When Edge Independence Fails: Joint Graph Diffusion with Latent Sociability Prior

Adarsh Jamadandi[1], **Nicolas Keriven**[1], **Aline Roumy**[2]
CNRS, IRISA, Rennes, France[1], INRIA, Rennes, France[2]
adarsh.jamadandi@irisa.fr, nicolas.keriven@cnrs.fr, aline.roumy@inria.fr

Abstract

Discrete graph diffusion models corrupt graph structure through independent edge updates with a Markovian noise model. Combined with the exchangeability requirement on the output distribution, the terminal prior is necessarily an Erdős–Rényi (E-R) random graph. This has a fundamental consequence for unconditional graph generation without node features: in the large graph limit, any GNN denoiser applied to an E-R graph approximately outputs an E-R graph, making effective denoising difficult. In practice, generated graphs fail to recover meaningful node heterogeneity and higher-order structure. Previous models introduced specialized node features to compensate, framing this as an expressivity problem, while the role of the prior remained unexamined. We instead address the root cause by introducing the *Latent Sociability Prior* (LSP) and a *joint* diffusion process that co-evolves latent node sociability variables and graph structure, removing edge independence while preserving exchangeability. Unlike existing latent graph diffusion methods, the sociability mechanism requires no complex encoder: its sole role is to guide edge updates and preserve node heterogeneity throughout the trajectory. We validate our approach on multiple graph generation datasets and show that the proposed framework more accurately captures graph distributions than baseline methods without node features.

1 Introduction

Diffusion models [48, 49, 18, 36] have recently emerged as a powerful class of generative models, with major improvements in image [55] and video generation [34]. Motivated by this success, diffusion-based methods have been extended to graph generation. The two most popular approaches are: embedding nodes into a continuous latent space and applying Gaussian diffusion [38, 22, 12], or operating directly in the graph space through a discrete corruption process on edges [3]. The latter approach has been shown to be more appropriate for generating realistic graph structures [17, 51], leading to a growing body of work on both graph diffusion and discrete flow matching [15, 41, 43].

Despite this recent progress, discrete diffusion models also have inherent problems. For instance, scaling to large graphs is a bottleneck as the computational complexity grows quadratically with increasing graph size. Recent methods [9, 31, 42] have made notable progress on addressing these practical limitations. In this work, however, we identify a more fundamental limitation at the core of existing graph diffusion models. We find that, in unconditional graph generation without node features, the combination of edge independence in the forward process and the requirement that the output graph distribution remains exchangeable tightly constrains the terminal distribution, leaving Erdős–Rényi (E-R) as the *only* admissible prior. We show theoretically and empirically that such graph diffusion models exhibit a *fixed-point* pathology in the large graph limit: for most architectures including attention-based GNNs, when initialized from an E-R prior, *any* denoiser approximately outputs constant edge predictions (Theorem 4.2) which is also an E-R random graph. Consequently, the generated graphs tend to look homogeneous and lack higher-order structure.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In practice, this failure is commonly attributed to insufficient model expressivity, with auxiliary node features introduced as a remedy even in the unconditional setting [51, 9, 42], leaving the role of the E-R prior unexamined. While such features may improve performance, they can be expensive to compute and obscure the real source of failure. Our claim is therefore not simply that existing models are insufficiently expressive, but that the standard edge-independent discrete diffusion pipeline is fundamentally mismatched with unconditional graph generation.

As a resolution, we propose a *joint diffusion* framework that co-evolves continuous latent variables together with the discrete graph structure, combining the strengths of latent diffusion models and graph-space discrete diffusion. Our framework is built around a new prior inspired by inhomogeneous random graph models [39, 4, 19, 5], termed the *Latent Sociability Prior* (LSP). Each node is endowed with a positive sociability weight, and edge probabilities are conditioned on these weights, inducing node-level heterogeneity. We jointly diffuse these variables and the graph edges during both noising and denoising, allowing them to guide edge updates throughout the diffusion trajectory. As a consequence, the terminal prior is no longer edge-independent and therefore does not induce the fixed-point behavior of standard discrete denoisers. Crucially, the sociability mechanism is much simpler than existing latent graph diffusion approaches, as its role is not to generate graphs alone but to guide the companion discrete diffusion process. In particular, it requires no pre-trained encoder, and the clean latent variables admit an approximate closed-form expression from the training data.

Our contributions can be summarized as follows:

1. We identify a fundamental limitation in existing discrete graph diffusion pipelines. Without node features, an edge-independent terminal prior together with independent edge updates leads to a fixed-point pathology: any denoiser applied to an E-R random graph approximately yields an E-R random graph, making it difficult to escape this regime and preventing recovery of meaningful node heterogeneity and higher-order structure.
2. We propose a joint diffusion framework for graph generation that co-evolves continuous latent variables together with discrete graph structure, allowing latent node heterogeneity to directly guide edge updates throughout both noising and denoising.
3. We instantiate this framework through the *Latent Sociability Prior* (LSP), a novel exchangeable latent-variable prior inspired by inhomogeneous random graph models. The LSP admits a closed-form approximation for the clean latent variables, bypassing the need for an encoder, while inducing beneficial edge dependencies and preserving node-level heterogeneity.
4. We validate our approach on multiple graph generation benchmarks and show that, even without node features, the proposed framework more accurately captures the target graph distribution compared to existing baselines.

2 Related work

Latent graph diffusion. Extending the original diffusion formulation [48, 49, 18, 36] for graph generation involves embedding the graphs into a low-dimensional continuous space using variational autoencoders (VAEs) and corrupting the graph structure using Gaussian noise in this latent space [38, 22, 12]. Although this approach is appealing since it allows for a computationally friendly diffusion process, it also presents unique challenges. For instance, VAEs are notorious for training instabilities [10] and small reconstruction errors in the encoding-decoding process can affect the structural validity of generated graphs. Further, generating large graphs is challenging since the dimensionality of the latent space itself becomes a bottleneck, limiting the structural detail that can be faithfully recovered.

Discrete diffusion for graphs. Another popular approach for graph diffusion is to adopt the discrete diffusion paradigm [3] where the corruption process is carried out directly in the space of graphs through independent edge additions and deletions. This framework has been shown to be more appropriate for generating realistic graph structures, resulting in large improvements in both diffusion models [17, 51, 9, 31, 40, 37, 23, 47] and discrete flow matching [15, 21, 41, 43]. However, most of these advancements address practical issues such as scalability, since the computational burden grows quadratically with increasing graph sizes. In this work, we identify a more fundamental limitation specific to discrete graph diffusion models: the edge-independent forward process forces the terminal

prior to be Erdős–Rényi, which leads to a fixed-point pathology that constrains generation quality independently of model capacity.

Influence of prior distributions. The role of the prior in graph generation has received some attention. [6, 7] study the limitations of edge-independent generative models and show they cannot simultaneously produce diverse graphs and capture higher-order statistics such as triangle and cycle counts a theoretical finding that directly motivates our work. In the continuous flow matching setting, [8] use embedding-based source distributions as informed starting points, and [52] propose motif-based priors via Gromov–Wasserstein-coupled flow matching, improving structural fidelity at the cost of expensive optimal transport and manual motif specification per graph family. These methods operate outside the discrete diffusion setting and do not address the fixed-point pathology we identify. Our approach requires no optimal transport and no domain-specific prior engineering: node heterogeneity emerges directly from the latent sociability variables by construction.

3 Background on discrete denoising diffusion framework

We recall the discrete denoising diffusion framework for graph generation following [3, 51]. Let $\mathcal{D}_{\text{data}} = \{G_1, \dots, G_M\}$ be a dataset of graphs drawn from an unknown distribution p_{data}. For a graph $G = (X, E)$ with node features X and edges E, the goal is to learn a distribution $p_\theta(G)$ from which new graphs can be sampled. Node features and edges are represented via one-hot encodings, giving tensors $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{E} \in \mathbb{R}^{n \times n \times b}$, where d is the number of node feature types and b the number of edge categories. Here we simply consider $b = 2$ edge categories, with $e_{ij} \in \mathcal{B} = \{0, 1\}$ denoting the scalar edge state (absence or presence) between nodes i and j, and $\mathbf{e}_{ij} \in \mathbb{R}^b$ its one-hot encoding, organized into the edge tensor $\mathbf{E} \in \mathbb{R}^{n \times n \times b}$.

Since we focus on unconditional graph generation *without node features* we set $\mathbf{X} = \mathbf{1}$ throughout, unless otherwise written. The forward process defines a sequence of increasingly noisy graphs $G^0, G^1, \dots, G^T$, where G^0 is a clean training graph and G^T is drawn from the terminal prior π_T. We denote by $e^0_{ij} \in \mathcal{B}$ the clean edge state between nodes i and j in G^0, and by $\mathbf{e}^0_{ij} \in \mathbb{R}^b$ its one-hot encoding. The probability of a noisy sequence $(G^1, \dots, G^T)$ given G^0 factorizes as

$$q(G^{1:T} \mid G^0) = \prod_{t=1}^{T} q(G^t \mid G^{t-1}) \tag{1}$$

The transition matrix $\mathbf{Q}^t_E \in \mathbb{R}^{b \times b}$ governs one-step edge corruption, with entries $[\mathbf{Q}^t_E]_{rs} = q(e^t = s \mid e^{t-1} = r)$, and $\bar{\mathbf{Q}}^t_E = \mathbf{Q}^1_E \cdots \mathbf{Q}^t_E$ is the cumulative transition matrix. Since transitions act independently on each edge, the per-edge forward marginal is the $1 \times b$ probability vector $q(e^t_{ij} \mid e^0_{ij}) = \mathbf{e}^0_{ij}\, \bar{\mathbf{Q}}^t_E$. The joint forward marginal over all edges is

$$q(\mathbf{E}^t \mid \mathbf{E}^0) = \prod_{1 \le i,j \le n} q(e^t_{ij} \mid e^0_{ij}) \tag{2}$$

For undirected graphs, noise is applied only to the upper-triangular part of $\mathbf{E}$ and then symmetrized. A denoising network ϕ_θ takes a noisy graph $G^t = (\mathbf{1}, \mathbf{E}^t)$ and timestep t as input, and outputs for each node pair (i, j) a probability vector $\hat{\mathbf{p}}^{E,t}_{ij} \in \mathbb{R}^b$ over edge types. It is optimized via cross-entropy against the clean edge states:

$$\mathcal{L}_{\text{CE}}(\hat{\mathbf{p}}^{E,t}, G^0) = \sum_{1 \le i,j \le n} \text{CE}\Big(e^0_{ij}, \hat{\mathbf{p}}^{E,t}_{ij}\Big) \tag{3}$$

New graphs are sampled by running the reverse process from $G^T \sim \pi_T$. Since node features remain fixed, the reverse distribution factorizes over edges:

$$p_\theta(\mathbf{E}^{t-1} \mid G^t) = \prod_{1 \le i,j \le n} p_\theta(e^{t-1}_{ij} \mid G^t) \tag{4}$$

Each factor is computed by marginalizing over the predicted clean edge distribution:

$$p_\theta(e^{t-1}_{ij} \mid G^t) = \sum_{r \in \mathcal{B}} q\big(e^{t-1}_{ij} \mid e^t_{ij}, e^0_{ij} = r\big)\, \hat{\mathbf{p}}^{E,t}_{ij}(r) \tag{5}$$

where $q(e_{ij}^{t-1} \mid e_{ij}^t, e_{ij} = r)$ is the forward posterior, available in closed form in [51] and $\hat{\mathbf{p}}_{ij}^{E,t}(r)$ denotes the network's estimate of the probability that the clean edge takes value r given the noisy graph G^t. As we will see later, by Theorem 4.2, when G^t is an E-R random graph, $\hat{\mathbf{p}}_{ij}^{E,t}$ is approximately constant across all pairs (i, j), so $p_\theta(e_{ij}^{t-1} \mid G^t)$ is the same for all pairs and $\mathbf{E}^{t-1}$ again follows an E-R distribution.

Transition matrices are chosen as in [51] to preserve the empirical marginal distribution of edge types $\mathbf{m}_E \in \mathbb{R}^b$ throughout diffusion through $\mathbf{Q}_E^t = \alpha_t I + \beta_t \mathbf{1}_b \mathbf{m}_E^\top$ where, $\beta_t = 1 - \alpha_t$. With probability α_t the edge state is preserved, and with probability β_t it is resampled from $\mathbf{m}_E$.

As $t \to T$, every edge is driven independently towards $\mathbf{m}_E$, so the terminal prior $\pi_T := q(G^T)$ is a product of independent marginals, an Erdős–Rényi random graph where every edge is drawn independently with probability equal to the empirical edge density of the training set.

Exchangeability. An important feature of any graph generation method is that it needs to be *exchangeable*: all permutations of the generated graphs need to be equiprobable. In other words, if we denote by $\sigma \cdot G$ the relabelling of a graph G by a permutation σ, our entire generation process needs to satisfy

$$p_\theta(\sigma \cdot G^0) = p_\theta(G^0)$$

for all σ, G^0. Fortunately, if a) the prior distribution $\pi_T = q(G^T)$ is exchangeable (which is naturally the case of E-R random graphs), and b) the denoiser ϕ_θ is *permutation equivariant*[1], then it is easy to see that the final distribution is exchangeable: we have $p_\theta(\sigma \cdot G^{t-1} \mid \sigma \cdot G^t) = p_\theta(G^{t-1} \mid G^t)$ by equivariance, and $q(\sigma \cdot G^T) = q(G^T)$ by exchangeability of the prior.

4 Issue with Erdős–Rényi prior

In this section, we show that the E-R prior introduces a fundamental limitation in the generation process: in the large-graph limit, E-R random graphs are approximate *fixed points* for *any* denoiser. That is, as the graph size grows, denoising an E-R graph approximately yields another E-R graph, making it increasingly difficult to escape this regime. We begin by asking whether a different prior could have been chosen within the discrete diffusion framework.

E-R prior is unavoidable. As we have seen in the previous section, the prior distribution π_T needs to be exchangeable to guarantee exchangeability of the whole generation process. Moreover, the discrete diffusion process is based on successive *independent* edge perturbations, such that the terminal distribution has fully independent edges. We then have the following result, shown in Appendix B.1 for completeness.

Lemma 4.1. *[2, 20] Any **exchangeable** random graph model with **independent edges** is an E-R model.*

In other words, as long as discrete diffusion operates with *independent edges*, the terminal distribution π_T is *necessarily* an E-R random graph.

E-R is a fixed point for denoisers. Consider a simple denoiser that performs link prediction by first computing node representations with a GNN $\hat{\mathbf{X}} = \phi_\theta(\mathbf{X}, \mathbf{E})$ (recall that we generally consider $\mathbf{X} = 1$ in this paper, but the result below is more generally valid for node features $\mathbf{X}$), then compute edge probabilities using these representations with a similarity function φ:

$$\hat{p}_{ij} = \varphi(\mathbf{x}_i^{(K)}, \mathbf{x}_j^{(K)}) \in [0, 1]$$

This method is akin to a Variational Graph Auto-Encoder (VGAE) [26, 27], but when the GNN is a GAT [50], this is close to the graph transformer [11] used in DiGress [51], removing some normalization layers and edge modulation for simplicity [2]. Denote by

$$\texttt{std}(\hat{p}) = \sqrt{\tfrac{1}{n^2}\textstyle\sum_{ij}\left|\hat{p}_{ij} - \tfrac{1}{n^2}\sum_{ij}\hat{p}_{ij}\right|^2}, \quad \texttt{std}(\mathbf{X}) = \sqrt{\tfrac{1}{n}\textstyle\sum_i \left\|\mathbf{X}_i - \tfrac{1}{n}\sum_i \mathbf{X}_i\right\|^2} \tag{6}$$

[1] that is, $\phi_\theta(\sigma \cdot G) = \sigma \cdot \phi_\theta(G)$

[2] Technically, in vanilla graph transformers the edge probabilities would also be modulated by the input graph $\hat{p}_{ij} = \varphi(\mathbf{x}_i^{(K)}, \mathbf{x}_j^{(K)}, e_{ij})$, in which case our result become: $\hat{\mathbf{p}}^E$ is approximately an *affine function* of $\mathbf{E}$ instead of being approximately constant. That is: the output graph is a probabilistic "sum" of the input E-R graph and another E-R graph. We choose our version for simplicity.

the standard deviation of the edge probabilities and input node features, respectively, such that E-R random graphs have exactly $\texttt{std}(\hat{p}) = 0$. We have the following result.

Theorem 4.2. *Consider an input graph* $\mathbf{E}$ *of size* n*, drawn from an Erdős–Rényi model with parameter* p*. Then, for the GNNs outlined below, there is* C *such that: for all* $\delta, \epsilon > 0$*, if* n *is large enough, then with probability at least* $1 - \delta$*,*

$$\texttt{std}(\hat{p}) \lesssim C\texttt{std}(\mathbf{X}) + \epsilon$$

where the multiplicative constant only depends on the parameters of the GNN. We have:

- *if the GNN is a GAT [50] with* K *layers, then* $C = (1/p)^K$*, and* n *is such that*

$$pn \gtrsim \log(n/\delta) \tag{7}$$

  In particular, one can choose $\epsilon = 0$ *in this case.*

- *if the GNN is a GCN [28], then* $C = O(1)$*, and* n *is such that*

$$pn \gtrsim \epsilon^{-2} \log(n/\delta) \tag{8}$$

- *if the GNN is GIN [53], for all* $r > 0$ *there exists* C_r *such that the result is valid with* $C = O(1)$*, and* n *such that*

$$pn \gtrsim \epsilon^{-2} \log(n/\delta) \quad \textit{and} \quad n \geq C_r \delta^{-r} \tag{9}$$

The details for this theorem and precise description of the considered GNNs are given in Appendix B.2. In short, we thus have the following: when the node features have low standard deviation, the denoiser outputs approximately constant edge probabilities. Constant features are an extreme version of that, since $\texttt{std}(\mathbf{1}) = 0$. In other words, the class of E-R random graphs represents an approximate *fixed-point* for denoisers, and as a result node heterogeneity remains low during the whole diffusion process (Figure 2), as all graphs will be almost E-R-like.

Applicability of Theorem 4.2 on Graph Transformer. We prove Theorem 4.2 on a simplified link predictor in Appendix B.2 while in practice DiGress uses a complex Graph Transformer with its edge predictions modulated by FiLM layers [51]. We argue this does not change the conclusion as this setup provides with a tractable model to study behavior. Further, the edge modulation in the DiGress transformer is applied per pair: each pair's modulation is computed from its own edge feature and from the global (timestep) vector, so it does not introduce dependence between different pairs. With constant node features, the predicted edge logits depend on each pair only through its own current state and a global summary of the graph, leading to E-R-like graphs and preventing node heterogeneity from emerging. Empirically, this is what we observe on the model used in practice: Fig. 2 shows the node propensity variance of the actual DiGress denoiser, with edge modulation, collapsing toward zero at high noise, while the LSP-based model retains heterogeneity throughout.

Structural node features. The above issue has often been interpreted as a lack of "expressivity" of GNNs, hampering their denoising abilities. To fix this, many works add structural node features computed *a posteriori* at each denoising step, such as degrees, cycle counts, and so on. While such node features indeed counteract the incapacity of GNNs to compute these quantities, they only offer limited fix to the E-R problem, as shown in the following Lemma proved in Appendix B.3.

Lemma 4.3. *Consider a dense E-R random graph with parameter* $p = O(1)$ *for simplicity. Consider a template graph* H *of size* r*, and denote by* C_i *the number of subgraphs equal to* H *containing node* i*. Then* $C = \mathbb{E}(C_i)$ *does not depend on* i*, and with probability at least* $1 - \delta$*,*

$$\texttt{std}(C_i/C) \lesssim \sqrt{\log(n/\delta)/n}$$

Hence, in the limit, this does not solve the problem, as the standard deviation of such features also vanishes. This is shown in Figure 7 in the Appendix.

5 Latent variable guided discrete diffusion

Lemma 4.1 establishes that the E-R prior is not merely a sub-optimal choice but an unavoidable consequence of edge independence and exchangeability, and Theorem 4.2 and Lemma 4.3 state that

E-R graphs become difficult to denoise in the large graph limit, even with structural features computed *a posteriori*. To resolve this, we propose a *new prior* that induces node-level heterogeneity directly through latent variables, along with the corresponding *joint* diffusion process on latent variables and discrete graph structure.

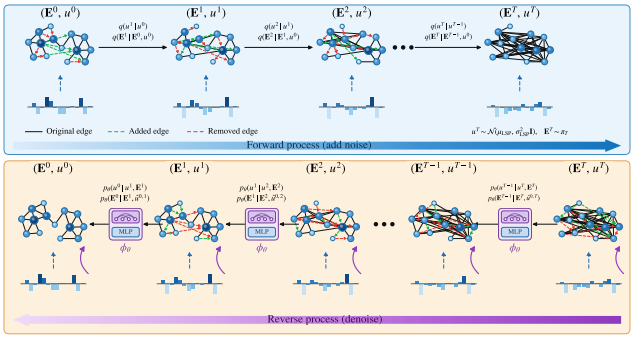


Figure 1: **Joint diffusion over edges and latent sociability.** *Top:* the latent sociability u^t and edge states $\mathbf{E}^t$ are jointly noised, with edge transitions conditioned on u^0, until reaching $u^T \sim \mathcal{N}(\mu_{\text{LSP}}, \sigma^2_{\text{LSP}})$ and $\mathbf{E}^T \sim \pi_T$. *Bottom:* during denoising, the graph transformer ϕ_θ predicts the less noisy latent $\hat{u}^{t-1}$, which conditions the reverse edge transition $p_\theta(\mathbf{E}^{t-1} \mid \mathbf{E}^t, \hat{u}^{t-1})$.

5.1 Latent Sociability Prior (LSP)

Unlike existing latent graph diffusion approaches [47], which encode the entire graph structure into high-dimensional latent space and may require complex autoencoders, the role of the latent variables here is strictly to enforce node heterogeneity along the discrete diffusion process. This allows our choice of latent model to be extremely simple.

Definition 1 (Latent Sociability Distribution [39])**.** *Let* $w_1, \dots, w_n \overset{i.i.d.}{\sim} p_W$ *be latent sociability weights, where* p_W *is a distribution on* $\mathbb{R}_{>0}$ *to be specified in Section 5.2. The latent sociability distribution defines a random undirected graph* G *with* conditionally *independent Bernoulli edges as*

$$\mathbb{P}(e_{ij} = 1 \mid w_i, w_j) = 1 - e^{-w_i w_j}, \qquad i < j, \tag{10}$$

We refer to this distribution of random graphs as Latent Sociability Prior (LSP). By construction, the edges are conditionally independent given $w_1, \dots, w_n$ but marginally dependent, while global exchangeability is preserved. Intuitively, a node with high weights w_i will tend to be more "connected": all the p_{ij} will be higher. Hence, a distribution p_W with a large variance *guarantees* a diverse profile of connection propensities (see Figure 2), and empirically a variety of higher-order structures as well (Figure 6), unlike the E-R prior. Integrating this latent-variable prior into the discrete denoising framework of Section 3 is not immediately obvious. We develop below a *joint* diffusion process that combines continuous Gaussian diffusion [18] on the latent weights with the discrete corruption process on the graph edges.

5.2 Joint diffusion with LSP

A direct diffusion on the weights w_i is not suitable, since Gaussian diffusion would not preserve the positivity constraint and could produce invalid intermediate states. We therefore reparameterize the weights as $w_i = \exp(u_i)$ with $u_i \in \mathbb{R}$ unconstrained. Under a Gaussian prior on u_i, the induced prior on w_i is log-normal. After this reparameterization, the edge probability in Equation 10 becomes

$$p_{ij} = 1 - e^{-e^{u_i + u_j}} \tag{11}$$

Forward process. We define a joint forward noising process over the continuous latent vector $\mathbf{u}$ and the discrete edge tensor $\mathbf{E}$. Given a clean graph G^0, one must first construct the corresponding clean latent vector $\mathbf{u}^0$. While usual latent approaches require training an encoder, in the LSP $\mathbf{u}^0$ admits a simple, approximate *closed-form* expression.

Lemma 5.1. *Consider an LSP random graph. Assuming a sparse regime where $u_i = O(-c_n)$ for some constant $c_n \gg 0$, we have for all i*

$$\mathbb{E}(d_i \mid \mathbf{u}) = Ce^{u_i} + O(ne^{-4c_n})$$

for some constant $C = O(ne^{-c_n})$, where d_i denotes the degree of node i.

Following this, we set $u_i^0 \approx \log(d_i + \delta) + \mu$ where $\delta > 0$ is a small smoothing constant. To better capture relative degree heterogeneity across different training graphs, we remove the graph-level mean and obtain

$$u_i^0 = \mu_{\mathrm{LSP}} + \log(d_i + \delta) - \frac{1}{|V|} \sum_{k \in V} \log(d_k + \delta) \tag{12}$$

where μ_{LSP} is the mean of the terminal Gaussian prior. Similarly to enforcing the marginal edge density in the E-R prior, we calibrate the parameters $(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}})$ from the training data, as detailed in Appendix A.1.

We corrupt the latent variables using the shifted and scaled DDPM forward process

$$\mathbf{u}^t = \mu_{\mathrm{LSP}} + \sqrt{\bar{\alpha}_t}\,(\mathbf{u}^0 - \mu_{\mathrm{LSP}}) + \sigma_{\mathrm{LSP}}\sqrt{1 - \bar{\alpha}_t}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{13}$$

so that $\mathbf{u}^t = \mathbf{u}^0$ at $t = 0$ and $\mathbf{u}^t$ converges in distribution to $\mathcal{N}(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}}^2 I)$ as $t \to T$. The derivation and reverse posterior are given in Appendix A.2.

For the discrete component, we modify the transition matrix of Section 3 by replacing the global stationary $\mathbf{m}_E$ with a *pair-specific* stationary

$\mathbf{p}_{ij}^0 = [\,1 - p_{ij}^0,\ p_{ij}^0\,] \in \mathbb{R}^b$, where $p_{ij}^0 = 1 - e^{-e^{u_i^0 + u_j^0}}$. This gives the pair-specific transition matrices

$$\mathbf{Q}_E^{t,ij} = \alpha_t I + \beta_t \mathbf{1}_b(\mathbf{p}_{ij}^0)^\top, \qquad \bar{\mathbf{Q}}_E^{t,ij} = \bar{\alpha}_t I + (1 - \bar{\alpha}_t)\mathbf{1}_b(\mathbf{p}_{ij}^0)^\top \tag{14}$$

with entries $[\mathbf{Q}_E^{t,ij}]_{rs} = q(e_{ij}^t = s \mid e_{ij}^{t-1} = r, \mathbf{u}^0)$. Since $\mathbf{u}^0$ is a deterministic function of G^0, all matrices are fixed given $(\mathbf{E}^0, \mathbf{u}^0)$, and the joint forward marginal factors over independent edges:

$$q(\mathbf{E}^t \mid \mathbf{E}^0, \mathbf{u}^0) = \prod_{1 \le i,j \le n} q(e_{ij}^t \mid e_{ij}^0, \mathbf{u}^0), \qquad q(e_{ij}^t \mid e_{ij}^0, \mathbf{u}^0) = e_{ij}^0\, \bar{\mathbf{Q}}_E^{t,ij} \tag{15}$$

This has the same closed-form structure as Equation 2, with pair-specific stationaries, so the forward posteriors needed for reverse sampling remain tractable [51]. Crucially, parameterizing the transitions by $\mathbf{u}^0$ rather than $\mathbf{u}^t$ is what makes this possible: if the transitions depended on $\mathbf{u}^t$, the matrices would be random variables and the product structure in Equation 15 would no longer hold.

Reverse process. Starting from $(\mathbf{u}^t, \mathbf{E}^t)$, the network ϕ_θ predicts the clean latent vector $\hat{\mathbf{u}}^{0,t} = \phi_\theta(\mathbf{u}^t, \mathbf{E}^t, t)$ and updates both components separately. The reverse kernel factorizes as

$$p_\theta(\mathbf{u}^{t-1}, \mathbf{E}^{t-1} \mid \mathbf{u}^t, \mathbf{E}^t) = p_\theta(\mathbf{u}^{t-1} \mid \mathbf{u}^t, \mathbf{E}^t)\, p_\theta(\mathbf{E}^{t-1} \mid \hat{\mathbf{u}}^{0,t}, \mathbf{E}^t) \tag{16}$$

For the continuous component, the reverse update follows the DDPM posterior in the shifted-and-scaled parameterization derived in Appendix A.2. For the discrete component, $\hat{\mathbf{u}}^{0,t}$ determines pair-specific reverse transition matrices

$$\hat{\mathbf{Q}}_E^{t,ij} = \alpha_t I + \beta_t \mathbf{1}_b(\hat{\mathbf{p}}_{ij}^{0,t})^\top, \qquad \hat{\bar{\mathbf{Q}}}_E^{t,ij} = \bar{\alpha}_t I + (1 - \bar{\alpha}_t)\mathbf{1}_b(\hat{\mathbf{p}}_{ij}^{0,t})^\top \tag{17}$$

where $\hat{p}_{ij}^{0,t} = 1 - e^{-e^{\hat{u}_i^{0,t} + \hat{u}_j^{0,t}}}$ and $\hat{\mathbf{p}}_{ij}^{0,t} = [\,1 - \hat{p}_{ij}^{0,t},\ \hat{p}_{ij}^{0,t}\,]$. The reverse edge distribution factorizes as in Equation 4, with each factor given by

$$p_\theta(e_{ij}^{t-1} \mid e_{ij}^t, \hat{\mathbf{u}}^{0,t}) = \sum_{r \in \mathcal{B}} q(e_{ij}^{t-1} \mid e_{ij}^t, e_{ij} = r;\, \hat{\mathbf{u}}^{0,t})\, \hat{\mathbf{p}}_{ij}^{E,t}(r) \tag{18}$$

where the posterior is computed from the pair-specific matrices in Equation 17.

Training objective. We retain the cross-entropy objective of Equation 3 for edge prediction and augment it with an MSE loss on the predicted clean latents:

$$\mathcal{L} = \sum_{1 \le i,j \le n} \mathrm{CE}(e_{ij},\, \hat{\mathbf{p}}_{ij}^{E,t}) + \lambda_u \left(\frac{1}{n} \sum_{i=1}^{n} \|\hat{u}_i^{0,t} - u_i^0\|_2^2 \right) \tag{19}$$

where $\lambda_u \in \mathbb{R}_+$ controls the relative weight of the latent regression term. The variational derivation and its scope are detailed in Appendix C.

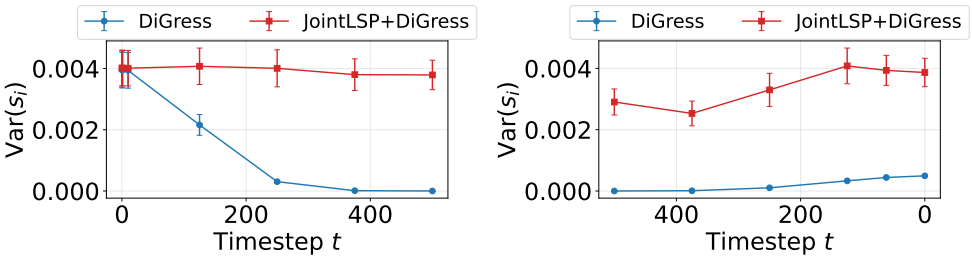
6 Experiments

Datasets. We introduce a new synthetic benchmark by sampling subgraphs from two real-world datasets, Cora [32] and Chameleon [44], which we refer to as CoraChameleon. Unlike commonly used synthetic benchmarks such as SBM or planar graphs, this better reflects the structural properties of real-world graphs; details are given in Appendix E. We generate 3000 graphs for $n \in \{10, 20, 50, 100\}$ and 1400 for $n = 500$, across node sizes $n \in \{10, 20, 50, 100, 500\}$. We also conduct experiments on the commonly used benchmarks Ego and Community datasets [54], with a $60/20/20$ train/test/validation split.

6.1 JointLSP preserves node heterogeneity at high noise

To study whether the proposed joint diffusion framework with LSP preserves node-level heterogeneity during the diffusion process, we define a node propensity score $\mathcal{S}_i^t$ using the predicted clean edge probabilities at diffusion step t. For each node i, we compute $\mathcal{S}_i^t = \frac{1}{n-1} \sum_{j \neq i} \hat{\mathbf{p}}_{ij}^{E,t}$, where $\hat{\mathbf{p}}_{ij}^{E,t}$ denotes the model's predicted clean edge probability between nodes i and j at time t. Intuitively, $\mathcal{S}_i^t$ measures the average connectivity propensity of node i at a given noise level. To quantify how much this varies across nodes, we compute $V^t = \mathrm{Var}_{i=1,\dots,n}(\mathcal{S}_i^t)$.

We evaluate this metric at different diffusion steps $t \in \{500, 50, 0\}$, where $t = 500$ corresponds to the maximally noisy state, over training graphs in a batch. For each graph, V^t is computed across all nodes, and the mean and standard deviation are reported over graphs in the batch. This is shown for CoraChameleon-Node100 in Figure 2. The results show that the node heterogeneity induced by the JointLSP is preserved even at high noise levels. In contrast, under the original DiGress marginal prior without node features, the predicted edge probabilities become increasingly uniform across node pairs as $t \to T$, causing the node propensity scores to collapse toward a common value. This indicates that, at highly noisy states, the model effectively starts from an ER graph and must rely on auxiliary handcrafted features to recover richer structure during denoising. By contrast, our framework allows the model to retain variation in node propensity throughout the reverse process, providing an intrinsic source of heterogeneity without requiring such external features.



(a) Forward Noising Process.

(b) Reverse Denoising Process.

Figure 2: We compare the variance of node propensity scores for both the original DiGress and JointLSP+DiGress (LSP in the figure). We observe at terminal noisy state, our prior helps maintain a healthy level of node heterogeneity as opposed to DiGress which loses node distinguishability during the diffusion process.

6.2 Experiments on graph generation

Evaluation metrics. We evaluate generated graphs using degree distribution, clustering coefficient, triangle count, and degree assortativity. We additionally report the power-law exponent of the degree sequence (PLE) and characteristic path length (CPL), following [7] and [9]. For each statistic, we report the MMD ratio $r = \frac{\mathrm{MMD}(\mathrm{generated}, \mathrm{test})^2}{\mathrm{MMD}(\mathrm{train}, \mathrm{test})^2}$ following [51]. Lower is better, with $r = 1$ as the natural baseline.

Ablation with auxiliary features. We first establish a DiGress [51] baseline using the full set of recommended auxiliary features. We find that the performance of existing graph diffusion models relies heavily on these additional node and graph-level features rather than on the diffusion process alone. These features fall into two categories: *structural features* such as k-cycle counts, which we address in Lemma 4.3, and *spectral features* such as eigenvectors and eigenvalues computed at every denoising step. We note however that the latter are expensive to compute, and prohibitive at scale. The full list is given in Appendix Table 11. In Table 1 we present results on CoraChameleon-Node100 for three methods: DiGress+k structural features, by randomly choosing feature subsets of size k, averaged over 3 runs, JointLSP+DiGress, which augments DiGress with our joint diffusion framework and LSP but uses *no* auxiliary features ($k = 0$), and finally Digress with the full set of structural and spectral features ($k = 17$), which is more computationally intensive. The training details and hyperparameters are given in Appendix E.1.

We can observe that structural features are consistently insufficient: even with $k = 10$ features, performance on higher-order statistics such as triangle count and assortativity remains poor. We also conduct another ablation study in Table 6, where we supply degree as both node and graph-level feature and observe similar results. This is consistent with Lemma 4.3: structural features computed on a near-ER graph are themselves nearly constant across nodes, and therefore provide little additional information to the denoiser at high noise levels. JointLSP+DiGress outperforms DiGress with up to $k = 10$ features on clustering coefficient, triangle count, and assortativity, demonstrating that improving the prior directly addresses these limitations. Nevertheless, JointLSP+DiGress is still outperformed by the use of *spectral features*, even if much more expensive to compute. Integrating such spectral components into our novel joint diffusion process is a major avenue for future work.

Unconditional graph generation without node features. In Table 2 we present results for CoraChameleon datasets with $n \in \{10, 20, 50, 100\}$ comparing DiGress without features against our proposed method. Our goal is not to claim state-of-the-art performance, but to isolate the no-node-feature setting in which the standard discrete diffusion pipeline is most constrained. Our method shows consistent improvements on triangle count and assortativity [35, 14]. At $n = 100$, DiGress performs better on degree distribution and PLE, which likely reflects occasional over-correction of the degree distribution under the stronger node heterogeneity induced by LSP. Additional results are reported in the appendix: results for JointLSP+SparseDiff on CoraChameleon across $n \in \{10, 20, 50, 100, 500\}$ and on the Ego and Community datasets are given in Appendix D. Finally, Table 6 examines whether supplying node degree directly as an auxiliary feature to DiGress can close the gap, showing that while degree features improve some metrics, they cannot recover the higher-order structural statistics that the LSP improves confirming that the benefit of our method is not reducible to the degree information used in constructing $\mathbf{u}^0$.

7 Conclusion

We identify a fundamental limitation in discrete graph diffusion models for unconditional graph generation without node features: the combination of independent edge updates and an E-R terminal prior leads to a fixed-point pathology in the large graph limit, where any GNN denoiser applied to an E-R graph approximately outputs an E-R graph. We propose the Latent Sociability Prior, a simple latent-variable prior that is co-diffused with discrete edge updates in a joint diffusion framework. By preserving node heterogeneity throughout the diffusion trajectory, LSP enables the model to recover higher-order graph structure without relying on auxiliary structural features. We validate this on graph generation benchmarks across multiple graph sizes.

Table 1: Feature subset ablation for DiGress on CoraChameleon-Node100. JointLSP+DiGress ($k = 0$, no features) is included for direct comparison. Bold: best result excluding the $k = 17$ ceiling.

	Degree ↓	Clustering ↓	Triangles ↓	Assortativity ↓	PLE ↓	CPL ↓
DiGress + Structural features						
$k = 2$	$9.32_{\pm 3.08}$	$5.08_{\pm 2.95}$	$11.75_{\pm 4.50}$	$118.18_{\pm 2.68}$	$7.43_{\pm 2.65}$	$47.23_{\pm 8.47}$
$k = 4$	$7.01_{\pm 0.53}$	$9.73_{\pm 7.67}$	$11.46_{\pm 4.55}$	$107.68_{\pm 12.37}$	$10.07_{\pm 5.75}$	$43.99_{\pm 4.43}$
$k = 8$	$10.88_{\pm 11.42}$	$5.89_{\pm 4.60}$	$7.45_{\pm 5.08}$	$79.81_{\pm 52.72}$	$7.04_{\pm 5.38}$	$36.55_{\pm 19.28}$
$k = 10$	$\mathbf{4.25}_{\pm 1.51}$	$10.72_{\pm 6.46}$	$10.06_{\pm 6.11}$	$70.54_{\pm 40.85}$	$\mathbf{6.99}_{\pm 3.69}$	$\mathbf{34.47}_{\pm 21.49}$
JointLSP+DiGress (no features)						
$k = 0$	$10.50_{\pm 0.16}$	$\mathbf{2.07}_{\pm 0.10}$	$\mathbf{2.33}_{\pm 0.15}$	$\mathbf{59.72}_{\pm 0.75}$	$10.79_{\pm 0.19}$	$44.99_{\pm 0.29}$
DiGress + Structural features + Spectral features						
$k = 17$	$0.93_{\pm 0.00}$	$1.95_{\pm 0.00}$	$1.68_{\pm 0.00}$	$3.55_{\pm 0.00}$	$1.53_{\pm 0.00}$	$1.92_{\pm 0.00}$

Table 2: MMD ratios for unconditional graph generation under no-node features.

	Degree ↓	Clustering ↓	Triangles ↓	Assortativity ↓	PLE ↓	CPL ↓
DiGress (no features)						
$n = 10$	$6.30_{\pm 1.57}$	$2.46_{\pm 0.34}$	$15.28_{\pm 7.27}$	$12.41_{\pm 1.55}$	$35.40_{\pm 6.41}$	$12.97_{\pm 4.19}$
$n = 20$	$22.47_{\pm 1.63}$	$1.44_{\pm 0.00}$	$58.45_{\pm 9.88}$	$49.80_{\pm 3.08}$	$\mathbf{15.84}_{\pm 2.39}$	$11.46_{\pm 2.22}$
$n = 50$	$21.09_{\pm 0.91}$	$1.51_{\pm 0.01}$	$16.90_{\pm 0.23}$	$70.67_{\pm 1.95}$	$18.57_{\pm 1.07}$	$55.49_{\pm 3.01}$
$n = 100$	$\mathbf{4.53}_{\pm 0.11}$	$11.33_{\pm 0.27}$	$14.82_{\pm 0.12}$	$101.95_{\pm 2.13}$	$\mathbf{4.64}_{\pm 0.17}$	$\mathbf{39.68}_{\pm 1.14}$
JointLSP+DiGress (no features)						
$n = 10$	$\mathbf{5.19}_{\pm 1.61}$	$\mathbf{1.79}_{\pm 0.06}$	$\mathbf{13.45}_{\pm 4.25}$	$\mathbf{1.25}_{\pm 0.25}$	$\mathbf{23.58}_{\pm 4.95}$	$\mathbf{12.93}_{\pm 3.90}$
$n = 20$	$\mathbf{11.13}_{\pm 1.29}$	$\mathbf{1.40}_{\pm 0.00}$	$\mathbf{8.38}_{\pm 0.92}$	$\mathbf{14.04}_{\pm 3.72}$	$34.19_{\pm 3.47}$	$\mathbf{7.82}_{\pm 1.24}$
$n = 50$	$\mathbf{8.01}_{\pm 0.42}$	$\mathbf{1.40}_{\pm 0.00}$	$\mathbf{4.87}_{\pm 0.80}$	$\mathbf{33.18}_{\pm 1.16}$	$\mathbf{10.77}_{\pm 0.69}$	$\mathbf{30.65}_{\pm 1.71}$
$n = 100$	$10.50_{\pm 0.16}$	$\mathbf{2.07}_{\pm 0.10}$	$\mathbf{2.33}_{\pm 0.15}$	$\mathbf{59.72}_{\pm 0.75}$	$10.79_{\pm 0.19}$	$44.99_{\pm 0.29}$

Limitations and future work. This work focuses on a key limitation in existing graph diffusion models and several questions remain. Our framework is most suited to graphs with absent or limited node features. For molecular generation, where rich semantic features are available, the fixed-point pathology is less of a concern and the benefit of LSP is expected to be minimal. Combining LSP with auxiliary node features is a natural next step, but is non-trivial since LSP is already informative at high noise levels contrary to E-R prior which is uninformative. Thus, there is no guarantee that the auxiliary features will compound the benefits of the LSP. Promising future directions could include designing auxiliary features that complement the LSP, rather than duplicating the degree heterogeneity already captured by the sociability variables. In particular, spectral features appear important for improving community-level structure, but explicitly computing eigenvectors or eigenvalues at every denoising step can be expensive. One possibility is to extend the LSP itself with a community-aware component, by replacing the sociability prior with a calibrated community-conditioned variant. This would make the terminal prior informative about degree heterogeneity and community structure.

Acknowledgments and Disclosure of Funding

This work was funded by the European Union ERC-2024- STG-101163069 MALAGA.

References

[1] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization, 2021. URL https://arxiv.org/abs/2010.01179.

[2] David J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981. ISSN 0047-259X. doi: https://doi.org/10.1016/0047-259X(81)90099-3. URL https://www.sciencedirect.com/science/article/pii/0047259X81900993.

[3] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

[4] Béla Bollobás, Svante Janson, and Oliver Riordan. The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms*, 31(1):3–122, June 2007. ISSN 1098-2418. doi: 10.1002/rsa.20168. URL http://dx.doi.org/10.1002/rsa.20168.

[5] François Caron and Emily B. Fox. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(5):1295–1366, 09 2017. ISSN 1369-7412. doi: 10.1111/rssb.12233. URL https://doi.org/10.1111/rssb.12233.

[6] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E. Tsourakakis. On the power of edge independent graph models. *CoRR*, abs/2111.00048, 2021. URL https://arxiv.org/abs/2111.00048.

[7] Sudhanshu Chanpuriya, Cameron N Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. On the role of edge dependency in graph generative models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6325–6345. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chanpuriya24a.html.

[8] Harvey Chen, Nicolas Zilberstein, and Santiago Segarra. Prior-informed flow matching for graph reconstruction, 2026. URL https://arxiv.org/abs/2601.22107.

[9] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling, 2023. URL https://arxiv.org/abs/2305.04111.

[10] David Dehaene and Rémy Brossard. Re-parameterizing vaes for stability. *CoRR*, abs/2106.13739, 2021. URL https://arxiv.org/abs/2106.13739.

[11] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

[12] Iakovos Evdaimon, Giannis Nikolentzos, Christos Xypolopoulos, Ahmed Kammoun, Michail Chatzianastasis, Hadi Abdine, and Michalis Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models, 2024. URL https://arxiv.org/abs/2403.01535.

[13] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[14] Jacob G. Foster, David V. Foster, Peter Grassberger, and Maya Paczuski. Edge direction and the structure of networks. *Proceedings of the National Academy of Sciences*, 107(24):10815–10820, 2010. doi: 10.1073/pnas.0912671107. URL https://www.pnas.org/doi/abs/10.1073/pnas.0912671107.

[15] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

[16] Lukas Gonon, Thilo Meyer-Brandis, and Niklas Weber. Universality and approximation rates of graph neural networks with random features, 2026. URL https://arxiv.org/abs/2607.26699.

[17] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces, 2023. URL https://arxiv.org/abs/2210.01549.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

[19] Remco van der Hofstad. *Random Graphs and Complex Networks*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2016.

[20] Douglas N. Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute for Advanced Study, Princeton, NJ, 1979. URL https://www.stat.berkeley.edu/~aldous/Research/hoover.pdf. Preprint.

[21] Keyue Jiang, Jiahao Cui, Xiaowen Dong, and Laura Toni. Bures-wasserstein flow matching for graph generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5Bl5qf3fON.

[22] 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 Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10362–10383. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

[23] Mahdi Karami. Higen: Hierarchical graph generative networks, 2025. URL https://arxiv.org/abs/2305.19337.

[24] Nicolas Keriven and Samuel Vaiter. Sparse and smooth: improved guarantees for spectral clustering in the dynamic stochastic block model. *Electronic Journal of Statistics*, 16:1330 – 1366, 2022.

[25] Nicolas Keriven and Samuel Vaiter. What functions can graph neural networks compute on random graphs? the role of positional encoding. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

[27] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016. URL https://arxiv.org/abs/1611.07308.

[28] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[29] Jing Lei and Alessandro Rinaldo. Consistency of spectral clustering in stochastic block models. *Annals of Statistics*, 43:215–237, 2015. ISSN 00905364. doi: 10.1214/14-AOS1274.

[30] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters, 2007. URL https://arxiv.org/abs/physics/0603229.

[31] Mufei Li, Eleonora Kreacic, Vamsi K. Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs?, 2024. URL https://openreview.net/forum?id=ledQ1BCrwc.

[32] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2): 127–163, 2000. doi: 10.1023/A:1009953814988. URL https://doi.org/10.1023/A:1009953814988.

[33] Colin McDiarmid. On the method of bounded differences. In *Surveys in combinatorics, 1989 (Norwich, 1989)*, volume 141 of *London Math. Soc. Lecture Note Ser.*, pages 148–188. Cambridge Univ. Press, Cambridge, 1989.

[34] Andrew Melnik, Michal Ljubljanac, Cong Lu, Qi Yan, Weiming Ren, and Helge Ritter. Video diffusion models: A survey, 2024. URL https://arxiv.org/abs/2405.03150.

[35] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2), February 2003. ISSN 1095-3787. doi: 10.1103/physreve.67.026126. URL http://dx.doi.org/10.1103/PhysRevE.67.026126.

[36] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/nichol21a.html.

[37] Matteo Ninniri, Marco Podda, and Davide Bacciu. Graph diffusion that can insert and delete, 2025. URL https://arxiv.org/abs/2506.15725.

[38] 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, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/niu20a.html.

[39] Ilkka Norros and Hannu Reittu. On a conditionally poissonian graph process. *Advances in Applied Probability*, 38(1):59–75, 2006. doi: 10.1239/aap/1143936140.

[40] Nagham Osman, Keyue Jiang, Davide Buffelli, Xiaowen Dong, and Laura Toni. Lgdc: Latent graph diffusion via spectrum-preserving coarsening, 2025. URL https://arxiv.org/abs/2512.01190.

[41] Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. In *International Conference on Machine Learning (ICML)*, 2025.

[42] Yiming QIN, Clement Vignac, and Pascal Frossard. Sparsediff: Sparse discrete diffusion for scalable graph generation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=kuJ3lpxnVC.

[43] Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233.

[44] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *CoRR*, abs/1909.13021, 2019. URL http://arxiv.org/abs/1909.13021.

[45] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks, 2021. URL https://arxiv.org/abs/2002.03155.

[46] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008. doi: 10.1609/aimag.v29i3.2157. URL https://ojs.aaai.org/index.php/aimagazine/article/view/2157.

[47] Antoine Siraudin and Christopher Morris. Principled latent diffusion for graphs via laplacian autoencoders. *arXiv preprint arXiv:2601.13780*, 2026.

[48] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[49] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020. URL https://arxiv.org/abs/1907.05600.

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

[51] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation, 2023. URL https://arxiv.org/abs/2209.14734.

[52] Asiri Wijesinghe, Sevvandi Kandanaarachchi, Daniel M. Steinberg, and Cheng Soon Ong. Flowette: Flow matching with graphette priors for graph generation, 2026. URL https://arxiv.org/abs/2602.23566.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[54] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *CoRR*, abs/1802.08773, 2018. URL http://arxiv.org/abs/1802.08773.

[55] Tianyi Zhang, Zheng Wang, Jing Huang, Mohiuddin Muhammad Tasnim, and Wei Shi. A survey of diffusion based image generation models: Issues and their solutions, 2023. URL https://arxiv.org/abs/2308.13142.

A Appendix

Algorithm 1 Calibrating the LSP terminal prior

Require: Training dataset $\mathcal{D}_{\text{data}}$, smoothing constant $\delta > 0$, number of quadrature nodes K

for each graph $G \in \mathcal{D}_{\text{data}}$ and each node $i \in V(G)$ **do**
  Compute $\tilde{u}_i = \log(d_i + \delta) - \frac{1}{|V|}\sum_{k \in V} \log(d_k + \delta)$
end for
$\sigma_{\text{LSP}} \leftarrow \text{std}\big(\{\tilde{u}_i : i \in V(G),\, G \in \mathcal{D}_{\text{data}}\}\big)$ ▷ std over all nodes across all training graphs
$\rho_{\text{data}} \leftarrow \frac{1}{|\mathcal{D}_{\text{data}}|}\sum_{G \in \mathcal{D}_{\text{data}}} \frac{2|E_G|}{n_G(n_G-1)}$ ▷ mean empirical edge density over training graphs
Initialize $\mu^0_{\text{LSP}} \leftarrow \frac{1}{2}(\log \rho_{\text{data}} - \sigma^2_{\text{LSP}})$ ▷ sparse-regime approximation, see Appendix A.1
Solve $\rho(\mu_{\text{LSP}}) = \rho_{\text{data}}$ for μ_{LSP} using K-node Gauss-Hermite quadrature to evaluate $\rho(\cdot)$, starting from μ^0_{LSP}
return $(\mu_{\text{LSP}},\, \sigma_{\text{LSP}})$

Algorithm 2 Training JointLSP

Require: A graph $G = (\mathbf{1}, \mathbf{E}^0)$, calibrated prior parameters $(\mu_{\text{LSP}}, \sigma_{\text{LSP}})$ from Algorithm 1

Sample $t \sim \mathcal{U}(\{1, \ldots, T\})$
Compute $\mathbf{u}^0$ from G using Equation 12
Sample $\mathbf{u}^t \sim q(\mathbf{u}^t \mid \mathbf{u}^0)$ using Equation 13
Compute $\mathbf{p}^0_{ij}$ for all pairs (i, j) using $p^0_{ij} = 1 - e^{-e^{u^0_i + u^0_j}}$
Sample $\mathbf{E}^t \sim q(\mathbf{E}^t \mid \mathbf{E}^0, \mathbf{u}^0)$ using Equation 15
$(\hat{\mathbf{u}}^{0,t}, \hat{\mathbf{p}}^{E,t}) \leftarrow \phi_\theta(\mathbf{u}^t, \mathbf{E}^t, t)$
Optimizer step on $\mathcal{L}(\hat{\mathbf{u}}^{0,t}, \hat{\mathbf{p}}^{E,t}; \mathbf{u}^0, \mathbf{E}^0)$ using Equation 19

Algorithm 3 Sampling from JointLSP

Require: Trained network ϕ_θ, calibrated prior parameters $(\mu_{\text{LSP}}, \sigma_{\text{LSP}})$

Sample n from the empirical node-count distribution of $\mathcal{D}_{\text{data}}$
Sample $\mathbf{u}^T \sim \mathcal{N}(\mu_{\text{LSP}}, \sigma^2_{\text{LSP}} I)$
Compute $\mathbf{p}^T_{ij} = [1 - p^T_{ij},\, p^T_{ij}]$ where $p^T_{ij} = 1 - e^{-e^{u^T_i + u^T_j}}$ for all pairs (i, j)
Sample $\mathbf{E}^T$: each edge (i, j) independently from $\text{Bernoulli}(p^T_{ij})$
for $t = T$ **down to** 1 **do**
  $(\hat{\mathbf{u}}^{0,t}, \hat{\mathbf{p}}^{E,t}) \leftarrow \phi_\theta(\mathbf{u}^t, \mathbf{E}^t, t)$
  Sample $\mathbf{u}^{t-1} \sim p_\theta(\mathbf{u}^{t-1} \mid \mathbf{u}^t, \mathbf{E}^t)$ using Equation 37
  Sample $\mathbf{E}^{t-1} \sim p_\theta(\mathbf{E}^{t-1} \mid \hat{\mathbf{u}}^{0,t}, \mathbf{E}^t)$ using Equation 18
end for
return $\mathbf{E}^0$

A.1 Constructing the Latent Sociability Prior

We derive the calibration procedure for the terminal prior parameters $(\mu_{\text{LSP}}, \sigma_{\text{LSP}})$, which are estimated from the training data and fixed before training begins. The goal is to choose these parameters so that graphs sampled from the LSP terminal prior match the empirical edge density and degree heterogeneity of $\mathcal{D}_{\text{data}}$.

Reparameterization. From Section 5.2, the latent sociability weights satisfy $w_i = \exp(u_i)$, so $u_i = \log w_i \in \mathbb{R}$. We place a Gaussian prior on the log-weights, $u_i \overset{i.i.d.}{\sim} \mathcal{N}(\mu_{\text{LSP}}, \sigma^2_{\text{LSP}})$, which induces a log-normal distribution on w_i. With $c = 1$, the edge probability becomes

$$p_{ij} = 1 - e^{-e^{u_i + u_j}} \tag{20}$$

Estimating σ_{LSP}. Since the true weights w_i are unobserved, we construct a proxy from the degree sequence. For each node i in each training graph G, define the centered log-degree

$$\tilde{u}_i = \log(d_i + \delta) - \frac{1}{|V|} \sum_{k \in V} \log(d_k + \delta) \tag{21}$$

where $\delta > 0$ is a small smoothing constant. Centering removes the graph-level mean so that $\tilde{u}_i$ captures relative degree heterogeneity within each graph. We then set

$$\sigma_{\mathrm{LSP}} = \mathrm{std}\big(\{\tilde{u}_i : i \in V(G),\, G \in \mathcal{D}_{\mathrm{data}}\}\big) \tag{22}$$

where the standard deviation is taken over all nodes across all training graphs.

Estimating μ_{LSP}. We calibrate μ_{LSP} by matching the expected edge probability under the LSP to the empirical edge density of the training data. The empirical edge density, averaged over training graphs, is

$$\rho_{\mathrm{data}} = \frac{1}{|\mathcal{D}_{\mathrm{data}}|} \sum_{G \in \mathcal{D}_{\mathrm{data}}} \frac{2|E_G|}{n_G(n_G - 1)} \tag{23}$$

Under the LSP with parameters $(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}})$, the expected edge probability between any two nodes is

$$\rho(\mu_{\mathrm{LSP}}) = \mathbb{E}[p_{ij}] = \mathbb{E}\big[1 - e^{-e^{u_i + u_j}}\big] \tag{24}$$

Note that c enters Equation 24 only through the product $c\, e^{u_i + u_j}$, which in the sparse regime reduces to $c\, e^{2\mu_{\mathrm{LSP}} + \sigma^2_{\mathrm{LSP}}}$ (see Equation 27). Any value of $c > 0$ can therefore be absorbed into μ_{LSP} via the substitution $\mu_{\mathrm{LSP}} \leftarrow \mu_{\mathrm{LSP}} + \frac{1}{2} \log c$, leaving the model unchanged. We fix $c = 1$ throughout without loss of generality, since μ_{LSP} is calibrated directly to match ρ_{data}.

Since $u_i, u_j \overset{i.i.d.}{\sim} \mathcal{N}(\mu_{\mathrm{LSP}}, \sigma^2_{\mathrm{LSP}})$, the sum $S = u_i + u_j \sim \mathcal{N}(2\mu_{\mathrm{LSP}}, 2\sigma^2_{\mathrm{LSP}})$. Writing $S = 2\mu_{\mathrm{LSP}} + \sqrt{2}\, \sigma_{\mathrm{LSP}}\, Z$ with $Z \sim \mathcal{N}(0, 1)$, Equation 24 becomes

$$\rho(\mu_{\mathrm{LSP}}) = 1 - \mathbb{E}_{Z \sim \mathcal{N}(0,1)} \Big[e^{-e^{2\mu_{\mathrm{LSP}} + \sqrt{2}\, \sigma_{\mathrm{LSP}}\, Z}} \Big] \tag{25}$$

We solve $\rho(\mu_{\mathrm{LSP}}) = \rho_{\mathrm{data}}$ for μ_{LSP} using root-finding with the expectation in Equation 25 evaluated via K-node Gauss-Hermite quadrature, which replaces the integral by a deterministic weighted sum.

Initialization of the root-finding. In the sparse regime where $e^{u_i + u_j}$ is small, $p_{ij} = 1 - e^{-e^{u_i + u_j}} \approx e^{u_i + u_j}$, so

$$\rho_{\mathrm{data}} \approx \mathbb{E}\big[e^{u_i + u_j}\big] = e^{2\mu_{\mathrm{LSP}} + \sigma^2_{\mathrm{LSP}}} \tag{26}$$

Solving for μ_{LSP} gives the initialization

$$\mu^0_{\mathrm{LSP}} = \tfrac{1}{2}\big(\log \rho_{\mathrm{data}} - \sigma^2_{\mathrm{LSP}}\big) \tag{27}$$

where we have used $c = 1$. Since real-world graphs are typically sparse, this approximation is mild and provides a close starting point for the root-finder in practice.

A.2 Shifted and scaled DDPM for latent sociability variables

The following derivation is stated for a single coordinate u_i; the result extends to the full vector $\mathbf{u}$ coordinatewise.

Forward process. The standard DDPM forward process [18] takes the form

$$x^t = \sqrt{\bar{\alpha}_t}\, x^0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{28}$$

In our setting the latent sociability variables should diffuse toward a Gaussian prior with mean μ_{LSP} and variance σ^2_{LSP}, rather than toward $\mathcal{N}(0, I)$.

To obtain this, we apply the DDPM construction to the centered and scaled variable

$(u - \mu_{\mathrm{LSP}})/\sigma_{\mathrm{LSP}}$:

$$\frac{u^t - \mu_{\mathrm{LSP}}}{\sigma_{\mathrm{LSP}}} = \sqrt{\bar{\alpha}_t}\,\frac{u^0 - \mu_{\mathrm{LSP}}}{\sigma_{\mathrm{LSP}}} + \sqrt{1-\bar{\alpha}_t}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{29}$$

Multiplying through by σ_{LSP} and adding μ_{LSP} gives

$$u^t = \mu_{\mathrm{LSP}} + \sqrt{\bar{\alpha}_t}(u^0 - \mu_{\mathrm{LSP}}) + \sigma_{\mathrm{LSP}}\sqrt{1-\bar{\alpha}_t}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{30}$$

so that $u^t = u^0$ at $t = 0$ and $u^t \to \mathcal{N}(\mu_{\mathrm{LSP}}, \sigma^2_{\mathrm{LSP}})$ as $t \to T$.

Reverse process. Since Equation 29 is the standard DDPM forward process applied to $\tilde{u}^t := (u^t - \mu_{\mathrm{LSP}})/\sigma_{\mathrm{LSP}}$, the reverse posterior in $\tilde{u}$-space is the standard DDPM Gaussian posterior [18]:

$$q(\tilde{u}^{t-1} \mid \tilde{u}^t, \tilde{u}^0) = \mathcal{N}(\tilde{u}^{t-1} \mid \tilde{\mu}_{t-1},\, \tilde{\beta}_t) \tag{31}$$

with mean

$$\tilde{\mu}_{t-1} = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\,\tilde{u}^0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,\tilde{u}^t \tag{32}$$

and variance

$$\tilde{\beta}_t = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}. \tag{33}$$

Substituting $\tilde{u}^t = (u^t - \mu_{\mathrm{LSP}})/\sigma_{\mathrm{LSP}}$ and $\tilde{u}^0 = (u^0 - \mu_{\mathrm{LSP}})/\sigma_{\mathrm{LSP}}$ into Equation 32 and multiplying through by σ_{LSP} gives the posterior mean in u-space:

$$\mu_{t-1} = \mu_{\mathrm{LSP}} + \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}(u^0 - \mu_{\mathrm{LSP}}) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}(u^t - \mu_{\mathrm{LSP}}) \tag{34}$$

The posterior variance scales by σ^2_{LSP}, giving

$$\tilde{\sigma}_t^2 = \sigma^2_{\mathrm{LSP}}\,\tilde{\beta}_t = \sigma^2_{\mathrm{LSP}}\,\frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \tag{35}$$

so that

$$q(u^{t-1} \mid u^t, u^0) = \mathcal{N}(u^{t-1} \mid \mu_{t-1},\, \tilde{\sigma}_t^2). \tag{36}$$

The reverse sampling step is therefore

$$u^{t-1} = \mu_{t-1} + \sigma_{\mathrm{LSP}}\sqrt{\frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}\,z, \qquad z \sim \mathcal{N}(0, 1), \tag{37}$$

which is the update used in Section 5.2, with u^0 replaced by the network prediction $\hat{u}^{0,t}$ at each timestep.

B Proofs

B.1 Proof of Lemma 4.1

Proof of Lemma 4.1. Since the edges are discrete in $\{0, 1\}$, they are independent Bernoulli random variables. We note $\mathbb{P}(e_{ij} = 1) = p_{ij}$ its parameters. Denote by $A = [e_{ij}]_{ij}$ the adjacency matrix of a drawn graph, we have $P = [p_{ij}]_{ij} = \mathbb{E}(A)$. By exchangeability, $\sigma A \sigma^\top$ has the same distribution as A for any permutation matrix σ, and therefore

$$\mathbb{E}[\sigma A \sigma^\top] = \mathbb{E}[A] \tag{38}$$

which is

$$\pi P \pi^\top = P \tag{39}$$

The only matrices invariant under conjugation by all permutation matrices have constant off-diagonal entries. Since the diagonal is zero for simple graphs, it follows that $P_{ij} = p$ for all $i \neq j$. Therefore the model is Erdős–Rényi. □

B.2 Proof of Thm. 4.2

GNNs. Denote by $\mathbf{X}^{(0)} = \mathbf{X}$ the input node features. The GNNs we are examining are all under the form

$$\mathbf{X}^{(k)} = \sigma\left(\mathbf{X}^{(k-1)}\theta_0^{(k)} + S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} + 1_n \mathbf{b}^{(k)}\right) \in \mathbb{R}^{n \times d_k}$$

where $S_k(\mathbf{X}^{(k-1)}) \in \mathbb{R}^{n \times n}$ are message-passing matrices that depends on the graph structure and may depend on the layer k and $\mathbf{X}^{(k-1)}$ for attention-based models, $\theta_i^{(k)} \in \mathbb{R}^{d_{k-1} \times d_k}$ are trainable weights, $\mathbf{b}^{(k)} \in \mathbb{R}^{d_k}$ are trainable biases, and σ is an activation function that we assume to be 1-Lipschitz. All models can then be obtained by varying the choice of $S_k(\mathbf{X})$, which we denote by S when it is fixed:

- for GCN,
$$S = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$
where $\tilde{A} = Id + A$ and $\tilde{D}$ is the corresponding degree matrix.
- GIN uses sum-aggregation, which would correspond to $S = A$. However for random graphs this choice may diverge when n increases if not properly normalized. We choose the classical normalization for random graphs [25] by the expected density of the graph: for the ER model considered in the result,
$$S = \frac{1}{pn}A$$
assuming that p is known for simplicity. Note that, for a fixed graph size n, this normalization can be absorbed by the weight θ_1, such that this is indeed equivalent to the vanilla GIN;
- for GAT, we consider attention coefficients $c_k(\mathbf{x}, \mathbf{x}')$ at layer k (typically exponentials of dot products), and normalized attention:
$$[S_k(\mathbf{X})]_{ij} = \frac{c_k(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{\ell \sim i} c_k(\mathbf{x}_i, \mathbf{x}_\ell)}$$
We will assume that $c_k(\cdot) \geq c_{\min} > 0$, which is for instance the case for exponential attention coefficients on bounded features. Graph Transformers are essentially the same model, when stripped of various normalization layers (which we do not consider here for simplicity).

After K layers, the node representations are used pairwise to obtain the edge prediction

$$\hat{p}_{ij} = [\varphi(\mathbf{x}_i^{(K)}, \mathbf{x}_j^{(K)})]_{i,j=1}^n \in [0,1]$$

where φ is a link function that we assume to be L_φ-Lipschitz in each component.

Proof of Thm. 4.2. Recall that $\texttt{std}(\hat{p}) = \sqrt{\frac{1}{n^2}\sum_{ij}(\hat{p}_{ij} - \bar{p})^2}$ where $\bar{p} = \sum_{ij}\hat{p}_{ij}$. First observe that

$$\sum_{ij}(\hat{p}_{ij} - \bar{p})^2 = \min_p \sum_{ij}(p_{ij} - p)^2$$

and similarly, for $\mathbf{X} \in \mathbb{R}^{n \times d}$,

$$\texttt{std}(\mathbf{X}) = \sqrt{\frac{1}{n}\sum_i \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2} = \min_{\mathbf{x}} \sqrt{\frac{1}{n}\sum_i \|\mathbf{x}_i - \mathbf{x}\|^2}$$

where $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}_i$. We therefore denote $\bar{\mathbf{x}}^{(k)} = \frac{1}{n}\sum_i \mathbf{x}_i^{(k)}$.

Hence, for any $\mathbf{x}$, by Lipschitz property of φ and a triangular inequality

$$\begin{aligned}
\texttt{std}(\hat{p}) &\leq \sqrt{\frac{1}{n^2}\sum_{ij}(\varphi(\mathbf{x}_i^{(K)},\mathbf{x}_j^{(K)}) - \varphi(\mathbf{x},\mathbf{x}))^2} \\
&\leq \sqrt{\frac{1}{n^2}\sum_{ij}(\varphi(\mathbf{x}_i^{(K)},\mathbf{x}_j^{(K)}) - \varphi(\mathbf{x}_i^{(K)},\mathbf{x}))^2} + \sqrt{\frac{1}{n^2}\sum_{ij}(\varphi(\mathbf{x}_i^{(K)},\mathbf{x}) - \varphi(\mathbf{x},\mathbf{x}))^2} \\
&\leq 2L_\varphi \sqrt{\frac{1}{n}\sum_i \left\|\mathbf{x}_i^{(K)} - \mathbf{x}\right\|^2}
\end{aligned}$$

and by minimizing over $\mathbf{x}$, we get $\texttt{std}(\hat{p}) \leq 2L_\varphi \texttt{std}(\mathbf{X}^{(K)})$.

Now, denote by $E_k = \texttt{std}(\mathbf{X}^{(k)})$. Define $\|\cdot\|_{F,n} = \|\cdot\|_F / \sqrt{n}$ for short, where $\|\cdot\|_F$ is the Frobenius norm, and observe that $\texttt{std}(\mathbf{X}) = \left\|\mathbf{X} - 1_n \bar{\mathbf{x}}^\top\right\|_{F,n}$. Moreover, observe that $\|S\mathbf{X}\theta\|_{F,n} \leq \|S\|_2 \|\mathbf{X}\|_{F,n} \|\theta\|_2$.

We have the recursion equation: since minimizing over $\mathbf{x}$ is more general than over $\sigma(\mathbf{x})$, and using the fact that σ is 1-Lipschitz and we can decompose $\min_x \|y + y' - x\| = \min_{x,x'} \|y + y' - (x + x')\| \leq \min_x \|y - x\| + \min_x \|y' - x\|$,

$$\begin{aligned}
\texttt{std}(\mathbf{X}^{(k)}) &= \min_{\mathbf{x}} \left\|\mathbf{X}^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n} \leq \min_{\mathbf{x}} \left\|\mathbf{X}^{(k)} - 1_n \sigma(\mathbf{x})^\top\right\|_{F,n} \\
&= \min_{\mathbf{x}} \left\|\sigma\left(\mathbf{X}^{(k-1)}\theta_0^{(k)} + S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} + 1_n \mathbf{b}^{(k)}\right) - \sigma(1_n \mathbf{x}^\top)\right\|_{F,n} \\
&\leq \min_{\mathbf{x}} \left\|\mathbf{X}^{(k-1)}\theta_0^{(k)} + S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} + 1_n \mathbf{b}^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n} \\
&\leq \min_{\mathbf{x}} \left\|\mathbf{X}^{(k-1)}\theta_0^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n} + \min_{\mathbf{x}} \left\|S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n}
\end{aligned}$$

For the first term, by picking $\mathbf{x} = {\theta_0^{(k)}}^\top \bar{\mathbf{x}}^{(k-1)}$, we get

$$\min_{\mathbf{x}} \left\|\mathbf{X}^{(k-1)}\theta_0^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n} \leq \left\|(\mathbf{X}^{(k-1)} - 1_n (\bar{\mathbf{x}}^{(k-1)})^\top)\theta_0^{(k)}\right\|_{F,n} \leq E_{k-1} \left\|\theta_0^{(k)}\right\|_2$$

We need to divide the second term, since the term $1_n \mathbf{x}^\top$ cannot necessarily be written as $S_k(\mathbf{X}^{(k-1)}) 1_n {\mathbf{x}'}^\top$ for some $\mathbf{x}'$. So we introduce a term $S_k(\mathbf{X}^{(k-1)}) 1_n {\mathbf{x}'}^\top$ for some $\mathbf{x}'$:

$$\begin{aligned}
\min_{\mathbf{x}} \left\|S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} - 1_n \mathbf{x}^\top\right\|_{F,n} \leq{}& \left\|S_k(\mathbf{X}^{(k-1)})\left(\mathbf{X}^{(k-1)}\theta_1^{(k)} - 1_n {\mathbf{x}'}^\top\right)\right\|_{F,n} \\
&+ \min_{\mathbf{x}} \left\|S_k(\mathbf{X}^{(k-1)}) 1_n {\mathbf{x}'}^\top - 1_n \mathbf{x}^\top\right\|_{F,n}
\end{aligned}$$

Then, as previously, by picking $\mathbf{x}' = \tilde{\mathbf{x}}_k = {\theta_1^{(k)}}^\top \bar{\mathbf{x}}^{(k-1)}$ the first term in this equation can be bounded by

$$\left\|S_k(\mathbf{X}^{(k-1)})\left(\mathbf{X}^{(k-1)}\theta_1^{(k)} - 1_n {\mathbf{x}'}^\top\right)\right\|_{F,n} \leq \left\|S_k(\mathbf{X}^{(k-1)})\right\|_2 E_{k-1} \left\|\theta_1^{(k)}\right\|_2$$

Note that we must then bound $\left\|S_k(\mathbf{X}^{(k-1)})\right\|_2$, which is random and depend on the random graph and chosen GNN architecture.

We denote the remaining term by

$$r_n^{(k)} = \min_{\mathbf{x}} \left\|S_k(\mathbf{X}^{(k-1)}) 1_n \tilde{\mathbf{x}}_k^\top - 1_n \mathbf{x}^\top\right\|_{F,n}$$

which we will again bound for each example.

At the end of the day, we obtain

$$E_k \leq \underbrace{\left(\left\|\theta_0^{(k)}\right\|_2 + \left\|S_k(\mathbf{X}^{(k-1)})\right\|_2 \left\|\theta_1^{(k)}\right\|_2\right)}_{u_k} E_{k-1} + r_n^{(k)}$$

$$= \underbrace{\sum_{\ell=1}^{k} \left[\prod_{p=\ell+1}^{k} u_p\right] r_n^{(\ell)}}_{r_n(\delta)} + \left[\prod_{\ell=1}^{k} u_p\right] E_0$$

For all example, we are going to show that $u_k = O(1)$ with probability $1-\delta$ if n is large enough, and bound $r_n(\delta)$ wrt n and δ. Our general strategy will be to write

$$r_n^{(k)} = \min_{\mathbf{x}} \left\|S_k(\mathbf{X}^{(k-1)})1_n\tilde{\mathbf{x}}_k^\top - 1_n\mathbf{x}^\top\right\|_{F,n} \leq \|\tilde{\mathbf{x}}_k\| \min_c \max_i \left|\sum_{j\sim i} s_{ij} - c\right| \tag{40}$$

for some constant $c \in \mathbb{R}$ (often 1 in our examples), and s_{ij} are the elements of $S_k(\mathbf{X}^{(k-1)})$ for short. We remark that $\|\tilde{\mathbf{x}}_k\| \leq \left\|\theta_1^{(k)}\right\|_2 \left\|\mathbf{X}^{(k-1)}\right\|_{F,n}$ and we can use Lemma B.2 to bound this term, which again involves u_k.

Our main tool will be the following Lemma.

Lemma B.1 (Lemma 3 in [24])**.** *Consider an ER model with parameter p. Then, with probability at least $1-\delta$, the degrees d_i satisfy*

$$\max_i \left|\frac{d_i}{pn} - 1\right| \lesssim \sqrt{\frac{\log(n/\delta)}{pn}} \tag{41}$$

In particular, if

$$n \gtrsim \frac{\log(n/\delta)}{p} \tag{42}$$

then with probability $1-\delta$, for all i

$$\frac{pn}{2} \leq d_i \leq \frac{3pn}{2}$$

We now bound $\left\|S_k(\mathbf{X}^{(k-1)})\right\|_2$ and r_n through Equation 40 for our examples.

GAT. GAT is actually the simplest model to bound. Indeed, due to the normalization of the coefficients $\sum_{j\sim i} s_{ij} = 1$ at all layers, we directly obtain $r_n^{(k)} = 0$.

For the bound on $\|S_k(\mathbf{X})\|_2$, we remark that under the bound Equation 42 on n, for all i, j, k

$$\frac{c_k(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{\ell\sim i} c_k(\mathbf{x}_i, \mathbf{x}_\ell)} \leq \frac{c_{\max}}{c_{\min}} \frac{2}{pn}$$

and thus for any layer and $\mathbf{X}$

$$\|S_k(\mathbf{X})\|_2 \leq \|S_k(\mathbf{X})\|_F = \sqrt{\sum_{ij} s_{ij}^2} \lesssim \frac{c_{\max}}{c_{\min}} \sqrt{\sum_i \frac{d_i}{(pn)^2}} \lesssim \frac{c_{\max}}{pc_{\min}}$$

hence the result $C = O(1/p^K)$.

GIN. For GIN, we have $S = A/(pn)$. Hence Equation 40 directly yields $r_n^{(k)} \leq \|\tilde{\mathbf{x}}_k\| \max_i \left|\frac{d_i}{pn} - 1\right|$, and the last term is smaller than ϵ when $pn \geq \epsilon^{-2}\log(n/\delta)$ by Lemma B.1.

For the bound on $\|S\|_2$, we invoke [29, Thm 5.2], such that for any $r > 0$, there is a constant C_r such that, if $n \geq C_r(1/\delta)^r$, then

$$\left\|\frac{A}{pn} - \frac{1_{n\times n}}{n}\right\|_2 \lesssim \frac{1}{\sqrt{pn}}$$

and $\left\|\frac{1_{n\times n}}{n}\right\|_2 = 1$ hence the result.

GCN. For GCN, if $pn \geq \epsilon^{-2}\log(n/\delta)$ by Lemma B.1 $|d_i - pn| = O(pn\epsilon)$ and thus since $\frac{1}{\sqrt{a}} = \frac{1}{\sqrt{b}} + O(\frac{|a-b|}{b^{3/2}})$,

$$\frac{1}{\sqrt{d_i}} = \frac{1}{\sqrt{pn}} + O\left(\frac{\epsilon}{\sqrt{pn}}\right)$$

From this we get

$$\begin{aligned}\max_i \left|\sum_{j\sim i} s_{ij} - 1\right| &= \max_i \left|\frac{1}{\sqrt{d_i}}\sum_{j\sim i}\frac{1}{\sqrt{d_j}} - 1\right| \\ &\leq \max_i \left|\frac{d_i}{pn} - 1\right| + O\left(\frac{\epsilon}{\sqrt{pn}}\right)\end{aligned}$$

and we go back to the GIN case.

For the bound on $\|S\|_2$ we have directly that $\|S\|_2 \leq 1$, since its a symmetric normalized Laplacian (up to identity term).

□

Lemma B.2.

$$\left\|\mathbf{X}^{(k)}\right\|_{F,n} \leq \sum_{\ell=1}^{k}\left[\prod_{p=\ell+1}^{k} u_p\right] v_\ell + \left[\prod_{\ell=1}^{k} u_p\right]\left\|\mathbf{X}^{(0)}\right\|_{F,n} \tag{43}$$

where

$$u_k = \left\|\theta_0^{(k)}\right\|_2 + \left\|S_k(\mathbf{X}^{(k-1)})\right\|_2 \left\|\theta_1^{(k)}\right\|_2, \quad v_k = \left\|\mathbf{b}^{(k)}\right\|$$

Proof. We write the recursion

$$\begin{aligned}\left\|\mathbf{X}^{(k)}\right\|_{F,n} &= \left\|\sigma\left(\mathbf{X}^{(k-1)}\theta_0^{(k)} + S_k(\mathbf{X}^{(k-1)})\mathbf{X}^{(k-1)}\theta_1^{(k)} + 1_n\mathbf{b}^{(k)}\right)\right\|_{F,n} \\ &\leq \underbrace{\left(\left\|\theta_0^{(k)}\right\|_2 + \left\|S_k(\mathbf{X}^{(k-1)})\right\|_2\left\|\theta_1^{(k)}\right\|_2\right)}_{u_k}\left\|\mathbf{X}^{(k-1)}\right\|_{F,n} + \underbrace{\left\|\mathbf{b}^{(k)}\right\|}_{v_k} \\ &= \sum_{\ell=1}^{k}\left[\prod_{p=\ell+1}^{k} u_p\right] v_\ell + \left[\prod_{\ell=1}^{k} u_p\right]\left\|\mathbf{X}^{(0)}\right\|_{F,n}\end{aligned}$$

□

B.3 Proof of Lemma 4.3

With the same reasoning as before, we have

$$\texttt{std}(C_i/C) \leq \max_i |C_i/C - 1|$$

we are going to bound the right hand side.

The feature C_i can be expressed as a sum over subsets of $r-1$ indices:

$$C_i = \sum_{\mathcal{S}\subset([1,n]\setminus\{i\})^{r-1}} 1_{G_{\{i\}\cup\mathcal{S}}\simeq H} \tag{44}$$

where $G_{\mathcal{S}}$ refers to the subgraph of G corresponding to subset of nodes $\mathcal{S}$, and $1_{F\simeq H} = 1$ if F and H are isomorphic and 0 otherwise.

By exchangeability, it is obvious that $C = \mathbb{E}(C_i)$ does not depend on the node i. For $\mathcal{S}$ of size r, since $G_{\mathcal{S}}$ is also an E-R graph of size r, we have $\mathbb{E}1_{G_{\{i\}\cup\mathcal{S}}\simeq H} = r!p^{m_H}(1-p)^{r(r-1)/2-m_H}$ where m_H is the number of edges in H, such that

$$C = \binom{n-1}{r-1} r!p^{m_H}(1-p)^{r(r-1)/2-m_H} = O(n^{r-1}) \tag{45}$$

We are going to bound the deviation of C_i/C from its expectation of 1 using McDiarmid's concentration inequality [33], which requires to bound the variation of C_i/C when one edge $e_{k\ell}$ flips, and sum the square of these variations over all indices k, ℓ.

If $e_{k\ell}$ flips and k, ℓ are different from i, then the terms affected are those where $\mathcal{S}$ contains both k, ℓ. There are $\binom{n-3}{r-3} = O(n^{r-3})$ such terms, such that the deviation of C_i/C is of order $1/n^2$. There are $O(n^2)$ such indices k, ℓ, such that the sum of the square of the deviation has a term $O(n^2/n^4) = O(1/n^2)$.

If $e_{k\ell}$ flips and $k = i$, then the terms affected are those where $\mathcal{S}$ contains ℓ. There are $\binom{n-2}{r-2} = O(n^{r-2})$ such terms, such that the deviation of C_i/C is of order $1/n$. There are $O(n)$ such indices k, ℓ, such that the sum of the square of the deviation has a term $O(n/n^2) = O(1/n)$. Similarly when $\ell = i$.

Hence, by applying McDiarmid's inequality, with probability $1 - \delta$, $|C_i/C - 1|$ is bounded by $\sqrt{\log(1/\delta)/n}$. Using a union bound over i, we obtain the result.

B.4 Proof of Lemma 5.1

A simple approximation of $1 - e^{-x}$ yields

$$p_{ij} = e^{u_i+u_j} + O((e^{u_i+u_j})^2) = e^{u_i+u_j} + O(e^{-4c_n})$$

Then,

$$\mathbb{E}(d_i \mid \mathbf{u}) = \sum_j p_{ij} = \sum_j e^{u_i+u_j} + O(ne^{-4c_n})$$
$$\underbrace{\left(\sum_j e^{u_j}\right)}_{C} e^{u_i} + O(ne^{-4c_n})$$

C Variational motivation for the training objective

The following derivation provides variational motivation for the joint training objective in Equation 19. We follow the DiGress framework [51] and highlight the additional terms arising from the continuous latent component $\mathbf{u}$.

A key structural feature of our forward process is that the discrete transition matrices $\bar{\mathbf{Q}}_E^{t,ij}$ are parameterized by the *clean* latent vector $\mathbf{u}^0$ rather than the noisy $\mathbf{u}^t$ this is what gives the forward process its closed-form factorization (Section 5.2). As a consequence, the terminal distribution of the forward process conditioned on a training graph G^0 has a discrete component that depends on $\mathbf{u}^0$: edges converge to $\mathrm{Bernoulli}(p_{ij}^0)$ where $p_{ij}^0 = 1 - e^{-e^{u_i^0+u_j^0}}$. The sampling process (Algorithm 3), on the other hand, draws $\mathbf{u}^T \sim \mathcal{N}(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}}^2 I)$ and then samples edges from $\mathrm{Bernoulli}(p_{ij}^T)$ where p_{ij}^T depends on $\mathbf{u}^T$. These two terminals coincide only in expectation over $\mathbf{u}^0$. We therefore do not claim that 19 is an exact ELBO for the generative model in Algorithm 3; rather, following the pragmatic stance of DiGress [51], we treat the objective as a well-motivated denoising loss whose variational structure we describe below.

Let the joint state at time t be $z^t = (\mathbf{u}^t, \mathbf{E}^t)$, with clean graph $z^0 = (\mathbf{u}^0, \mathbf{E}^0)$ and terminal state $z^T = (\mathbf{u}^T, \mathbf{E}^T)$.

Joint forward factorization. Because the continuous and discrete components are corrupted independently, the forward process factorizes as

$$q(z^t \mid z^0) = q(\mathbf{u}^t \mid \mathbf{u}^0)\, q(\mathbf{E}^t \mid \mathbf{E}^0, \mathbf{u}^0) \tag{46}$$

The continuous component follows the shifted-scaled Gaussian diffusion of Appendix A.2:

$$q(\mathbf{u}^t \mid \mathbf{u}^0) = \mathcal{N}\big(\mathbf{u}^t \;\big|\; \mu_{\mathrm{LSP}} + \sqrt{\bar{\alpha}_t}(\mathbf{u}^0 - \mu_{\mathrm{LSP}}),\ \sigma_{\mathrm{LSP}}^2(1 - \bar{\alpha}_t)I\big) \tag{47}$$

The discrete component follows from the pair-specific forward marginals of Equation 15:

$$q(\mathbf{E}^t \mid \mathbf{E}^0, \mathbf{u}^0) = \prod_{1 \le i,j \le n} q(e_{ij}^t \mid e_{ij}^0,\, \mathbf{u}^0) \tag{48}$$

Evidence lower bound. Following [51], applying Jensen's inequality with $q(z^{1:T} \mid z^0)$ as the variational distribution gives the ELBO:

$$\log p_\theta(G) \ge \log p(n_G) - D_{\mathrm{KL}}\big[q(z^T \mid G) \,\|\, p(z^T \mid n_G)\big] - \sum_{t=2}^{T} L_t(G) + \mathbb{E}_{q(z^1 \mid G)}\big[\log p_\theta(G \mid z^1)\big] \tag{49}$$

where

$$L_t(G) = \mathbb{E}_{q(z^t \mid G)}\left[D_{\mathrm{KL}}\big(q(z^{t-1} \mid z^t, z^0) \,\|\, p_\theta(z^{t-1} \mid z^t)\big)\right] \tag{50}$$

Terminal prior KL. With the $\mathbf{u}^0$-keyed forward process, the training terminal factors as

$$p(z^T \mid n_G) = \mathcal{N}(\mu_{\mathrm{LSP}},\, \sigma_{\mathrm{LSP}}^2 I) \times \prod_{1 \le i,j \le n} \mathrm{Bernoulli}(p_{ij}^0) \tag{51}$$

where $p_{ij}^0 = 1 - \exp(-e^{u_i^0 + u_j^0})$ depends on the clean graph. The prior KL decomposes as

$$\begin{aligned} D_{\mathrm{KL}}\big[q(z^T \mid G) \,\|\, p(z^T \mid n_G)\big] &= D_{\mathrm{KL}}\big[q(\mathbf{u}^T \mid \mathbf{u}^0) \,\|\, \mathcal{N}(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}}^2 I)\big] \\ &\quad + D_{\mathrm{KL}}\left[q(\mathbf{E}^T \mid \mathbf{E}^0, \mathbf{u}^0) \,\|\, \prod_{1 \le i,j \le n} \mathrm{Bernoulli}(p_{ij}^0)\right] \end{aligned} \tag{52}$$

Both terms vanish as $\bar{\alpha}_T \to 0$ for the $\mathbf{u}^0$-keyed forward process: the continuous component converges to $\mathcal{N}(\mu_{\mathrm{LSP}}, \sigma_{\mathrm{LSP}}^2 I)$ and each row of $\bar{\mathbf{Q}}_E^{T,ij}$ converges to $(\mathbf{p}_{ij}^0)^\top$, so $q(e_{ij}^T \mid e_{ij}^0, \mathbf{u}^0)$ converges to $\mathrm{Bernoulli}(p_{ij}^0)$ independently of e_{ij}^0. Note that this vanishing holds for the training terminal conditioned on $\mathbf{u}^0$, not for the unconditional sampling terminal.

Decomposition of the diffusion loss. Because the forward posterior factorizes as

$$q(z^{t-1} \mid z^t, z^0) = q(\mathbf{u}^{t-1} \mid \mathbf{u}^t, \mathbf{u}^0)\, q(\mathbf{E}^{t-1} \mid \mathbf{E}^t, \mathbf{E}^0, \mathbf{u}^0) \tag{53}$$

the per-step loss $L_t(G)$ splits into independent continuous and discrete terms:

$$L_t(G) = L_t^u(G) + L_t^{\mathbf{E}}(G) \tag{54}$$

where $L_t^{\mathbf{E}}(G)$ is identical in form to the DiGress per-step loss [51], and $L_t^u(G)$ is the additional term:

$$L_t^u(G) = \mathbb{E}_{q(\mathbf{u}^t \mid \mathbf{u}^0)}\left[D_{\mathrm{KL}}\big(q(\mathbf{u}^{t-1} \mid \mathbf{u}^t, \mathbf{u}^0) \,\|\, p_\theta(\mathbf{u}^{t-1} \mid z^t)\big)\right] \tag{55}$$

Practical training objective. As in DDPM [18] and DiGress [51], we do not optimize the ELBO term-by-term. Instead, we optimize the simplified joint objective of Equation 19, replacing the per-step KL terms with cross-entropy and MSE losses respectively. The prior KL terms are dropped since they vanish for the $\mathbf{u}^0$-keyed forward process by construction.

D Additional experiments

D.1 DiGress on Tree dataset

In Table 3 we also present results on the Tree dataset [54] comparing DiGress with no aux features and our JointLSP with DiGress. We can observe that our proposed method outperforms DiGress on 3 out of 4 metrics.

Table 3: Results on Tree dataset

Method	Degree	Assortativity	PLE	CPL
DiGress	1073.41 ± 38.35	33.89 ± 0.81	242.14 ± 0.63	**222.38 ± 6.51**
JointLSP+DiGress	**204.17 ± 26.83**	**15.45 ± 1.58**	**134.26 ± 2.03**	231.38 ± 1.19

Table 4: MMD ratios on CoraChameleon datasets for SparseDiff and JointLSP+SparseDiff under no-node features. Results are mean±std over 3 random seeds. Bold: better of SparseDiff vs. JointLSP+SparseDiff.

	Degree ↓	Clustering ↓	Triangles ↓	Assortativity ↓	PLE ↓	CPL ↓
SparseDiff (no features)						
$n = 10$	$17.23_{\pm 0.87}$	$\mathbf{1.99}_{\pm 0.02}$	$6.68_{\pm 0.07}$	$69.77_{\pm 18.17}$	$431.13_{\pm 133.46}$	$114.33_{\pm 21.79}$
$n = 20$	$49.10_{\pm 2.08}$	$1.41_{\pm 0.00}$	$3.67_{\pm 0.43}$	$30.17_{\pm 4.03}$	$27.25_{\pm 2.67}$	$0.92_{\pm 0.12}$
$n = 50$	$17.51_{\pm 0.98}$	$1.55_{\pm 0.04}$	$25.87_{\pm 0.77}$	$60.65_{\pm 4.84}$	$\mathbf{4.98}_{\pm 0.59}$	$\mathbf{3.69}_{\pm 0.39}$
$n = 100$	$30.43_{\pm 0.15}$	$3.43_{\pm 0.10}$	$66.65_{\pm 0.76}$	$50.44_{\pm 2.34}$	$15.33_{\pm 0.10}$	$18.92_{\pm 0.10}$
$n = 500$	$3.93_{\pm 0.01}$	$3.23_{\pm 0.02}$	$113.81_{\pm 7.41}$	$39.04_{\pm 0.14}$	$\mathbf{2.13}_{\pm 0.00}$	$\mathbf{7.93}_{\pm 0.00}$
JointLSP+SparseDiff (no features)						
$n = 10$	$\mathbf{14.18}_{\pm 0.64}$	$2.00_{\pm 0.01}$	$\mathbf{6.43}_{\pm 0.14}$	$\mathbf{11.38}_{\pm 7.16}$	$\mathbf{37.70}_{\pm 14.13}$	$\mathbf{37.19}_{\pm 3.19}$
$n = 20$	$\mathbf{29.08}_{\pm 1.11}$	$\mathbf{1.40}_{\pm 0.00}$	$\mathbf{2.19}_{\pm 0.23}$	$\mathbf{15.31}_{\pm 0.95}$	$\mathbf{15.75}_{\pm 1.29}$	$\mathbf{0.54}_{\pm 0.12}$
$n = 50$	$\mathbf{13.29}_{\pm 0.32}$	$\mathbf{1.49}_{\pm 0.01}$	$\mathbf{15.11}_{\pm 0.34}$	$\mathbf{36.18}_{\pm 1.16}$	$6.29_{\pm 0.19}$	$5.76_{\pm 0.16}$
$n = 100$	$\mathbf{19.07}_{\pm 0.49}$	$\mathbf{3.07}_{\pm 0.07}$	$\mathbf{36.00}_{\pm 0.84}$	$\mathbf{24.88}_{\pm 1.18}$	$\mathbf{11.14}_{\pm 0.23}$	$\mathbf{16.24}_{\pm 0.21}$
$n = 500$	$\mathbf{3.82}_{\pm 0.01}$	$\mathbf{2.99}_{\pm 0.02}$	$\mathbf{35.41}_{\pm 4.21}$	$\mathbf{28.89}_{\pm 0.27}$	$2.20_{\pm 0.00}$	$8.37_{\pm 0.00}$

D.2 Graph generation using JointLSP+SparseDiff

We augment SparseDiff [42] with our prior and joint diffusion process, referring to the result as JointLSP+SparseDiff. SparseDiff is particularly efficient at larger graph sizes, processing only randomly selected node pairs at each step, which is why we use it as the baseline for $n = 500$ and the Ego and Community datasets.

The results on CoraChameleon across $n \in \{10, 20, 50, 100, 500\}$ are reported in Table 4. JointLSP+SparseDiff consistently improves over SparseDiff on most metrics, with gains most pronounced on triangle count and assortativity. Notably, the improvement holds at $n = 500$, where the fixed-point pathology is most severe and the edge-independent prior has the greatest influence on generated statistics.

The results on Ego and Community are reported in Table 5. These datasets remain challenging for both methods, as indicated by the uniformly large MMD ratios. This is likely due in part to the presence of self-loops in Ego and the lack of auxiliary spectral features needed to capture community structure. Nevertheless, JointLSP+SparseDiff improves over SparseDiff on all metrics except clustering on Ego and assortativity on Community. Taken together, these results support our main claim: the benefit of the proposed method is not that it universally optimizes generation quality, but that it addresses a fundamental limitation of edge-independent diffusion in the no-node feature setting.

Table 5: Comparison of graph generation with JointLSP+SparseDiff and SparseDiff on Ego and Community datasets. Results are mean±std over 3 random seeds.

	Degree ↓	Clustering ↓	Triangles ↓	Assortativity ↓	PLE ↓	CPL ↓
Ego						
SparseDiff	$768.06_{\pm 107.44}$	$\mathbf{14.78}_{\pm 0.25}$	$85.20_{\pm 12.21}$	$1715.86_{\pm 53.10}$	$1696.14_{\pm 183.27}$	$183.07_{\pm 31.15}$
JointLSP+SparseDiff	$\mathbf{302.50}_{\pm 51.36}$	$15.48_{\pm 0.57}$	$\mathbf{12.41}_{\pm 1.25}$	$\mathbf{785.72}_{\pm 141.52}$	$\mathbf{750.94}_{\pm 145.86}$	$\mathbf{28.95}_{\pm 6.31}$
Community						
SparseDiff	$91.57_{\pm 24.34}$	$6.86_{\pm 0.63}$	$8.12_{\pm 0.96}$	$0.99_{\pm 1.27}$	$226.18_{\pm 49.36}$	21917.62±157.43
JointLSP+SparseDiff	$\mathbf{38.70}_{\pm 14.18}$	$\mathbf{5.08}_{\pm 0.44}$	$\mathbf{1.54}_{\pm 0.44}$	$\mathbf{9.02}_{\pm 6.79}$	$\mathbf{105.09}_{\pm 39.66}$	$\mathbf{18251.36}_{\pm 550.46}$

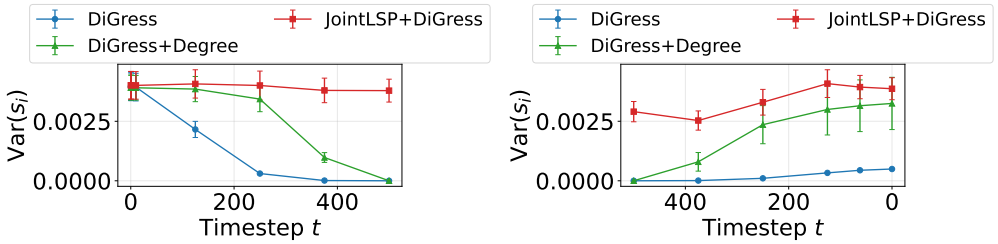
D.3 DiGress with degree features

In the JointDiffusion+LSP proposal we derive the latent weights for each node by using the node degree as a proxy. This naturally leads to the question: is this similar to supplying node degree as auxiliary features to the denoiser, and whether DiGress [51] could also benefit from such augmentation? To answer this question, we conduct an experiment where we compute node degrees from the noisy graph at each denoising step and feed them as auxiliary features. More specifically, we compute the normalized degree $\tilde{d}_i = d_i/(n-1)$ as a node-level feature, and the mean μ_d and standard deviation σ_d of $\tilde{d}_i$ across nodes as graph-level features, recomputed from the noisy adjacency A_t at every reverse step. The results are shown in Table 6. We can observe that JointLSP+DiGress outperforms DiGress+Degree features on metrics like degree, triangles, PLE and CPL. Thus, DiGress augmented with degree features does not recover the performance of JointLSP. This is consistent with Lemma 4.3: at high noise levels A_t is approximately E-R, so all $\tilde{d}_i \approx \bar{p}$ and the features collapse to a near-constant, carrying no structural signal. The key distinction is that the LSP does not compute degree on the noisy graph, it supplies this information inherently by construction. The degree-informed heterogeneity information is baked into the terminal prior itself via u^0, making node-level variation available throughout the entire reverse trajectory rather than only when the graph is sufficiently clean.

Table 6: DiGress + degree features on CoraChameleon-Node100. JointLSP+DiGress ($k = 0$, no features) is included for direct comparison. Bold: best result excluding the $k = 17$ ceiling.

	Degree ↓	Clustering ↓	Triangles ↓	Assortativity ↓	PLE ↓	CPL ↓
DiGress+Degree						
$k = 1$	$10.67_{\pm 0.14}$	**2.04**$_{\pm 0.02}$	$2.41_{\pm 0.14}$	**57.66**$_{\pm 0.76}$	$10.81_{\pm 0.17}$	$45.54_{\pm 0.56}$
JointLSP+DiGress						
$k = 0$	**10.50**$_{\pm 0.16}$	$2.07_{\pm 0.10}$	**2.33**$_{\pm 0.15}$	$59.72_{\pm 0.75}$	**10.79**$_{\pm 0.19}$	**44.99**$_{\pm 0.29}$
DiGress + 17 features						
$k = 17$	$0.93_{\pm 0.00}$	$1.95_{\pm 0.00}$	$1.68_{\pm 0.00}$	$3.55_{\pm 0.00}$	$1.53_{\pm 0.00}$	$1.92_{\pm 0.00}$

DiGress+degree does not preserve node heterogeneity. We compute the variance in node propensity scores as defined in Section 6.1, comparing JointLSP+DiGress, DiGress, and DiGress+Degree — where node degree is supplied as both a node-level and graph-level auxiliary feature — in Figure 3. JointLSP+DiGress maintains consistently high node heterogeneity throughout the reverse trajectory, starting from $t = T$ where the LSP prior already induces diverse connection propensities. DiGress+Degree, by contrast, starts near zero at high noise levels and builds heterogeneity only as denoising progresses. This confirms that degree features are reactive: they can only provide structural signal once the graph is already partially denoised, whereas LSP provides it from initialization. The two mechanisms are therefore fundamentally different, even though both draw on degree information.



(a) Forward Noising Process.

(b) Reverse Denoising Process.

Figure 3: We compare the variance in node propensity scores for JointLSP+DiGress, DiGress+degree and DiGress. We can observe that our proposal yields node heterogeneity throughout the diffusion trajectory even at highly noisy states as opposed to DiGress with degree features which only improve the node heterogeneity when the graph is already denoised.

Random node features Many works such as [1, 45, 16] suggest incorporating random node features improve GNN expressivity, we test if this also helps improve generation quality in graph diffusion models. We verify this empirically on the CoraChameleon-Node100 dataset and compare DiGress with no aux features, DiGress+RND which denotes the random node features and JointLSP+Digress. The setup is as follows: The diffused node state is unchanged ($X = \mathbf{1}$), and at each denoising step we concatenate random features $r \in \mathbb{R}^{n \times d}$, drawn i.i.d. from $\mathcal{N}(0, I)$ with dim=8 to the node input. The results are presented in Table 7, adding random node features does not help with generation because firstly RND leaves the corruption process and the terminal distribution unchanged: Lemma 4.1 still applies and the terminal distribution is still E-R, and sampling still begins from a graph with no node-level structure. RND alters only the denoiser input, not where the reverse trajectory starts. Second, the heterogeneity it supplies is uncalibrated, as the features are drawn independently of the data, whereas LSP is calibrated from the training data.

Table 7: Results on CoraChameleon-Node100 dataset comparing DiGress with random node features

Method	Degree	Clustering	Triangles	Assortativity	PLE	CPL
DiGress	**4.53±0.1**	11.33±0.27	14.82±0.12	101.95±2.13	4.64±0.17	39.68±1.14
DiGress+RND	5.48±0.13	12.50±1.18	10.13±0.04	127.06±6.20	**4.14±0.41**	**24.01±1.57**
JointLSP+DiGress	**10.50±0.16**	**2.07±0.10**	**2.33±0.15**	**59.72±0.75**	10.79±0.19	44.99±0.29

D.4 LSP improves generation quality throughout diffusion trajectory

To further demonstrate the effectiveness of the proposed JointLSP framework, we track the MMD ratio (r) presented in Section 6.2 at various timesteps $t \in \{500, 50, 0\}$ for both JointLSP+DiGress and DiGress on CoraChameleon-Node100 dataset. At each timestep t, we use the network's predicted clean edge probabilities $\hat{\mathbf{p}}_{ij}^{E,t}$ to construct a graph and evaluate MMD against the test set. The results are shown in Figure 4. JointLSP+DiGress consistently achieves lower MMD ratios than DiGress at all timesteps across all three structural statistics except Degree, where we surmise the LSP over-corrects the degree when inducing node heterogeneity. At $t = 500$, where the input is close to the terminal prior, DiGress produces graphs with very high MMD as the ER prior provides no structural information making it difficult for the denoiser. JointLSP+DiGress already achieves substantially lower MMD at this stage, confirming that the LSP along with the joint diffusion process provides a more informative starting point. As $t \to 0$ both models improve, but JointLSP+DiGress maintains a consistent advantage, particularly on assortativity and triangle count. This supports our main claim that our proposal indeed improves the diffusion pipeline and does not result in the fixed-point pathology we have pointed out.

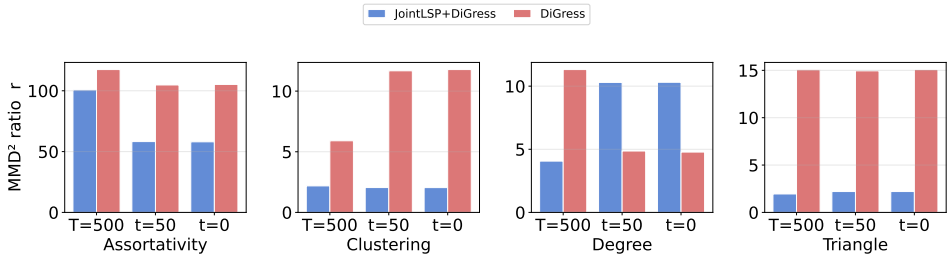


Figure 4: We track the MMD ratio across various diffusion timesteps for JointLSP+DiGress and DiGress. Our proposed framework consistently generates good quality graphs even at intermediate steps.

D.5 LSP induces node heterogeneity

To verify that the proposed prior preserves node distinguishability, we simulate graphs ($m = 100$ graphs) with increasing node sizes $n = \{10, 20, 50, 100, 500, 1000\}$ under three random graph models:

1. Dense Bernoulli baseline: $A_{ij} \sim \texttt{Bern}(\rho_{\text{dense}})$, where $\rho_{\text{dense}} = 0.1$.

2. Sparse mean-field Bernoulli: $A_{ij} \sim \texttt{Bern}(\rho_{\text{sparse}})$, where $\rho_{\text{sparse}} = \frac{\bar{k}}{n-1}$ and $\bar{k} = 5$, and
3. Latent Sociability Prior: $A_{ij} \sim \texttt{Bern}(P_{ij})$, where $P_{ij} = 1 - e^{(-ce^{u_i+u_j})}$.

To quantify node-level heterogeneity, we measure the within-graph degree variance $\text{Var}_i(d_i^{(m)})$ and within-graph degree mean $\mathbb{E}_i[d_i^{(m)}]$, and define the dispersion index

$$\varphi = \frac{\text{Var}_i[d_i^{(m)}]}{\mathbb{E}_i[d_i^{(m)}]}, \tag{56}$$

which we then average across all sampled graphs. This statistic captures how much node degrees vary within a graph. The results are shown in Figure 5. As $n \to 1000$, both the dense and sparse Bernoulli baselines yield $\varphi \approx 1$, indicating that edge-independent models remain tightly constrained by their mean degree and therefore, by design, cannot induce meaningful node heterogeneity. In contrast, LSP consistently yields $\varphi > 1$, showing that it preserves non-trivial degree variation as graph size increases. This confirms that, unlike ER-type models, LSP does not collapse to a homogeneous fixed-point.

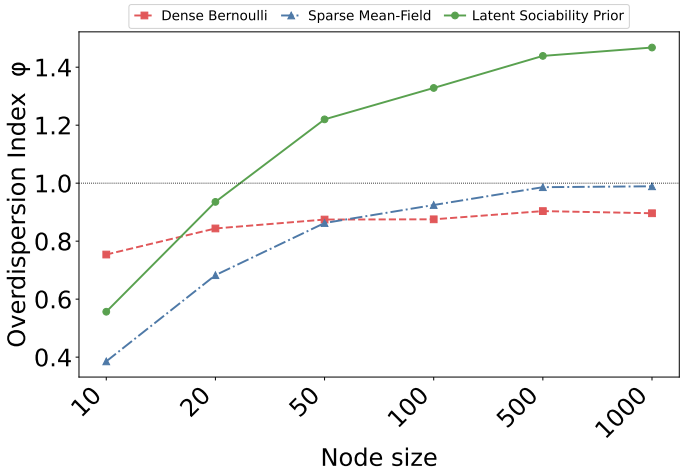


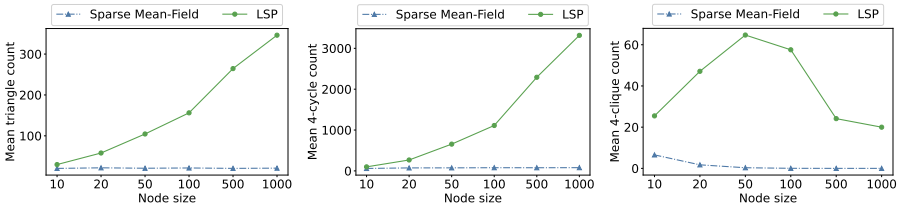
Figure 5: Dispersion index φ as a function of node size under three prior models. Unlike edge-independent Bernoulli baselines, LSP preserves non-trivial node heterogeneity as graph size increases.

D.6 LSP can generate higher triangle and cycle counts

We conduct a second experiment to study how higher-order graph statistics scale with increasing graph size under a fixed mean-degree constraint. This distinction is important, since large triangle or cycle counts can be obtained trivially by generating dense graphs. Specifically, for node sizes $n = \{10, 20, 50, 100, 500, 1000\}$, we simulate 100 graphs from each random graph model while fixing the mean degree to the same target value. For each graph, we measure the number of triangles, 4-cycles, and 4-cliques, and study how these quantities scale as n increases. The results, shown in Figure 6, indicate that LSP consistently produces substantially larger higher-order structure counts than a sparse mean-field Bernoulli prior under the same mean-degree budget. This shows that the gain is not a consequence of simply generating denser graphs, but of inducing non-trivial dependencies that support richer local structure.

D.7 Structural features are uninformative for E-R graphs

In Figure 7 we empirically validate Lemma 4.3. We simulate 100 graphs of node sizes $n \in \{100, 500, 1000, 2000, 5000\}$ under two priors at matched edge density $\rho = 0.1$. For E-R, each edge is drawn independently with probability $p = 0.1$. For the LSP, we fix $\sigma_{\text{LSP}} = 1.0$ and solve for μ_{LSP} such that $\mathbb{E}[p_{ij}] = 0.1$ using the calibration procedure of Algorithm 1. Graphs are then sampled directly from the prior by drawing $u_i \sim \mathcal{N}(\mu_{\text{LSP}}, \sigma^2_{\text{LSP}})$ and connecting each pair (i, j)



(a) Triangle Count. (b) 4-Cycle Count. (c) 4-Clique Count.

Figure 6: We simulate graphs using the sparse Bernoulli and our proposed LSP and compute how many triangles, 4-cycles and 4-cliques can the prior generate at a fixed mean-degree resolution while the number of nodes is increased.

with probability $p_{ij} = 1 - e^{-e^{u_i+u_j}}$. Note that no diffusion process is involved; we sample from the terminal prior directly to isolate the effect of the prior on structural feature statistics. For each graph, we compute the per-node triangle count C_i and measure $\mathrm{std}(C_i/\bar{C})$ where $\bar{C} = \mathrm{mean}_i(C_i)$. As n grows, this quantity vanishes for E-R graphs, confirming that structural features become increasingly uniform across nodes and therefore uninformative to a denoiser. The LSP maintains a stable, non-trivial spread across all graph sizes.

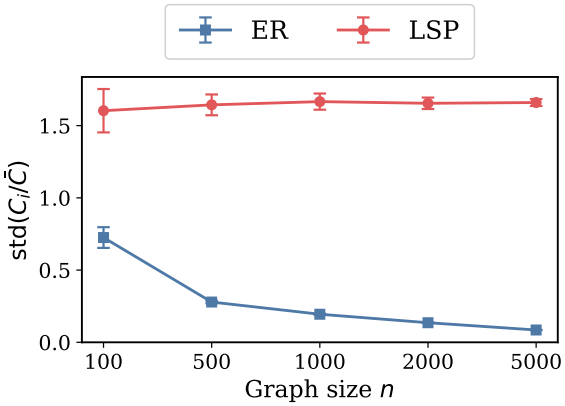


Figure 7: We empirically validate the hypothesis that in the large graph limit the structural features derived from an E-R are uninformative for the denoiser.

E Dataset generation

CoraChameleon datasets. Existing datasets like Community and Planar graphs which are usually used in evaluation of the graph diffusion models do not accurately reflect the characteristics of real-world graphs. Hence, we generate 5 new datasets by sampling from two real-world datasets Cora [32] and Chameleon [44].

We apply 3 sampling strategies to the source graphs: random walk based where a biased random walk starts from a random node. The walk continues until ~ 1.5x the target node size has been visited, then the BFS-largest connected component is taken. After half of the allowed attempts, the starting node is biased toward high-degree nodes to improve connectivity. Forest-fire [30] which starting from a seed node, each active node stochastically ignites a geometrically-distributed number of its un-burned neighbors with forward probability $p_{\mathrm{forward}} = 0.7$ and backward probability $p_{\mathrm{backward}} = 0.3$. And a dense-core sampling strategy that randomly picks top-10% of highest degree nodes in the source graph as candidate neighbours and greedily scores how many edges are already in the currently selected set and picks the highest-scoring candidate and adds it. A density check is done once target size nodes are collected, rejects the subgraph if it isn't connected or if its average degree is below minimum average degree (default 4.0). We use a ratio of 40% Random walk, 40% Forest-fire and 20% Dense core to generate our synthetic graphs. Additionally 100 graphs are generated which are

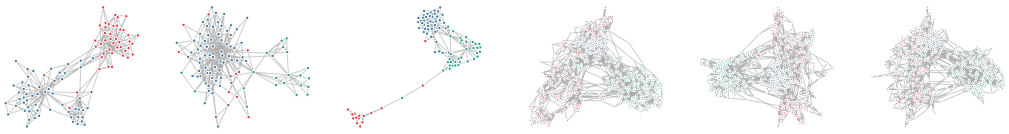
divided into 60/40 as test and validation sets respectively. The hyperparameters are tuned on the validation set.

Ego network Similar to SparseDiff [42] we also evaluate our models on the Ego dataset [54] which is a 3-hop ego network extracted from the Citeseer dataset [46].

Community dataset We also evaluate on the community dataset [54] which consists of 2 communities generated by an E-R model with $n = |V|/2$ nodes and $p = 0.3$. The inter-community edges are added with probability $0.05|V|$ with uniform probability. In Table 8 we provide the data statistics of the graphs used for the generation task.

Table 8: Statistics for the datasets used in the graph generation experiments

Dataset	#Graphs	#Nodes	#Edges
CoraChameleon-Node10	3000	10	[14,45]
CoraChameleon-Node20	3000	20	[27,120]
CoraChameleon-Node50	3000	50	[71,300]
CoraChameleon-Node100	3000	100	[172,600]
CoraChameleon-Node500	1400	500	[1282,4630]
EGO	757	[50,399]	[57,1071]
Community	510	[60,160]	[231,1956]



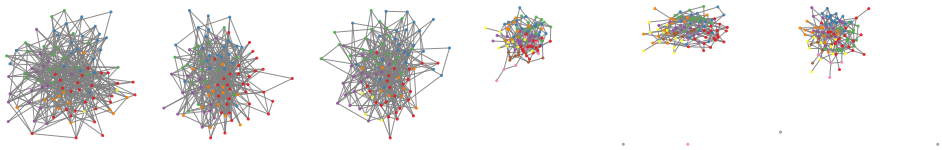
(a) CoraChameleon-Node100

(b) CoraChameleon-Node500



(c) Community dataset

Figure 8: Sample graphs from the training set.



(a) Sample CoraChameleon-Node100 graphs generated using JointLSP+DiGress.

(b) Sample CoraChameleon-Node100 graphs generated using DiGress.

Figure 9: We visualize the generated graphs from both JointLSP+DiGress (a) and DiGress (b) on CoraChameleon-Node100 dataset. Clearly, we can observe that without node features the generation quality is impacted, but DiGress generates graphs which are dense and hairball-like even generating isolated nodes.

E.1 Training details

All experiments were implemented in PyTorch and PyTorch Geometric [13]. Our model architecture follows DiGress and SparseDiff, using a Graph Transformer [11] as the denoising network. All

experiments were run on a single A40 GPU. Hyperparameters are provided in Table 9. For each configuration, we train a single model and run inference three times with different random seeds; reported mean and standard deviation are computed over these three inference runs. This evaluation setup is consistent with the protocol followed in existing graph diffusion papers such as DeFoG [41] which reports the mean and std deviation averaged over 5 sampling runs and not 5 independent runs as it is computationally infeasible. For instance, training DiGress [51] on Planar graphs requires nearly 50K epochs with a 10 layer transformer and DeFoG is trained for 100000 epochs with a 10 layer transformer. Additionally, computing the auxiliary features such as spectral adds further overhead. In our setup, we report the mmd ratios averaged over 3 sampling runs. Our code is available here `https://github.com/AdarshMJ/JointDiffusionwithLSP`.

Table 9: Hyperparameters used for the graph generation task.

Dataset	#Layers	LR	Weight Decay	#Training Epochs	Diffusion Steps
JointLSP+DiGress					
CoraChameleon-Node10	6	0.0001	0.00012	5000	500
CoraChameleon-Node20	6	0.0001	0.00012	5000	500
CoraChameleon-Node50	6	0.0001	0.00012	5000	500
CoraChameleon-Node100	6	0.0001	0.00012	5000	500
JointLSP+SparseDiff					
CoraChameleon-Node10	6	0.0001	0.00001	100	500
CoraChameleon-Node20	6	0.0001	0.00001	100	500
CoraChameleon-Node50	6	0.0001	0.00001	100	500
CoraChameleon-Node100	6	0.0001	0.00001	100	500
CoraChameleon-Node500	4	0.0001	0.00001	100	500
EGO	6	0.0001	0.000012	2000	500
Community	6	0.0001	0.000012	1000	500

Runtime analysis LSP does not change the complexity of the underlying model. The denoiser forward pass is $O(Kn^2d^2)$ for both DiGress and JointLSP+DiGress; the operations LSP adds is computing $p^0_{ij} = 1 - e^{-e^{u^0_i + u^0_j}}$ across node pairs and the DDPM update on the n latent variables which are $O(n^2)$ and $O(n)$ respectively, smaller than the base cost by a factor $O(Kd^2)$. Prior calibration (Algorithm 1) is a one time pre-processing step, not per-iteration.

Table 10: Runtime analysis

Metric	DiGress	JointLSP+DiGress
Model parameters	8,888,579	8,888,579
Training step (in ms)	493.3±4.7	489.8±5.0
Peak GPU Memory (GiB)	13.40	13.39
Sampling (graph/sec)	9.50	9.33

List of auxiliary features In 11 the details of node level and graph level features used as auxiliary features for DiGress is provided.

F Broader impact

Graph diffusion models have potential applications across a range of domains, including drug discovery, protein interaction networks, and social network analysis, where the ability to generate realistic graph structures could accelerate scientific discovery or inform network design. Like all generative models, these methods are not free from certain risks, for instance, realistic graph generation could in principle be misused to generate fake synthetic social network structures or in the long term, to design molecules with adversarial properties. However, we note that these risks are not specific to our contribution. This work addresses a theoretical limitation in existing discrete graph diffusion pipelines and proposes a prior and a joint diffusion framework that improves graph generation in the no-node-feature setting.

Table 11: Auxiliary features used in DiGress.

Feature	Description
`k3_node`	Node triangle count
`k4_node`	Node 4-cycle count
`k5_node`	Node 5-cycle count
`k3_graph`	Graph triangle count
`k4_graph`	Graph 4-cycle count
`k5_graph`	Graph 5-cycle count
`k6_graph`	Graph 6-cycle count
`n_comp`	Number of connected components
`nonLCC`	Indicator for nodes outside the largest connected component
`eigvec1`	First nontrivial Laplacian eigenvector
`eigvec2`	Second nontrivial Laplacian eigenvector
`eigval1`	First nonzero Laplacian eigenvalue
`eigval2`	Second nonzero Laplacian eigenvalue
`eigval3`	Third nonzero Laplacian eigenvalue
`eigval4`	Fourth nonzero Laplacian eigenvalue
`eigval5`	Fifth nonzero Laplacian eigenvalue
`n/max_n`	Normalized graph size ($n/n_{\max}$)

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: We identify a fixed-point pathology in existing graph diffusion models applied to unconditional graph generation without node features. As a resolution we propose a novel prior called the LSP and a joint diffusion process that co-evolves latent variables along with discrete graph diffusion that allows for better graph generation.

   Guidelines:

   - The answer [N/A] 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 [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include a Limitations section 7 that discusses the scope of the current work and also includes possible future directions to extend the proposed framework.

   Guidelines:

   - The answer [N/A] 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: Our work includes several theoretical results which are substantiated with detailed proofs. All the proofs are available in Appendix B.

Guidelines:

- The answer [N/A] 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: We provide access to our code base in the supplementary materials and all the training details and hyperparameters are provided in Appendix E.1 to ensure reproducibility.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The datasets used are publicly available. The synthetic datasets used are provided as part of the code base that allows for reproducing the results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: The code and the datasets used are provided as part of the supplementary materials, while the hyperparameters are provided in the Appendix E.1.

   Guidelines:

   - The answer [N/A] 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 present results with mean$\pm$ std for all the experiments.

   Guidelines:

   - The answer [N/A] 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: We provide the training details in Appendix E.1.

   Guidelines:

   - The answer [N/A] 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 adhere to the NeurIPS Code of Ethics and our submission does not need any exceptions.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have included a broader impact section in Appendix F.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This work poses no such risks.

    Guidelines:

    - The answer [N/A] 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 assets used in the work including datasets and code bases are properly attributed in the citations.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not introduce any new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not involve any crowdsourcing.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This work does not involve any study participants.

    Guidelines:

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

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this work does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.