Discrete-state Continuous-time Diffusion for Graph Generation

Zhe Xu* **Ruizhong Qiu*** **Yuzhong Chen†** **Huiyuan Chen†** **Xiran Fan†** **Menghai Pan†**

Zhichen Zeng* **Mahashweta Das†** **Hanghang Tong***

Abstract

Graph is a prevalent discrete data structure, whose generation has wide applications such as drug discovery and circuit design. Diffusion generative models, as an emerging research focus, have been applied to graph generation tasks. Overall, according to the space of *states* and *time* steps, diffusion generative models can be categorized into discrete-/continuous-state discrete-/continuous-time fashions. In this paper, we formulate the graph diffusion generation in a discrete-state continuous-time setting, which has never been studied in previous graph diffusion models. The rationale of such a formulation is to preserve the discrete nature of graph-structured data and meanwhile provide flexible sampling trade-offs between sample quality and efficiency. Analysis shows that our training objective is closely related to the generation quality and our proposed generation framework enjoys ideal invariant/equivariant properties concerning the permutation of node ordering. Our proposed model shows competitive empirical performance against state-of-the-art graph generation solutions on various benchmarks and at the same time can flexibly trade off the generation quality and efficiency in the sampling phase.

1 Introduction

Graph generation has been studied for a long time with broad applications, based on either the one-shot (i.e., one-step) [50, 39, 56, 51, 72, 32] or auto-regressive generation paradigm [82, 29, 42, 52]. The former generates all the graph components at once and the latter does that sequentially. A recent trend of applying diffusion generative models [67, 23, 70] to graph generation tasks attracts increasing attentions because of its excellent performance and solid theoretical foundation. In this paper, we follow the one-shot generation paradigm, the same as most graph diffusion generative models.

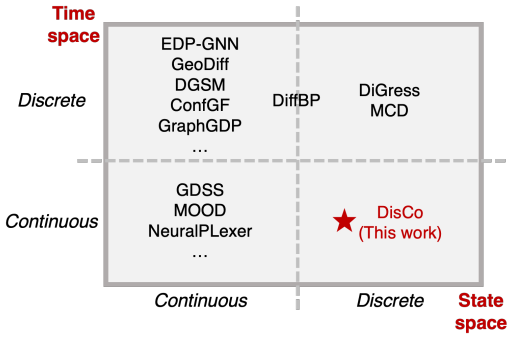


Figure 1: A taxonomy of graph diffusion models.

Some earlier attempts at graph diffusion models treat the graph data in a continuous state space by viewing the graph topology and features as continuous variables [56]. Such a formulation departs from the discrete nature of graph-structured data; e.g., topological sparsity is lost and the discretization in the generation process requires extra hyper-parameters. DiGress [73] is one of the early efforts

*University of Illinois Urbana-Champaign. {zhexu3, rq5, zhichenz, htong}@illinois.edu
†Visa Research. {yuzchen, hchen, xirafan, menpan, mahdas}@visa.com

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-2531

applying discrete-state diffusion models to graph generation tasks and is the current state-of-the-art graph diffusion generative model. However, DiGress is defined in the discrete time space whose generation is inflexible. This is because, its number of sampling steps must match the number of forward diffusion steps, which is a fixed hyperparameter after the model finishes training. A unique advantage of the continuous-time diffusion models [70, 32] lies in their flexible sampling process, and its simulation complexity is proportional to the number of sampling steps, determined by the step size of various numerical approaches (e.g., τ-leaping [18, 8, 71]) and decoupled from the models' training. Thus, a discrete-state continuous-time diffusion model is highly desirable for graph generation tasks.

Driven by the recent advance of continuous-time Markov Chain (CTMC)-based diffusion generative model [8], we incorporate the ideas of CTMC into the corruption and denoising of graph data and propose the first discrete-state continuous-time graph diffusion generative model. It shares the same advantages as DiGress by preserving the discrete nature of graph data and meanwhile overcomes the drawback of the nonadjustable sampling process in DiGress. This Discrete-state Continuous-time graph diffusion model is named DISCO.

DISCO bears several desirable properties and advantages. First, despite its simplicity, the training objective has a rigorously proved connection to the sampling error. Second, its formulation includes a parametric graph-to-graph mapping, named backbone model, whose input-output architecture is shared between DISCO and DiGress. Therefore, the graph transformer (GT)-based backbone model [54] from DiGress can be seamlessly plugged into DISCO. Third, a concise message-passing neural network backbone model is explored with DISCO, which is simpler than the GT backbone and has decent empirical performance. Last but not least, our analyses show that the forward and reverse diffusion process in DISCO can retain the permutation-equivariant/invariant properties for its training loss and sampling distribution, both of which are critical and practical inductive biases on graph data.

Comprehensive experiments on plain and molecule graphs show that DISCO can obtain competitive or superior performance against state-of-the-art graph generative models and provide additional sampling flexibility. Our main contributions are summarized:

- **Model.** We propose the first discrete-state continuous-time graph diffusion model, DISCO. We utilize the successful graph-to-graph neural network architecture from DiGress and further explore a new lightweight backbone model with decent efficacy.
- **Analysis.** Our analysis reveals (1) the key connection between the training loss and the approximation error (Theorem 3.3) and (2) invariant/equivariant properties of DISCO in terms of the permutation of nodes (Theorems 3.8 and 3.9).
- **Experiment.** Extensive experiments validate the empirical performance of DISCO.

2 Preliminaries

2.1 Discrete-State Continuous-time Diffusion Models

A D-dimensional discrete state space is represented as $\mathcal{X} = \{1, \dots, C\}^D$. A continuous-time Markov Chain (CTMC) $\{\mathbf{x}_t = [x_t^1, \cdots x_t^D]\}_{t\in[0,T]}$ is characterized by its (time-dependent) rate matrix $\mathbf{R}_t \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$. Here $\mathbf{x}_t$ is the state at the time step t. The transition probability $q_{t|s}$ between from time s to t satisfies the Kolmogorov forward equation, for $s < t$,

$$\frac{d}{dt} q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = \sum_{\xi\in\mathcal{X}} q_{t|s}(\xi|\mathbf{x}_s)\mathbf{R}_t(\xi, \mathbf{x}_t), \tag{1}$$

The marginal distribution can be represented as $q_t(\mathbf{x}_t) = \sum_{\mathbf{x}_0\in\mathcal{X}} q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\pi_{\texttt{data}}(\mathbf{x}_0)$ where $\pi_{\texttt{data}}(\mathbf{x}_0)$ is the data distribution. If the CTMC is defined in time interval $[0, T]$ and if the rate matrix $\mathbf{R}_t$ is well-designed, the final distribution $q_T(\mathbf{x}_T)$ can be close to a tractable reference distribution $\pi_{\texttt{ref}}(\mathbf{x}_T)$, e.g., uniform distribution. We notate the reverse stochastic process as $\tilde{\mathbf{x}}_t = \mathbf{x}_{T-t}$; a well-known fact (e.g., Section 5.9 in [63]) is that the reverse process $\{\tilde{\mathbf{x}}_t\}_{t\in[0,T]}$ is also a CTMC, characterized by the reverse rate matrix: $\tilde{\mathbf{R}}_t(\mathbf{x}, \mathbf{y}) = \frac{q(\mathbf{y})}{q(\mathbf{x})}\mathbf{R}_t(\mathbf{y}, \mathbf{x})$. The goal of the CTMC-based diffusion models is an accurate estimation of the reverse rate matrix $\tilde{\mathbf{R}}_t$ so that new data can be generated by sampling the reference distribution $\pi_{\texttt{ref}}$ and then simulating the reverse CTMC [16, 17, 18, 1]. However, the complexity of the rate matrix is prohibitively high because there are C^D possible states. A reasonable simplification is to factorize the process over dimensions [8, 71, 73, 2]. Specifically,

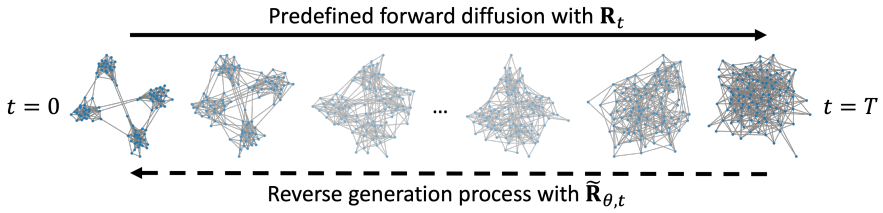


Figure 2: An overview of DISCO. A transition can happen at any time in $[0, T]$.

the forward process is factorized as $q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = \prod_{d=1}^{D} q_{t|s}(x_t^d|x_s^d)$, for $s < t$. Then, the forward diffusion of each dimension is independent and is governed by dimension-specific forward rate matrices $\{\mathbf{R}_t^d\}_{d=1}^D$. With such a factorization, the goal is to estimate the dimension-specific reverse rate matrices $\{\tilde{\mathbf{R}}_t^d\}_{d=1}^D$.

The dimension-specific reverse rate is represented as $\tilde{\mathbf{R}}_t^d(x^d, y^d) = \sum_{x_0^d} \mathbf{R}_t^d(y^d, x^d) \frac{q_{t|0}(y^d|x_0^d)}{q_{t|0}(x^d|x_0^d)} q_{0|t}(x_0^d|\mathbf{x})$. Campbell et al. [8] estimate $q_{0|t}(x_0^d|\mathbf{x})$ via a neural network p_θ such that $p_\theta(x_0^d|\mathbf{x}, t) \approx q_{0|t}(x_0^d|\mathbf{x})$; Sun et al. [71] propose another singleton conditional distribution-based objective $\frac{p_\theta(y^d|\mathbf{x}^{\setminus d},t)}{p_\theta(x^d|\mathbf{x}^{\setminus d},t)} \approx \frac{q(y^d|\mathbf{x}^{\setminus d})}{q(x^d|\mathbf{x}^{\setminus d})}$ whose rationale is Brook's Lemma [5, 49].

2.2 Graph Generation and Notations

We study the graphs with *categorical* node and edge attributes. A graph with n nodes is represented by its edge type matrix and node type vector: $\mathcal{G} = (\mathbf{E}, \mathbf{F})$, where $\mathbf{E} = (e^{(i,j)})_{i,j \in \mathbb{N}^+_{\leq n}} \in \{1, \dots, a+1\}^{n \times n}$, $\mathbf{F} = (f^i)_{i \in \mathbb{N}^+_{\leq n}} \in \{1, \dots, b\}^n$, a and b are the numbers of node and edge types, respectively. Notably, the absence of an edge is viewed as a special edge type, so there are $(a+1)$ edge types in total. The problem we study is graph generation where N graphs $\{\mathcal{G}^i\}_{i \in \mathbb{N}^+_{\leq N}}$ from an inaccessible graph data distribution $\mathfrak{G}$ are given and we aim to generate M graphs $\{\mathcal{G}^i\}_{i \in \mathbb{N}^+_{\leq M}}$ from $\mathfrak{G}$.

3 Method

This section presents the proposed discrete-state continuous-time graph diffusion model, DISCO whose overview is Figure 2. Section 3.1 introduces the necessity to factorize the diffusion process and Section 3.2 details the forward process. Our training objective and its connection to sampling are introduced in Sections 3.3 and 3.4, respectively. Last but not least, a specific neural architecture of the graph-to-graph backbone model and its properties regarding the permutation of node ordering are introduced in Sections 3.5 and 3.6, respectively. *All proofs are in Appendix.*

3.1 Factorized Discrete Graph Diffusion Process

The number of possible states of an n-node graph is $(a+1)^{n^2} \times b^n$ which is intractably large. Thus, we follow existing discrete models [2, 8, 71, 73] and formulate the forward processes on every node/edge to be independent. Mathematically, the forward diffusion process for $s < t$ is factorized as

$$q_{t|s}(\mathcal{G}_t|\mathcal{G}_s) = \prod_{i,j=1}^{n} q_{t|s}(e_t^{(i,j)}|e_s^{(i,j)}) \prod_{i=1}^{n} q_{t|s}(f_t^i|f_s^i) \tag{2}$$

where the edge type transition probabilities $\{q_{t|s}(e_t^{(i,j)}|e_s^{(i,j)})\}_{i,j \in \mathbb{N}^+_{\leq n}}$ and node type transition probabilities $\{q_{t|s}(f_t^i|f_s^i)\}_{i \in \mathbb{N}^+_{\leq n}}$ are characterized by their forward rate matrices $\{\mathbf{R}_t^{(i,j)}\}_{i,j \in \mathbb{N}^+_{\leq n}}$ and $\{\mathbf{R}_t^i\}_{i \in \mathbb{N}^+_{\leq n}}$, respectively. The forward processes, i.e., the forward rate matrices in our context, are predefined, which will be introduced in Section 3.2. Given the factorization of forward transition probability in Eq. (2), a question is raised: *what is the corresponding factorization of the forward rate matrix ($\mathbf{R}_t$) and the reverse rate matrix ($\tilde{\mathbf{R}}_t$)?* Remark 3.1 shows such a factorization.

https://doi.org/10.52202/079017-2531

Algorithm 1 Training of DISCO

1: A training graph $\mathcal{G}_0 = (\{f_0^i\}, \{e_0^{(i,j)}\})$ is given.
2: Sample $t \sim \mathcal{U}_{(0,T)}$; sample $\mathcal{G}_t$ based on transition probabilities $q_{t|0}(f_t = v|f_0 = u) = (e^{\int_0^t \beta(s)\mathbf{R}_f ds})_{uv}$ and $q_{t|0}(e_t = v|e_0 = u) = (e^{\int_0^t \beta(s)\mathbf{R}_e ds})_{uv}$, given $\mathcal{G}_0 = (\{f_0^i\}, \{e_0^{(i,j)}\})$.
3: Predict the clean graph $\hat{\mathcal{G}}_0 = (\{\hat{f}_0^i\}, \{\hat{e}_0^{(i,j)}\}) \leftarrow \Big(\{p^\theta_{0|t}(f^i|\mathcal{G}_t)\}, \{p^\theta_{0|t}(e^{(i,j)}|\mathcal{G}_t)\}\Big)$, given $\mathcal{G}_t$.
4: Compute cross-entropy loss between $\mathcal{G}_0$ and $\hat{\mathcal{G}}_0$ based on Eq. (6) and update θ.

Remark 3.1. (Factorization of rate matrices, extended from Proposition 3 of [8]) Given the factorized forward process Eq. (2), the overall rate matrices are factorized as

$$\mathbf{R}_t(\bar{\mathcal{G}}, \mathcal{G}) = \sum_i A_t^i + \sum_{i,j} B_t^{(i,j)} \tag{3}$$

$$\tilde{\mathbf{R}}_t(\mathcal{G}, \bar{\mathcal{G}}) = \sum_i A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} q_{0|t}(f_0^i|\mathcal{G}) + \sum_{i,j} B_t^{(i,j)} \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} q_{0|t}(e_0^{(i,j)}|\mathcal{G}) \tag{4}$$

where $A_t^i = \mathbf{R}_t^i(\bar{f}^i, f^i)\delta_{\bar{\mathcal{G}}\setminus\bar{f}^i, \mathcal{G}\setminus f^i}$, $B_t^{(i,j)} = \mathbf{R}_t^{(i,j)}(\bar{e}^{(i,j)}, e^{(i,j)})\delta_{\bar{\mathcal{G}}\setminus\bar{e}^{(i,j)}, \mathcal{G}\setminus e^{(i,j)}}$, the operator $\delta_{\bar{\mathcal{G}}\setminus\bar{f}^i, \mathcal{G}\setminus f^i}$ (or $\delta_{\bar{\mathcal{G}}\setminus\bar{e}^{(i,j)}, \mathcal{G}\setminus e^{(i,j)}}$) checks whether two graphs $\bar{\mathcal{G}}$ and $\mathcal{G}$ are exactly the same except for node i (or the edge between nodes i and j).

Note that this factorization itself is not our contribution but a necessary part of our framework, so we mention it here for completeness. Its full derivation is in Appendix - Section A. Next, we detail the design of forward rate matrices.

3.2 Forward Process

A proper choice of the forward rate matrices $\{\mathbf{R}_t^{(i,j)}\}_{i,j\in\mathbb{N}^+_{\leq n}}$ and $\{\mathbf{R}_t^i\}_{i\in\mathbb{N}^+_{\leq n}}$ is important because (1) the probability distributions of node and edge types, $\{q(f_t^i)\}_{i\in\mathbb{N}^+_{\leq n}}$ and $\{q(e_t^{(i,j)})\}_{i,j\in\mathbb{N}^+_{\leq n}}$, should converge to their reference distributions within $[0, T]$ and (2) the reference distributions should be easy to sample (e.g., uniform distribution). We follow [8] to formulate $\mathbf{R}_t^{(i,j)} = \beta(t)\mathbf{R}_e^{(i,j)}$, $\forall i, j$ and $\mathbf{R}_t^i = \beta(t)\mathbf{R}_f^i$, $\forall i$, where $\beta(t)$ is a corruption schedule, $\{\mathbf{R}_e^{(i,j)}\}$ and $\{\mathbf{R}_f^i\}$ are the base rate matrices. For brevity, we set all the nodes/edges to share a common node/edge rate matrix, i.e., $\mathbf{R}_e^{(i,j)} = \mathbf{R}_e$ and $\mathbf{R}_f^i = \mathbf{R}_f$, $\forall i, j$. Then, the forward transition probability for all the nodes and edges are $q_{t|0}(f_t = v|f_0 = u) = (e^{\int_0^t \beta(s)\mathbf{R}_f ds})_{uv}$ and $q_{t|0}(e_t = v|e_0 = u) = (e^{\int_0^t \beta(s)\mathbf{R}_e ds})_{uv}$, respectively. We omit the superscript i (or (i, j)) because the transition probability is shared by all the nodes (or edges). The detailed derivation of the above analytic forward transition probability is provided in Appendix - Section B.

For categorical data, a reasonable reference distribution is a uniform distribution, i.e., $\pi_f = \frac{\mathbf{1}}{b}$ for nodes and $\pi_e = \frac{\mathbf{1}}{a+1}$ for edges. In addition, inspired by [73], we find that node and edge marginal distributions $\mathbf{m}_f$ and $\mathbf{m}_e$ are good choices as the reference distributions. Concretely, an empirical estimation of $\mathbf{m}_f$ and $\mathbf{m}_e$ is to count the number of node/edge types and normalize them. The following proposition shows how to design the rate matrices to guide the forward process to converge to uniform and marginal distributions.

Proposition 3.2. *The forward processes for nodes and edges converge to uniform distributions if* $\mathbf{R}_f = \mathbf{1}\mathbf{1}^\top - b\mathbf{I}$ *and* $\mathbf{R}_e = \mathbf{1}\mathbf{1}^\top - (a+1)\mathbf{I}$*; they converge to marginal distributions* $\mathbf{m}_f$ *and* $\mathbf{m}_e$ *if* $\mathbf{R}_f = \mathbf{1}\mathbf{m}_f^\top - \mathbf{I}$ *and* $\mathbf{R}_e = \mathbf{1}\mathbf{m}_e^\top - \mathbf{I}$*.* $\mathbf{1}$ *is an all-one vector and* $\mathbf{I}$ *is an identity matrix.*

Regarding the selection of $\beta(t)$, we follow [23, 70, 8] and set $\beta(t) = \alpha\gamma^t \log(\gamma)$ for a smooth change of the rate matrix. α and γ are hyperparameters. Detailed settings are in Appendix F.3.

3.3 Parameterization and Optimization Objective

Next, we introduce the estimation of the reverse process from its motivation. The reverse process is essentially determined by the reverse rate matrix $\tilde{\mathbf{R}}_t$ in Eq. (4), whose computation needs $q_{0|t}(f_0^i|\mathcal{G})$ and $q_{0|t}(e_0^{(i,j)}|\mathcal{G})$, $\forall i, j$; their exact estimation is expensive because according to Bayes' rule, $p_t(\mathcal{G})$ is needed, whose computation needs to enumerate all the given graphs: $p_t(\mathcal{G}) = \sum_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0)\pi_{\texttt{data}}(\mathcal{G}_0)$.

Thus, we propose parameterizing the reverse transition probabilities via a neural network θ whose specific architecture is introduced in Section 3.5. The terms $\{q_{0|t}(f_0^i|\mathcal{G})\}_{i\in\mathbb{N}^+_{\leq n}}$ and $\{q_{0|t}(e_0^{(i,j)}|\mathcal{G})\}_{i,j\in\mathbb{N}^+_{\leq n}}$ in Eq. (4) are replaced with the parameterized $\{p^\theta_{0|t}(f^i|\mathcal{G})\}_{i\in\mathbb{N}^+_{\leq n}}$ and $\{p^\theta_{0|t}(e^{(i,j)}|\mathcal{G})\}_{i,j\in\mathbb{N}^+_{\leq n}}$. Thus, a parameterized reverse rate matrix $\tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}})$ is represented as $\tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}}) = \sum_i \tilde{\mathbf{R}}^i_{\theta,t}(f^i, \bar{f}^i) + \sum_{i,j} \tilde{\mathbf{R}}^{(i,j)}_{\theta,t}(e^{(i,j)}, \bar{e}^{(i,j)})$ where $\tilde{\mathbf{R}}^i_{\theta,t}(f^i, \bar{f}^i) = A^i_t \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} p^\theta_{0|t}(f_0^i|\mathcal{G})$, $\tilde{\mathbf{R}}^{(i,j)}_{\theta,t}(e^{(i,j)}, \bar{e}^{(i,j)}) = B^{(i,j)}_t \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} p^\theta_{0|t}(e_0^{(i,j)}|\mathcal{G})$, and the remaining notations are the same as Eq. (4). Note that all the terms $\{p^\theta_{0|t}(f^i|\mathcal{G})\}_{i\in\mathbb{N}^+_{\leq n}}$ and $\{p^\theta_{0|t}(e^{(i,j)}|\mathcal{G})\}_{i,j\in\mathbb{N}^+_{\leq n}}$ can be viewed together as a graph-to-graph mapping $\theta : \mathfrak{G} \mapsto \mathfrak{G}$, whose input is the noisy graph $\mathcal{G}_t$ and its output is the predicted clean graph probabilities, concretely, the node/edge type probabilities of all the nodes and edges.

Intuitively, the discrepancy between the groundtruth $\tilde{\mathbf{R}}_t$ (from Eq. (4)) and the parametric $\tilde{\mathbf{R}}_{\theta,t}$ should be small. Theorem 3.3 establishes a cross-entropy (CE)-based upper bound of such a discrepancy, where the estimated probability vectors (sum is 1) are notated as $\hat{f}_0^i = [p^\theta_{0|t}(f^i = 1|\mathcal{G}_t), \ldots, p^\theta_{0|t}(f^i = b|\mathcal{G}_t)]^\top \in [0,1]^b$ and $\hat{e}_0^{(i,j)} = [p^\theta_{0|t}(e^{(i,j)} = 1|\mathcal{G}_t), \ldots, p^\theta_{0|t}(e^{(i,j)} = a + 1|\mathcal{G}_t)]^\top \in [0,1]^{a+1}$.

Theorem 3.3 (Approximation error)**.** *for $\mathcal{G} \neq \bar{\mathcal{G}}$*

$$\left|\tilde{\mathbf{R}}_t(\mathcal{G}, \bar{\mathcal{G}}) - \tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}})\right|^2 \leq C_t + C_t^{\texttt{node}} \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}\big(\texttt{One-Hot}(f_0^i), \hat{f}_0^i\big) + C_t^{\texttt{edge}} \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_{i,j} \mathcal{L}_{\texttt{CE}}\big(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)}\big) \tag{5}$$

where C_t, $C_t^{\texttt{node}}$, and $C_t^{\texttt{edge}}$ are constants independent on θ but dependent on t, $\mathcal{G}$, and $\bar{\mathcal{G}}$; One-Hot *transforms f_0^i and $e_0^{(i,j)}$ into one-hot vectors.*

The bound in Theorem 3.3 is tight, i.e., the right-hand side of Eq. (5) is 0, whenever $\hat{f}_0^i = q_{0|t}(f_0^i|\mathcal{G}_t), \forall i$ and $\hat{e}_0^{(i,j)} = q_{0|t}(e_0^{(i,j)}|\mathcal{G}_t), \forall i, j$. Guided by Theorem 3.3, we (1) take expectation of t by sampling t from a uniform distribution $t \sim \mathcal{U}_{(0,T)}$ and (2) simplify the right-hand side of Eq. (5) by using the unweighted CE loss as our training objective:

$$\min_\theta \; T\mathbb{E}_t \mathbb{E}_{\mathcal{G}_0} \mathbb{E}_{q_{t|0}(\mathcal{G}_t|\mathcal{G}_0)} \Big[\sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) + \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)}) \Big] \tag{6}$$

A step-by-step training algorithm is in Algorithm 1. Note that the above CE loss has been used in some diffusion models (e.g., [2, 8]) but lacks a good motivation, especially in the continuous-time setting. We motivate it based on the rate matrix discrepancy, as a unique contribution of this paper.

3.4 Sampling Reverse Process

Given the parametric reverse rate matrix $\tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}})$, the graph generation process can be implemented by two steps: (1) sampling the reference distribution $\pi_{\texttt{ref}}$ (i.e., π_f for nodes and π_e for edges) and (2) numerically simulating the CTMC from time T to 0. The exact simulation of a CTMC has been studied for a long time, e.g., [16, 17, 1]. However, their simulation strategies only allow one transition (e.g., one edge/node type change) per step, which is highly inefficient for graphs as the number of nodes and edges is typically large; once a(n) node/edge is updated, $\tilde{\mathbf{R}}_{\theta,t}$ requires recomputation. A

https://doi.org/10.52202/079017-2531

practical approximation is to assume $\tilde{\mathbf{R}}_{\theta,t}$ is fixed during a time interval $[t-\tau, t]$, i.e., delaying the happening of transitions in $[t-\tau, t]$ and triggering them all together at the time $t-\tau$; this strategy is also known as τ-leaping [18, 8, 71], and DISCO adopts it.

We elaborate on τ-leaping for transitions of node types; the transitions of edge types are similar. The rate matrix of the i-th node is fixed as $\tilde{\mathbf{R}}^i_{\theta,t}(f^i, \bar{f}^i) = \mathbf{R}^i_t(\bar{f}^i, f^i) \sum_{f^i_0} \frac{q_{t|0}(\bar{f}^i|f^i_0)}{q_{t|0}(f^i|f^i_0)} p^\theta_{0|t}(f^i|\mathcal{G}_t)$, during $[t-\tau, t]$. According to the definition of rate matrix, in $[t-\tau, t]$, the number of transitions from f^i to $\bar{f}^i$, namely $J_{f^i,\bar{f}^i}$, follows the Poisson distribution, i.e., $J_{f^i,\bar{f}^i} \sim \text{Poisson}(\tau \tilde{\mathbf{R}}^i_{\theta,t}(f^i, \bar{f}^i))$. For categorical data (e.g., node type), multiple transitions in $[t-\tau, t]$ are invalid and meaningless. In other words, for the i-th node, if the total number of transitions $\sum_{\bar{f}^i} J_{f^i,\bar{f}^i} > 1$, f^i keeps unchanged in $[t-\tau, t]$; otherwise, if $\sum_{\bar{f}^i} J_{f^i,\bar{f}^i} = 1$ and $J_{f^i,s} = 1$, i.e., there is exact 1 transition, f^i jumps to s. A step-by-step sampling algorithm (Algorithm 2) is in Appendix.

Remark 3.4. The sampling error of τ-leaping is linear to $C_{\texttt{err}}$ [8], the approximation error of the reverse rates: $\sum_{\mathcal{G}\neq\bar{\mathcal{G}}} \left|\tilde{\mathbf{R}}_t(\mathcal{G}, \bar{\mathcal{G}}) - \tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}})\right| \leq C_{\texttt{err}}$. Interested readers are referred to Theorem 1 from [8]. Our Theorem 3.3 shows the connection between our training loss and $C_{\texttt{err}}$, which further verifies the correctness of our training loss.

3.5 Model Instantiation

As mentioned in Section 3.3, the parametric backbone $p^\theta_{0|t}(\mathcal{G}_0|\mathcal{G}_t)$ is a graph-to-graph mapping whose input is the noisy graph $\mathcal{G}_t$ and its output is the predicted denoised graph $\mathcal{G}_0$. There exists a broad range of neural network architectures. Notably, DiGress [73] uses a graph Transformer (GT) as $p^\theta_{0|t}$, a decent reference for our continuous-time framework. We name our model with the GT backbone as DISCO-GT and its detailed configuration is in Appendix F.3. The main advantage of the GT is its long-range interaction thanks to the complete self-attention graph; however, the architecture is very complex and includes multi-head self-attention modules, leading to expensive computation.

Beyond GTs, in this paper, we posit that a regular message-passing neural network (MPNN) [19] should be a promising choice for $p^\theta_{0|t}(\mathcal{G}_0|\mathcal{G}_t)$. It is recognized that the MPNNs' expressiveness might not be as good as GTs' [33, 7], e.g., in terms of long-range interactions. However, in our setting, the absence of an edge is viewed as a special type of edge and the whole graph is complete; therefore, such a limitation of MPNN is naturally mitigated, which is verified by our empirical evaluations.

Concretely, an MPNN-based graph-to-graph mapping is presented as follows, and DISCO with MPNN backbone is named DISCO-MPNN. Given a graph $\mathcal{G} = (\mathbf{E}, \mathbf{F})$, where $\mathbf{E} \in \{1, \ldots, a, a+1\}^{n\times n}$, $\mathbf{F} \in \{1, \ldots, b\}^n$, we first transform both the matrix $\mathbf{E}$ and $\mathbf{F}$ into one-hot embeddings $\mathbf{E}_{\texttt{OH}} \in \{0,1\}^{n\times n\times(a+1)}$ and $\mathbf{F}_{\texttt{OH}} \in \{0,1\}^{n\times b}$. Then, some auxiliary features (e.g., the # of specific motifs) are extracted: $\mathbf{F}_{\texttt{aux}}, \mathbf{y}_{\texttt{aux}} = \texttt{Aux}(\mathbf{E}_{\texttt{OH}})$ to overcome the expressiveness limitation of MPNNs [11]. Here $\mathbf{F}_{\texttt{aux}}$ and $\mathbf{y}_{\texttt{aux}}$ are the node and global auxiliary features, respectively. Note that a similar auxiliary feature engineering is also applied in DiGress [73]. More details about the Aux can be found in Appendix E. Then, three multi-layer perceptrons (MLPs) are used to map node features $\mathbf{F}_{\texttt{OH}} \oplus \mathbf{F}_{\texttt{aux}}$, edge features $\mathbf{E}_{\texttt{OH}}$, and global features $\mathbf{y}_{\texttt{aux}}$ into a common hidden space as $\mathbf{F}_{\texttt{hidden}} = \texttt{MLP}(\mathbf{F}_{\texttt{OH}} \oplus \mathbf{F}_{\texttt{aux}})$, $\mathbf{E}_{\texttt{hidden}} = \texttt{MLP}(\mathbf{E}_{\texttt{OH}})$, $\mathbf{y}_{\texttt{hidden}} = \texttt{MLP}(\mathbf{y}_{\texttt{aux}})$, where $\oplus$ is a concatenation operator. The following formulas present the update of node embeddings (e.g., $\mathbf{r}^i = \mathbf{F}(i,:)$), edge embedding (e.g., $\mathbf{r}^{(i,j)} = \mathbf{E}(i,j,:)$), and global embedding $\mathbf{y}$ in an MPNN layer, where we omit the subscript hidden if it does not cause ambiguity:

$$\mathbf{r}^i \leftarrow \texttt{FiLM}\left(\texttt{FiLM}\left(\mathbf{r}^i, \texttt{MLP}\left(\sum_{j=1}^n \mathbf{r}^{(j,i)}/n\right)\right), \mathbf{y}\right), \quad \mathbf{r}^{(i,j)} \leftarrow \texttt{FiLM}\big(\texttt{FiLM}(\mathbf{r}^{(i,j)}, \mathbf{r}^i \odot \mathbf{r}^j), \mathbf{y}\big), \tag{7}$$

$$\mathbf{y} \leftarrow \mathbf{y} + \texttt{PNA}\big(\{\mathbf{r}^i\}_{i=1}^n\big) + \texttt{PNA}\big(\{\mathbf{r}^{(i,j)}\}_{i,j=1}^n\big). \tag{8}$$

The edge embeddings are aggregated by mean pooling (i.e., $\sum_{j=1}^n \mathbf{r}^{(j,i)}/n$); the node pair embeddings are passed to edges by Hadamard product (i.e., $\mathbf{r}^i \odot \mathbf{r}^j$); edge/node embeddings are merged to the global embedding $\mathbf{y}$ via the PNA module [12]; Some FiLM modules [57] are used for the interaction between node/edge/global embeddings. More details about the PNA and FiLM are in Appendix E. In this paper, we name Eqs. (7) and (8) on all nodes/edges together as an MPNN layer, $\mathbf{F}, \mathbf{E}, \mathbf{y} \leftarrow \texttt{MPNN}(\mathbf{F}, \mathbf{E}, \mathbf{y})$. Stacking multiple MPNN layers leads to larger model capacity.

Finally, two readout MLPs are used to project the node/edge embeddings into input dimensions, $\texttt{MLP}(\mathbf{F}) \in \mathbb{R}^{n \times b}$ and $\texttt{MLP}(\mathbf{E}) \in \mathbb{R}^{n \times n \times (a+1)}$, which are output after wrapped with `softmax`.

Both the proposed MPNN and the GT from DiGress [73] use the PNA and FiLM to merge embeddings, but MPNN does not have multi-head self-attention layers so that the computation overhead is lower.

3.6 Permutation Equivariance and Invariance

Reordering the nodes keeps the property of a given graph, which is known as permutation invariance. In addition, for a given function if its input is permuted and its output is permuted accordingly, such a behavior is known as permutation equivariance. In this subsection, we analyze permutation-equivariance/invariance of the (1) diffusion framework (Lemmas 3.5, 3.6, and 3.7), (2) sampling density (Theorem 3.8), and (3) training loss (Theorem 3.9).

Lemma 3.5 (Permutation-equivariant layer)**.** *The proposed MPNN layer (Eqs.* (7) *and* (8)*) is permutation-equivariant.*

The auxiliary features from the `Aux` are also permutation-equivariant (see Appendix E). Thus, the whole MPNN-based backbone $p^{\theta}_{0|t}$ is permutation-equivariant. Note that the GT-based backbone from DiGress [73] is also permutation-equivariant whose proof is omitted as it is not our contribution. Next, we show the permutation invariance of the rate matrices.

Lemma 3.6 (Permutation-invariant rate matrices)**.** *The forward rate matrix of* DisCo *is permutation-invariant if it is factorized as Eq.* (3)*. The parametric reverse rate matrix of* DisCo *($\tilde{\mathbf{R}}_{\theta,t}$) is permutation-invariant whenever the graph-to-graph backbone $p^{\theta}_{0|t}$ is permutation-equivariant.*

Lemma 3.7 (Permutation-invariant transition probability)**.** *For CTMC satisfying the Kolmogorov forward equation (Eq.* (1)*), if the rate matrix is permutation-invariant (i.e., $\mathbf{R}_t(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{R}_t(\mathcal{P}(\mathbf{x}_i), \mathcal{P}(\mathbf{x}_j))$, the transition probability is permutation-invariant (i.e., $q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = q_{t|s}(\mathcal{P}(\mathbf{x}_t)|\mathcal{P}(\mathbf{x}_s))$, where $\mathcal{P}$ is a permutation.*

Based on Lemmas 3.6 and 3.7, DisCo's parametric reverse transition probability is permutation-invariant. The next theorem shows the permutation-invariance of the sampling probability.

Theorem 3.8 (Permutation-invariant sampling probability)**.** *If both the reference distribution $\pi_{\texttt{ref}}$ and the reverse transition probability are permutation-invariant, the parametric sampling distribution $p^{\theta}_0(\mathcal{G}_0)$ is permutation-invariant.*

In addition, the next theorem shows the permutation invariance of the training loss.

Theorem 3.9 (Permutation-invariant training loss)**.** *The proposed training loss Eq.* (6) *is invariant to any permutation of the input graph $\mathcal{G}_0$ if $p^{\theta}_{0|t}$ is permutation-equivariant.*

4 Experiments

This section includes: an effectiveness evaluation on plain graphs (Section 4.1) and molecule graphs (Section 4.2), an efficiency study (Section 4.3), and an ablation study (Section 4.4). Detailed settings (Sections F.1-F.3), additional effectiveness evaluation (Sections F.4, additional ablation study (Section F.5), convergence study (Section F.6), and visualization (Section F.7) are in Appendix. Our code is released [3].

4.1 Plain Graph Generation

Datasets and metrics. Datasets SBM, Planar [51], and Community [82] are used. The relative squared Maximum Mean Discrepancy (MMD) for degree distributions (Deg.), clustering coefficient distributions (Clus.), and orbit counts (Orb.) distributions (the number of occurrences of substructures with 4 nodes), Uniqueness(%), Novelty(%), and Validity(%) are chosen as metrics. Details about the datasets, metrics, baselines (Section F.2.2), and results on Community (Table 8) are in Appendix.

Results. Table 1 shows the effectiveness evaluation on SBD and Planar from which we observe:

[3] `https://github.com/pricexu/DisCo`

 https://doi.org/10.52202/079017-2531

Table 1: Performance (mean±std) on SBM and Planar datasets.

Dataset	Model	Deg.↓	Clus.↓	Orb.↓	Unique ↑	Novel ↑	Valid ↑
SBM	GraphRNN [82]	6.9	1.7	3.1	**100.0**	**100.0**	5.0
	GRAN [42]	14.1	1.7	2.1	**100.0**	**100.0**	25.0
	GG-GAN [37]	4.4	2.1	2.3	**100.0**	**100.0**	0.0
	MolGAN [9]	29.4	3.5	2.8	95.0	**100.0**	10.0
	SPECTRE [51]	1.9	1.6	**1.6**	**100.0**	**100.0**	52.5
	ConGress [73]	34.1	3.1	4.5	0.0	0.0	0.0
	DiGress [73]	1.6	1.5	1.7	**100.0**	**100.0**	**67.5**
	DisCo-MPNN	1.8±0.2	**0.8±0.1**	2.7±0.4	**100.0±0.0**	**100.0±0.0**	41.9±2.2
	DisCo-GT	**0.8±0.2**	**0.8±0.4**	2.0±0.5	**100.0±0.0**	**100.0±0.0**	66.2±1.4
Planar	GraphRNN [82]	24.5	9.0	2508.0	**100.0**	**100.0**	0.0
	GRAN [42]	3.5	1.4	1.8	85.0	2.5	**97.5**
	GG-GAN [37]	315.0	8.3	2062.6	**100.0**	**100.0**	0.0
	MolGAN [9]	4.5	10.2	2346.0	25.0	**100.0**	0.0
	SPECTRE [51]	2.5	2.5	2.4	**100.0**	**100.0**	25.0
	ConGress [73]	23.8	8.8	2590.0	0.0	0.0	0.0
	DiGress [73]	1.4	**1.2**	**1.7**	**100.0**	**100.0**	85.0
	DisCo-MPNN	1.4±0.3	1.4±0.4	6.4±1.6	**100.0±0.0**	**100.0±0.0**	33.8±2.7
	DisCo-GT	**1.2**±0.5	1.3±0.5	**1.7±0.7**	**100.0±0.0**	**100.0±0.0**	83.6±2.1

Table 2: Performance (mean±std%) on QM9 dataset. V., U., and N. mean Valid, Unique, and Novel.

Model	Valid ↑	V.U. ↑	V.U.N. ↑
CharacterVAE [20]	10.3	7.0	6.3
GrammarVAE[38]	60.2	5.6	4.5
GraphVAE [66]	55.7	42.0	26.1
GT-VAE [55]	74.6	16.8	15.8
Set2GraphVAE [72]	59.9	56.2	-
GG-GAN [37]	51.2	24.4	24.4
MolGAN [9]	98.1	10.2	9.6
SPECTRE [51]	87.3	31.2	29.1
GraphNVP [50]	83.1	82.4	-
GDSS [32]	95.7	94.3	-
EDGE [10]	99.1	**99.1**	-
ConGress [73]	98.9	95.7	38.3
DiGress [73]	99.0	95.2	31.8
GRAPHARM [36]	90.3	86.3	-
DisCo-MPNN	98.9±0.7	98.7±0.5	**68.7±0.2**
DisCo-GT	**99.3±0.6**	98.9±0.6	56.2±0.4

- DisCo-GT can obtain competitive performance against the SOTA, DiGress, which is reasonable because both models share the graph Transformer backbone. Note that DiGress's performance in terms of Validity is not the statistics reported in the paper but from their latest model checkpoint [4]. In fact, we found it very hard for DiGress and DisCo-GT to learn to generate valid SBM/Planar graphs. These two datasets have only 200 graphs, but sometimes only after $> 10,000$ epochs training, the Validity percentage can be $> 50\%$. Additionally, DisCo-GT provides extra flexibility during sampling by adjusting the τ. This is important: our models can still trade-off between the sampling efficiency and quality even after the model is trained and frozen.
- In general, DisCo-MPNN has competitive performance against DisCo-GT in terms of Deg., Clus., and Orb. However, its performance is worse compared to DisCo-GT in terms of Validity, which might be related to the different model expressiveness. Studying the graph-to-graph model expressiveness would be an interesting future direction, e.g., generating valid Planar graphs.

[4] `https://github.com/cvignac/DiGress/blob/main/README.md`

Table 3: Performance on MOSES. VAE, JT-VAE, and GraphINVENT have hard-coded rules to ensure high validity.

Model	Valid ↑	Unique ↑	Novel ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
VAE [21]	97.7	98.8	69.5	99.7	0.57	0.58	5.9
JT-VAE [29]	100.0	100.0	99.9	97.8	1.00	0.53	10.0
GraphINVENT [53]	96.4	99.8	N/A	95.0	1.22	0.54	12.7
ConGress [73]	83.4	99.9	96.4	94.8	1.48	0.50	16.4
DiGress [73]	85.7	100.0	95.0	97.1	1.19	0.52	14.8
DisCo-MPNN	83.9	100.0	98.8	87.3	1.63	0.48	13.5
DisCo-GT	88.3	100.0	97.7	95.6	1.44	0.50	15.1

Table 4: Performance on GuacaMol. LSTM, NAGVAE, and MCTS are tailored for molecule datasets; ConGress, DiGress, and DisCo are general graph generation models.

Model	Valid ↑	Unique ↑	Novel ↑	KL div ↑	FCD ↑
LSTM [64]	95.9	100.0	91.2	99.1	91.3
NAGVAE [40]	92.9	95.5	100.0	38.4	0.9
MCTS [28]	100.0	100.0	95.4	82.2	1.5
ConGress [73]	0.1	100.0	100.0	36.1	0.0
DiGress [73]	85.2	100.0	99.9	92.9	68.0
DisCo-MPNN	68.7	100.0	96.4	77.0	36.4
DisCo-GT	86.6	100.0	99.9	92.6	59.7

4.2 Molecule Graph Generation

Dataset and metrics. The datasets QM9 [62], MOSES [58], and GuacaMol [6] are chosen. For MOSES, metrics including Uniquess, Novelty, Validity, Filter, FCD, SNN, and Scaf are reported in Table 3. For QM9, metrics include Uniqueness, Novelty, and Validity. For GuacaMol, metrics include Valid, Unique, Novel, KL div, and FCD. Details about the datasets, metrics, and baseline methods are in Appendix F.2.3.

Results. Table 2 shows the performance on QM9 dataset. Our observation is consistent with the performance comparison on plain datasets: (1) DisCo-GT obtains slightly better or at least competitive performance against DiGress due to the shared graph-to-graph backbone, but our framework offers extra flexibility in the sampling process; (2) DisCo-MPNN obtains decent performance in terms of Validity, Uniqueness, and Novelty comparing with DisCo-GT.

Tables 3 and 4 show the performance on MOSES and GuacaMol which further verifies that (1) performance of DisCo-GT is on par with the SOTA general graph generative models, DiGress and (2) DisCo-MPNN has decent performance, but worse than DisCo-GT and DiGress.

4.3 Efficiency Study

A major computation bottleneck is the graph-to-graph backbone $p^{\theta}_{0|t}$, which is GT or MPNN. We compare the number of parameters, the forward and backpropagation time of GT and MPNN in Table 5. For a fair comparison, we set all the hidden dimensions of GT and MPNN as 256 and the number of layers as 5. We use the Community [82] dataset and set the batch size as 64. Table 5 shows that GT has a larger capacity and more parameters at the expense of more expensive training.

Table 5: Efficiency comparison in terms of number of parameters, forward and backpropagation time (second/iteration).

	GT	MPNN
# Parameters	14×10^6	7×10^6
Forward	0.065	0.022
Backprop.	0.034	0.018

https://doi.org/10.52202/079017-2531

4.4 Ablation Study

An ablation study on DISCO-GT for reference distributions (marginal vs. uniform), and sampling steps (1 to 500) is presented in Table 6. The number of sampling steps is $\texttt{round}(\frac{1}{\tau})$ if $T = 1$. QM9 dataset is chosen. A similar ablation study on DISCO-MPNN is in Table 9 in Appendix. We observe that first, generally, the fewer sampling steps, the lower the generation quality. In some cases (e.g., the marginal distribution) with the sampling steps decreasing significantly (e.g., from 500 to 30), the performance degradation is still very slight, implying our method's high robustness in sampling steps. Second, the marginal reference distribution is better than the uniform distribution, consistent with the observation from DiGress [73].

Table 6: Ablation study (mean±std%) with GT backbone. V., U., and N. mean Valid, Unique, and Novel.

Ref. Dist.	Steps	Valid ↑	V.U. ↑	V.U.N. ↑
Marginal	500	99.3±0.6	98.9±0.6	56.2±0.4
	100	98.7±0.5	98.5±0.4	58.8±0.4
	30	97.9±1.2	97.6±1.1	59.2±0.8
	10	95.3±1.9	94.8±1.6	62.1±0.9
	5	93.0±1.7	92.4±1.3	64.9±1.1
	1	76.1±2.3	73.9±1.6	62.9±1.8
Uniform	500	94.1±0.9	92.9±0.5	56.6±0.4
	100	91.5±1.0	90.3±0.9	54.4±1.2
	30	88.7±1.6	86.9±1.0	58.6±2.1
	10	84.5±2.3	80.4±1.7	59.8±1.8
	5	77.0±2.5	69.9±1.5	56.1±3.5
	1	44.9±3.1	35.1±3.4	29.6±2.5

5 Related Work

Diffusion models [80] can be interpreted from both the score-matching [69, 70] or the variational autoencoder perspective [23, 35, 34]. Pioneering efforts on diffusion generative modeling study the process in continuous-state [67, 23, 68] whose typical reference distribution is Gaussian. Beyond that, some efforts propose discrete-state models [24] to . E.g., D3PM [2] designs the discrete diffusion process by multiplication of transition matrices; τ-LDR [8] generalizes D3PM by formulating a continuous-time Markov chain; [71] proposes a singleton conditional distribution-based objective for the continuous-time Markov chain-based model whose rationale is Brook's Lemma [5, 49].

Diffusion models are widely used in graph generation tasks [44, 13, 85, 86, 84, 15, 83, 14, 61, 74, 47, 79, 78, 59, 3] such as molecule design [65, 25, 27, 43]. Pioneering works such as EDP-GNN [56] and GDSS [32] diffuse graph data in a continuous state space [26]. DiscDDPM [22] is an early effort to modify the DDPM architecture into a discrete state. In addition, DiGress [73] is also a one-shot discrete-state diffusion model, followed by a very recent work MCD [46], both in the discrete-time setting. Beyond the above-mentioned efforts, DruM [31] proposes to mix the diffusion process. EDGE [10] proposes an interesting process: diffusing graphs into empty graphs. Besides, GRAPHARM [36] proposes an autoregressive graph diffusion model, and [45] applies the diffusion models for molecule property prediction tasks. In addition to the above-mentioned general graph diffusion models, there are many other task-tailored graph diffusion generative models [48, 30, 41, 60, 77, 76, 4, 75], which incorporate more in-depth domain expertise into the model design. Interested readers are referred to this survey [44].

6 Conclusion

This paper introduces the first discrete-state continuous-time graph diffusion generative model, DISCO. Our model effectively marries continuous-time Markov Chain formulation with the discrete nature of graph data, addressing the fundamental sampling limitation of prior models. DISCO's training objective is concise with a solid theoretical foundation. We also propose a simplified message-passing architecture to serve as the graph-to-graph backbone, which theoretically has desirable properties against permutation of node ordering and empirically demonstrates decent performance against existing graph generative models in tests on various datasets.

Acknowledgments and Disclosure of Funding

ZX, RQ, ZZ, and HT are partially supported by NSF (2324770). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] David F Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *The Journal of chemical physics*, 127(21), 2007.

[2] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17981–17993, 2021.

[3] Yikun Ban, Yunzhe Qi, and Jingrui He. Neural contextual bandits for personalized recommendation. In Tat-Seng Chua, Chong-Wah Ngo, Roy Ka-Wei Lee, Ravi Kumar, and Hady W. Lauw, editors, *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, pages 1246–1249. ACM, 2024.

[4] Fan Bao, Min Zhao, Zhongkai Hao, Peiyao Li, Chongxuan Li, and Jun Zhu. Equivariant energy-guided sde for inverse molecular design. In *The eleventh international conference on learning representations*, 2023.

[5] D Brook. On the distinction between the conditional probability and the joint probability approaches in the specification of nearest-neighbour systems. *Biometrika*, 51(3/4):481–483, 1964.

[6] Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *J. Chem. Inf. Model.*, 59(3):1096–1108, 2019.

[7] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between MPNN and graph transformer. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3408–3430. PMLR, 2023.

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

[9] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018.

[10] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *Proceedings of the 40th International Conference on Machine Learning*, pages 4585–4610, 2023.

[11] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

 https://doi.org/10.52202/079017-2531

[13] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *SIGKDD Explor.*, 24(2):61–77, 2022.

[14] Dongqi Fu and Jingrui He. Natural and artificial dynamics in graphs: Concept, progress, and future. *Frontiers Big Data*, 5, 2022.

[15] Dongqi Fu, Zhe Xu, Hanghang Tong, and Jingrui He. Natural and artificial dynamics in gnns: A tutorial. In Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas, editors, *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 1252–1255. ACM, 2023.

[16] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

[17] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[18] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of chemical physics*, 115(4):1716–1733, 2001.

[19] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[20] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *CoRR*, abs/1610.02415, 2016.

[21] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[22] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *CoRR*, abs/2210.01549, 2022.

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[24] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12454–12465, 2021.

[25] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8867–8887. PMLR, 2022.

[26] Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation. In Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 201–210. IEEE, 2022.

[27] Lei Huang, Hengtong Zhang, Tingyang Xu, and Ka-Chun Wong. MDM: molecular diffusion model for 3d molecule generation. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 5105–5112. AAAI Press, 2023.

[28] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

[29] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer G. Dy and Andreas Krause, editors, *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*, pages 2328–2337. PMLR, 2018.

[30] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi S. Jaakkola. Torsional diffusion for molecular conformer generation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[31] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with destination-predicting diffusion mixture. *OpenReview*, 2023.

[32] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10362–10383. PMLR, 2022.

[33] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[34] Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[35] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *CoRR*, abs/2107.00630, 2021.

[36] Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B. Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17391–17408. PMLR, 2023.

[37] Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. Gg-gan: A geometric graph generative adversarial network, 2021. In *URL https://openreview. net/forum*, 2020.

[38] Matt J. Kusner and José Miguel Hernández-Lobato. GANS for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051, 2016.

[39] Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. Compressed graph representation for scalable molecular graph generation. *J. Cheminformatics*, 12(1):58, 2020.

 https://doi.org/10.52202/079017-2531

[40] Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. Compressed graph representation for scalable molecular graph generation. *Journal of Cheminformatics*, 12:1–8, 2020.

[41] Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 18872–18892. PMLR, 2023.

[42] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4257–4267, 2019.

[43] Haitao Lin, Yufei Huang, Meng Liu, Xuanjing Li, Shuiwang Ji, and Stan Z. Li. Diffbp: Generative diffusion of 3d molecules for target protein binding. *CoRR*, abs/2211.11214, 2022.

[44] Chengyi Liu, Wenqi Fan, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative diffusion models on graphs: Methods and applications. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 6702–6711. ijcai.org, 2023.

[45] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Data-centric learning from unlabeled graphs with diffusion model. *Advances in neural information processing systems*, 36, 2023.

[46] Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Inverse molecular design with multi-conditional diffusion guidance. *arXiv preprint arXiv:2401.13858*, 2024.

[47] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Class-imbalanced graph learning without class rebalancing. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[48] Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19784–19795, 2021.

[49] Siwei Lyu. Interpretation and generalization of score matching. In Jeff A. Bilmes and Andrew Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 359–366. AUAI Press, 2009.

[50] Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *CoRR*, abs/1905.11600, 2019.

[51] Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15159–15179. PMLR, 2022.

[52] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Mach. Learn. Sci. Technol.*, 2(2):25023, 2021.

[53] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023, 2021.

[54] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *CoRR*, abs/2202.08455, 2022.

[55] Joshua Mitton, Hans M. Senn, Klaas Wynne, and Roderick Murray-Smith. A graph VAE and graph transformer approach to generating molecular graphs. *CoRR*, abs/2104.04345, 2021.

[56] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 2020.

[57] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951. AAAI Press, 2018.

[58] Daniil Polykovskiy, Alexander Zhebrak, Benjamín Sánchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Sergey I. Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *CoRR*, abs/1811.12823, 2018.

[59] Yunzhe Qi, Yikun Ban, and Jingrui He. Graph neural bandits. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1920–1931. ACM, 2023.

[60] Zhuoran Qiao, Weili Nie, Arash Vahdat, Thomas F. Miller III, and Anima Anandkumar. Dynamic-backbone protein-ligand structure prediction with multiscale generative diffusion models. *CoRR*, abs/2209.15171, 2022.

[61] Ruizhong Qiu, Dingsu Wang, Lei Ying, H. Vincent Poor, Yifang Zhang, and Hanghang Tong. Reconstructing graph diffusion history from a single snapshot. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 1978–1988. ACM, 2023.

[62] 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.

[63] Sidney I Resnick. *Adventures in stochastic processes*. Springer Science & Business Media, 1992.

[64] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

[65] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In Marina Meila and Tong Zhang, editors, *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*, pages 9558–9568. PMLR, 2021.

[66] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In Vera Kurková, Yannis Manolopoulos, Barbara Hammer, Lazaros S. Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I*, volume 11139 of *Lecture Notes in Computer Science*, pages 412–422. Springer, 2018.

 https://doi.org/10.52202/079017-2531

[67] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015.

[68] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[69] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11895–11907, 2019.

[70] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[71] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[72] Clément Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[73] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[74] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. Networked time series imputation via position-aware graph enhanced variational autoencoders. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 2256–2268. ACM, 2023.

[75] Tomer Weiss, Eduardo Mayo Yanes, Sabyasachi Chakraborty, Luca Cosmo, Alex M Bronstein, and Renana Gershoni-Poranne. Guided diffusion for inverse molecular design. *Nature Computational Science*, 3(10):873–882, 2023.

[76] Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

[77] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[78] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye, editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 2850–2861. ACM, 2023.

[79] Zhe Xu, Boxin Du, and Hanghang Tong. Graph sanitation with application to node classification. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini, editors, *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1136–1147. ACM, 2022.

[80] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Ming-Hsuan Yang, and Bin Cui. Diffusion models: A comprehensive survey of methods and applications. *CoRR*, abs/2209.00796, 2022.

[81] Jiaxuan You, Jonathan Michael Gomes Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 10737–10745. AAAI Press, 2021.

[82] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In Jennifer G. Dy and Andreas Krause, editors, *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*, pages 5694–5703. PMLR, 2018.

[83] Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. Graph mixup on approximate gromov-wasserstein geodesics. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[84] Lecheng Zheng, Dawei Zhou, Hanghang Tong, Jiejun Xu, Yada Zhu, and Jingrui He. Fairgen: Towards fair graph generation. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*, pages 2285–2297. IEEE, 2024.

[85] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A data-driven graph generative model for temporal interaction networks. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 401–411. ACM, 2020.

[86] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. Misc-gan: A multi-scale generative model for graphs. *Frontiers Big Data*, 2:3, 2019.

 https://doi.org/10.52202/079017-2531

Appendix

The organization of this appendix is as follows

- Section A: the derivation of the factorized rate matrices.
- Section B: the forward transition probability matrix given a rate matrix.
- Section C: all the proofs.
  - Section C.1: proof of Proposition 3.2
  - Section C.2: proof of Theorem 3.3
  - Section C.3: proof of Lemma 3.5
  - Section C.4: proof of Lemma 3.6
  - Section C.5: proof of Lemma 3.7
  - Section C.6: proof of Theorem 3.8
  - Section C.7: proof of Theorem 3.9
- Section D: a step-by-step sampling algorithm.
- Section E: the auxiliary features and neural modules used by DISCO.
- Section F: detailed experimental settings and additional experimental results.
  - Section F.1: hardware and software
  - Section F.2: dataset setup
  - Section F.3: hyperparameter settings
  - Section F.4: additional effectiveness evaluation on Community Dataset
  - Section F.5: additional ablation study with the MPNN backbone
  - Section F.6: convergence study
  - Section F.7: visualization
- Section G: this paper's limitations and future work.
- Section H: the broad impact of this paper.

A Details of the Factorization of Rate Matrices

In this section, we detail the derivation of Remark 3.1, which is extended from the following Proposition 3 of [8].

Proposition A.1 (Factorization of the rate matrix, Proposition 3 from [8])**.** *If the forward process factorizes as* $q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = \prod_{d=1}^{D} q_{t|s}(x_t^d|x_s^d), t > s$*, then the forward and reverse rates are of the form*

$$\mathbf{R}_t(\bar{\mathbf{x}}, \mathbf{x}) = \sum_{d=1}^{D} \mathbf{R}_t^d(\bar{x}^d, x^d)\delta_{\bar{\mathbf{x}}\backslash\bar{x}^d, \mathbf{x}\backslash x^d} \tag{9}$$

$$\tilde{\mathbf{R}}_t(\mathbf{x}, \bar{\mathbf{x}}) = \sum_{d=1}^{D} \mathbf{R}_t^d(\bar{x}^d, x^d)\delta_{\bar{\mathbf{x}}\backslash\bar{x}^d, \mathbf{x}\backslash x^d} \sum_{x_0^d} q_{0|t}(x_0^d|\mathbf{x}) \frac{q_{t|0}(\bar{x}^d|x_0^d)}{q_{t|0}(x^d|x_0^d)} \tag{10}$$

where $\delta_{\bar{\mathbf{x}}\backslash\bar{x}^d, \mathbf{x}\backslash x^d} = 1$ *when all dimensions except for* d *are equal.*

As all the nodes and edges are categorical, applying the above proposition of all the nodes and edges leads to our Remark 3.1.

B Details of Forward Transition Probability

In this section, we present the derivation of the forward transition probability for nodes; the forward process for edges can be derived similarly. Note that this derivation has been mentioned in [8] for generic discrete cases; we graft it to the graph settings and include it here for completeness. The core derivation of the forward transition probability is to prove the following proposition.

Proposition B.1 (Analytical forward process for commutable rate matrices, Proposition 10 from [8])**.** *if* $\mathbf{R}_t$ *and* $\mathbf{R}_{t'}$ *commute* $\forall t, t'$, $q_{t|0}(x_t = j|x_0 = i) = (e^{\int_0^t \mathbf{R}_s ds})_{ij}$

Proof. If $q_{t|0} = \exp\left(\int_0^t \mathbf{R}_s ds\right)$ is the forward transition probability matrix, it should satisfy the Kolmogorov forward equation $\frac{d}{dt} q_{t|0} = q_{t|0}\mathbf{R}_s$. The transition probability matrix

$$q_{t|0} = \sum_{k=0}^{\infty} \frac{1}{k!}\left(\int_0^t \mathbf{R}_s ds\right)^k, \tag{11}$$

and, based on the fact that $\mathbf{R}_t$ and $\mathbf{R}'_t$ commute $\forall t, t'$, its derivative is

$$\frac{d}{dt} q_{t|0} = \sum_{k=1}^{\infty} \frac{1}{(k-1)!}\left(\int_0^t \mathbf{R}_s ds\right)^{(k-1)} = q_{t|0}\mathbf{R}_t. \tag{12}$$

Thus, $q_{t|0} = \exp\left(\int_0^t \mathbf{R}_s ds\right)$ is the solution of Kolmogorov forward equation. □

For the node i, if its forward rate matrix is set as $\mathbf{R}_t^i = \beta(t)\mathbf{R}_f$, we have $\mathbf{R}_t^i$ and $\mathbf{R}_{t'}^i$ commute, $\forall t, t'$. Thus, the transition probability for node i is $q_{t|0}(f_t^i = v|f_0^i = u) = (e^{\int_0^t \beta(s)\mathbf{R}_f ds})_{uv}$. Based on similar derivation, we have the transition probability for the edge (i, j) as $q_{t|0}(e_t^{(i,j)} = v|e_0^{(i,j)} = u) = (e^{\int_0^t \beta(s)\mathbf{R}_e ds})_{uv}$.

C Proofs

C.1 Proof of Proposition 3.2

Proposition 3.2 claims the forward process converges to uniform distributions if $\mathbf{R}_f = \mathbf{1}\mathbf{1}^\top - b\mathbf{I}$ and $\mathbf{R}_e = \mathbf{1}\mathbf{1}^\top - (a+1)\mathbf{I}$ and it converges to marginal distributions $\mathbf{m}_f$ and $\mathbf{m}_e$ if $\mathbf{R}_f = \mathbf{1}\mathbf{m}_f^\top - \mathbf{I}$ and $\mathbf{R}_e = \mathbf{1}\mathbf{m}_e^\top - \mathbf{I}$.

Proof. If we formulate the rate matrices for nodes and edges as $\mathbf{R}_t^{(i,j)} = \beta(t)\mathbf{R}_e,\ \forall i, j$ and $\mathbf{R}_t^i = \beta(t)\mathbf{R}_f,\ \forall i$, every rate matrix is commutable for any time steps t and t'. In the following content, we show the proof for the node rate matrix $\mathbf{R}_t^i = \beta(t)\mathbf{R}_f$; the converged distribution of edge can be proved similarly. Based on Proposition B.1, the transition probability matrix between time steps t and $t + \Delta t$ is

$$q_{t+\Delta t|t} = \mathbf{I} + \int_t^{t+\Delta t} \beta(s)\mathbf{R}_f ds + O((\Delta t)^2) \tag{13}$$

$$\overset{(*)}{=} \mathbf{I} + \Delta t\beta(\xi)\mathbf{R}_f + O((\Delta t)^2), \tag{14}$$

where (*) is based on the Mean Value Theorem. If the high-order term $O((\Delta t)^2)$ is omitted and we short $\beta_{\Delta t} = \Delta t\beta(\xi)$, for $\mathbf{R}_f = \mathbf{1}\mathbf{1}^\top - b\mathbf{I}$, we have

$$q_{t+\Delta t|t} \approx \beta_{\Delta t}\mathbf{1}\mathbf{1}^\top + (1 - \beta_{\Delta t} b)\mathbf{I}, \tag{15}$$

which is the transition matrix of the uniform diffusion in the discrete-time diffusion models [67, 2]. Thus, with $T \to \infty$ and $q_{t+\Delta t|t}$ to the power of infinite, the converged distribution is a uniform distribution. Similarly, for $\mathbf{R}_f = \mathbf{1}\mathbf{m}_f^\top - \mathbf{I}$ the transition matrix is

$$q_{t+\Delta t|t} \approx \beta_{\Delta t}\mathbf{1}\mathbf{m}_f^\top + (1 - \beta_{\Delta t})\mathbf{I} \tag{16}$$

which is a generalized transition matrix of the 'absorbing state' diffusion [2]. The difference lies at for the 'absorbing state' diffusion [2], $\mathbf{m}_f$ is set as a one-hot vector for the absorbing state, and here we set it as the marginal distribution. Thus, with $T \to \infty$ and $q_{t+\Delta t|t}$ to the power of infinite, the converged distribution is a marginal distribution $\mathbf{m}_f$. □

https://doi.org/10.52202/079017-2531

C.2 Proof of Theorem 3.3

Theorem 3.3 says for $\mathcal{G} \neq \bar{\mathcal{G}}$,

$$\left|\tilde{\mathbf{R}}_t(\mathcal{G},\bar{\mathcal{G}}) - \tilde{\mathbf{R}}_{\theta,t}(\mathcal{G},\bar{\mathcal{G}})\right|^2 \leq C_t + C_t^{\texttt{node}}\mathbb{E}_{\mathcal{G}_0 q_{t|0}(\mathcal{G}|\mathcal{G}_0)} \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) + C_t^{\texttt{edge}}\mathbb{E}_{\mathcal{G}_0 q_{t|0}(\mathcal{G}|\mathcal{G}_0)} \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)}) \tag{17}$$

where the node and edge estimated probability vector (sum is 1) is notated as $\hat{f}_0^i = [p^\theta_{0|t}(f^i = 1|\mathcal{G}_t), \ldots, p^\theta_{0|t}(f^i = b|\mathcal{G}_t)]^\top \in [0,1]^b$ and $\hat{e}_0^{(i,j)} = [p^\theta_{0|t}(e^{(i,j)} = 1|\mathcal{G}_t), \ldots, p^\theta_{0|t}(e^{(i,j)} = a+1|\mathcal{G}_t)]^\top \in [0,1]^{a+1}$.

Proof.

$$\left|\tilde{\mathbf{R}}_t(\mathcal{G},\bar{\mathcal{G}}) - \tilde{\mathbf{R}}_{\theta,t}(\mathcal{G},\bar{\mathcal{G}})\right| \tag{18}$$

$$= \Big| \sum_i A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} (q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G})) + \sum_{i,j} B_t^{(i,j)} \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} (q_{0|t}(e_0^{(i,j)}|\mathcal{G}) - p^\theta_{0|t}(e_0^{(i,j)}|\mathcal{G}))\Big| \tag{19}$$

$$\leq \Big| \sum_i A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} (q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G}))\Big| + \Big| \sum_{i,j} B_t^{(i,j)} \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} (q_{0|t}(e_0^{(i,j)}|\mathcal{G}) - p^\theta_{0|t}(e_0^{(i,j)}|\mathcal{G}))\Big| \tag{20}$$

We check the first term of Eq. (20):

$$\Big| \sum_i A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} (q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G}))\Big| \tag{21}$$

$$\leq \sum_i A_t^i \sup_{f_0^i} \left\{ \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} \right\} \sum_{f_0^i} \Big| q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G}) \Big| \tag{22}$$

$$= \sum_i C_i \sum_{f_0^i} \Big| q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G}) \Big| \tag{23}$$

$$\overset{(*)}{\leq} \sum_i C_i \sqrt{2 \sum_{f_0^i} \Big(C_{f_0^i} - q_{0|t}(f_0^i|\mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \Big)} \tag{24}$$

$$\overset{(**)}{\leq} C_1 \sqrt{\sum_i \sum_{f_0^i} \Big(C_{f_0^i} - q_{0|t}(f_0^i|\mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \Big)} \tag{25}$$

$$= C_1 \sqrt{C_2 - \sum_i \sum_{f_0^i} q_{0|t}(f_0^i|\mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G})} \tag{26}$$

where $C_i = A_t^i \sup_{f_0^i} \left\{ \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} \right\}$, $C_{f_0^i} = q_{0|t}(f_0^i|\mathcal{G}) \log q_{0|t}(f_0^i|\mathcal{G})$, (*) is based on the Pinsker's inequality, (**) is based on Cauchy–Schwarz inequality: $\sum_{i=1}^n \sqrt{x_i} \leq \sqrt{n \sum_{i=1}^n x_i}$, $C_1 = \sqrt{2n} \sup_i \{C_i\}$, $C_2 = \sum_i \sum_{f_0^i} C_{f_0^i}$. Next, the term $-\sum_i \sum_{f_0^i} q_{0|t}(f_0^i|\mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G})$ is equiva-

lent to:

$$-\sum_i \sum_{f_0^i} q_{0|t}(f_0^i|\mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \tag{27}$$

$$= -\frac{1}{p_t(\mathcal{G})} \sum_i \sum_{f_0^i} p_{0,t}(f_0^i, \mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \tag{28}$$

$$= -\frac{1}{p_t(\mathcal{G})} \sum_i \sum_{f_0^i} \sum_{\mathcal{G}_0(f_0^i)} p_{0,t}(\mathcal{G}_0, \mathcal{G}) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \tag{29}$$

$$= -\frac{1}{p_t(\mathcal{G})} \sum_i \sum_{f_0^i} \sum_{\mathcal{G}_0(f_0^i)} \pi_{\texttt{data}}(\mathcal{G}_0) q_{t|0}(\mathcal{G}|\mathcal{G}_0) \log p^\theta_{0|t}(f_0^i|\mathcal{G}) \tag{30}$$

$$= \frac{1}{p_t(\mathcal{G})} \sum_i \sum_{f_0^i} \sum_{\mathcal{G}_0(f_0^i)} \pi_{\texttt{data}}(\mathcal{G}_0) q_{t|0}(\mathcal{G}|\mathcal{G}_0) \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{31}$$

$$= \frac{1}{p_t(\mathcal{G})} \sum_{\mathcal{G}_0} \pi_{\texttt{data}}(\mathcal{G}_0) q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{32}$$

$$= \frac{1}{p_t(\mathcal{G})} \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{33}$$

where $\sum_{\mathcal{G}_0(f_0^i)}$ marginalizing all the graphs at time 0 whose i-th node is f_0^i; $p_{0,t}(f_0^i, \mathcal{G})$ is the joint probability of a graph whose i-th node is f_0^i at time 0 and it is $\mathcal{G}$ at time t; $p_{0,t}(\mathcal{G}_0, \mathcal{G})$ is the joint probability of a graph which is $\mathcal{G}_0$ at time 0 and it is $\mathcal{G}$ at time t. Plugging Eq. (33) into Eq. (26):

$$\begin{aligned} &\Big| \sum_i A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} (q_{0|t}(f_0^i|\mathcal{G}) - p^\theta_{0|t}(f_0^i|\mathcal{G})) \Big| \\ \leq & C_1 \sqrt{C_2 + C_5 \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i)} \end{aligned} \tag{34}$$

where $C_5 = \frac{1}{p_t(\mathcal{G})}$. A similar analysis can be conducted about the second term of Eq. (20) and we directly present it here:

$$\begin{aligned} &\Big| \sum_{i,j} B_t^{(i,j)} \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} (q_{0|t}(e_0^{(i,j)}|\mathcal{G}) - p^\theta_{0|t}(e_0^{(i,j)}|\mathcal{G})) \Big| \\ \leq & C_3 \sqrt{C_4 + C_5 \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)})} \end{aligned} \tag{35}$$

where $C_3 = \sqrt{2} n \sup_{i,j}\{C_{i,j}\}$, $C_4 = \sum_{i,j} \sum_{e_0^{(i,j)}} C_{e_0^{(i,j)}}$, $C_{i,j} = B_t^{(i,j)} \sup_{e_0^{(i,j)}} \left\{ \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} \right\}$, $C_{e_0^{(i,j)}} = q_{0|t}(e_0^{(i,j)}|\mathcal{G}) \log q_{0|t}(e_0^{(i,j)}|\mathcal{G})$.

Plugging Eqs. (34) and (35) into Eq. (20), being aware that C_1, C_2, C_3, C_4, C_5 are all t-related:

$$\begin{aligned} \left| \tilde{\mathbf{R}}_t(\mathcal{G}, \bar{\mathcal{G}}) - \tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}}) \right| \leq & C_1 \sqrt{C_2 + C_5 \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i)} \\ & + C_3 \sqrt{C_4 + C_5 \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)})} \end{aligned} \tag{36}$$

$$\begin{aligned} \overset{(*)}{\leq} & \Big(C_t + C_t^{\texttt{node}} \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \\ & + C_t^{\texttt{edge}} \mathbb{E}_{\mathcal{G}_0} q_{t|0}(\mathcal{G}|\mathcal{G}_0) \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)}) \Big)^{1/2} \end{aligned} \tag{37}$$

https://doi.org/10.52202/079017-2531

where (*) is based on Cauchy–Schwarz inequality, $C_t = 2C_1^2C_2 + 2C_3^2C_4$, $C_t^{\texttt{node}} = 2C_1^2C_5$, $C_t^{\texttt{edge}} = 2C_3^2C_5$. □

C.3 Proof of Lemma 3.5

We clarify that the term "permutation" in this paper refers to the reordering of the node indices, i.e., the first dimension of $\mathbf{F}$ and the first two dimensions of $\mathbf{E}$.

Proof. The input of an MPNN layer is $\mathbf{F} = \{\mathbf{r}_i\}_{i=1}^n \in \mathbb{R}^{n\times d}, \mathbf{E} = \{\mathbf{r}_{i,j}\}_{i,j=1}^n \in \mathbb{R}^{n\times n\times d}, \mathbf{y} \in \mathbb{R}^d$, where d is the hidden dimension. The updating formulas of an MPNN layer can be presented as

$$\mathbf{r}^i \leftarrow \texttt{FiLM}\left(\texttt{FiLM}\left(\mathbf{r}^i, \texttt{MLP}\left(\frac{\sum_{j=1}^n \mathbf{r}^{(j,i)}}{n}\right)\right), \mathbf{y}\right), \tag{38}$$

$$\mathbf{r}^{(i,j)} \leftarrow \texttt{FiLM}\big(\texttt{FiLM}(\mathbf{r}^{(i,j)}, \mathbf{r}^i \odot \mathbf{r}^j), \mathbf{y}\big), \tag{39}$$

$$\mathbf{y} \leftarrow \mathbf{y} + \texttt{PNA}\big(\{\mathbf{r}^i\}_{i=1}^n\big) + \texttt{PNA}\big(\{\mathbf{r}^{(i,j)}\}_{i,j=1}^n\big), \tag{40}$$

The permutation $\mathcal{P}$ of the input of an MPNN layer can be presented as $\mathcal{P}\big(\mathbf{F} = \{\mathbf{r}_i\}_{i=1}^n, \mathbf{E} = \{\mathbf{r}_{i,j}\}_{i,j=1}^n, \mathbf{y}\big) = \big(\{\mathbf{r}_{\sigma(i)}\}_{i=1}^n, \{\mathbf{r}_{\sigma(i),\sigma(j)}\}_{i,j=1}^n, \mathbf{y}\big)$ where $\sigma : \{1,\dots,n\} \mapsto \{1,\dots,n\}$ is a bijection.

For PNA (Eq. (70)), it includes operations `max`, `min`, `mean`, and `std` which are all permutation-invariant and thus, the PNA module is permutation-invariant. Then,

$$\mathbf{y} + \texttt{PNA}\big(\{\mathbf{r}^i\}_{i=1}^n\big) + \texttt{PNA}\big(\{\mathbf{r}^{(i,j)}\}_{i,j=1}^n\big) = \mathbf{y} + \texttt{PNA}\big(\{\mathbf{r}^{\sigma(i)}\}_{i=1}^n\big) + \texttt{PNA}\big(\{\mathbf{r}^{(\sigma(i),\sigma(j))}\}_{i,j=1}^n\big) \tag{41}$$

Because $\sum_{j=1}^n \mathbf{r}^{(j,i)} = \sum_{j=1}^n \mathbf{r}^{(\sigma(j),\sigma(i))}$, $\mathbf{r}^i \odot \mathbf{r}^j = \mathbf{r}^{\sigma(i)} \odot \mathbf{r}^{\sigma(j)}$, and the FiLM module (Eq. (71)) is not related to the node ordering,

$$\mathbf{r}^{(\sigma(i),\sigma(j))} \leftarrow \texttt{FiLM}\big(\texttt{FiLM}(\mathbf{r}^{(\sigma(i),\sigma(j))}, \mathbf{r}^{\sigma(i)} \odot \mathbf{r}^{\sigma(j)}), \mathbf{y}\big) = \texttt{FiLM}\big(\texttt{FiLM}(\mathbf{r}^{(i,j)}, \mathbf{r}^i \odot \mathbf{r}^j), \mathbf{y}\big) \tag{42}$$

$$\mathbf{r}^{\sigma(i)} \leftarrow \texttt{FiLM}\left(\texttt{FiLM}\left(\mathbf{r}^{\sigma(i)}, \texttt{MLP}\left(\frac{\sum_{j=1}^n \mathbf{r}^{(\sigma(j),\sigma(i))}}{n}\right)\right), \mathbf{y}\right) \tag{43}$$

$$= \texttt{FiLM}\left(\texttt{FiLM}\left(\mathbf{r}^i, \texttt{MLP}\left(\frac{\sum_{j=1}^n \mathbf{r}^{(j,i)}}{n}\right)\right), \mathbf{y}\right) \tag{44}$$

Thus, we proved that

$$\texttt{MPNN}\Big(\mathcal{P}\big(\mathbf{F}, \mathbf{E}, \mathbf{y}\big)\Big) = \mathcal{P}\Big(\texttt{MPNN}(\mathbf{F}, \mathbf{E}, \mathbf{y})\Big) \tag{45}$$

□

C.4 Proof of Lemma 3.6

Proof. The forward rate matrix (Eq. (3)) is the sum of component-specific forward rate matrices ($\{\mathbf{R}_t^{(i,j)}\}_{i,j\in\mathbb{N}_{\le n}^+}$ and $\{\mathbf{R}_t^i\}_{i\in\mathbb{N}_{\le n}^+}$). It is permutation-invariant because the summation is permutation-invariant.

The parametric reverse rate matrix is

$$\tilde{\mathbf{R}}_{\theta,t}(\mathcal{G}, \bar{\mathcal{G}}) = \sum_i \tilde{\mathbf{R}}_{\theta,t}^i(f^i, \bar{f}^i) + \sum_{i,j} \tilde{\mathbf{R}}_{\theta,t}^{(i,j)}(e^{(i,j)}, \bar{e}^{(i,j)}) \tag{46}$$

where $\tilde{\mathbf{R}}_{\theta,t}^i(f^i, \bar{f}^i) = A_t^i \sum_{f_0^i} \frac{q_{t|0}(\bar{f}^i|f_0^i)}{q_{t|0}(f^i|f_0^i)} p_{0|t}^\theta(f_0^i|\mathcal{G}_t)$, $\tilde{\mathbf{R}}_{\theta,t}^{(i,j)}(e^{(i,j)}, \bar{e}^{(i,j)}) = B_t^{(i,j)} \sum_{e_0^{(i,j)}} \frac{q_{t|0}(\bar{e}^{(i,j)}|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})} p_{0|t}^\theta(e_0^{(i,j)}|\mathcal{G}_t)$. If we present the permutation $\mathcal{P}$ on every node

as a bijection $\sigma : \{1, \dots, n\} \mapsto \{1, \dots, n\}$, the term

$$\tilde{\mathbf{R}}^i_{\theta,t}(f^i, \bar{f}^i) = A^i_t \sum_{f^i_0} \frac{q_{t|0}(\bar{f}^i|f^i_0)}{q_{t|0}(f^i|f^i_0)} p^\theta_{0|t}(f^i_0|\mathcal{G}_t) \tag{47}$$

$$= \mathbf{R}^i_t(\bar{f}^i, f^i)\delta_{\bar{\mathcal{G}}\setminus\bar{f}^i, \mathcal{G}\setminus f^i} \sum_{f^i_0} \frac{q_{t|0}(\bar{f}^i|f^i_0)}{q_{t|0}(f^i|f^i_0)} p^\theta_{0|t}(f^i_0|\mathcal{G}_t) \tag{48}$$

$$\overset{(*)}{=} \mathbf{R}^{\sigma(i)}_t(\bar{f}^{\sigma(i)}, f^{\sigma(i)})\delta_{\mathcal{P}(\bar{\mathcal{G}})\setminus\bar{f}^{\sigma(i)}, \mathcal{P}(\mathcal{G})\setminus f^{\sigma(i)}} \sum_{f^{\sigma(i)}_0} \frac{q_{t|0}(\bar{f}^{\sigma(i)}|f^{\sigma(i)}_0)}{q_{t|0}(f^{\sigma(i)}|f^{\sigma(i)}_0)} p^\theta_{0|t}(f^i_0|\mathcal{G}_t) \tag{49}$$

$$\overset{(**)}{=} \mathbf{R}^{\sigma(i)}_t(\bar{f}^{\sigma(i)}, f^{\sigma(i)})\delta_{\mathcal{P}(\bar{\mathcal{G}})\setminus\bar{f}^{\sigma(i)}, \mathcal{P}(\mathcal{G})\setminus f^{\sigma(i)}} \sum_{f^{\sigma(i)}_0} \frac{q_{t|0}(\bar{f}^{\sigma(i)}|f^{\sigma(i)}_0)}{q_{t|0}(f^{\sigma(i)}|f^{\sigma(i)}_0)} p^\theta_{0|t}(f^{\sigma(i)}_0|\mathcal{P}(\mathcal{G}_t)) \tag{50}$$

$$= \tilde{\mathbf{R}}^{\sigma(i)}_{\theta,t}(f^{\sigma(i)}, \bar{f}^{\sigma(i)}) \tag{51}$$

where (*) is based on the permutation invariant of the forward process and its rate matrix; (**) is based on the permutation equivariance of the graph-to-graph backbone $p^\theta_{0|t}$. □

C.5 Proof of Lemma 3.7

Recall the Kolmogorov forward equation, for $s < t$,

$$\frac{d}{dt} q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = \sum_{\xi\in\mathcal{X}} q_{t|s}(\xi|\mathbf{x}_s)\mathbf{R}_t(\xi, \mathbf{x}_t). \tag{52}$$

Proof. We aim to show that $q_{t|s}(\mathcal{P}(\mathbf{x}_t)|\mathcal{P}(\mathbf{x}_s))$ is a solution of Eq. (52). Because the permutation $\mathcal{P}$ is a bijection, we have

$$\frac{d}{dt} q_{t|s}(\mathcal{P}(\mathbf{x}_t)|\mathcal{P}(\mathbf{x}_s)) \tag{53}$$

$$= \sum_{\xi\in\mathcal{X}} q_{t|s}(\mathcal{P}(\xi)|\mathcal{P}(\mathbf{x}_s))\mathbf{R}_t(\mathcal{P}(\xi), \mathcal{P}(\mathbf{x}_t)) \tag{54}$$

$$\overset{(*)}{=} \sum_{\xi\in\mathcal{X}} q_{t|s}(\mathcal{P}(\xi)|\mathcal{P}(\mathbf{x}_s))\mathbf{R}_t(\xi, \mathbf{x}_t) \tag{55}$$

where (*) is because $\mathbf{R}_t$ is permutation-invariant. As Eq. 55 and Eq. 52 share the same rate matrix, and the rate matrix completely determines the CTMC (and its Kolmogorov forward equation) [63], thus, their solutions are the same: $q_{t|s}(\mathbf{x}_t|\mathbf{x}_s) = q_{t|s}(\mathcal{P}(\mathbf{x}_t)|\mathcal{P}(\mathbf{x}_s))$, i.e., the transition probability is permutation-invariant. □

C.6 Proof of Theorem 3.8

Proof. We start from a simple case where the parametric rate matrix is fixed all the time,

$$p^\theta_0(\mathcal{G}_0) = \sum_{\mathcal{G}_T} q^\theta_{0|T}(\mathcal{G}_0|\mathcal{G}_T)\pi_{\texttt{ref}}(\mathcal{G}_T), \tag{56}$$

where the transition probability is by solving the Kolmogorov forward equation

$$\frac{d}{dt} q^\theta_{t|s}(\mathcal{G}_t|\mathcal{G}_s) = \sum_{\xi} q^\theta_{t|s}(\xi|\mathcal{G}_s)\tilde{\mathbf{R}}_\theta(\xi, \mathcal{G}_t). \tag{57}$$

https://doi.org/10.52202/079017-2531

Thus, the sampling probability of permuted graph $\mathcal{P}(\mathcal{G}_0)$

$$p_0^\theta(\mathcal{P}(\mathcal{G}_0)) = \sum_{\mathcal{G}_T} q_{0|T}^\theta(\mathcal{P}(\mathcal{G}_0)|\mathcal{P}(\mathcal{G}_T))\pi_{\texttt{ref}}(\mathcal{P}(\mathcal{G}_T)) \tag{58}$$

$$\overset{(*)}{=} \sum_{\mathcal{G}_T} q_{0|T}^\theta(\mathcal{G}_0|\mathcal{G}_T)\pi_{\texttt{ref}}(\mathcal{P}(\mathcal{G}_T)) \tag{59}$$

$$\overset{(**)}{=} \sum_{\mathcal{G}_T} q_{0|T}^\theta(\mathcal{G}_0|\mathcal{G}_T)\pi_{\texttt{ref}}(\mathcal{G}_T) \tag{60}$$

$$= p_0^\theta(\mathcal{G}_0) \tag{61}$$

where (*) is based on Lemma 3.6 and Lemma 3.7, the transition probability of DISCO is permutation-invariant and (**) is from the assumption that the reference distribution $\pi_{\texttt{ref}}(\mathcal{G}_T)$ is permutation-invariant. Thus, we proved that for the simple case, $\tilde{\mathbf{R}}_{\theta,t}$ fixed $\forall t$, the sampling probability is permutation-invariant.

For the practical sampling, as we mentioned in Section 3.4, the τ-leaping algorithm assumes that the time interval $[0, T]$ is divided into various length-τ intervals $[0, \tau), [\tau, 2\tau), \ldots, [T - \tau, T]$ (here both close sets or open sets work) and assume the reverse rate matrix is fixed as $\tilde{\mathbf{R}}_{\theta,t}$ within every length-τ interval, such as $(t - \tau, t]$. Thus, the sampling probability can be computed as

$$p_0^\theta(\mathcal{G}_0) = \sum_{\mathcal{G}_T, \mathcal{G}_{T-\tau}, \ldots, \mathcal{G}_\tau} q_{0|\tau}(\mathcal{G}_0|\mathcal{G}_\tau) \ldots q_{T-\tau|T}(\mathcal{G}_{T-\tau}|\mathcal{G}_T)\pi_{\texttt{ref}}(\mathcal{G}_T). \tag{62}$$

The conclusion from the simple case can be generalized to this τ-leaping-based case because all the transition probability $q_{t-\tau|t}(\mathcal{G}_{t-\tau}|\mathcal{G}_t)$ and the reference distribution are permutation-invariant. □

Note that Xu et al. [77] have a similar analysis in their Proposition 1 on a DDPM-based model.

C.7 Proof of Theorem 3.9

Recall our training objective is

$$\min_\theta T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}}\mathbb{E}_{\mathcal{G}_0}\mathbb{E}_{q_{t|0}(\mathcal{G}_t|\mathcal{G}_0)}\left[\sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) + \sum_{i,j} \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}), \hat{e}_0^{(i,j)})\right] \tag{63}$$

where $\hat{f}_0^i = [p_{0|t}^\theta(f^i = 1|\mathcal{G}_t), \ldots, p_{0|t}^\theta(f^i = b|\mathcal{G}_t)]^\top \in [0, 1]^b$ and $\hat{e}_0^{(i,j)} = [p_{0|t}^\theta(e^{(i,j)} = 1|\mathcal{G}_t), \ldots, p_{0|t}^\theta(e^{(i,j)} = a + 1|\mathcal{G}_t)]^\top \in [0, 1]^{a+1}$

Proof. We follow the notation and present the permutation $\mathcal{P}$ on every node as a bijection $\sigma : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$. We first analyze the cross-entropy loss on the nodes for a single training graph $\mathcal{G}_0$ and taking expectation $\mathbb{E}_{\mathcal{G}_0}$ keeps the permutation invariance:

$$\mathcal{L}_{\texttt{node}}(\mathcal{G}_0) = T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}}\mathbb{E}_{q_{t|0}(\mathcal{G}_t|\mathcal{G}_0)} \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{64}$$

$$= T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}} \sum_{\mathcal{G}_t} q_{t|0}(\mathcal{G}_t|\mathcal{G}_0) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{65}$$

$$\overset{(*)}{=} T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}} \sum_{\mathcal{G}_t} q_{t|0}(\mathcal{P}(\mathcal{G}_t)|\mathcal{P}(\mathcal{G}_0)) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^i), \hat{f}_0^i) \tag{66}$$

$$\overset{(**)}{=} T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}} \sum_{\mathcal{G}_t} q_{t|0}(\mathcal{P}(\mathcal{G}_t)|\mathcal{P}(\mathcal{G}_0)) \sum_i \mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(f_0^{\sigma(i)}), \hat{f}_0^{\sigma(i)}) \tag{67}$$

$$= \mathcal{L}_{\texttt{node}}(\mathcal{P}(\mathcal{G}_0)) \tag{68}$$

where (*) is from the permutation invariance of the forward process and (**) is from the permutation equivariance of the graph-to-graph backbone and the permutation invariance of the cross-entropy

loss. A similar result can be analyzed on the cross-entropy loss on the edges

$$\mathcal{L}_{\texttt{edge}}(\mathcal{G}_0) = T\mathbb{E}_{t\sim\mathcal{U}_{(0,T)}}\mathbb{E}_{q_{t|0}(\mathcal{G}_t|\mathcal{G}_0)}\sum_{i,j}\mathcal{L}_{\texttt{CE}}(\texttt{One-Hot}(e_0^{(i,j)}),\hat{e}_0^{(i,j)}) = \mathcal{L}_{\texttt{edge}}(\mathcal{P}(\mathcal{G}_0)) \tag{69}$$

and we omit the proof here for brevity. □

D Sampling Algorithm

A Step-by-step procedure about the τ-leaping graph generation is presented in Algorithm 2.

Algorithm 2 τ-Leaping Graph Generation

$t \leftarrow T$
$\mathcal{G}_t = (\{e^{(i,j)}\}_{i,j\in\mathbb{N}^+_{\leq n}}, \{f^i\}_{i\in\mathbb{N}^+_{\leq n}}) \leftarrow \pi_{\texttt{ref}}(\mathcal{G})$
while $t > 0$ **do**
  for $i = 1,\dots,n$ **do**
    for $s = 1,\dots,b$ **do**
      $\tilde{\mathbf{R}}^i_{\theta,t}(f^i, s) = \mathbf{R}^i_t(s, f^i)\sum_{f^i_0}\frac{q_{t|0}(s|f^i_0)}{q_{t|0}(f^i|f^i_0)}p_\theta(f^i|\mathcal{G}_t, t)$
      $J_{f^i,s} \leftarrow \text{Poisson}(\tau\mathbf{R}^i_t(s, f^i))$ ▷ # of transition for every node
    end for
  end for
  for $i, j = 1,\dots,n$ **do**
    for $s = 1,\dots,a$ **do**
      $\tilde{\mathbf{R}}^{(i,j)}_{\theta,t}(e^{(i,j)}, s) = \mathbf{R}^{(i,j)}_t(s, e^{(i,j)})\sum_{e_0^{(i,j)}}\frac{q_{t|0}(s|e_0^{(i,j)})}{q_{t|0}(e^{(i,j)}|e_0^{(i,j)})}p_\theta(e^{(i,j)}|\mathcal{G}_t, t)$
      $J_{e^{(i,j)},s} \leftarrow \text{Poisson}(\tau\mathbf{R}^{(i,j)}_t(s, e^{(i,j)}))$ ▷ # of transition for every edge
    end for
  end for
  for $i = 1,\dots,n$ **do**
    if $\sum_{s=1}^b J_{f^i,s} > 1$ or $\sum_{s=1}^b J_{f^i,s} = 0$ **then**
      $f^i \leftarrow f^i$ ▷ stay the same
    else
      $s^* = \arg\max_s\{J_{f^i,s}\}_{s=1}^b$
      $f^i \leftarrow s^*$ ▷ update node
    end if
  end for
  for $i, j = 1,\dots,n$ **do**
    if $\sum_{s=1}^a J_{e^{(i,j)},s} > 1$ or $\sum_{s=1}^a J_{e^{(i,j)},s} = 0$ **then**
      $e^{(i,j)} \leftarrow e^{(i,j)}$ ▷ stay the same
    else
      $s^* = \arg\max_s\{J_{e^{(i,j)},s}\}_{s=1}^b$
      $e^{(i,j)} \leftarrow s^*$ ▷ update edge
    end if
  end for
  $t \leftarrow t - \tau$
end while

E Auxiliary Features, PNA and FiLM Modules

For learning a better graph-to-graph mapping $p^\theta_{0|t}(\mathcal{G}_0|\mathcal{G}_t)$, artificially augmenting the node-level features and graph-level features is proved effective to enhance the expressiveness of graph learning models [81, 73]. For this setting, we keep consistent with the state-of-the-art model, DiGress [73], and extract the following three sets of auxiliary features. Note that the following features are extracted on the noised graph $\mathcal{G}_t$.

https://doi.org/10.52202/079017-2531

We binarize the edge tensor $\mathbf{E}$ into an adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$ whose 1 entries denote the corresponding node pair is connected by any type of edge.

Motif features. The number of length-$3/4/5$ cycles every node is included in is counted as the topological node-level features; also, the total number of length-$3/4/5/6$ cycles is the topological graph-level features.

Spectral features. The graph Laplacian is decomposed. The number of connected components and the first 5 non-zero eigenvalues are selected as the spectral graph-level features. An estimated indicator of whether a node is included in the largest connected component and the first 2 eigenvectors of the non-zero eigenvalues are selected as the spectral node-level features.

Molecule features. On molecule datasets, the valency of each atom is selected as the node-level feature, and the total weight of the whole molecule is selected as the graph-level feature.

The above node-level features and graph-level features are concatenated together as the auxiliary node-level features $\mathbf{F}_{\texttt{aux}}$ and graph-level features $\mathbf{y}$. An important property is that the above node-level features are permutation-equivariant and the above graph-level features are permutation-invariant, whose proof is straightforward so we omit it here.

Next, two important modules used in the MPNN backbone: PNA and FiLM are detailed.

PNA module. The PNA module [12] is implemented as follows,

$$\texttt{PNA}(\{\mathbf{x}_i\}_{i=1}^n) = \texttt{MLP}(\texttt{min}(\{\mathbf{x}_i\}_{i=1}^n) \oplus \texttt{max}(\{\mathbf{x}_i\}_{i=1}^n) \oplus \texttt{mean}(\{\mathbf{x}_i\}_{i=1}^n) \oplus \texttt{std}(\{\mathbf{x}_i\}_{i=1}^n)) \quad (70)$$

where $\oplus$ is the concatenation operator, $\mathbf{x}_i \in \mathbb{R}^d$; `min`, `max`, `mean`, and `std` are coordinate-wise, e.g., $\texttt{min}(\{\mathbf{x}_i\}_{i=1}^n) \in \mathbb{R}^d$.

FiLM module. FiLM [57] is implemented as follows,

$$\texttt{FiLM}(\mathbf{x}_i, \mathbf{x}_j) = \texttt{Linear}(\mathbf{x}_i) + \texttt{Linear}(\mathbf{x}_i) \odot \mathbf{x}_j + \mathbf{x}_j \quad (71)$$

where `Linear` is a single fully-connected layer without activation function and $\odot$ is the Hadamard product.

F Supplementary Details about Experiments

F.1 Hardware and Software

We implement DISCO in PyTorch[5] and PyTorch-geometric[6]. All the efficiency study results are from one NVIDIA Tesla V100 SXM2-32GB GPU on a server with 96 Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz processors and 1.5T RAM. The training on QM9 and Community can be finished in 2 hours. For the training on SBM, Planar, it can be finished within 48 hours to get decent validity. The training on MOSES and GuacaMol can be finished within 96 hours.

F.2 Dataset Setup

F.2.1 Dataset Statistics

The statistics about all the datasets used in this paper are presented in Table 7, where a is the number of edge types, b is the number of node types, $|\mathbf{E}|$ is the number of edges and $|\mathbf{F}|$ is the number of nodes.

F.2.2 Detailed Settings on Plain Graph Datasets

Dataset Split. We follow the settings of SPECTRE [51] and DiGress [73] to split the SBM, Planar [51], and Community [82] datasets into $64/16/20\%$ for training/validation/test set.

[5] `https://pytorch.org`

[6] `https://pytorch-geometric.readthedocs.io/en/latest`

Table 7: Dataset statistics.

Name	# Graphs	Split	a	b	Avg. $\lvert\mathbf{E}\rvert$	Max $\lvert\mathbf{E}\rvert$	Avg. $\lvert\mathbf{F}\rvert$	Max $\lvert\mathbf{F}\rvert$
SBM	200	128/32/40	1	1	1000.8	2258	104.0	187
Planar	200	128/32/40	1	1	355.7	362	64.0	64
Community	100	64/16/20	1	1	74.0	122	15.7	20
QM9	130831	97734/20042/13055	4	4	18.9	28	8.8	9
MOSES	1733214	1419512/156176/157526	4	8	46.3	62	21.6	27
GuacaMol	1398213	1118633/69926/209654	4	12	60.4	176	27.8	88

Metrics. The Maximum Mean Discrepancy (MMD) [82] measures the discrepancy between two sets of distributions. The relative squared MMD [73]is defined as follows

$$score = \frac{\text{MMD}^2(\{\mathcal{G}\}_{\texttt{gen}}||\{\mathcal{G}\}_{\texttt{test}})}{\text{MMD}^2(\{\mathcal{G}\}_{\texttt{train}}||\{\mathcal{G}\}_{\texttt{test}})}, \tag{72}$$

where $(\{\mathcal{G}\}_{\texttt{gen}}$, $(\{\mathcal{G}\}_{\texttt{train}}$, and $(\{\mathcal{G}\}_{\texttt{test}}$ are the sets of generated graphs, training graphs, and test graphs, respectively. We report the above relative squared MMD for degree distributions (Deg.), clustering coefficient distributions (Clus.), and average orbit counts (Orb.) statistics (the number of occurrences of all substructures with 4 nodes). In addition, the Uniqueness, Novelty, and Validity are chosen. Uniqueness reports the fraction of the generated nonisomorphic graphs; Novelty reports the fraction of the generated graphs not isomorphic with any graph from the training set; Validity checks the fraction of the generated graphs following some specific rules. For the SBM dataset, we follow the validity check from [51] whose core idea is to check whether real SBM graphs are statistically indistinguishable from the generated graphs; for the Planar dataset, we check whether the generated graphs are connected and are indeed planar graphs. Because the Community dataset does not have the Validity metric, we only report the Uniqueness, Novelty, and Validity results on the SBM and Planar datasets.

We report mean$\pm$std in 5 runs.

Baseline methods. GraphRNN [82], GRAN [42], GG-GAN [37], MolGAN [9], SPECTRE [51], EDP-GNN [56], GraphGDP [26], DiscDDPM [22], EDGE [10], ConGress [73], DiGress [73] are chosen.

F.2.3 Detailed Settings on Molecule Graph Datasets

Dataset Split. We follow the split of QM9 from DiGress [73] and follow the split of MOSES [58] and GuacaMol [6] according to their benchmark settings. Their statistics are presented in Table 7.

Metrics. For QM9, Uniqueness, Novelty, and Validity are chosen as metrics. The first two are the same as introduced in Section F.2.2. The Validity is computed by building a molecule with RdKit [7] and checking if we can obtain a valid SMILES string from it.

For MOSES, the chosen metrics include Uniqueness, Novelty, Validity, Filters, Fréchet ChemNet Distance (FCD), Similarity to a nearest neighbor (SNN), and Scaffold similarity (Scaf), which is consistent with DiGress [73]. The official evaluation code [8] is used to report the performance.

For GuacaMol, the chosen metrics include Uniqueness, Novelty, Validity, KL Divergence, and Frećhet ChemNet Distance (FCD), which is consistent with DiGress [73]. The official evaluation code [9] is used to report the performance.

We report mean$\pm$std in 5 runs except MOSES and GuacaMol, whose computations are too expensive to repeat multiple times.

Baseline methods. CharacterVAE [20], GrammarVAE [38], GraphVAE [66], GT-VAE [55], Set2GraphVAE [72], GG-GAN [37], MolGAN [9], SPECTRE [51], GraphNVP [50], GDSS [32],

[7] `https://www.rdkit.org/`

[8] `https://github.com/molecularsets/moses`

[9] `https://github.com/BenevolentAI/guacamol`

https://doi.org/10.52202/079017-2531

Table 8: Generation performance (mean±std) on the Community dataset.

Model	Deg.↓	Clus.↓	Orb.↓
GraphRNN [82]	4.0	1.7	4.0
GRAN [42]	3.0	1.6	1.0
EDP-GNN [56]	2.5	2.0	3.0
GraphGDP [26]	2.0	1.1	-
DiscDDPM [22]	1.2	**0.9**	1.5
EDGE [10]	1.0	1.0	2.0
GG-GAN [37]	4.0	3.1	8.0
MolGAN [9]	3.0	1.9	1.0
SPECTRE [51]	0.5	2.7	2.0
DiGress [73]	1.0	**0.9**	1.0
DISCO-MPNN	1.4±0.5	**0.9**±0.2	**0.9**±0.3
DISCO-GT	**0.9**±0.2	**0.9**±0.3	1.1±0.4

EDGE [10], ConGress [73], DiGress [73], GRAPHARM [36],VAE [21], JT-VAE [29], GraphINVENT [53], LSTM [64], NAGVAE [40], and MCTS [28] are chosen.

F.3 Hyperparameter Settings

Forward Diffusion Settings. As we introduced in Proposition 3.2, we tried two sets of rate matrices for the node and edge forward diffusion, so that the converged distribution is either uniform or marginal distribution. We found the marginal distribution leads to better results than the uniform distribution. Thus, the reference distribution is the marginal distribution for all the main results, except Tables 6 and 9. The performance comparison between the marginal diffusion and uniform diffusion is presented in the ablation study in Sections 4.4 andF.5. The $\beta(t)$ controls how fast the forward process converges to the reference distribution, which is set as $\beta(t) = \alpha\gamma^t log(\gamma)$, which is consistent with many existing works [23, 70, 8]. In our implementation, we assume the converged time $T = 1$ and for the forward diffusion hyperparameters (α, γ) we tried two sets: $(1.0, 5.0)$ and $(0.8, 2.0)$ where the former one can ensure at $T = 1$ the distribution is very close to the reference distribution, and the latter one does not fully corrupt the raw data distribution so the graph-to-graph model $p^{\theta}_{0|t}$ is easier to train.

Reverse Sampling Settings. The number of sampling steps is determined by τ, which is $\texttt{round}(\frac{1}{\tau})$ if we set the converged time $T = 1$. We select the number of sampling steps from $\{50, 100, 500\}$, which is much smaller the number of sampling steps of DiGress [73] from $\{500, 1000\}$. For the number of nodes n in every generated graph, we compute a graph size distribution of the training set by counting the number of graphs for different sizes (and normalize the counting to sum it up to 1). Then, we will sample the number of nodes from this graph size distribution for graph generation.

Neural Network Settings. For DISCO-GT, the parametric graph-to-graph model $p^{\theta}_{0|t}$ is graph transformer (GT). We use the exactly same GT architecture as DiGress [73] and adopt their recommended configurations [10]. The reason is that this architecture is not our contribution, and setting the graph-to-graph model $p^{\theta}_{0|t}$ same can ensure a fair comparison between the discrete-time graph diffusion framework (from DiGress) and the continuous-time graph diffusion framework (from this work). For DISCO-MPNN, we search the number of MPNN layers from $\{3, 5, 8\}$, set all the hidden dimensions the same, and search it from $\{256, 512\}$. For both variants, the dropout is set as 0.1, the learning rate is set as $2e^{-4}$, and the weight decay is set as 0.

F.4 Additional Results on Community

Additional Community plain graph dataset results are in Table 8. Our observation is consistent with the main content: both variants of DISCO are on par with, or even better than the SOTA general graph diffusion generative model, DiGress.

[10] `https://github.com/cvignac/DiGress/tree/main/configs/experiment`

Table 9: Ablation study (mean±std%) with MPNN backbone. V., U., and N. mean Valid, Unique, and Novel.

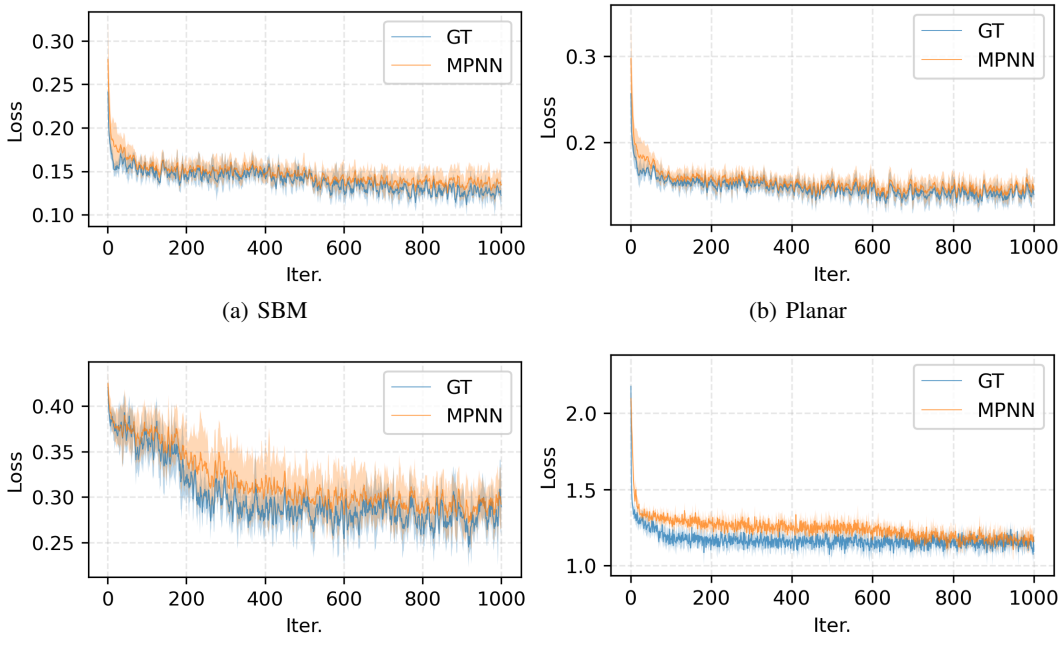
Ref. Dist.	Steps	Valid ↑	V.U. ↑	V.U.N. ↑
Marginal	500	98.9±0.7	98.7±0.5	68.7±0.2
	100	98.4±1.1	98.0±1.0	69.1±0.6
	30	97.7±1.2	97.5±0.8	70.4±1.1
	10	92.3±1.9	91.9±2.2	66.4±1.7
	5	88.8±3.3	87.1±2.8	67.3±2.9
	1	64.4±2.7	63.2±1.9	55.8±1.4
Uniform	500	93.5±1.7	93.2±1.1	64.9±1.0
	100	93.1±2.1	92.6±1.7	66.2±1.9
	30	87.1±1.8	86.8±1.1	64.0±1.0
	10	83.7±3.2	81.9±2.1	61.3±2.0
	5	81.5±2.9	75.4±3.4	64.6±2.3
	1	71.3±2.3	42.2±4.0	36.9±3.2

F.5 Additional Ablation Study

Table 9 shows the ablation study of DISCO-MPNN on QM9 dataset. Our observations are consistent with the main content: (1) generally, the fewer sampling steps, the lower the generation quality but method's performance is robust in terms of the decreasing of sampling steps; (2) the marginal reference distribution is better than the uniform distribution, consistent with the observation from DiGress [73].

F.6 Convergence Study

Figure 3 shows the training loss of DISCO-GT and DISCO-MPNN on four datasets, whose X-axis is the number of iterations (i.e., the number of epochs $\times$ the number of training samples / batch size). We found that overall the training losses converge smoothly on 4 datasets.



(a) SBM (b) Planar

(c) Community (d) QM9

Figure 3: Training loss of DISCO on different datasets and backbone models.

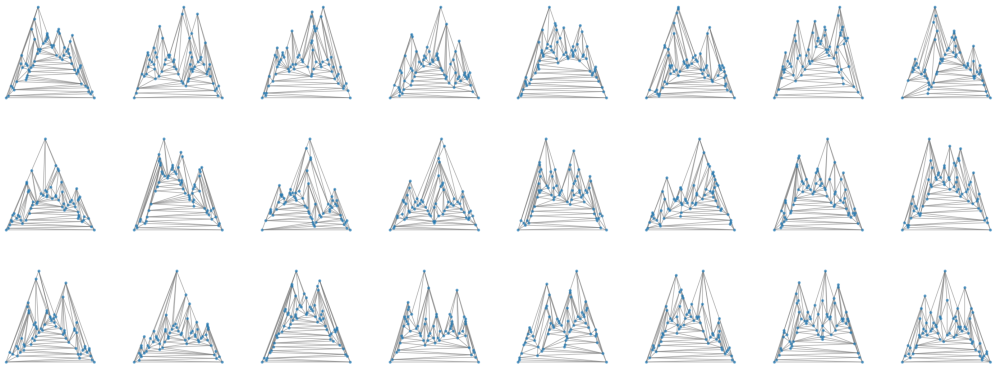
https://doi.org/10.52202/079017-2531

F.7 Visualization

The generated graphs on the SBM and Planar datasets are presented in Figure 4. We clarify that the generated planar graphs are *selected to be valid* because, as Table 1 shows, not all the generated graphs are valid planar graphs, but the planar layout can only visualize valid planar graphs in our setting [11]. The generated SBM graphs are not selected; even if a part of them cannot pass the strict SBM statistic test (introduced in Section F.2.2 - Metrics), most, if not all, of them still form $2-5$ densely connected clusters.

The generation trajectory of SBM graphs is presented in Figure 5 which demonstrates the reverse denoising process visually.

(a) SBM



(b) Planar

Figure 4: Generated graphs.

G Limitation and Future Work

In this paper, we study the generation of graphs with categorical node and edge types. The current model DISCO cannot be applied to generate graphs with multiple node/edge features (e.g., multiplex networks) and this is an important future work to study. Also, we view the absence of edge as a special type of edge, which forms a complete graph and promotes the expressiveness of our MPNN backbone model. However, it will lead to quadratic complexity concerning the number of nodes. For our current dataset (e.g. graphs with < 1000 nodes) the complexity is still acceptable but for future studies on generating *large* graphs, we aim to design more efficient diffusion generative models.

[11] `https://networkx.org/documentation/stable/reference/generated/networkx.drawing.layout.planar_layout.html`

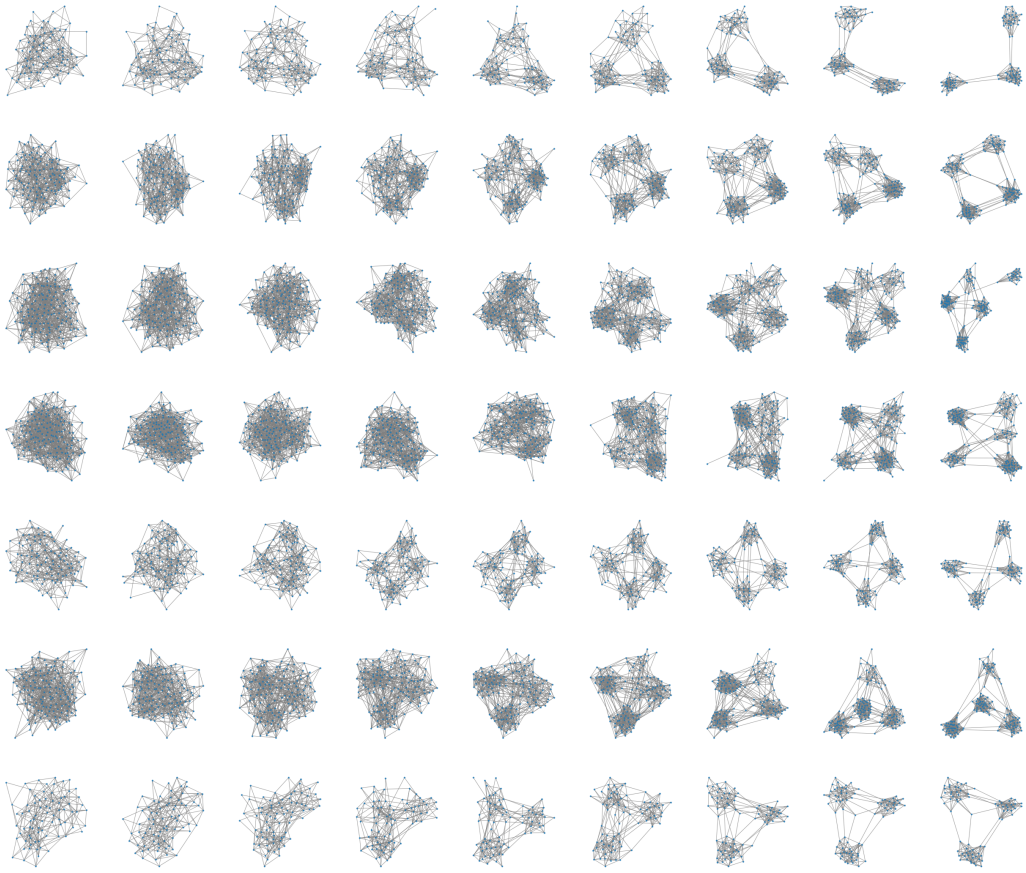


Figure 5: Generation trajectory of SBM graphs with different sizes. Every row is the generation trajectory of one graph from time $t = T$ (left) to $t = 0$ (right) with equal time intervals.

H Broader Impact

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

https://doi.org/10.52202/079017-2531

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction summarize all the theoretical and experimental contributions of this paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitation is mentioned in Section G.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: Detailed assumptions and proofs are included in the Section C.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: Detailed experimental settings are included in Section F. Also, the code of this paper is released in the supplementary materials.

   Guidelines:

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

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

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

https://doi.org/10.52202/079017-2531

Answer: [Yes]

Justification: All the datasets are publicly available and their links are included in Section F. The code is released in the supplementary materials; we will formally release the code after acceptance.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: All the training details are included in Section F and the released codes.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: We report average results with standard deviation on all the datasets, except MOSES and GuacaMol, whose computations are too expensive to repeat multiple times.

   Guidelines:

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

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: The compute resources are detailed in Section F.1.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: We checked the code of Ethics and the paper conforms with the Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: It is mentioned in Section H. Our work is general graph generative modeling, which shares potential societal consequences with many established graph generative models.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-2531

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: We did not scrape any dataset from Internet and our released model is of low risk for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All the datasets and codes of baseline methods are publicly available and for the academic purpose.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: We did not release any new dataset and the code released will be well documented after the acceptance.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

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

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-2531