 2D molecule datasets.** We report the average of 3 different runs except for the results of Vignac et al. (2022) on ZINC250k dataset that takes more than 1 week. The best results are highlighted in bold. The baseline results are taken from Jo et al. (2022) or attained by running the open-source codes. We provide the results of uniqueness, novelty and variance in Appendix D.1

	QM9 ($\|V\| \leq 9$)				ZINC250k ($\|V\| \leq 38$)			
Method	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑
Training set	100.0	0.0398	0.0001	0.9719	100.0	0.0615	0.0001	0.8395
MoFlow Zang & Wang (2020)	91.36	4.467	0.0169	0.1447	63.11	20.931	0.0455	0.0133
GraphAF (Shi et al., 2020)	74.43	5.625	0.0207	0.3046	68.47	16.023	0.0442	0.0672
GraphDF (Luo et al., 2021)	93.88	10.928	0.0636	0.0978	90.61	33.546	0.1770	0.0000
EDP-GNN Niu et al. (2020)	47.52	2.680	0.0046	0.3270	82.97	16.737	0.0485	0.0000
GDSS (Jo et al., 2022)	95.72	2.900	0.0033	0.6983	97.01	14.656	0.0195	0.0467
DiGress (Vignac et al., 2022)	98.19	**0.095**	0.0003	0.9353	92.13	3.606	0.0025	0.3549
DruM (Ours)	**99.69**	0.108	**0.0002**	**0.9449**	**98.65**	**2.257**	**0.0015**	**0.5299**

	QM9 ($\|V\| \leq 29$)		GEOM-DRUGS ($\|V\| \leq 181$)	
Method	Atom Stab.(%)	Mol. Stab.(%)	Atom Stab.(%)	Mol. Stab.(%)
G-Schnet (Gebauer et al., 2019)	95.7	68.1	-	-
EN-Flow (Satorras et al., 2021)	85.0	4.9	75.0	0.0
GDM (Hoogeboom et al., 2022)	97.0	63.2	75.0	0.0
EDM (Hoogeboom et al., 2022)	98.7 ±0.1	82.0 ±0.4	81.3	0.0
Bridge (Wu et al., 2022)	98.7 ±0.1	81.8 ±0.2	81.0 ±0.7	0.0
DruM (Ours)	**98.81** ±0.03	**87.34** ±0.19	**82.96** ±0.12	**0.51** ±0.03

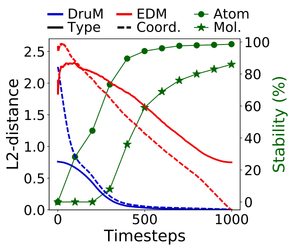


Figure 3: **(Left) Generation results on the 3D molecule datasets.** Best results are highlighted in bold which is the average of 3 different runs. The baseline results are taken from Hoogeboom et al. (2022) and Wu et al. (2022). **(Right) Convergence of the generative process.** We compare the convergence of the destination mixture from DruM and the trajectory of EDM, measured with L2 distance from the final result of each process. Atom and molecule stabilities are measured with the destination mixture from DruM.

we plot the MMD results of the estimated destination mixture from DruM through the diffusion timesteps in Figure 2 (Middle), which shows that the local characteristics of graphs are improved progressively through the generation process converging to that of the training set.

Exploiting the Inductive Bias To validate that exploiting the inductive bias of the data is critical for reducing the hypothesis space, we compare DruM against a variant of it without a transformation function at the last layer in the model. Figure 2 (Right) shows the complexity of the models s_θ trained on the Planar dataset, which verify that the transformation at the last layer significantly reduces the complexity of the model for predicting the destination. Especially, the larger complexity gap at the end of the process suggests that using the inductive bias is crucial for learning the exact destination.

4.2 2D Molecule Generation

We further validate DruM on 2D molecule generation tasks to show that it can accurately generate graphs with both the node features and the topologies of the target graphs.

Datasets and Metrics We evaluate the quality of generated 2D molecules on two molecule datasets used as benchmarks in Jo et al. (2022). **QM9** (Ramakrishnan et al., 2014) consists of small molecules with up to 9 atoms while **ZINC250k** (Irwin et al., 2012) consists of larger molecules up to 38 atoms. Following the evaluation setting of Jo et al. (2022), we evaluate the models with four metrics: (1) **Validity**, which is the percentage of the valid molecules among the generated molecules without any post-hoc correction such as valency correction or edge resampling. (2) **Fréchet ChemNet Distance** (FCD) (Preuer et al., 2018), which measures the distance between the feature vectors of generated molecules and the test set using ChemNet, evaluating the chemical properties of the molecules. (3) **Neighborhood subgraph pairwise distance kernel** (NSPDK) MMD (Costa & De Grave, 2010), which measures the MMD between the generated molecular graphs and the molecular graphs from the test set incorporating both the node and edge features, assessing the quality of the graph structure. (4) **Scaffold similarity** (Scaf.) which measures the cosine similarity of the frequencies of Bemis-Murcko scaffolds (Bemis & Murcko, 1996) that evaluates the ability to generate similar substructures. See Appendix C.2 for more details.

Baselines We compare DruM against the following molecular graph generative models: **MoFlow** (Zang & Wang, 2020) is a one-shot flow-based model. **GraphAF** (Shi et al., 2020) and **GraphDF** (Luo et al., 2021) are autoregressive flow-based model. **EDP-GNN**, **GDSS**, and

DiGress are diffusion models previously explained in the general graph generation tasks. See further implementation details in Appendix C.2.

Results Table 2 shows that our method achieves the highest validity on all datasets verifying that DruM can generate valid molecules without correction. Further, DruM outperforms the baselines in FCD and NSPDK metrics which demonstrates that the molecules synthesized by DruM are similar to the molecule from the training set in both chemical and graph-structure aspects. Especially, DruM achieves the highest scaffold similarity indicating that DruM is able to generate similar substructures, for instance ring structures, from that of the training set. We visualize the generated molecules in Appendix E where they share similar substructures with the molecules from the training set.

4.3 3D Molecule Generation

To show that DruM is able to generate graphs with both continuous and discrete features, we validate it on 3D molecule generation tasks, which come with discrete atom types and continuous coordinates.

Datasets and Metrics We evaluate the generated 3D molecules on two molecule datasets used in Hoogeboom et al. (2022). **QM9** (Ramakrishnan et al., 2014) consists of small molecules up to 29 atoms, while **GEOM-DRUGS** (Axelrod & Gomez-Bombarelli, 2022) consists of larger molecules up to 181 atoms. We evaluate the quality of the generated molecules with two stability metrics: **Atom stability** is the percentage of the atoms with valid valency. **Molecule stability** is the percentage of the generated molecules that consist of stable atoms. For more details, please see Appendix C.3.

Baselines We compare DruM against 3D molecule generative models: **G-Schnet** (Gebauer et al., 2019) is an autoregressive model based on the 3d point sets. **EN-Flow** (Satorras et al., 2021) is a flow-based model. **GDM** and **EDM** (Hoogeboom et al., 2022) are denoising diffusion models. **Bridge** (Wu et al., 2022) is a diffusion model based on the diffusion mixture that learns to approximate the drift. For DruM, we follow the training setting of Hoogeboom et al. (2022) using EGNN (Satorras et al., 2021) model. We describe further implementation details in Appendix C.3.

Results As shown in the table of Figure 3, our method yields the highest atom stability compared to all the baselines on both datasets. Furthermore, DruM achieves significantly higher molecule stability compared to the state-of-the-art diffusion model EDM, since DruM learns to predict the destination directly instead of learning to denoise. To the best of our knowledge, DruM is the first generative model to achieve non-zero molecule stability in the GEOM-DRUGS dataset which consists of large molecules. We visualize the generated molecules that are stable and the generative process of DruM in Appendix E, which demonstrates that the estimated destination mixture predicts the final molecule at an early stage of the process, leading to stable molecules. We further observe that DruM is able to generate more graphs that are connected in GEOM-DRUGS, shown in Table 5 of the Appendix.

Stability Analysis To further investigate the superior performance of DruM in generating more stable molecules, we conduct an analysis on the convergence and the stability of DruM. Figure 3 (Right) shows the convergence of the destination mixture from DruM and the trajectory of EDM through the generative process. We observe that for DruM, both the atom types and the coordinates converge rapidly to the final destination. After the convergence, the stability increases as it has sufficient steps to calibrate the details to produce valid molecules, which can be observed in the visualized generative process in Figure 18 of Appendix. As for EDM, the coordinates linearly converge while the resulting atom types do not. Thus, the generated molecules of EDM can have a mismatch between the atom types and the positions, which leads to lower molecule stability.

5 Conclusion

In this work, we proposed a novel graph generation framework, DruM, that explicitly models the topological properties of the graphs. Unlike existing graph diffusion models that estimate the score or the noise, our framework directly predicts the destination of the generative process, thereby accurately capturing the topologies of the final graphs. Specifically, DruM constructs the generation process as a mixture of diffusion bridges, that drives the generation process toward the destination in the data distribution. We extensively validated DruM on diverse graph generation tasks, including 2D/3D molecular generation, on which ours significantly outperforms previous graph generation methods.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021.

Simon Axelrod and Rafael Gomez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):1–14, 2022.

Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17695–17709, 2021.

Damiano Brigo. The general mixture-diffusion sde and its relationship with an uncertain-volatility option model with volatility-asset decorrelation. *arXiv:0812.4052*, 2008.

Marc Brockschmidt, Miltiadis Allamanis, Alexander L. Gaunt, and Oleksandr Polozov. Generative code modeling with graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *ICLR*, 2021.

Tianrong Chen, Guan-Horng Liu, and Evangelos A. Theodorou. Likelihood training of schrödinger bridge using forward-backward sdes theory. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 255–262. Omnipress; Madison, WI, USA, 2010.

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

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv:2105.05233*, 2021.

Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

Joseph L Doob and JI Doob. *Classical potential theory and its probabilistic counterpart*, volume 549. Springer, 1984.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv:2012.09699*, 2020.

Niklas W. A. Gebauer, Michael Gastegger, and Kristof Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7564–7576, 2019.

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

Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12454–12465, 2021.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887. PMLR, 2022.

John Ingraham, Vikas K. Garg, Regina Barzilay, and Tommi S. Jaakkola. Generative models for graph-based protein design. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net, 2019.

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.

Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-tts: A denoising diffusion model for text-to-speech. In *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pp. 3605–3609. ISCA, 2021.

Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2328–2337. PMLR, 2018.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383. PMLR, 2022.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021.

Greg Landrum et al. Rdkit: Open-source cheminformatics software, 2016. *URL http://www. rdkit. org/, https://github. com/rdkit/rdkit*, 2016.

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. *Advances in Neural Information Processing Systems*, 32:4255–4265, 2019.

Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. Let us build bridges: Understanding and extending diffusion generative models. *arXiv:2208.14699*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. *International Conference on Machine Learning*, 2021.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *ICML*, 2022.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Stefano Peluchetti. Non-denoising forward-time diffusions. *Openreview*, 2021.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,*, pp. 3942–3951. AAAI Press, 2018.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):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*, 1(1):1–7, 2014.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 4181–4192, 2021.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

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

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *NeurIPS*, 2020.

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.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019.

Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil D. Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.

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

Gefei Wang, Yuling Jiao, Qian Xu, Yang Wang, and Can Yang. Deep generative learning via schrödinger bridge. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10794–10804. PMLR, 2021.

Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *arXiv:2209.00865*, 2022.

Mao Ye, Lemeng Wu, and Qiang Liu. First hitting diffusion models. *arXiv:2209.01170*, 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*, pp. 5708–5717. PMLR, 2018.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 617–626, 2020.

Bernt Øksendal. *Stochastic Differential Equations*. Universitext. Springer Berlin Heidelberg, 2003.

Appendix

Organization The appendix is organized as follows: In Section A, we provide the derivations of the results from the main paper. In Section B, we explain the details of our generative framework including the training objectives, the sampling method, and the model architectures. In Section C, we provide experimental details for the generation tasks and further present additional experimental results in Section D. Lastly, in Section E, we visualize the generated graphs and molecules, with visualized diffusion processes.

A DERIVATIONS

A.1 DIFFUSION BRIDGE PROCESSES

Here we derive the Ornstein-Uhlenbeck (OU) bridge process using Doob's h-transform, and show that the Brownian bridge process is a special case of the OU bridge process.

Ornstein-Uhlenbeck Bridge Process First, we consider the simple case when the reference process is given as a standard OU process without a time-dependent diffusion coefficient:

$$\hat{\mathbb{Q}} \;:\; \mathrm{d}\boldsymbol{G}_t = \alpha \boldsymbol{G}_t \mathrm{d}t + \mathrm{d}\mathbf{W}_t, \tag{10}$$

where α is a constant. Then the Doob's h-transform on $\hat{\mathbb{Q}}$ yields the representation of an endpoint-conditioned process $\hat{\mathbb{Q}}^{\boldsymbol{g}} := \hat{\mathbb{Q}}(\cdot|\boldsymbol{G}_T = \boldsymbol{g})$ defined by the following SDE:

$$\hat{\mathbb{Q}}^{\boldsymbol{g}} \;:\; \mathrm{d}\boldsymbol{G}_t = \Big[\alpha \boldsymbol{G}_t + \nabla_{\boldsymbol{G}_t} \log \hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\Big]\mathrm{d}t + \mathrm{d}\mathbf{W}_t, \tag{11}$$

where $\hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)$ is the transition probability from time t to T of the standard OU process in Eq. (10). Since the standard OU process has a linear drift, the transition probability is Gaussian, i.e. $\hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t) = \mathcal{N}(\boldsymbol{g}; \mu_t, \boldsymbol{\Sigma}_t)$, where the mean μ_t and the covariance $\boldsymbol{\Sigma}_t$ satisfies the following ODEs (from the results of Eq.(5.50) and Eq.(5.51) of Särkkä & Solin (2019)):

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}t} = \alpha\mu_t \;\;,\;\; \frac{\mathrm{d}\boldsymbol{\Sigma}_t}{\mathrm{d}t} = \mathbf{I} + 2\alpha\boldsymbol{\Sigma}_t. \tag{12}$$

The ODE with respect to $\boldsymbol{\Sigma}_t$ can be modified as:

$$\frac{\mathrm{d}}{\mathrm{d}t} e^{-2\alpha t}\boldsymbol{\Sigma}_t = e^{-2\alpha t}\mathbf{I}, \tag{13}$$

which has the following closed-form solutions:

$$\mu_t = \hat{u}_t \boldsymbol{G}_t \;\;,\;\; \boldsymbol{\Sigma}_t = \frac{1}{2\alpha}\left(\hat{u}_t^2 - 1\right)\mathbf{I} \;\text{ for }\; \hat{u}_t = e^{\alpha(T-t)}. \tag{14}$$

Therefore, the SDE representation of the standard OU bridge process with fixed endpoint $\boldsymbol{g}$ is given as follows:

$$\mathrm{d}\boldsymbol{G}_t = \left[\alpha\boldsymbol{G}_t + \frac{2\alpha}{1-\hat{u}_t^{-2}}\left(\frac{\boldsymbol{g}}{\hat{u}_t} - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \mathrm{d}\mathbf{W}_t. \tag{15}$$

Now to derive the bridge process for the general OU process with a time-dependent diffusion coefficient defined by the following SDE:

$$\mathbb{Q}\colon\; \mathrm{d}\boldsymbol{G}_t = \alpha\sigma_t^2\boldsymbol{G}_t\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t, \tag{16}$$

where σ_t is a scalar function, we leverage the time change (Section 8.5. of Øksendal (2003)) with $\beta_t = \int_0^t \sigma_\tau^2 \mathrm{d}\tau$. Since the time change with β_t of the diffusion process in Eq. (10) is equivalent to the diffusion process of Eq. (16), the transition probability $\tilde{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)$ of the general OU process satisfies the following:

$$\tilde{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t) = \hat{p}_{\beta_T|\beta_t}(\boldsymbol{g}|\boldsymbol{G}_t) \tag{17}$$

Thereby, the general OU bridge process conditioned on the endpoint $\boldsymbol{g}$ is defined by the following SDE:

$$\mathbb{Q}^{\boldsymbol{g}} \; : \; \mathrm{d}\boldsymbol{G}_t = \left[\alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t \right) \right] \mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{18}$$

where the scalar function u_t and v_t are given as:

$$u_t = e^{\alpha(\beta_T - \beta_t)} = \exp\left(\alpha \int_t^T \sigma_\tau^2 \mathrm{d}\tau \right) \;\; , \;\; v_t = \frac{1}{2\alpha}(1 - u_t^{-2}). \tag{19}$$

Brownian Bridge Process We show that the Brownian bridge process is a special case of the OU bridge process. When the constant α of the OU bridge process approaches 0, the scalar function u_t converges to 1 that leads to the convergence of v_t as follows:

$$v_t = \frac{1}{2\alpha}(1 - u_t^{-2}) = \frac{1}{2\alpha}\left(1 - e^{-2\alpha(\beta_T - \beta_t)} \right) \rightarrow \beta_T - \beta_t,$$

which is due to the Taylor expansion of the exponential function. Therefore, the OU bridge process for $\alpha \rightarrow 0$ is modeled by the following SDE:

$$\mathrm{d}\boldsymbol{G}_t = \frac{\sigma_t^2}{\beta_T - \beta_t}(\boldsymbol{g} - \boldsymbol{G}_t)\,\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{20}$$

which is equivalent to the SDE representation of the Brownian bridge process. Compared to the OU bridge process in Eq. (18), the Brownian bridge process has a simpler SDE representation with less flexibility for the design.

Note that the Brownian bridge is an endpoint-conditioned process with respect to a reference Brownian Motion defined by the following SDE:

$$\mathrm{d}\boldsymbol{G}_t = \sigma_t \mathrm{d}\mathbf{W}_t, \tag{21}$$

which is a diffusion process without drift, and also a special case of the OU process that is used for the reference process of the OU bridge process.

A.2 Diffusion Mixture Representation

In this section, we provide the formal definition of the diffusion mixture representation (Brigo, 2008; Peluchetti, 2021).

Consider a collection of diffusion processes $\{\boldsymbol{Z}_t^\lambda\}_{\lambda\in\Lambda}$ defined by the SDEs:

$$\mathbb{Q}^\lambda : \mathrm{d}\boldsymbol{Z}_t^\lambda = \eta^\lambda(\boldsymbol{Z}_t, t)\mathrm{d}t + \sigma_t^\lambda \mathrm{d}\mathbf{W}_t^\lambda \;\; , \;\; \boldsymbol{Z}_0^\lambda \sim p_0^\lambda \tag{22}$$

where p_0^λ are the initial distributions and $\mathbf{W}_t^\lambda$ are independent standard Wiener processes, and p_t^λ denotes the marginal distribution of the process $\boldsymbol{Z}_t^\lambda$. Define the mixture of marginal distributions and the mixture of initial distributions with respect to a mixing distribution $\mathcal{L}$ on the collection Λ as follows:

$$p_t(z) = \int_\Lambda p_t^\lambda(z)\mathcal{L}(\mathrm{d}\lambda) \;\; , \;\; p_0(z) = \int_\Lambda p_0^\lambda(z)\mathcal{L}(\mathrm{d}\lambda), \tag{23}$$

Then there exists a diffusion process $\boldsymbol{Z}_t$ that induces a marginal distribution p_t defined by the following SDE:

$$\mathbb{Q}^\mathcal{L} : \mathrm{d}\boldsymbol{Z}_t = \eta(\boldsymbol{Z}_t, t)\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t \;\; , \;\; \boldsymbol{Z}_0 \sim p_0, \tag{24}$$

where the drift and diffusion coefficients are given as the weighted mean of the corresponding coefficients of $\mathbb{Q}^\lambda$:

$$\eta(z,t) = \int_\Lambda \eta^\lambda(z,t)\frac{p_t^\lambda(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda) \;\; , \;\; \sigma_t^2 = \int_\Lambda (\sigma_t^\lambda)^2 \frac{p_t^\lambda(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda), \tag{25}$$

which is the full statement of the diffusion mixture representation.

In order to prove this, it is enough to show that p_t defined in Eq. (23) is the solution to the Fokker-Planck equation of Eq. (24), which is given as follows:

$$\frac{\partial q_t(z)}{\partial t} = -\nabla_z \cdot \left(q_t(z)\eta(z,t) - \frac{1}{2}\sigma_t^2 \nabla_z q_t(z) \right) \tag{26}$$

Using the definition of Eq. (23) and the Fokker-Planck equations with respect to $\mathbb{Q}^\lambda$ for $\lambda \in \Lambda$, we can derive the following result:

$$\begin{aligned}
\frac{\partial p_t(z)}{\partial t} &= \frac{\partial}{\partial t}\int_\Lambda p_t^\lambda(z)\mathcal{L}(\mathrm{d}\lambda) = \int_\Lambda \frac{\partial}{\partial t} p_t^\lambda(z)\mathcal{L}(\mathrm{d}\lambda) \\
&= \int_\Lambda \left[-\nabla_z \cdot \left(\eta^\lambda(z,t)p_t^\lambda(z) - \frac{1}{2}(\sigma^\lambda)_t^2 \nabla_z p_t^\lambda(z) \right) \right] \mathcal{L}(\mathrm{d}\lambda) \\
&= -\nabla_z \cdot \int_\Lambda \left[\eta^\lambda(z,t)p_t^\lambda(z) - \frac{1}{2}(\sigma^\lambda)_t^2 \nabla_z p_t^\lambda(z) \right] \mathcal{L}(\mathrm{d}\lambda) \\
&= -\nabla_z \cdot \left(p_t(z) \int_\Lambda \eta^\lambda(z,t) \frac{p_t^\lambda(z)}{p_t(z)} \mathcal{L}(\mathrm{d}\lambda) - \frac{1}{2}\nabla_z \left[p_t(z) \int_\Lambda (\sigma_t^\lambda)^2 \frac{p_t^\lambda(z)}{p_t(z)} \mathcal{L}(\mathrm{d}\lambda) \right] \right) \\
&= -\nabla_z \cdot \left(p_t(z)\eta(z,t) - \frac{1}{2}\sigma_t^2 \nabla_z p_t(z) \right),
\end{aligned} \tag{27}$$

which proves that p_t is the solution to the Fokker-Planck equation of Eq. (24).

A.3 OU Bridge Mixture

Now we use the diffusion mixture representation described in Appendix A.2 to derive the OU bridge mixture. Consider a mixture of the collection of OU bridge processes with endpoints in the data distribution, i.e. $\{\mathbb{Q}^{\boldsymbol{g}}\}_{\boldsymbol{g}\in\Pi^*}$. We mix this collection of processes with the data distribution Π^* as the mixing distribution, which is represented by the following SDE:

$$\begin{aligned}
\mathbb{Q}^{\Pi^*} : \mathrm{d}\boldsymbol{G}_t &= \left[\int \left(\alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t \right) \right) \frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)} \Pi^*(\mathrm{d}\boldsymbol{g}) \right] \mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t \\
&= \left[\alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)} \Pi^*(\mathrm{d}\boldsymbol{g}) - \boldsymbol{G}_t \right) \right] \mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t \\
&= \left[\alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) - \boldsymbol{G}_t \right) \right] \mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t
\end{aligned} \tag{28}$$

where the fact that $p_t(z) = \int p_t^{\boldsymbol{g}}(z)\Pi^*(\mathrm{d}\boldsymbol{g})$ is used for the second equality.

A.4 Derivation of the Loss Objective

Further, we provide the derivation of the objectives in Eq. (8) and Eq. (9) in the main paper. First, we leverage the Girsanov theorem Øksendal (2003) to upper bound the KL divergence between the data distribution Π^* and the terminal distribution of $\mathbb{P}^\theta$ denoted as p_T^θ:

$$D_{KL}(\Pi^* \| p_T^\theta) \le D_{KL}(\mathbb{Q}^{\Pi^*} \| \mathbb{P}^\theta) \tag{29}$$

$$= D_{KL}(\mathbb{Q}_0^{\Pi^*} \| \mathbb{P}_0^\theta) + \mathbb{E}_{\mathbb{Q}^{\Pi^*}}\left[\log \frac{\mathrm{d}\mathbb{Q}^{\Pi^*}}{\mathrm{d}\mathbb{P}^\theta} \right] \tag{30}$$

$$= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[-\log p_0^\theta(\boldsymbol{G}_0) + \frac{1}{2}\int_0^T \left\| \sigma_t^{-1}\left(\eta_\theta(\boldsymbol{G}_t, t) - \eta(\boldsymbol{G}_t, t) \right) \right\|^2 \mathrm{d}t \right] + C_1 \tag{31}$$

$$= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[-\log p_0^\theta(\boldsymbol{G}_0) + \frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) \right\|^2 \mathrm{d}t \right] + C_1, \tag{32}$$

where the first inequality is known as the data processing inequality, p_0^θ is a predetermined prior distribution that is easy to sample from, for instance, standard Gaussian distribution, and C_1 is a constant independent of θ. The Eq. (32) corresponds to the objective of Eq. (8) in the main paper.

Furthermore, using the property of the conditional expectation, minimizing the expectation of Eq. (32) is equivalent to minimizing the following loss:

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t,t)-\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\right\|^2 \mathrm{d}t\right]\\ &=\mathbb{E}_{\boldsymbol{g}\sim\Pi^*,\boldsymbol{G}\sim\mathbb{Q}^{\boldsymbol{g}}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t,t)-\boldsymbol{g}\right\|^2 \mathrm{d}t\right]+C_2 \\ &=\mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t,t)-\boldsymbol{G}_T\right\|^2 \mathrm{d}t\right]+C_2.\end{aligned} \tag{33}$$

where C_2 is a constant independent of θ. Thereby, we obtain the loss presented in Eq. (9).

A.5 Analytical Computation of the Probability

In order to practically use the loss in Eq. (9), we provide the analytical form for the probability $p_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$ of the following OU bridge process:

$$\mathbb{Q}^{\boldsymbol{g}}\ :\ \mathrm{d}\boldsymbol{G}_t=\left[\alpha\sigma_t^2\boldsymbol{G}_t+\frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t}-\boldsymbol{G}_t\right)\right]\mathrm{d}t+\sigma_t\mathrm{d}\mathbf{W}_t. \tag{34}$$

First, we use the Bayes theorem as follows:

$$p(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)=\frac{p(\boldsymbol{G}_t,\boldsymbol{G}_T|\boldsymbol{G}_0)}{p(\boldsymbol{G}_T|\boldsymbol{G}_0)}=\frac{p(\boldsymbol{G}_T|\boldsymbol{G}_t,\boldsymbol{G}_0)\ p(\boldsymbol{G}_t|\boldsymbol{G}_0)}{p(\boldsymbol{G}_T|\boldsymbol{G}_0)}=\frac{p(\boldsymbol{G}_T|\boldsymbol{G}_t)\ p(\boldsymbol{G}_t|\boldsymbol{G}_0)}{p(\boldsymbol{G}_T|\boldsymbol{G}_0)}, \tag{35}$$

where the last equality is due to the Markov property of $\boldsymbol{G}$. The transition distributions in this equation are Gaussian with the mean and the covariance satisfying the ODEs of Eq. (12), which could be further computed as follows:

$$\begin{aligned}&p_{b|a}(\boldsymbol{G}_b|\boldsymbol{G}_a)=\mathcal{N}(\boldsymbol{G}_b;u_{b|a}\boldsymbol{G}_a,u_{b|a}^2 v_{b|a}\mathbf{I})\ \text{ for }\ 0\le a<b\le T,\\ &\text{where }\ u_{b|a}\coloneqq\exp\left(\alpha\int_a^b\sigma_\tau^2\mathrm{d}\tau\right)\ ,\ v_{b|a}\coloneqq\frac{1}{2\alpha}\left(1-u_{b|a}^{-2}\right).\end{aligned} \tag{36}$$

Thereby, the probability $p(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$ is also Gaussian from the results of Eq. (35):

$$p(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)=\frac{p(\boldsymbol{G}_T|\boldsymbol{G}_t)\ p(\boldsymbol{G}_t|\boldsymbol{G}_0)}{p(\boldsymbol{G}_T|\boldsymbol{G}_0)}=\mathcal{N}(\boldsymbol{G}_t;\tilde{\mu}_t,\tilde{\boldsymbol{\Sigma}}_t), \tag{37}$$

where the mean and the covariance have analytical form computed from the product of Gaussian distributions as follows:

$$\tilde{\mu}_t=\frac{v_{T|t}}{u_{t|0}v_{T|0}}\boldsymbol{G}_0+\frac{v_{t|0}}{u_{T|t}v_{T|0}}\boldsymbol{G}_T\ \ ,\ \ \tilde{\boldsymbol{\Sigma}}_t=\frac{v_{T|t}v_{t|0}}{v_{T|0}}\mathbf{I}. \tag{38}$$

A.6 Destination Mixture for Attributed Graphs

We extend the framework of DruM for the generation of *attributed graphs* with node features. A graph $\boldsymbol{G}$ with N nodes is defined by the node features $\boldsymbol{X}\in\mathbb{R}^{N\times F}$ and the adjacency matrix $\boldsymbol{A}\in\mathbb{R}^{N\times N}$, where F is the dimension of the node features and the adjacency represents the topology as well as the edge features. Representing the trajectory of the diffusion process as $\boldsymbol{G}_t=(\boldsymbol{X}_t,\boldsymbol{A}_t)\in\mathbb{R}^{N\times F}\times\mathbb{R}^{N\times N}$ for time $t\in[0,T]$, we can derive the OU bridge mixture $\mathbb{Q}^{\Pi^*}$ for attributed graphs from Eq. (4) as follows:

$$\begin{pmatrix}\mathrm{d}\boldsymbol{X}_t\\ \mathrm{d}\boldsymbol{A}_t\end{pmatrix}=\begin{pmatrix}\eta_X(\boldsymbol{G}_t,t)\\ \eta_A(\boldsymbol{G}_t,t)\end{pmatrix}\mathrm{d}t+\begin{pmatrix}\sigma_{1,t}\ \mathrm{d}\mathbf{W}_{1,t}\\ \sigma_{2,t}\ \mathrm{d}\mathbf{W}_{2,t}\end{pmatrix}, \tag{39}$$

where the noise schedules $\sigma_{1,t}$ and $\sigma_{2,t}$ are scalar functions, $\mathbf{W}_{1,t}$ and $\mathbf{W}_{2,t}$ are independent Wiener processes, and the drift of the OU bridge mixture is defined as follows:

$$\begin{aligned}\eta_X(\boldsymbol{G}_t,t)&=\alpha_1\sigma_{1,t}^2\boldsymbol{X}_t+\frac{\sigma_{1,t}^2}{v_{1,t}}\left(\frac{\boldsymbol{D}_X(\boldsymbol{G}_t,t)}{u_{1,t}}-\boldsymbol{X}_t\right)\\ \eta_A(\boldsymbol{G}_t,t)&=\alpha_2\sigma_{2,t}^2\ \boldsymbol{A}_t+\frac{\sigma_{2,t}^2}{v_{2,t}}\left(\frac{\boldsymbol{D}_A(\boldsymbol{G}_t,t)}{u_{2,t}}-\boldsymbol{A}_t\right)\end{aligned} \tag{40}$$

Algorithm 1 Training of DruM

```
Input: Graph $\boldsymbol{G}$
1: Sample $t \sim [0, T-\epsilon]$ and $G_0 \sim \mathcal{N}(0, \mathbf{I})$
2: Sample $G_t \sim p_{t|0,T}(\boldsymbol{G}_t | \boldsymbol{G}_0, \boldsymbol{G})$    ▷ Section A.5
3: $\gamma_t \leftarrow \sigma_t / u_t v_t$
4: $\mathcal{L}_\theta \leftarrow \gamma_t^2 \|\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}\|^2$    ▷ Eq. (9)
5: Update $\theta$ using $\mathcal{L}_\theta$
```

Algorithm 2 Sampling from DruM

```
Input: Trained model $\boldsymbol{s}_\theta$, number of sampling steps $K$, step size $\mathrm{d}t$
1: Sample number of nodes $N$ from the training set.
2: $\boldsymbol{G}_0 \sim \mathcal{N}(0, \mathbf{I}_N)$    ▷ Start from noise
3: $t \leftarrow 0$
4: for $k = 1$ to $K$ do
5:     $\eta \leftarrow \alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}(\frac{1}{u_t}\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_t)$
6:     $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$
7:     $\boldsymbol{G}_{t+\mathrm{d}t} \leftarrow \eta \mathrm{d}t + \sigma_t \sqrt{\mathrm{d}t}\mathbf{w}$    ▷ Predictor
8:     $t \leftarrow t + \mathrm{d}t$
9: end for
10: $\boldsymbol{G} \leftarrow \texttt{quantize}(\boldsymbol{G}_t)$    ▷ Quantize if necessary
11: Return: Graph $\boldsymbol{G}$
```

Algorithm 3 PC Sampler for DruM

```
Input: Trained model $\boldsymbol{s}_\theta$, number of sampling steps $K$, number of correction steps per prediction $M$, step size $\mathrm{d}t$, score-to-noise ratio $r$, scaling coefficient $\epsilon_s$
Output: Sampled graph $\boldsymbol{G}$
1: Sample number of nodes $N$ from the training set.
2: $\boldsymbol{G}_0 \sim \mathcal{N}(0, \mathbf{I}_N)$    ▷ Start from noise
3: $t \leftarrow 0$
4: for $k = 1$ to $K$ do
5:     $\eta \leftarrow \alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_t\right)$
6:     $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
7:     $\tilde{\boldsymbol{G}}_t \leftarrow \eta \mathrm{d}t + \sigma_t \sqrt{\mathrm{d}t}\mathbf{w}$    ▷ Predictor
8:     for $m = 1$ to $M$ do
9:         $\tilde{\eta} \leftarrow \alpha\tilde{\boldsymbol{G}}_t + \frac{1}{v_t}\left(\frac{1}{u_t}\boldsymbol{s}_\theta(\tilde{\boldsymbol{G}}_t, t) - \tilde{\boldsymbol{G}}_t\right)$
10:        $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
11:        $\epsilon \leftarrow 2\left(r\|\mathbf{w}\|_2 / \|\tilde{\eta}\|_2\right)^2$
12:        $\boldsymbol{G}_{t+\mathrm{d}t} \leftarrow \tilde{\boldsymbol{G}}_t + \tilde{\eta}\epsilon + \epsilon_s\sqrt{2\epsilon}\mathbf{w}$    ▷ Corrector
13:    end for
14:    $t \leftarrow t + \mathrm{d}t$
15: end for
16: $\boldsymbol{G} \leftarrow \texttt{quantize}(\boldsymbol{G}_t)$    ▷ quantize if necessary
17: Return: graph $\boldsymbol{G}$
```

with the destination mixture for this process given as

$$\begin{pmatrix} \boldsymbol{D}_X(\boldsymbol{G}_t, t) \\ \boldsymbol{D}_A(\boldsymbol{G}_t, t) \end{pmatrix} = \int \boldsymbol{g} \frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)} \Pi^*(\mathrm{d}\boldsymbol{g}). \tag{41}$$

Notably, $\boldsymbol{D}_X(\boldsymbol{G}_t, t)$ and $\boldsymbol{D}_A(\boldsymbol{G}_t, t)$ are the destination mixtures of the node features and the adjacency matrices, respectively for given graph $\boldsymbol{G}_t$. Thereby, our extended framework allows us to directly model the graph topology via learning $\boldsymbol{D}_A(\cdot, t)$ as well as the node features with $\boldsymbol{D}_X(\cdot, t)$. Especially, the generation process of Eq. (39) is applicable to both the continuous and discrete features. We provide the training objective for the attributed graphs in Appendix B.2.

B Details of DruM

In this section, we provide the details of the training and sampling methods of DruM and the models used in our experiments. We provide the pseudo-codes for training and sampling with an explicit form of loss objectives. Further, we discuss the hyperparameters of DruM.

B.1 Overview

We provide the pseudo-code of the training and sampling of our generative framework in Algorithm 1 and 2, respectively. We further provide the implementation details of the training and sampling for each generation task in Section C.

B.2 Training Objective

Random Permutation The general graph datasets, namely Planar and SBM, contain only 200 graphs. Thus to ensure the permutation equivariant nature of the graph dataset, we apply random permutation to the graphs of the training set during training. To be specific, for a graph data $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{A})$ in the trianing set and random permutation matirx $\boldsymbol{P}$, we use the permuted data $(\boldsymbol{P}^T\boldsymbol{X}, \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P})$ for training, where $\boldsymbol{P}^T$ denotes the transposed matrix. We empirically found that this leads to more stable training.

Simplified Loss We provide the explicit form of simplified loss explained in Section 3.2, which uses constant loss coefficient c instead of the time dependent γ_t as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T c^2 \left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_T\right\|^2 \mathrm{d}t\right]. \tag{42}$$

We empirically found that using this loss is beneficial for the generation of continuous features such as eigenvectors of the graph Laplacian or 3D coordinates.

Attributed Graphs Especially for the generation of attributed graphs $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{A})$, the training loss for learning the destination mixture for $\boldsymbol{X}$ and $\boldsymbol{A}$ can be derived from the objective in Eq. (9) using our extended framework of Eq. (39). Specifically, for the model $\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) = (\boldsymbol{s}_\theta^X, \boldsymbol{s}_\theta^A)$, we use the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_{1,t}^2 \left\|\boldsymbol{s}_\theta^X(\boldsymbol{G}_t, t) - \boldsymbol{X}_T\right\|^2 \mathrm{d}t + \frac{\lambda}{2}\int_0^T \gamma_{2,t}^2 \left\|\boldsymbol{s}_\theta^A(\boldsymbol{G}_t, t) - \boldsymbol{A}_T\right\|^2 \mathrm{d}t\right] \tag{43}$$

where λ is the hyperparameter. We use $\lambda = 5$ for all our experiments, and empirically observed that changing λ did not make much difference for sufficient training epochs.

B.3 Model Architecture

For the general graph and 2D molecule generation tasks, we leverage the graph transformer network introduced in Dwivedi & Bresson (2020) and Vignac et al. (2022). The node features and the adjacency matrices are updated with multiple attention layers with global features obtained by the self-attention-based FiLM layers (Perez et al., 2018). We additionally use the higher-order adjacency matrices following Jo et al. (2022). For general graph generation tasks, we add the sigmoid function to the output of the model since the entries of the weighted mean of the data are supported in the interval $[0, 1]$. For 2D molecule generation tasks, we apply the softmax function to the output of the node features to model the one-hot encoded atom types, and further apply the sigmoid function to the output of the adjacency matrices. Note that we do not use the structural augmentation as in Vignac et al. (2022). For the 3D molecule generation task, we use EGNN (Satorras et al., 2021) in order to model the E(3) equivariance of the molecule data. Specifically, EGNN computes the messages with the E(3) equivariant objects such as length two coordinates. We additionally add the softmax function at the last layer to model the one-hot encoded atom types.

B.4 Prediction-Correction Sampler for DruM

Sampling from the generative model of Eq. (7) requires solving the SDE. If our model $\boldsymbol{s}_\theta$ can perfectly approximate the destination mixture of Eq. (5), a simple Euler-Maruyama method would be enough to simulate the generative model which is true for most of the experiments. However, for some cases, since $\boldsymbol{s}_\theta$ is trained on the marginal distribution p_t, $\boldsymbol{G}_t$ outside of p_t could cause incorrect predictions that lead to an undesired destination. To address the limitation, we leverage the predictor-corrector (PC) sampling method introduced in Song et al. (2021) for solving Eq. (7). Using the corrector method such as Langevin dynamics (Song et al., 2021), we force $\boldsymbol{G}_t$ to be drawn from p_t which ensures a more accurate estimation of the destination mixture. We provide the pseudo-code of the PC sampler for our DruM in Algorithm 3.

B.5 Hyperparameters of DruM

Since the generative process of DruM is a mixture of OU bridge processes in Eq. (2), it is uniquely determined with the noise schedule σ_t and constant α. Through our experiments, we use $\alpha = -1/2$ and a decreasing linear noise schedule, starting from σ_0^2 and ends in σ_0^2 defined as follows:

$$\sigma_t^2 = (1-t)\sigma_0^2 + t\sigma_1^2 \;\text{ for }\; 0 < \sigma_1 < \sigma_0 < 1 \tag{44}$$

We perform a grid search for the hyperparameters σ_0 and σ_1 in $\{0.4, 0.6, 0.8, 1.0\}$ and $\{0.1, 0.2, 0.3\}$, respectively, where the search space slightly differs for each generation task.

C Experimental Details

C.1 General Graph Generation

Datasets and Evaluation Metrics Planar graph dataset consists of 200 synthetic planar graphs where each graph has 64 nodes. We determine that a graph is a valid Planar graph if it is connected and planar. SBM graph dataset consists of 200 synthetic stochastic block model graphs with the number of communities uniformly sampled between 2 and 5 where the number of nodes in each community is uniformly sampled between 20 and 40. The probability of the inter-community edges and the intra-community edges are 0.3 and 0.05, respectively. We determine that a graph is a valid SBM graph if it has the number of communities between 2 and 5, and the number of nodes in each community between 20 and 40, and further using the statistical test introduced in Martinkus et al. (2022). Proteins graph dataset (Dobson & Doig, 2003) consists of 918 real protein graphs with up to 500 nodes in each graph. The protein graph is constructed by considering each amino acid as a node and connecting two nodes if the corresponding amino acids are less than 6 Angstrom. For our experiments, we use the datasets provided by Martinkus et al. (2022). We follow the evaluation setting of Liao et al. (2019), using total variation (TV) distance for measuring MMD which is considerably fast, especially for large graphs compared to using the earth mover's distance (EMD) kernel. Moreover, we use the V.U.N. metric following Martinkus et al. (2022) that measures the proportion of valid, unique, and novel graphs among the generated graphs where the validness is defined as satisfying the specific property of the dataset explained above.

Implementation Details We follow the standard setting of Martinkus et al. (2022) using the same data splitting and evaluation procedures. We report the baseline results taken from Martinkus et al. (2022) and Vignac et al. (2022), or the results obtained from running the open-source codes using the hyperparameters given by the original work. We could not report the results of EDP-GNN (Niu et al., 2020) and DiGress (Vignac et al., 2022) on the Proteins dataset as they took more than 2 weeks using their official codes. For the diffusion models including our proposed method, we set the diffusion steps to 1000 for a fair comparison.

For our proposed DruM, we train our model s_θ for 30,000 epochs for all datasets using a constant learning rate with AdamW optimizer (Loshchilov & Hutter, 2017) and weight decay 10^{-12}, applying the exponential moving average (EMA) to the parameters (Song & Ermon, 2020). We perform the hyperparameter search explained in Section B.5 for the lowest MMD and highest V.U.N. results. We initialize the node features with the eigenvectors of the graph Laplacian of the adjacency matrices, which we further scale with constant. During training (Algorithm 1), we sample the noise for the adjacency matrices to be symmetric with zero diagonals. For the generation (Algorithm 2), we generate both the node features and adjacency matrices starting from noise, and we quantize the entries of the resulting adjacency matrices to obtain 0-1 adjacency matrices. Empirically, we found that the entries of the resulting sample lie very close to the desired values 0 and 1, for which the L1 distance between the resulting sample and the quantized sample is smaller than 10^{-2}.

C.2 2D Molecule Generation

Datasets and Evaluation Metircs QM9 Ramakrishnan et al. (2014) dataset consists of 133,885 molecules with up to 9 heavy atoms of four types. ZINC250k Irwin et al. (2012) dataset consists of 249,455 molecules with up to 38 heavy atoms of 9 types. Molecules in both datasets have 3 edge types, namely single bond, double bond, and triple bond. For our experiments, we follow the standard procedure Shi et al. (2020); Luo et al. (2021); Jo et al. (2022) of kekulizing the molecules using the RDKit library (Landrum et al., 2016) and removing the hydrogen atoms from the molecules in the QM9 and ZINC250k datasets. To evaluate the generated molecules, we measure the validity (before any post-hoc correction), FCD Preuer et al. (2018), NSPDK MMD Costa & De Grave (2010), and scaffold similarity. Among these, FCD and NSPDK MMD metrics measure the distribution similarity between the test dataset and generated samples while scaffold similarity evaluates the similarity of the generated scaffolds.

Implementation Details We report the results of the baselines taken from Jo et al. (2022), except for the results of the Scaffold similarity (Scaf.) and the results of DiGress, which we obtained by running the open-source codes. Especially, the 2D molecule generation results of DiGress on QM9 dataset are different compared to the results reported in its original paper, since we have used the preprocessed dataset following the setting of Jo et al. (2022) for a fair comparison with other baselines. Similar to the general graph generation tasks, we set the diffusion steps to 1000 for all the diffusion models.

For our DruM, we train our model $\boldsymbol{s}_\theta$ with a constant learning rate with AdamW optimizer and weight decay 10^{-12}, applying the exponential moving average (EMA) to the parameters. We perform the hyperparameter search similar to that of the general graph generation tasks. Especially for DruM, we follow Jo et al. (2022) by using the adjacency matrices in the form of $\boldsymbol{A} \in \{0, 1, 2, 3\}^{N\times N}$ where N is the maximum number of atoms in a molecule and each entries indicating the bond types: 0 for no bond, 1 for the single bond, 2 for the double bond and 3 for the triple bond. Further, we scale the entries with a constant scale of 3 in order to bound the input of the model in the interval $[0, 1]$, and rescale the final sample of the generation process to recover the bond types. We also sample the noise for the adjacency matrices to be symmetric with zero diagonals during training. We quantize the entries of the resulting adjacency matrices to obtain the discrete bond types $\{0, 1, 2, 3\}$. Empirically, we found that the entries of the resulting sample lie very close to the desired bond types $\{0, 1, 2, 3\}$, for which the L1 distance between the resulting sample and the quantized sample is approximately 0.

C.3 3D Molecule Generation

Datasets and Evaluation Metircs QM9 Ramakrishnan et al. (2014) dataset consists of 133,885 molecules with up to 29 atoms of five types including hydrogen atoms. The node features of QM9 dataset include the one-hot representation of atom type and atom number. GEOM-DRUGS Axelrod & Gomez-Bombarelli (2022) dataset consists of 430,000 molecules with up to 181 atoms of fifteen types including hydrogen atoms. GEOM-DRUGS dataset contains different conformations for each molecule. Among many conformations, the 30 lowest energy conformations for each molecule are retained. For the GEOM-DRUGS dataset, we utilize the one-hot representation of atom type without the atom number. To evaluate the generated molecules, we measure the atom and molecule stability by predicting the bond type between atoms with the standard bond lengths, and then checking the valency.

Implementation Details We follow the standard setting of Hoogeboom et al. (2022) for a fair comparison: for the QM9 experiment, we use EGNN with 256 hidden features and 9 layers and train the model, and for the GEOM-DRUGS experiment, we use EGNN with 256 hidden features and 4 layers and train the model. We report the results of the baselines taken from Hoogeboom et al. (2022) and Wu et al. (2022).

For our DruM, we train our model $\boldsymbol{s}_\theta$ for 1,300 epochs with batch size 256 for the QM9 experiment, and for 13 epochs with batch size 64 for the GEOM-DRUGS experiment. We apply EMA to the parameters of the model with a coefficient of 0.999 and use AdamW optimizer with learning rate 10^{-4} and weight decay 10^{-12}. The 3D coordinates and charge values are scaled as $\times 4$ and $\times 0.1$, respectively, and we use the simplified loss with a constant $c = 100$. We perform the hyperparameter search with smaller values for coordinates in {0.1, 0.2, 0.3} and higher values for node features in {0.6, 0.7, 0.8, 0.9, 1.0}. For the generation, we use the Euler-Maruyama predictor.

C.4 Computing Resources

For all experiments, we use NVIDIA GeForce RTX 3090 and 2080 Ti and implement the source code with PyTorch Paszke et al. (2019).

D Additional Experimental Results

D.1 2D Molecule Genreation

We provide the standard deviation results in Table 3 and Table 4. We additionally report the following two metrics: **Novelty** is the proportion of the molecules generated that are valid and not in the training set, and **Uniqueness** is the proportion of the molecules generated that are valid and unique, where valid molecules are the ones that do not violate the chemical valency rule.

Table 3: **2D molecule generation results on the QM9 dataset.** The baseline results are taken from Jo et al. (2022) or obtained by running the open-source codes. Best results are highlighted in bold.

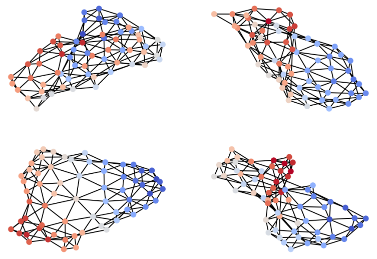
Method	Valid (%)↑	FCD ↓	NSPDK ↓	Scaf. ↑	Uniq (%) ↑	Novelty (%) ↑
MoFlow (Zang & Wang, 2020)	91.36 ±1.23	4.467 ±0.595	0.017 ±0.003	0.1447 ±0.0521	98.65 ±0.57	**94.72** ±0.77
GraphAF'(Shi et al., 2020)	74.43 ±2.55	5.625 ±0.259	0.021 ±0.003	0.3046 ±0.0556	88.64 ±2.37	86.59 ±1.95
GraphDF (Luo et al., 2021)	93.88 ±4.76	10.928 ±0.038	0.064 ±0.000	0.0978 ±0.1058	98.58 ±0.25	98.54 ±0.48
EDP-GNN (Niu et al., 2020)	47.52 ±3.60	2.680 ±0.221	0.005 ±0.001	0.3270 ±0.1151	**99.25** ±0.05	86.58 ±1.85
GDSS (Jo et al., 2022)	95.72 ±1.94	2.900 ±0.282	0.003 ±0.000	0.6983 ±0.0197	98.46 ±0.61	86.27 ±2.29
DiGress (Vignac et al., 2022)	98.19 ±0.23	**0.095** ±0.008	0.0003 ±0.000	0.9353 ±0.0025	96.67 ±0.24	25.58 ±2.36
DruM (ours)	**99.69** ±0.19	0.108 ±0.006	**0.0002** ±0.000	**0.9449** ±0.0054	96.90 ±0.15	24.15 ±0.80

Table 4: **2D molecule generation results on the ZINC250k dataset.** Baseline results are taken from Jo et al. (2022) or obtained by running the open-source codes. Best results are highlighted in bold. Since DiGress (Vignac et al., 2022) takes more than 1 week on ZINC250k dataset, we report the result from a single run.

Method	Valid (%)↑	FCD ↓	NSPDK ↓	Scaf. ↑	Uniq (%) ↑	Novelty (%) ↑
MoFlow (Zang & Wang, 2020)	63.11 ±5.17	20.931 ±0.184	0.046 ±0.002	0.0133 ±0.0052	**99.99** ±0.01	**100.00** ±0.00
GraphAF (Shi et al., 2020)	68.47 ±0.99	16.023 ±0.451	0.044 ±0.005	0.0672 ±0.0156	98.64 ±0.69	99.99 ±0.01
GraphDF (Luo et al., 2021)	90.61 ±4.30	33.546 ±0.150	0.177 ±0.001	0.0000 ±0.0000	99.63 ±0.01	**100.00** ±0.00
EDP-GNN Niu et al. (2020)	82.97 ±2.73	16.737 ±1.300	0.049 ±0.006	0.0000 ±0.0000	99.79 ±0.08	**100.00** ±0.00
GDSS (Jo et al., 2022)	97.01 ±0.77	14.656 ±0.680	0.019 ±0.001	0.0467 ±0.0054	99.64 ±0.13	**100.00** ±0.00
DiGress (Vignac et al., 2022)	92.13	3.606	0.0025	0.3549	99.96	100.00
DruM (ours)	**98.65** ±0.25	**2.257** ±0.084	**0.0015** ±0.0003	**0.5299** ±0.0441	99.97 ±0.03	99.98 ±0.02

Figure 4: **(Left) Generation results on the Planar dataset.** Best results are highlighted in bold, where smaller MMD and larger V.U.N. indicates better results. **(Right) Generated graphs by learning the drift.** The visualized graphs are randomly sampled from the set of generated graphs.

	Planar				
	Deg.	Clus.	Orbit	Spec.	V.U.N.
Training set	0.0002	0.0310	0.0005	0.0052	100.0
GraphRNN (You et al., 2018)	0.0049	0.2779	1.2543	0.0459	0.0
SPECTRE (Martinkus et al., 2022)	0.0005	0.0785	0.0012	0.0112	25.0
EDP-GNN (Niu et al., 2020)	0.0044	0.3187	1.4986	0.0813	0.0
GDSS (Jo et al., 2022)	0.0041	0.2676	0.1720	0.0370	0.0
DiGress (Vignac et al., 2022)	**0.0003**	0.0372	**0.0009**	0.0106	75
Drift	0.0008	0.0845	0.0075	0.0126	15
DruM (Ours)	0.0005	**0.0353**	**0.0009**	**0.0062**	**90.0**



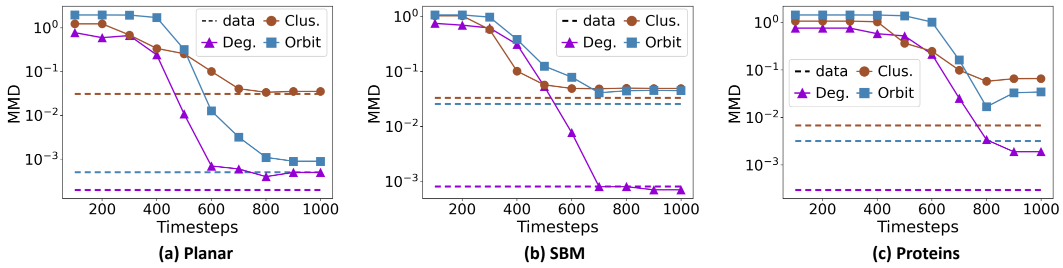
D.2 Further Analysis

Comparison with Learning the Drift We additionally report the generation result of learning the drift similar to the previous work (Wu et al., 2022), on the Planar dataset. Table in Figure 4 shows that learning the drift performs poorly, generating only 15% valid, novel, and unique graphs. The generated Planar graphs in Figure 4 demonstrate that it fails to capture the correct topology, resulting in non-planar graphs.

Early Stopping for Generation Process We provide additional MMD results of diffusion processes in Figure 5, which show that the estimated destination mixture converges to the exact destination around 800 diffusion steps for all datasets.

Role of the Diffusion Coefficient Here, we demonstrate the effect of the diffusion coefficient σ_t on the convergence of the generative process. Figure 6 (Sampled Graph) shows that the smaller values of σ_t (i.e. 0.2∼0.1) lead to fast convergence of the trajectory to the actual destination, compared to the larger σ_t. On the other hand, as shown in Figure 6 (Destination Mixture), the larger σ_t for the continuous features makes it hard to predict the actual destination since it destroys the topology of graphs with strong noise, and leads to slow convergence of the destination mixture.

Figure 5: **Additional MMD results of the destination mixture of DruM** through the generative process. The dotted lines indicate the MMD results of the training set.



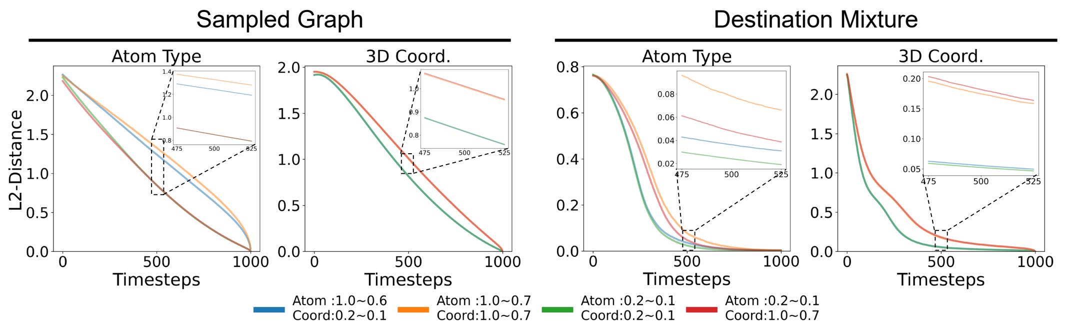


Figure 6: **Convergence of sampled graphs and destination mixtures with varying σ_0 and σ_1 values.**

E VISUALIZATION

In this section, we visualize the generated graphs and molecules of DruM, and further provide visualization of the diffusion processes for diverse generation tasks.

E.1 GENERATED SAMPLES OF DRUM

General Graphs Graphs from the training set and the generated graphs of DruM are visualized in Figure 7,8, and 9. The visualized graphs are randomly selected from the training set and the generated graph set. Note that we visualize the entire graph for the Proteins dataset, unlike Martinkus et al. (2022) which visualizes the largest connected component since the baselines fail to consistently generate connected graphs. For DruM, we found that 92% of the generated graphs are connected.

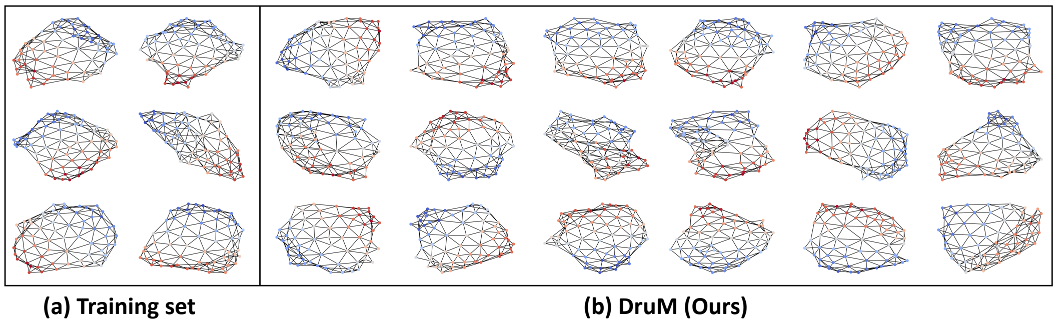


Figure 7: **Visualization of graphs from the Planar dataset and the generated graphs of DruM.**

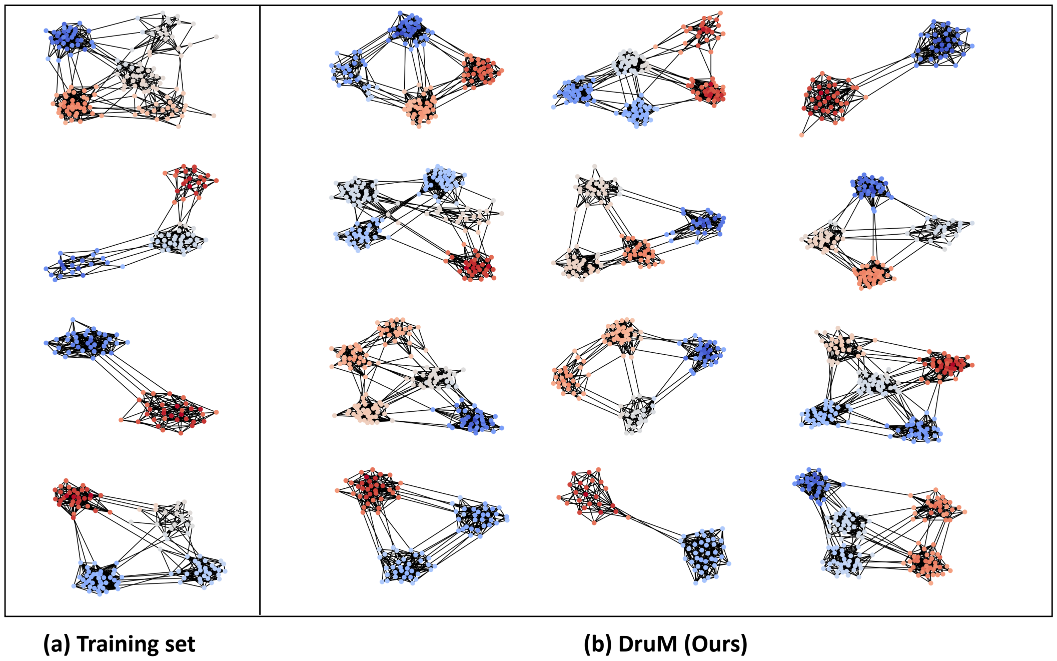


Figure 8: **Visualization of graphs from the SBM dataset and the generated graphs of DruM.**

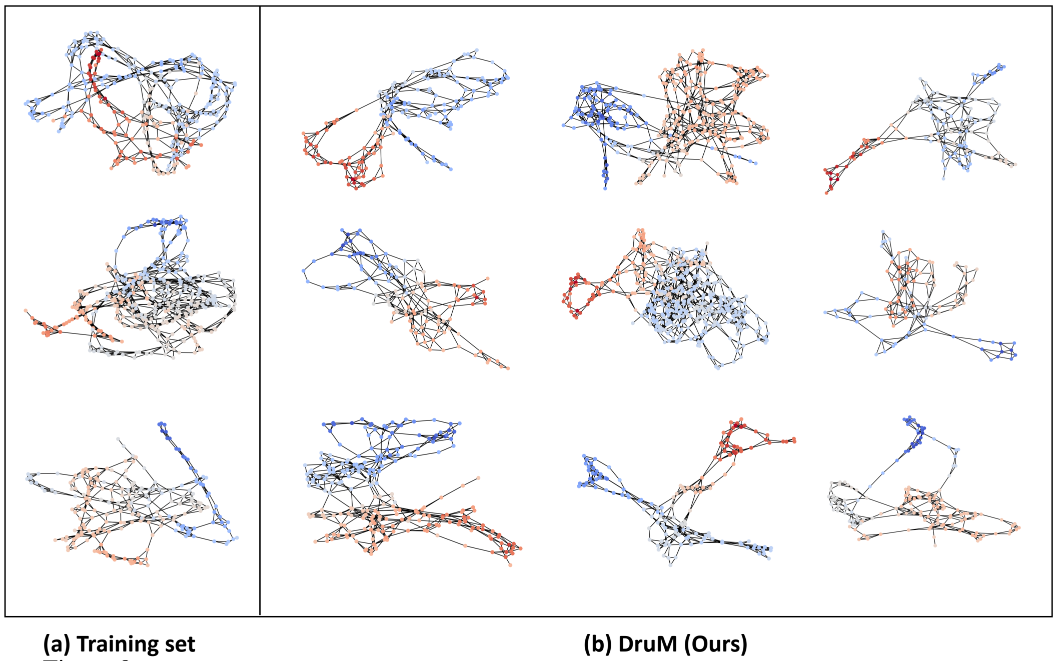


Figure 9: **Visualization of graphs from the Proteins dataset and the generated graphs of DruM.**

2D Molecules We provide the visualization of the molecules from the training set and the generated 2D molecules in Figure 10 and 11. These molecules are randomly selected from the training set and the generated molecule set.

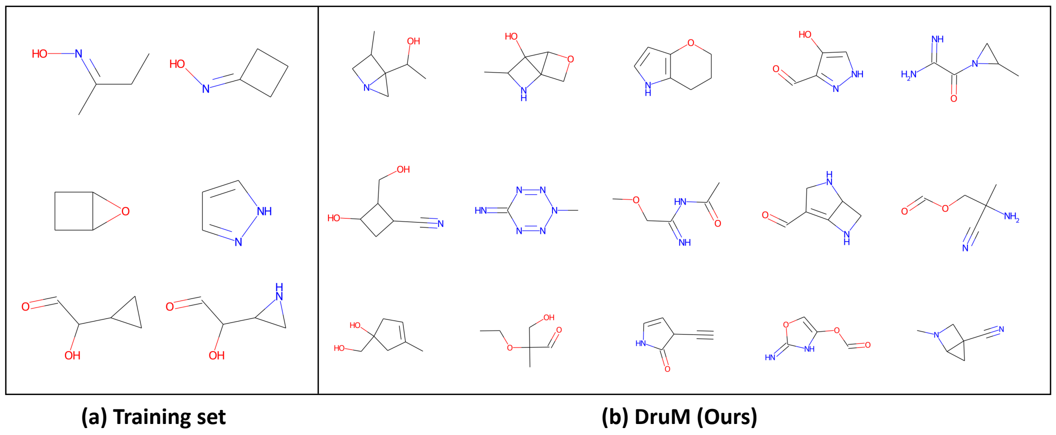


Figure 10: **Visualization of molecules from the QM9 dataset and the generated molecules of DruM for the 2D molecule generation experiment.**

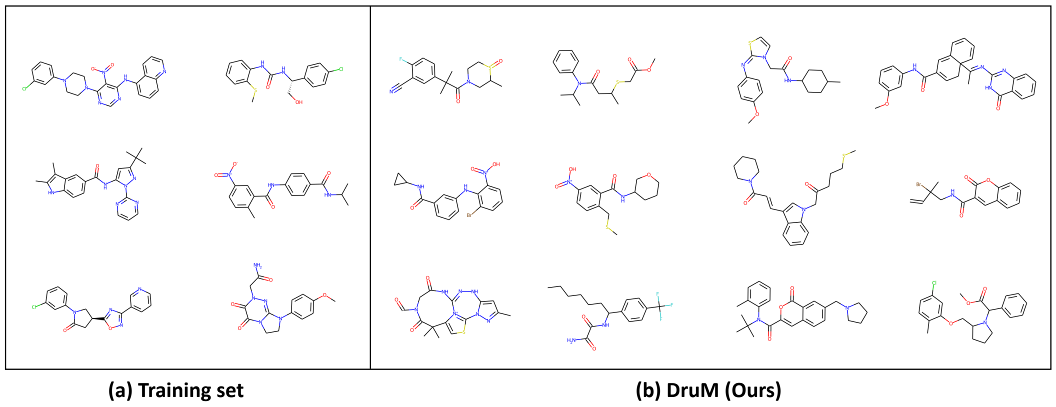


Figure 11: **Visualization of the molecules from the ZINC250k dataset and the generated molecules of DruM for the 2D molecule generation experiment.**

3D Molecules We visualize the generated molecule for the 3D molecule generation experiment in Figure 12, and 13. Note that the visualized molecules are all stable. For the GEOM-DRUGS experiment, we observe that a few of the generated molecules are not connected as pointed out in Hoogeboom et al. (2022). To measure how many graphs are connected, we report the percentage of the connected graphs. Table 5 shows that DruM can generate a significantly larger number of connected molecules compared to EDM Hoogeboom et al. (2022).

Table 5: Percentage of connected graphs on GEOM-DRUGS experiment with 3 different runs.

Methods	Connected (%)
EDM	37.70 $\pm$0.79
DruM (Ours)	**56.57** $\pm$0.31

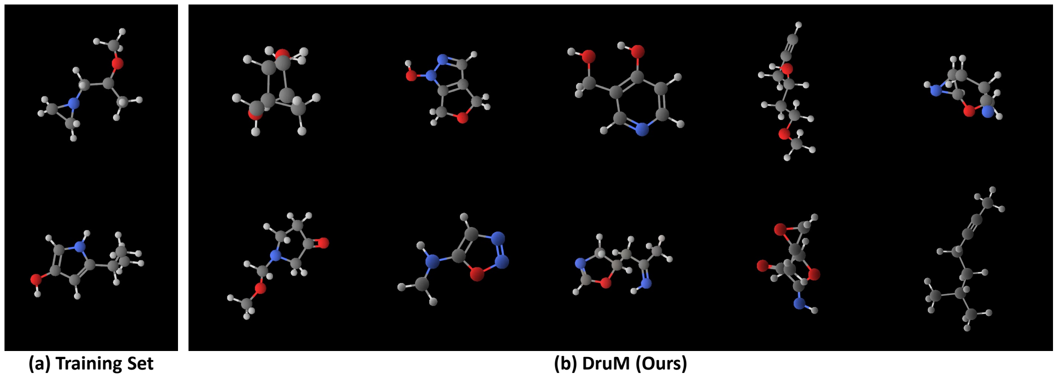


Figure 12: **Visualization of the molecules from the QM9 dataset and the generated molecules of DruM for the 3D molecule generation experiment.**

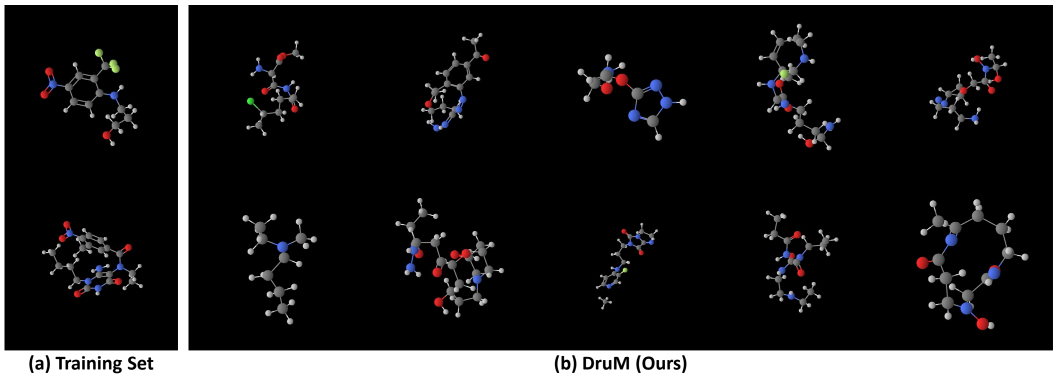


Figure 13: **Visualization of the molecules from the GEOM-DRUGS dataset and the generated molecules of DruM for the 3D molecule generation experiment.**

E.2 Generative Process of DruM

Here we visualize the generative process of DruM. We visualize the generative process of general graphs in Figure 14, 15, and 16. We also visualize the generative process of the 3D molecules in Figure 17 and 18. We further provide the animation of the generative process in the attached supplementary file.

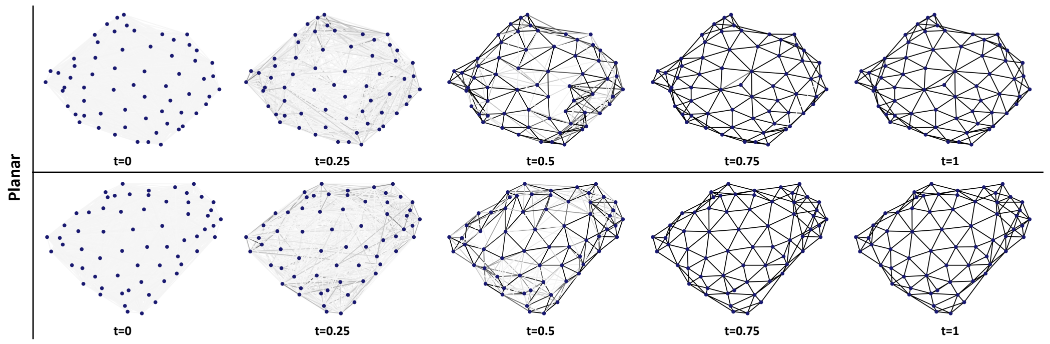


Figure 14: **Visualization of the generative process of DruM.** We visualize the destination mixture from DruM on the Planar dataset.

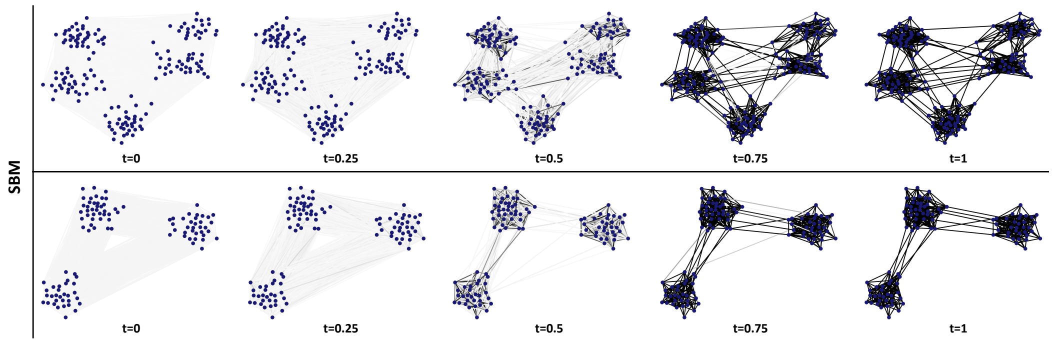


Figure 15: **Visualization of the generative process of DruM.** We visualize the destination mixture from DruM on the SBM dataset.

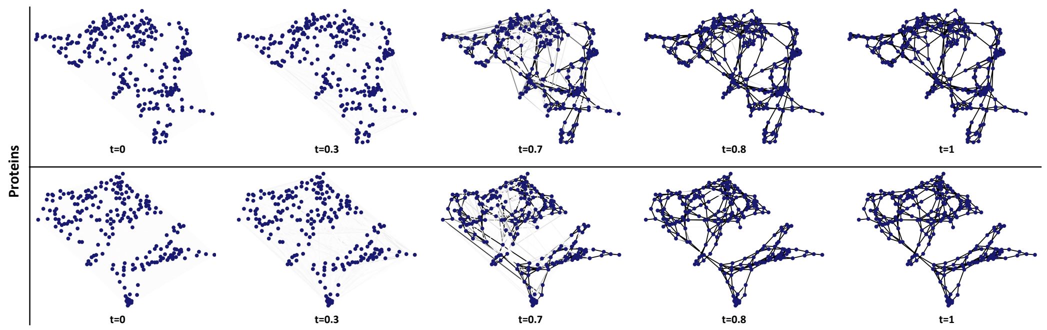


Figure 16: **Visualization of the generative process of DruM.** We visualize the destination mixture from DruM on the Proteins dataset.

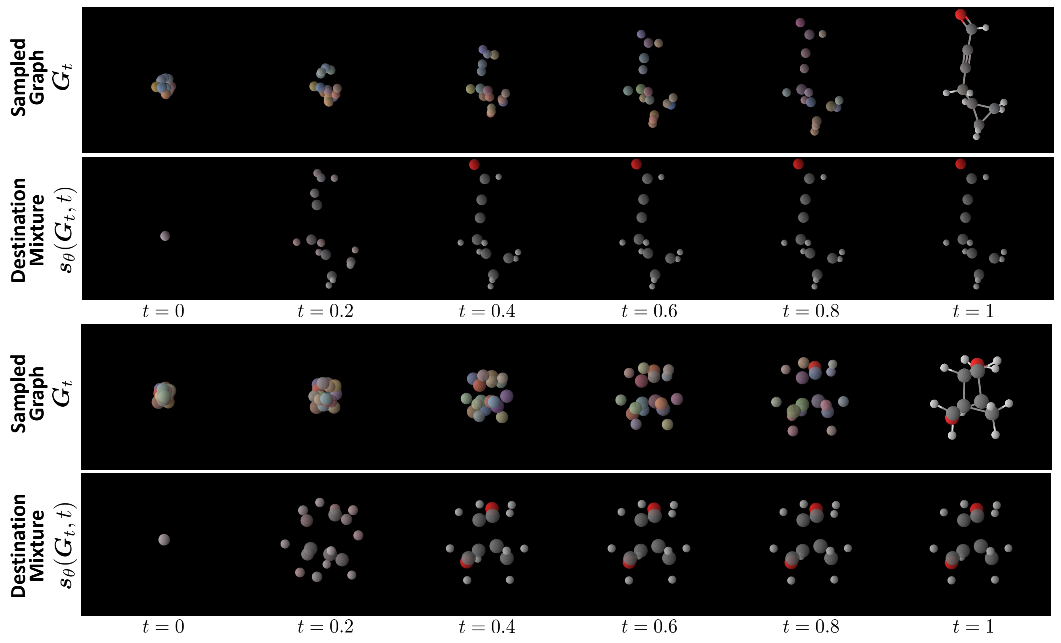


Figure 17: **Visualization of the 3D molecule generative process** of DruM on QM9 dataset. In the first row, we visualize the trajectory $\boldsymbol{G}_t$, and in the second row, we visualize the estimated destination mixtures from DruM. Note that the visualized molecules are stable.

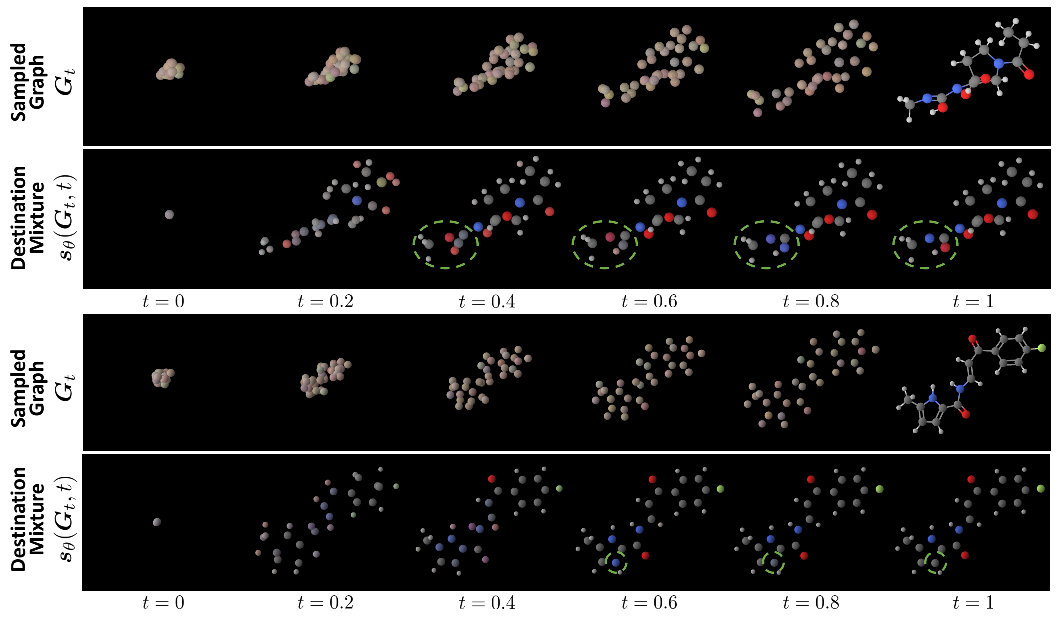


Figure 18: **Visualization of the 3D molecule generative process** of DruM on GEOM-DRUGS dataset. In the first row, we visualize the trajectory $\boldsymbol{G}_t$, and in the second row, we visualize the destination mixtures from DruM. Note that the visualized molecules are stable. The atom types and 3D coordinates in the green circle are calibrated after the estimated destination mixtures are converged at an early stage.