DISCRETE DIFFUSION FOR LARGE GRAPH GENERATION VIA STRUCTURAL CANDIDATE RESTRICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthesizing realistic graphs at scale is vital when the graphs of interest are large and real-world samples are limited or access-sensitive. Diffusion-based generators have recently driven much of the progress, offering high modeling capacity, but most such methods have quadratic computational complexity and are hence restricted to small-scale networks, currently up to 3k nodes. Existing non-quadratic methods remain limited by memorization issues and a trade-off between scalability and generation quality. Our goal is to generate large graphs whose structural statistics — e.g., degree distribution, clustering, and path length — faithfully reflect those of real-world sparse graphs, without resorting to memorizing the training data. We introduce a discrete graph diffusion model that restricts training to a structurally motivated subset of node pairs — observed edges and their wedge non-edges — reducing training complexity below quadratic in the number of nodes. To keep the noisy graph informative throughout both the forward and reverse trajectories, we design a three-class absorbing forward process governed by a *degree-aware, floored* cosine noise schedule: unlike a structure-blind schedule that adds noise to every pair identically, ours adapts to each node's degree and never fully erases the graph's structure, keeping the noisy graph informative at every step. Experiments across diverse datasets show that our model consistently ranks among the top methods for structural fidelity against existing discrete diffusion baselines in large graph generation. Our code is available at https://anonymous.4open.science/r/Graph-Diffusion-Generative-Model-E52E.

1 INTRODUCTION

Graphs in a natural way represent data in domains where a large number of entities interact in structured ways, such as financial networks, social platforms, and communication systems. In many use cases, such as fraud detection using anomaly detection, infrastructure planning by structural analysis, and behavior modeling through scenario simulation, learning representations for the structural relationships is essential. When the structural data in real-world tasks are incomplete, sensitive, or costly to obtain, generating instances of representative large graphs is crucial.

Graph diffusion-based generative models (GDGMs) offer strong training stability and architectural flexibility, but most target small graphs – molecules and similarly-sized structures – where the reverse denoising process is computationally cheap (Mo et al., 2024; You et al., 2024; Wen et al., 2024; Ayadi et al., 2024). DiGress (Vignac et al., 2023) and its generalized variant (Wang et al., 2025) produce realistic topologies and node features, but only for graphs with fewer than 300 nodes. Scaling GDGMs to large sparse graphs remains an open problem: existing attempts limit each denoising step to a smaller selection of nodes or edges (Chen et al., 2023; Qin et al., 2023), at the cost of complexity that can still scale quadratically when that selection itself grows with graph size; decouple structure and feature generation (Li et al., 2024), assuming rich node features are not always available in practice; or replace iterative denoising with autoregressive generation over a compact representation (Jang et al., 2024; Karami, 2024; Bernecker et al., 2025). This last approach shifts rather than removes the underlying cost: k2tree and Higen still incur quadratic encoding/decoding and cross-community prediction costs, respectively, while ARROW-Diff instead lacks formal convergence guarantees.

We address two limitations of existing graph diffusion models on large, sparse graphs. First, existing discrete diffusion schedules converge to a stationary distribution that is structurally uninformative for

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sparse data – typically empty (Chen et al., 2023) or fully dense (Vignac et al., 2023; Qin et al., 2023) – forcing the reverse process to reconstruct structure from one of these two extremes. Second, since the overwhelming majority of node pairs in a sparse graph are never connected, training over the full pair space is dominated by low-signal absent-edge predictions. We address the first by designing a noise schedule whose stationary distribution is instead a sparse random graph matching the target degree sequence. The second we address by restricting training to a compact, structurally motivated candidate set – observed edges and their wedge-connected non-edges, together with a small randomly sampled coverage set – combined with a class-balance reweighting scheme. Our contributions are as follows:

- **Degree-aware noise schedule.** A masking schedule that reveals node pairs according to node degree keeps the noisy graph structurally informative throughout the reverse process.
- **Training with sub-quadratic complexity.** We restrict training to edges, their wedge non-edges, and a small random coverage set, and prove that, under chosen conditions, this yields an unbiased estimator of the full-pair objective with substantially improved signal-to-noise, reducing training cost from $O(N^2)$ to $O(N\mu_2)$, where μ_2 is the second moment of the degree distribution.
- **Validated structural fidelity.** We evaluate our model on generative tasks across datasets of different sizes and domains, achieving consistently strong large-graph structural fidelity relative to other diffusion models, and though small-graph generation is not our *primary* objective still reasonable performance on small graphs.

2 Related work

Graph generation has been a long-studied problem in statistics and network science, well before the emergence of deep generative models. Classical statistical models – the Erdős–Rényi model (Erdos et al., 1960), the Stochastic Block Model (Holland et al., 1983), Chung-Lu (Chung & Lu, 2006), and the more general Exponential Random Graph Model (ERGM) (Frank & Strauss, 1986) – generate graphs from a fixed probabilistic form fit to observed statistics, ranging from simple edge-independent Bernoulli models (ER, SBM, Chung-Lu) to ERGM, which allows edge dependence via sufficient statistics such as triangle counts at the cost of an intractable normalizing constant requiring MCMC for both fitting and sampling. These models motivate our approach; for example, we adopt Chung–Lu's degree-based edge probabilities as a terminal distribution.

Neural-based models learn a representation of the graph distribution via gradient-based training. Early approaches such as GVAE (Kipf & Welling, 2016) and NetGAN (Bojchevski et al., 2018) adapted variational and adversarial generative frameworks to graphs. Autoregressive models, such as GraphRNN (You et al., 2018), generate graphs sequentially by modeling the adjacency matrix under a fixed node ordering, but typically incur at least $O(N^2)$ complexity, limiting scalability. BiGG (Dai et al., 2020) and its weighted variant BiGG-E (Williams et al., 2025) instead achieve $O((N+m)\log N)$ via a recursive binary-tree decomposition of the adjacency matrix. Jang et al. (2024) compress the adjacency matrix into a K^2-tree sequence and generate it autoregressively with a transformer, achieving a sub-quadratic sequence length; however, encoding a graph into this representation and decoding a generated sequence back into an adjacency matrix are each still $O(N^2)$ operations, shifting the quadratic cost to a pre/post-processing step rather than eliminating it.

Recently, graph diffusion-based generative models (GDGMs) have emerged as a promising direction, with *continuous* variants (Jo et al., 2022; Niu et al., 2020), which map graphs into Euclidean space for noise injection and denoising, and *discrete* variants (Xu et al., 2024; Haefeli et al., 2022), which inject noise directly into the graph's discrete structure. Applying discrete GDGMs to large sparse graphs surfaces two distinct problems, not always shared by every method. First, whether intermediate noisy graphs grow denser as the forward process progresses depends on the transition's marginal: under a uniform, resampling-style transition with a dense marginal – as in DiGress – absent pairs are resampled into edges far more often than existing edges are removed, so sparse graphs trend toward becoming dense as $t \to T$, forcing the denoiser to consider a quadratic number of potential edges at intermediate steps. This does not hold for absorbing, deletion-only processes, whose forward trajectory only removes edges and thus never increases density. Second, and independently of the first, since most node pairs are absent edges even in the clean graph, training, regardless of process

type, is dominated by low-signal absent-edge predictions degrading the signal-to-noise ratio during optimization.

To address the quadratic complexity of denoising over all $\binom{N}{2}$ node pairs, several methods restrict what each denoising step computes. EDGE (Chen et al., 2023) restricts edge prediction to K active nodes per step, but its transformer-based attention makes this cost quadratic in K, $O(\max(m, K^2))$, and its empty-graph stationary distribution still forces the reverse process to reconstruct structure from scratch. SparseDiff (Qin et al., 2023) samples a fixed-ratio subset of query edges per step, giving $O(m)$ training cost, but total compute per diffusion step at sampling time still covers all $\binom{N}{2}$ pairs, and, like DiGress, its Erdős–Rényi stationary distribution is far denser than most real graphs. ARROW-Diff (Bernecker et al., 2025) avoids the full adjacency matrix altogether by diffusing over sampled random walks with a separate GNN-based edge classifier, but lacks formal convergence guarantees and can become unstable over long sequences. GraphMaker (Li et al., 2024) instead improves efficiency via decoupled, asynchronous mini-batch training of structure and features, but assumes rich node features not always available outside attributed-graph domains.

3 Preliminaries and Background

Graph representation. We represent a graph as $G = (V, E, \mathbf{X}, \mathbf{Y})$, with node set V ($|V| = N$) and edge set E. Node attributes are given by $\mathbf{X} \in \{0,1\}^{N \times K_X}$, the stacked one-hot node representation over an alphabet of size K_X, and edge attributes by $\mathbf{Y} \in \{0,1\}^{N^2 \times K_Y}$, the stacked one-hot edge representation over alphabet $\mathcal{Y}$ of size K_Y, with single-step transition entries $[\mathbf{Q}_t^Y]_{ab} = q_e(e_t{=}b \mid e_{t-1}{=}a)$, $a, b \in \mathcal{Y}$. When no edge attribute information is modeled – i.e. edges carry only existence, not type – $\mathbf{Y}$ reduces to the adjacency matrix $A \in \{0,1\}^{N \times N}$ ($K_Y{=}2$: no-edge/edge), where $A_{ij} = 1$ iff an edge exists between i and j. This is the setting our own model operates in.

Graph diffusion process. Graph diffusion generative models (GDGMs) define a forward process $q(G_t \mid G_{t-1})$, $t = 1, \ldots, T$, that gradually corrupts G_0 into a noisy graph G_T, and train a denoiser $p_\theta(G_{t-1} \mid G_t)$ iteratively generating graphs from G_T to G_0 to reverse it. In discrete graph diffusion, this is typically written as $q(G_t \mid G_0) \propto (\mathbf{X}\bar{\mathbf{Q}}_t^X, \mathbf{Y}\bar{\mathbf{Q}}_t^Y)$, with cumulative transition $\bar{\mathbf{Q}}_t^Y = \mathbf{Q}_1^Y \cdots \mathbf{Q}_t^Y$ (analogously for $\mathbf{Q}_t^X, \bar{\mathbf{Q}}_t^X$). A common parameterization is $\mathbf{Q}_t = \alpha_t \mathbf{I} + (1 - \alpha_t)\mathbf{1}\mathbf{p}^\top$, preserving the current state with probability α_t and resampling from marginal $\mathbf{p}$ otherwise. The reverse process predicts from t to $t-1$ via

$$p_\theta^x(x_{t-1}^i \mid G_t) = \sum_{x^i \in \mathcal{X}} q^x(x_{t-1}^i \mid x_t^i, x^i, G_t)\, p_\theta^x(x^i \mid G_t), \tag{1}$$

and analogously for edges.

4 Method

Problem statement. Given a dataset of large, sparse graphs $\{G_0^{(k)}\}$ (a single large, sparse graph G_0 being the special case of a dataset with one element), our goal is a discrete diffusion model that generates new graphs whose structural statistics — e.g., degree distribution, clustering, path length, among others — faithfully match those of the training data without memorizing any $G_0^{(k)}$, and that scales to large graphs by achieving sub-quadratic training complexity in N, in contrast to the $O(N^2)$ cost of processing all node pairs.

We propose that GDDMs can be scaled to large, sparse graphs through two complementary design choices: a forward process that restricts training to a compact, structurally meaningful subset of node pairs rather than all $\binom{N}{2}$, and a denoiser conditioned on local structural signal rather than edge state alone. We condition the denoiser on the common-neighbor count, τ^{ij}, a local closure statistic related to clustering coefficient that gives a direct, per-pair signal of local topology at every diffusion step – following, in spirit, the ERGM view that conditioning generation on structural statistics, rather than treating edges as independent, better captures the dependencies real graphs exhibit. Since $\tau^{ij} = 0$ by definition for any pair with no shared neighbor, such pairs carry no additional information beyond this known zero value, so we restrict the diffusion process to a compact candidate set $\mathcal{Q}$ – observed edges and pairs with $\tau^{ij} > 0$, improving signal-to-noise during training and reducing computational cost. We define:

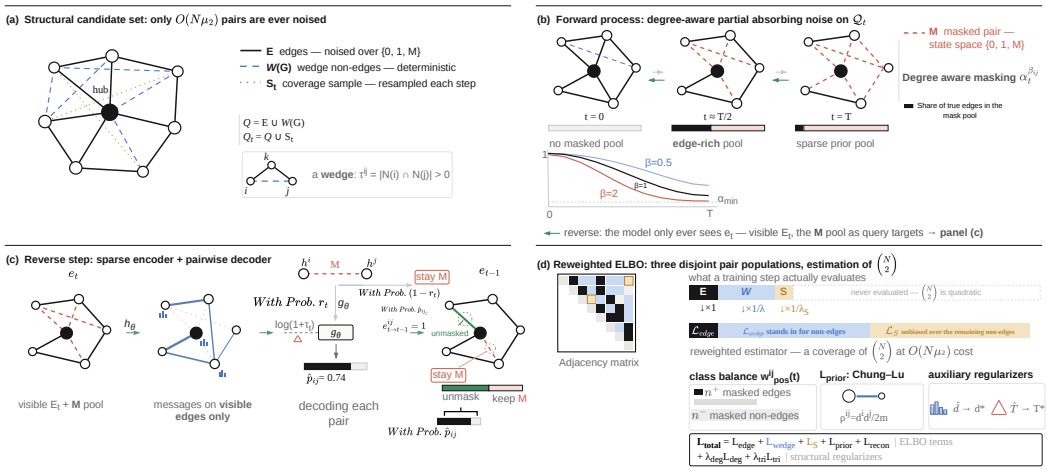


Figure 1: Overview of the proposed method. (a) Diffusion is confined to the structural candidate set. (b) Pairs in Q_t follow a partial absorbing process over $\{0, 1, \mathrm{M}\}$ with a degree-aware masking exponent β^{ij}. (c) The predictor passes messages only over visible edges E_t, then decodes each queried pair independently. (d) The three pair populations are recombined, giving an estimator that covers all $\binom{N}{2}$ pairs at $O(N\mu_2)$ cost per step.

Definition 4.1 (Wedge-Completion Set) *The wedge-completion set is the set of non-edges with at least one common neighbour,*

$$\mathcal{W}(G) = \big\{(i,j) \notin E \;:\; \exists k \text{ s.t. } (i,k) \in E,\, (j,k) \in E\big\}. \tag{2}$$

The structural candidate set is:

$$\mathcal{Q}(G) \;=\; E \;\cup\; \mathcal{W}(G). \tag{3}$$

During training, the candidate set is extended at each step to

$$\mathcal{Q}_t \;:=\; \mathcal{Q} \cup S_t, \tag{4}$$

where S_t is a set of non-edges resampled uniformly at each step t (Eq. 23). Fig. 1 provides an overview of our model.

In the following, we introduce the components of our diffusion model, namely, the forward and reverse process and the objective function. All algorithms, from optimization to sampling, are available in Appendix A.

Forward Process: Partial Absorbing Noise on $\mathcal{Q}_t$. Pairs $(u, v) \in \bar{E}$ and outside $\mathcal{Q}_t$ are fixed to 0 (non-edge) throughout. For pairs $(i, j) \in \mathcal{Q}_t$ the edge state space is $\mathcal{S} = \{0, 1, M\}$, where M is a mask token.

Noise schedule and degree-aware masking. We want the noisy graph to stay sparse and non-empty at every diffusion step, so the reverse process always begins from a partially informative state rather than an empty or fully masked one. Two mechanisms achieve this: restricting training to the candidate set $\mathcal{Q}$ keeps the graph sparse, and a floor $\alpha_{\min} > 0$ on the noise schedule keeps it non-empty, giving the *floored cosine* schedule:

$$\bar{\alpha}_t \;=\; \alpha_{\min} + (1 - \alpha_{\min}) \cos^2\!\left(\frac{\pi}{2} \cdot \frac{t}{T}\right), \quad \alpha_0 = 1, \quad \alpha_T = \alpha_{\min} \in (0, 1). \tag{5}$$

Within this sparse, non-empty pool, we additionally mask high-degree pairs faster than low-degree ones, giving the denoiser more reverse-process steps to recover low-degree connectivity, which a uniform masking rate would otherwise resolve too quickly. For any pair (i, j), we define:

$$\beta^{ij} \;=\; \min\!\left(\beta_{\max},\; \left(\frac{d^{i*}\, d^{j*}}{\bar{d}^{\,2}}\right)^{\gamma/2}\right), \quad \gamma, \beta_{\max} > 0. \tag{6}$$

Since β^{ij} depends only on node degrees, it is cached for $(i, j) \in \mathcal{Q}$ (Alg. 1) and computed on demand for $(i, j) \in S_t$ (Alg. 2).

Forward marginal. The forward marginal distribution is:

$$q(e_t^{(i,j)} = e_0^{(i,j)} \mid e_0^{(i,j)}) \;=\; {\alpha_t}^{\beta^{ij}}, \tag{7}$$

$$q(e_t^{(i,j)} = \mathtt{M} \mid e_0^{(i,j)}) \;=\; 1 - {\alpha_t}^{\beta^{ij}}. \tag{8}$$

The Markov structure gives the one-step transition:

$$q(e_t^{(i,j)} \mid e_{t-1}^{(i,j)}) \;=\; \frac{{\alpha_t}^{\beta^{ij}}}{{\alpha_{t-1}}^{\beta^{ij}}}\,\delta(e_t^{(i,j)}, e_{t-1}^{(i,j)}) + \left(1 - \frac{{\alpha_t}^{\beta^{ij}}}{{\alpha_{t-1}}^{\beta^{ij}}}\right)\delta(e_t^{(i,j)}, \mathtt{M}). \tag{9}$$

Reverse Process and Parametrisation. Following D3PM (Austin et al., 2021), we parametrise the reverse process via a neural network f_θ that outputs, independently for each pair $(i,j) \in \mathcal{Q}_t$, a Bernoulli distribution over the predicted class $\tilde{e}_0^{ij} \in \{1, 0\}$; the mask state is never a target of this prediction, since it is instead governed by the schedule-only unmask probability $r^{ij}_{t \to t_{\text{prev}}}$ below. Thus $f_\theta(\tilde{\mathbf{e}}_0; \mathbf{e}_t, t) = \prod_{(i,j)\in\mathcal{Q}_t} f_\theta(\tilde{e}_0^{ij}; \mathbf{e}_t, t)$ is a valid distribution over the full edge-state vector, and

$$p_\theta(\mathbf{e}_{t-1} \mid \mathbf{e}_t) \;=\; \sum_{\tilde{\mathbf{e}}_0} q(\mathbf{e}_{t-1} \mid \mathbf{e}_t, \tilde{\mathbf{e}}_0)\; f_\theta(\tilde{\mathbf{e}}_0; \mathbf{e}_t, t). \tag{10}$$

Concretely, f_θ decomposes into a sparse graph encoder h_θ and a pairwise decoder g_θ:

$$\mathbf{H}_t \;=\; h_\theta\big(\mathbf{X}_t, E_t, \{p^{ij}_{\text{vis}}\}_{(i,j)\in E_t}, \mathbf{d}^*, \mathbf{d}_t\big) \tag{11}$$

$$\hat{p}_{ij} \;=\; g_\theta\big(\mathbf{h}_t^i, \mathbf{h}_t^j, \log(1 + \tau_t^{ij})\big), \qquad (i,j) \in \mathcal{Q}_t \tag{12}$$

where $E_t = \{(i,j) : e_t^{(i,j)} = 1\}$ is the sparse visible edge set (message passing is restricted to E_t); $p^{ij}_{\text{vis}} \in (0,1)$ weights each message during convolution; $\mathbf{d}^* = \{d^{i*}\}$ is the true target degree sequence, supplied as a permanent per-node conditioning feature (distinct from its role as a regression target in $\mathcal{L}_{\text{deg}}$ below); $\mathbf{d}_t$ is the current visible degree under E_t. During training, pairs outside $\mathcal{Q}_t$ are not queried and remain fixed non-edges throughout.

For the absorbing process the posterior is analytic:

$$q(e_{t_{\text{prev}}}^{(i,j)} \mid e_t^{(i,j)}, e_0^{(i,j)}) \;=\; \begin{cases} r^{ij}_{t\to t_{\text{prev}}}\,\delta(e_0^{(i,j)}) + (1 - r^{ij}_{t\to t_{\text{prev}}})\,\delta(\mathtt{M}) & \text{if } e_t^{(i,j)} = \mathtt{M}, \\ \delta(e_t^{(i,j)}) & \text{if } e_t^{(i,j)} \neq \mathtt{M}, \end{cases} \tag{13}$$

where the unmask probability for a transition from t to t_{prev} is

$$r^{ij}_{t\to t_{\text{prev}}} \;=\; \frac{{\alpha_{t_{\text{prev}}}}^{\beta^{ij}} - {\alpha_t}^{\beta^{ij}}}{1 - {\alpha_t}^{\beta^{ij}}}, \qquad 0 \;\le\; r^{ij}_{t\to t_{\text{prev}}} \;\le\; 1. \tag{14}$$

When $t_{\text{prev}} = t - 1$, Eq. 14 gives the single-step probability; at the final step ($t_{\text{prev}} = 0$, $\alpha_0 = 1$), $r^{ij}_{t\to 0} = 1$, so all remaining masked pairs are decoded. Alg. 4 presents the sampling procedure.

Deriving the loss function. To optimize the parameters of our model, we use the evidence lower bound (ELBO) as our training loss. Since the candidate edge set is restricted to $\mathcal{Q}$, only a subset of non-edges is explicitly considered during training. To obtain an *unbiased* estimate of the ELBO over all $\binom{N}{2}$ node pairs, we additionally draw a random coverage sample S_t of non-edges at each step (Eq. 4) and apply a Horvitz–Thompson (HT) correction, $1/\lambda_S$, since S_t is drawn uniformly at random with a known inclusion probability λ_S. The wedge non-edges in $\mathcal{Q}$, by contrast, are included *deterministically* rather than sampled, so no probabilistic correction applies to them; instead, their aggregate loss is rescaled by $1/\lambda$ (Eq. 23). Summing over E, the λ-rescaled wedge non-edges, and the HT-corrected coverage sample S_t yields a tractable, unbiased estimator of the full ELBO; Alg. 2 summarizes the optimization procedure. To further correct the severe imbalance between edges and non-edges, each cross-entropy term is additionally scaled by a time-dependent edge-presence weight $w^{ij}_{\text{pos}}(t)$ (Eq. 27). Combining both corrections gives the final training objective:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{edge}} + \mathcal{L}_{\text{wedge}} + \mathcal{L}_S + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{recon}} + \lambda_{\text{deg}}\,\mathcal{L}_{\text{deg}} + \lambda_{\text{tri}}\,\mathcal{L}_{\text{tri}} \tag{15}$$

where $\mathcal{L}_{\text{edge}}$, $\mathcal{L}_{\text{wedge}}$, and $\mathcal{L}_S$ are masked cross-entropy losses evaluated over true edges, wedge candidates, and a random coverage sample, respectively:

$$\mathcal{L}_{\text{edge}} = \frac{1}{T}\sum_{t=1}^{T-1} \mathbb{E}_{\mathbf{e}_t|\mathbf{e}_0}\left[\sum_{(i,j)\in E} w^{ij}_{\text{pos}}(t)\, r_t^{ij}\, \mathbf{1}[e_t^{(i,j)}{=}\mathbf{M}]\, \text{CE}\big(e_0^{(i,j)}, \hat{p}_{ij}\big)\right] \tag{16}$$

$$\mathcal{L}_{\text{wedge}} = \frac{1}{T}\sum_{t=1}^{T-1} \mathbb{E}_{\mathbf{e}_t|\mathbf{e}_0}\left[\frac{1}{\lambda}\sum_{(i,j)\in \mathcal{W}} w^{ij}_{\text{pos}}(t)\, r_t^{ij}\, \mathbf{1}[e_t^{(i,j)}{=}\mathbf{M}]\, \text{CE}\big(e_0^{(i,j)}, \hat{p}_{ij}\big)\right] \tag{17}$$

$$\mathcal{L}_S = \frac{1}{T}\sum_{t=1}^{T-1} \mathbb{E}_{S_t}\, \mathbb{E}_{\mathbf{e}_t|\mathbf{e}_0}\left[\frac{1}{\lambda_S}\sum_{(i,j)\in S_t} w^{ij}_{\text{pos}}(t)\, r_t^{ij}\, \mathbf{1}[e_t^{(i,j)}{=}\mathbf{M}]\, \text{CE}\big(e_0^{(i,j)}, \hat{p}_{ij}\big)\right] \tag{18}$$

rescaled by $1/\lambda$ and $1/\lambda_S$ respectively to correct for subsampling, so all three terms estimate the reconstruction loss over the full edge set at comparable scale. $\mathcal{L}_{\text{deg}}$ and $\mathcal{L}_{\text{tri}}$ regularize node degree and triangle count toward their true values d^{i*} and T^* (defined below):

$$\mathcal{L}_{\text{deg}} = \frac{1}{T}\sum_{t=1}^{T-1} \mathbb{E}_{\mathbf{e}_t|\mathbf{e}_0}\left[\frac{1}{N}\sum_{i=1}^{N}\Big(\hat{d}^i(\mathbf{e}_t) - d^{i*}\Big)^2\right] \tag{19}$$

$$\mathcal{L}_{\text{tri}} = \frac{1}{T}\sum_{t=1}^{T-1} \mathbb{E}_{\mathbf{e}_t|\mathbf{e}_0}\left[\left(\frac{\hat{T}_\theta(\mathbf{e}_t) - T^*}{\max(T^*,\,1)}\right)^2\right] \tag{20}$$

with

$$d^{i*} = \sum_{(i,j)\in\mathcal{Q}} e_0^{(i,j)}, \quad \tau_t^{ij} = \big|\mathcal{N}(i;G_t)\cap\mathcal{N}(j;G_t)\big|, \quad T^* = T(G_0) = \frac{1}{3}\sum_{(i,j)\in E}\tau_0^{ij}, \tag{21}$$

the true degree of node i over candidate pairs $\mathcal{Q}$, the common-neighbor count of (i,j) on G_t, and the triangle count of G_0. d^{i*} is shared, unmodified, across all its uses in this section: as an encoder input feature (Eq. 11), as the regression target of $\mathcal{L}_{\text{deg}}$, and as a component of the Chung–Lu prior ρ^{ij} (Eq. 24). The predicted counterparts are

$$\hat{d}^i(\mathbf{e}_t) = \sum_{(i,j)\in\mathcal{Q}_t}\hat{p}_{ij}, \qquad \hat{T}_\theta(\mathbf{e}_t) = \frac{1}{3}\sum_{(i,j)\in\mathcal{Q}_t}\hat{p}_{ij}\,\tau_t^{ij}. \tag{22}$$

λ is the fraction of non-edges falling in the wedge set, and λ_S the known random inclusion rate of the coverage sample S_t:

$$\lambda = \frac{|\mathcal{W}(G)|}{\binom{N}{2} - m} \in (0,1], \qquad \lambda_S = \frac{|S_t|}{\binom{N}{2} - m - |\mathcal{W}(G)|}. \tag{23}$$

Reconstruction term. $\mathcal{L}_{recon} = \sum_{(i,j)\in\mathcal{Q}} w^{ij}\, w^{ij}_{\text{pos}}(0)\ \text{CE}\big(e_0^{(i,j)}, \hat{p}_{ij}\big)$ with $w^{ij} = 1$ for edges and $w^{ij} = 1/\lambda$ for wedge non-edges.

Prior term via Chung-Lu. This term measures the mismatch between the forward marginal at $t = T$ and the Chung–Lu prior used to initialize sampling. L_T has no masked pool and is therefore not scaled by w^{ij}_{pos}.

$$\mathcal{L}_{prior} = \sum_{(i,j)\in\mathcal{Q}} \bar{\alpha}_T^{\beta^{ij}}\ \text{CE}\big(e_0^{(i,j)},\, \rho^{ij}\big), \quad \rho^{ij} = \min\left(1,\ \frac{d^{i*}d^{j*}}{2m}\right) \in [0,1]. \tag{24}$$

Class-balance reweighting. To prevent the vast majority of non-edges from dominating the masked-pool loss, we reweight each term above by the current positive-to-negative class ratio. Let $y^{ij} := e_0^{(i,j)} \in \{0,1\}$ denote the ground-truth edge label, and let

$$\mathbf{1}^{ij}_{\text{pool}}(t) \;:=\; \begin{cases} \mathbf{1}[e_t^{(i,j)} = \mathtt{M}] & t > 0 \\ 1 & t = 0 \end{cases}, \qquad (i,j)\in\mathcal{Q}_t, \tag{25}$$

denote pool membership under the current mask. Since $S_t \subset \mathcal{Q}_t$ consists exclusively of non-edges ($y^{ij} = 0$ for $(i,j) \in S_t$), the positive count reduces to a sum over the static candidate set $\mathcal{Q}$, while the negative count does not (maskable S_t pairs can contribute):

$$n_t^+ = \sum_{(i,j)\in\mathcal{Q}} y^{ij}\, \mathbf{1}^{ij}_{\text{pool}}(t), \qquad n_t^- = \sum_{(i,j)\in\mathcal{Q}_t} (1-y^{ij})\, \mathbf{1}^{ij}_{\text{pool}}(t), \tag{26}$$

$$\rho_t = \frac{n_t^-}{\max(n_t^+,\, 1)}, \qquad w^{ij}_{\text{pos}}(t) = \rho_t\, y^{ij} + (1 - y^{ij}). \tag{27}$$

$w^{ij}_{\text{pos}}(t)$ is recomputed at every sampled t from the current batch and is not cached per graph.

Analysis. Training already restricts each graph to its candidate set $\mathcal{Q}$ rather than the full $\binom{N}{2}$ pairs; controlling the size of the randomly sampled coverage set S_t (Eq. 30) alongside this restriction yields a combined candidate set of size $|\mathcal{Q}_t| = O(N\mu_2)$ (Prop. B.2), where $\mu_2 = \frac{1}{N}\sum_v d_v^2$ is the second moment of the degree sequence — a fraction $O(\mu_2/N)$ of all possible pairs, which vanishes as $N \to \infty$ whenever $\mu_2 = o(N)$. During sampling, in contrast, the reverse process has no access to the ground-truth candidate set $\mathcal{Q}$ — defined from the clean graph G_0 — and must resolve all $\binom{N}{2}$ pairs at each step, introducing an unavoidable quadratic dependency on the number of nodes at sampling time (Table 4); full detail is in Appendix B.

Improving SNR. Restricting training to $\mathcal{Q}_t$ improves the class-mass signal-to-noise ratio (Def. B.6) over training on all $\binom{N}{2}$ pairs: at fixed masking fraction, the unweighted SNR improves from $\Theta(m/N^2)$ to $\Theta(m/N\mu_2)$ under $\mathcal{Q}_t$ (Rem. B.8), a gain of $\Theta(N/\mu_2)$. The positive-class weight $w_b^+(t)$ (Eq. 27) then exactly balances the remaining class imbalance within $\mathcal{Q}_t$, achieving a class-mass ratio of exactly 1 regardless of the pool (Prop. B.7), a further gain of $\Theta(N\mu_2/m)$. Together, restriction and reweighting improve the class-mass SNR by $\Theta(N^2/m) = \Theta(1/\rho)$ over an unweighted, dense-pool baseline, where ρ is the graph's edge density: sparser graphs benefit more (Rem. B.8). The coverage-sampled term S_t is HT corrected, so the full objective remains an unbiased estimate of the full-graph objective overall (Prop. B.3); full derivations are in Appendix B.

5 Experiments

We evaluate our model's performance on graph generation to assess how realistic the generated graphs are and how well they match the distribution of the training data. To check the fidelity of the generated graphs, we consider characteristics related to the structure of the training data and compare them accordingly while also screening for memorization indicators to avoid mere duplication of the training data.

Table 1: Dataset statistics.

Statistic	Polblogs	Bitcoin-Alpha	Bitcoin-OTC	Cora	Citeseer	Pubmed	Community	Ego	ZINC
Nodes	1,222	3,775	5,875	2,485	2,120	19,717	[60, 160]	[50, 399]	[9, 37]
Edges	16,714	14,120	21,489	5,069	3,679	44,324	[231, 1,956]	[56, 1,062]	[8, 41]
Graphs	1	1	1	1	1	1	510	757	12,000

5.1 Datasets and metrics

We consider a diverse range of graph datasets, from citation networks Sen et al. (2008) to financial networks Kumar et al. (2016; 2018); Elmougy & Liu (2023). Table 1 reports dataset statistics; see Appendix E for details.

A generated graph should reproduce the statistical properties of the training distribution while remaining structurally distinct from any individual training graph — i.e., matching the population, not memorizing any instance. We therefore evaluate distributional fidelity alongside explicit memorization checks. For fidelity, we compare generated and input graphs on statistics capturing both local and global structure. To capture local connectivity patterns and neighborhood cohesion, we use

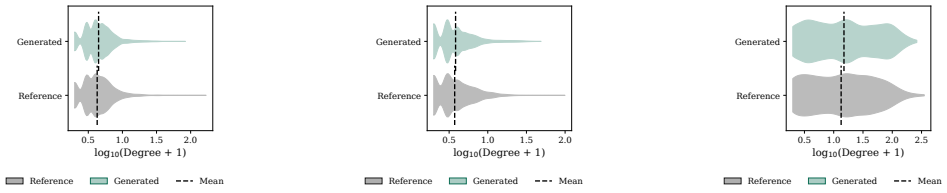
Table 2: Statistics of generated vs. ground-truth graphs. Proximity to ground truth is optimal for all metrics except EO (where ground truth denotes self-overlap). Bold/underlined indicate best/second-best.

	Cora					Citeseer					Pubmed				
	EO↓	PLE	CPL	CC	AC	EO↓	PLE	CPL	CC	AC	EO↓	PLE	CPL	CC	AC
ground truth	100.00	1.89	6.36	0.090	-0.07	100.00	2.07	9.35	0.130	0.01	100.00	2.18	6.31	0.050	-0.04
GraphRNN 2018	0.01	1.77	4.80	0.040	-0.01	0.10	2.30	4.10	0.002	0.23	0.10	1.59	3.30	0.200	0.10
BiGG 2020	10.40	1.50	3.10	0.005	0.10	12.40	1.60	3.40	0.004	0.11	17.30	1.55	3.40	0.350	0.09
EDGE 2023	2.10	1.76	4.90	0.030	-0.04	6.10	1.72	5.80	0.010	0.05	8.20	1.60	4.23	0.220	**0.02**
ARROW-Diff 2025	41.20	1.66	4.80	0.001	**-0.08**	64.50	1.77	5.40	0.062	**0.01**	51.30	1.58	3.27	0.370	0.12
SCR (ours)	**0.00**	**1.83**	**5.10**	**0.076**	0.10	**0.00**	**2.04**	**6.20**	**0.067**	0.35	**0.00**	**1.77**	**4.50**	**0.180**	0.03

	Alpha					OTC					Polblogs				
	EO↓	PLE	CPL	CC	AC	EO↓	PLE	CPL	CC	AC	EO↓	PLE	CPL	CC	AC
ground truth	100.00	1.92	3.58	0.080	-0.17	100.00	1.97	3.60	0.060	-0.16	100.00	1.41	2.74	0.230	-0.22
GraphRNN 2018	0.10	1.10	1.04	0.005	0.12	**0.00**	0.68	1.73	0.170	0.12	1.60	1.26	2.50	0.010	0.10
BiGG 2020	9.70	0.80	1.20	0.001	0.15	10.20	0.70	1.20	0.150	0.13	19.30	1.19	2.20	0.010	0.10
EDGE 2023	3.70	1.10	2.10	0.040	**-0.05**	5.60	0.87	1.90	0.180	0.10	3.10	1.37	**2.60**	**0.200**	-0.18
ARROW-Diff 2025	43.80	1.20	2.05	0.050	0.05	53.70	0.82	2.02	0.230	0.11	61.30	1.32	2.10	0.170	**-0.19**
SCR (ours)	**0.00**	**1.80**	**3.60**	**0.100**	0.10	**0.00**	**1.82**	**3.65**	**0.120**	**0.07**	**0.00**	**1.39**	**2.60**	**0.200**	-0.11

Maximum Mean Discrepancy (MMD) Gretton et al. (2012) over node degree and clustering coefficient distributions, respectively (lower is better). We also compare the power-law exponent (PLE) and characteristic path length (CPL) Chanpuriya et al. (2021), reflecting how well the model preserves scale-free degree behavior and small-world connectivity, along with the assortativity coefficient (AC), which captures the tendency of nodes to connect to others of similar degree. We further visualize the complementary cumulative distribution function (CCDF) of node degree, $P(\text{Degree} + 1 \geq x)$ — obtained by rank-sorting the pooled degree sequence and plotting on log-log axes — to inspect how closely the heavy-tailed degree distribution of generated graphs tracks the training data (Fig. 10).

To rule out memorization, we report edge overlap (EO), since a high value can indicate the model is reproducing training graphs rather than generalizing Chen et al. (2023). For smaller datasets, we additionally report uniqueness and novelty (Tab. 6, 7) quantifying how many generated instances are unique and how many are novel relative to the training data. Implementation details are in Appendix[1] D.



(a) Cora (b) Citeseer (c) Polblogs

Figure 2: Degree distribution comparison between generated and training graphs, restricted to generated samples with zero edge overlap with the training data (i.e., no memorized edges). See Fig. 9 for further datasets.

Results. As reported in table 2, across the four degree- and closure-based statistics (EO, PLE, CPL, CC), our model is consistently the closest or tied-closest to the ground truth on all six datasets. The clustering-coefficient result is worth highlighting specifically: our model is closest or second-closest to ground truth on all six datasets, which is consistent with the wedge-completion set $\mathcal{W}(G)$ and the triangle-count regularizer $\mathcal{L}_{\text{tri}}$ both explicitly targeting local closure structure — though we note this connection is suggestive rather than established, since we do not isolate these components' individual contribution to CC recovery here (see § 5.2). Assortativity is the clear exception to this pattern — performance is uneven across datasets, plausibly because every mechanism we introduce operates on local pairs or neighborhoods, with no component directly targeting this global degree correlation. Visualizing generated large graphs directly is uninformative at scale; instead, we visualize degree distributions alongside the numerical results (Fig. 2). Further results, including those for smaller datasets (Tabs. 6, 7) and empirical sampling time (Fig. 11), are provided in Appendix C.

[1]For baselines, we select efficient models capable of large-graph generation and exclude runs that result in out-of-memory failures.

Table 3: Ablation of Prior Scaffolds and Regularizers. Proximity to Ground Truth is optimal. Bold/underlined indicate best/second-best within each group.

	Cora				Polblogs				Alpha			
Variant	PLE	CPL	CC	AC	PLE	CPL	CC	AC	PLE	CPL	CC	AC
Ground Truth	1.89	6.36	0.090	-0.07	1.41	2.74	0.230	-0.22	1.92	3.58	0.080	-0.17
Prior Scaffold												
SCR (Ours)	**1.83**	**5.10**	**0.076**	0.10	1.39	2.60	0.200	**-0.11**	**1.80**	**3.60**	**0.100**	0.10
w/ Erdős–Rényi	**1.83**	4.87	0.115	0.33	1.32	2.57	0.096	0.33	1.55	3.65	0.640	0.43
w/ SBM	1.81	4.92	0.005	**0.01**	**1.40**	**2.66**	**0.204**	-0.08	1.58	3.09	0.558	0.27
Loss Terms												
SCR (Ours)	**1.83**	**5.10**	**0.076**	**0.10**	1.39	**2.60**	**0.200**	**-0.11**	**1.80**	**3.60**	**0.100**	0.10
w/o $\mathcal{L}_{\text{deg}}$	1.81	5.07	0.032	0.17	**1.40**	2.55	0.181	-0.03	1.47	3.05	0.162	**-0.05**
w/o $\mathcal{L}_{\text{tri}}$	1.77	4.81	0.120	0.34	1.39	2.58	0.177	-0.02	1.66	3.16	0.102	0.13

5.2 Ablation study

Table 3 isolates the contributions of our prior scaffold and regularizers. Extended derivations, hyperparameters, and inference step-reduction analyses are provided in Appendices C.3–C.5. In addition, we validate this SNR improvement empirically in Appendix B.1.

Terminal Prior Alignment. We test whether the terminal scaffold prior must match the inductive bias of our degree-aware masking schedule (β^{ij}). Replacing our degree-based Chung–Lu prior with flat Erdős–Rényi (ER) or a class-based SBM damages topological fidelity across datasets lacking explicit community structure: on Alpha, ER and SBM degrade clustering coefficient (CC) by $5\times$ to $6\times$. SBM only remains competitive on Polblogs, whose binary ideological communities explicitly fit the block-model prior. This confirms that matching the prior to the degree schedule is essential.

Role of Structural Losses. Removing $\mathcal{L}_{\text{deg}}$ systematically deteriorates CC (worsening error by 2–4$\times$), as CC is normalized by node degree and therefore distorted when degree distributions shift. Conversely, ablating $\mathcal{L}_{\text{tri}}$ primarily damages degree assortativity (AC error nearly doubles on Cora and Polblogs). Meanwhile, PLE remains stable under both ablations ($|\Delta_{\text{PLE}}| \leq 0.02$), confirming that the global degree distribution is anchored by the forward schedule β^{ij}, while $\mathcal{L}_{\text{deg}}$ and $\mathcal{L}_{\text{tri}}$ provide localized topological corrections.

Inference Efficiency. Since our reverse posterior (Eq. 14) is well-defined for arbitrary step skips, we can reduce the number of inference steps without modifying the sampling procedure. Empirically (Table 9, Fig. 12), inference steps can be reduced to 50% of the training schedule ($T{=}50\%$) with negligible metric degradation, cutting sampling time in half; performance drops noticeably only at extreme reductions ($T{=}10\%$). Complete curves across datasets appear in Appendix C.3.

6 Conclusion

We introduced a discrete diffusion framework for graph generation that restricts training to a structurally motivated subset of node pairs, together with a small random coverage sample, yielding an unbiased training objective with sub-quadratic training complexity in the number of nodes. A degree-aware forward process keeps the noisy graph structurally informative throughout the reverse process, while a dynamic common-neighbor feature conditions the denoiser on local structure at every noise level. Together, these design choices improve structural fidelity over existing discrete diffusion baselines without memorizing the training graphs. Across datasets of varying size and domain, our method is designed for, and achieves the best or second-best fidelity on, large graphs, while remaining comparable to baselines built specifically for small or molecular graphs.

Limitations. Our model does not achieve ideal performance on degree assortativity, a metric that captures whether nodes tend to connect to others of similar degree. This is not in tension with our model's degree-awareness: our mechanisms condition on each node's own degree individually, whereas assortativity reflects a correlation across connected pairs, a distinct, relational property that none of our mechanisms directly targets. In addition, our complexity gains apply to training; sampling remains quadratic in the number of nodes. Addressing both is a natural direction for future work.

Ethics Statement. This work presents a diffusion-based method for generating large graphs, using only public benchmark datasets and no human or sensitive data[2].

Reproducibility statement. We use publicly available datasets, detailed in Appendix E. Full definitions of our evaluation metrics are provided in Appendix E.1, alongside implementation and hyperparameter details in Appendix D. Proofs of all propositions stated in the main text are provided in Appendix B. Finally, our code is available in this anonymized repository to support reproducibility.

[2]AI (LLMs) were used exclusively for writing polishing.

REFERENCES

Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43. ACM, 2005.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

Sirine Ayadi, Leon Hetzel, Johanna Sommer, Fabian J. Theis, and Stephan Günnemann. Unified guidance for geometry-conditioned molecular generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/faa6276ea12d7afeb3e42b210c86f688-Abstract-Conference.html.

Tobias Bernecker, Ghalia Rehawi, Francesco Paolo Casale, Janine Knauer-Arloth, and Annalisa Marsico. Random walk diffusion for efficient large-scale graph generation. *Trans. Mach. Learn. Res.*, 2025. URL https://openreview.net/forum?id=tSFpsfndE7.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International conference on machine learning*, pp. 610–619. PMLR, 2018.

Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. On the power of edge independent graph models. *Advances in Neural Information Processing Systems*, 34:24418–24429, 2021.

Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4585–4610. PMLR, 2023. URL https://proceedings.mlr.press/v202/chen23k.html.

Fan RK Chung and Linyuan Lu. *Complex graphs and networks*. Number 107. American Mathematical Soc., 2006.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, 2009. doi: 10.1137/070710111. URL https://doi.org/10.1137/070710111.

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

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2302–2312. PMLR, 2020. URL http://proceedings.mlr.press/v119/dai20b.html.

Youssef Elmougy and Ling Liu. Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics. *arXiv preprint arXiv:2306.06108*, 2023.

Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

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

Ove Frank and David Strauss. Markov graphs. *Journal of the american Statistical association*, 81 (395):832–842, 1986.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022.

Tomaž Hočevar and Janez Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

John J. Irwin, T. Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52:1757 – 1768, 2012. URL https://api.semanticscholar.org/CorpusID:9759396.

Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Graph generation with k2-trees. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=RIEW6M9YoV.

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

Mahdi Karami. Higen: Hierarchical graph generative networks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=KNvubydSB5.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 221–230. IEEE, 2016.

Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 333–341. ACM, 2018.

Mufei Li, Eleonora Kreacic, Vamsi K. Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs? *Trans. Mach. Learn. Res.*, 2024. URL https://openreview.net/forum?id=0q4zjGMKoA.

Zhanfeng Mo, Tianze Luo, and Sinno Jialin Pan. Graph principal flow network for conditional graph generation. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pp. 768–779. ACM, 2024. doi: 10.1145/3589334.3645570. URL https://doi.org/10.1145/3589334.3645570.

Mark EJ Newman. Mixing patterns in networks. *Physical review E*, 67(2):026126, 2003.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International conference on artificial intelligence and statistics*, pp. 4474–4484. PMLR, 2020.

Yiming Qin, Clement Vignac, and Pascal Frossard. Sparse training of discrete diffusion models for graph generation. *arXiv preprint arXiv:2311.02142*, 2023.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W. Taylor. On evaluation metrics for graph generative models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Yu Wang, Ryan A. Rossi, Namyong Park, Huiyuan Chen, Nesreen K. Ahmed, Puja Trivedi, Franck Dernoncourt, Danai Koutra, and Tyler Derr. A large-scale training paradigm for graph generative models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=c01YB8pF0s.

Lingfeng Wen, Xuan Tang, Mingjie Ouyang, Xiangxiang Shen, Jian Yang, Daxin Zhu, Mingsong Chen, and Xian Wei. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 15823–15831. AAAI Press, 2024. doi: 10.1609/AAAI.V38I14.29512. URL https://doi.org/10.1609/aaai.v38i14.29512.

Richard Williams, Eric T. Nalisnick, and Andrew Holbrook. Scalable generative modeling of weighted graphs. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=yWKkBOcD18.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/91813e5ddd9658b99be4c532e274b49c-Abstract-Conference.html.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Yuning You, Ruida Zhou, Jiwoong Park, Haotian Xu, Chao Tian, Zhangyang Wang, and Yang Shen. Latent 3d graph diffusion. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=cXbnGtO0NZ.

APPENDIX OVERVIEW

CONTENTS

A Training and Sampling Algorithms **15**

B Analysis and Discussions **16**

B.1 Practical analysis 21

C Extended Results **25**

C.1 Results on Small Graphs 26

C.2 Empirical sampling time 27

C.3 Efficiency of sampling: reduced diffusion steps 27

C.4 Effect of the prior scaffold. 29

C.5 Effect of the regularizers. 29

D Implementation Details **30**

E Datasets **31**

E.1 Evaluation Metrics 32

A TRAINING AND SAMPLING ALGORITHMS

Here we provide the pre-computation 1, optimization during training 2, graph initialization during sampling 3 and sampling steps 4 algorithms.

Algorithm 1 Pre-computation (once per graph dataset)

Require: $G = (X_0, E_0) \sim Dataset$
1: Compute degree sequence $\mathbf{d}^*$, triangle count T^*, and the candidate set $\mathcal{Q} = E_0 \cup \mathcal{W}(G)$ ▷ Eq. 3
2: Compute masking exponents $\{\beta^{ij}\}$, HT weights $\{w^{ij}\}$, and Chung-Lu parameters $\{\rho^{ij}\}$ for $(i,j) \in \mathcal{Q}$ ▷ Eqs. 6,24
3: $L_T \leftarrow$ prior term (Eq. 24) ▷ θ- and t-independent scalar, cached once
4: **return** $\mathcal{Q},\ \mathbf{d}^*,\ T^*,\ \{\beta^{ij}\},\ \{w^{ij}\},\ \{\rho^{ij}\},\ L_T$

Algorithm 2 Optimization step of $\mathcal{L}(\theta)$, Eq. equation 15

Require: $\mathcal{Q}, \mathbf{d}^*, T^*, \{\beta^{ij}\}, \{w^{ij}\}, L_T$ from Alg. 1; model $f_\theta = (h_\theta, g_\theta)$
1: $t \sim \{1, ..., \mathrm{T}\}$
2: Sample coverage set $S_t \subseteq \mathcal{Q}^c$ of size $\lceil \lambda_S \, |\mathcal{Q}^c| \rceil$; set $\mathcal{Q}_t \leftarrow \mathcal{Q} \cup S_t$ ▷ Eq. 4
3: $G_t \leftarrow q(G_t|G_0)$
4: Compute $\{\tau_t^{ij}\}$ on $\mathcal{Q}_t$ (Eq. 21)
5: $\mathbf{H}_t \leftarrow h_\theta(G_t)$ ▷ sparse encoder, message passing restricted to visible edges E_t, Eq. 11
6: $\hat{p}_{ij} \leftarrow g_\theta(h_t^i, h_t^j, \log(1+\tau_t^{ij}))$ for $(i,j) \in \mathcal{Q}_t$ ▷ pairwise decoder, Eq. 12
7: Compute the class-balance weight $w_b^+(t)$ ▷ Eq. 27
8: Compute $\mathcal{L}_\theta$ ▷ Eq. 15
9: $\mathcal{L}_\theta$.backward()

Sampling. Since β^{ij}, the encoder, and the prior all require node degrees, generation is conditioned on a degree sequence: we draw a training graph uniformly at random and take its size N and degree sequence $\mathbf{d}^* = \{d^{i*}\}$, which stay fixed for the whole reverse chain and replace the true degrees in Eqs. 6, 11, and 24. The edges are generated by the model, so $\mathbf{d}^*$ acts as a target that the generated graph follows in expectation rather than exactly. The chain starts from the partial-absorbing Chung–Lu prior, drawn independently for every pair (i,j), $i<j$:

$$p\big(e_T^{(i,j)} = \mathtt{M}\big) = 1 - \alpha_{\min}^{\beta^{ij}}, \qquad p\big(e_T^{(i,j)} = 1\big) = \alpha_{\min}^{\beta^{ij}}\,\rho^{ij}, \qquad p\big(e_T^{(i,j)} = 0\big) = \alpha_{\min}^{\beta^{ij}}\,\big(1-\rho^{ij}\big), \tag{28}$$

with ρ^{ij} as in Eq. 24. Each pair is thus visible with the same probability as under the forward marginal at $t = T$ (Eq. 7), and its visible state is drawn from Chung–Lu; accordingly, the KL between the forward marginal at T and this prior is, per pair, $\alpha_{\min}^{\beta^{ij}}\ \mathrm{CE}\big(e_0^{(i,j)}, \rho^{ij}\big)$, the summand of $\mathcal{L}_{\text{prior}}$. We then run the reverse process along a decreasing step sequence $T = t_K > t_{K-1} > \cdots > t_0 = 0$, $K \le T$, which Eq. 14 permits for any $t_{\text{prev}} < t$. At each step, τ_t^{ij} and $\mathbf{d}_t$ are recomputed from the visible edges E_t, and every masked pair is revealed with probability $r_{t\to t_{\text{prev}}}^{ij}$ and then set to $\tilde{e}_0^{ij} \sim \mathrm{Bernoulli}(\hat{p}_{ij})$, while visible pairs keep their state (Eq. 13). At the last step, $r_{t\to 0}^{ij} = 1$ decodes all remaining masked pairs, and the generated graph has edge set $\{(i,j) : e_0^{(i,j)} = 1\}$. Unlike training, sampling cannot restrict the queries to $\mathcal{Q}$, which is defined by the unknown clean graph. A dynamic set built from the visible edges would miss hub pairs, which have β^{ij} close to $\beta_{\max}$, are almost surely masked at $t = T$, and rarely share a visible neighbor early in the chain. The denoiser therefore scores all $\binom{N}{2}$ pairs at every step, evaluated in chunks (Algs. 3 and 4).

Algorithm 3 Sampling initialization ($t = T$, once per generated graph)

1: Sample graph size N and degree sequence $\mathbf{d}^*$ from the empirical training distribution
2: Compute $\{\beta^{ij}\}$ and $\{\rho^{ij}\}$ for all $\binom{N}{2}$ pairs (Eqs. 6,24)
3: Sample E_T: each pair is visible w.p. $\alpha_{\min}^{\beta^{ij}}$, with visible edges drawn Chung-Lu Bernoulli(ρ^{ij}); all other pairs are masked
4: **return** $E_T,\ \{\beta^{ij}\},\ \{\rho^{ij}\}$

Algorithm 4 Reverse step $t \to t_{\text{prev}}$

Require: current state E_t; $\{\beta^{ij}\}$; model $f_\theta = (h_\theta, g_\theta)$
1: compute $\{\tau_t^{ij}\}$ for all pairs
2: $\mathbf{H}_t \leftarrow h_\theta(G_t)$
3: $\hat{p}_{ij} \leftarrow g_\theta(\mathbf{h}_i, \mathbf{h}_j, \log(1+\tau_t^{ij}))$ for all pairs ▷ query set is every pair, since sampling must decide each pair's final state.
4: **for** each $(i,j) \in \mathcal{Q}_t$ with $e_t^{ij} = \texttt{M}$ **do**
5: Compute $r_{t\to t-1}^{ij}$ via Eq.14
6: With prob. $r_{t\to t-1}^{ij}$: $e_{t-1}^{ij} \sim \text{Categorical}(\hat{p}_{ij})$; else $e_{t-1}^{ij} \leftarrow \texttt{M}$
7: **end for**
8: **return** $E_{t_{\text{prev}}}$ ▷ after $t-1=-1$: output edge set $= \{(i,j) : e_0^{(i,j)} = 1\}$

B Analysis and Discussions

As discussed above, by controlling the size of the randomly sampled coverage set S_t (Eq. 30), our model operates on a sub-quadratic (in N) fraction of all possible node pairs during training — $O(N\mu_2)$ rather than $O(N^2)$ — enabling scalability to large sparse graphs. During sampling, by contrast, the model must evaluate the state of every potential edge at each reverse step, since the reverse process has no access to the ground-truth candidate set $\mathcal{Q}$ and must therefore resolve all $\binom{N}{2}$ pairs; this introduces an unavoidable quadratic dependency on the number of nodes at sampling time (Table 4).

As shown in Tables 4 and 5, our training complexity compares favorably with EDGE (Chen et al., 2023) in a specific sense: our bound $O(N\mu_2)$ depends only on static graph structure and holds unconditionally by construction, whereas EDGE's bound $O(\max(m, K^2))$ depends on the number of "active" nodes K at each step, a stochastic, schedule-dependent quantity rather than a proven general bound. In this sense, our training-time efficiency is a guaranteed worst-case property of the graph, while EDGE's is an empirical regularity of its noise schedule. The comparison reverses at sampling time, however: EDGE's active-node restriction applies to its reverse process as well, plausibly preserving subquadratic sampling cost (to the extent its empirical $K \ll N$ observation holds), while our sampling procedure — lacking an analogous restriction — incurs the full $O(N^2)$ cost of dense diffusion baselines such as DiGress at every reverse step. This training/sampling asymmetry is a genuine limitation of the current method relative to EDGE. The asymmetry follows from a structural difference in convergent distributions: EDGE's forward process converges to the literal empty graph $G(N,0)$, which makes "active" (currently changing) nodes well-defined and lets the reverse process monotonically grow the graph from nothing, decoding only those nodes at each step. Our forward process instead uses a *partial* absorbing prior, $\alpha_{\min} \in (0,1)$ (Eq. 5), which does not converge to a fully empty or fully masked state; this choice is motivated by avoiding the SNR collapse of sparse graphs at the absorbing limit and by training convergence considerations, but it forfeits the empty-graph-anchored monotonic restriction that EDGE exploits. Recovering a comparable sampling-time restriction under a partial absorbing prior — e.g., restricting decoding to the currently masked pool, which shrinks monotonically over the reverse schedule even though it starts non-empty — is a natural direction for future work, left outside the scope of the current formulation.

Dependence on degree heterogeneity. Unlike SparsDiff, which sets $\lambda := m/N^2$ so its query count $\lambda N^2 = m$ is $O(m)$ unconditionally, our per-step cost $O(m + N\mu_2)$ depends on the second moment of the degree distribution, $\mu_2 := \frac{1}{N}\sum_v d_v^2$, since $\mathcal{W}(G)$ is enumerated deterministically rather than sampled. For each node v, computing its wedges costs $O(d_v^2)$, giving $|\mathcal{W}(G)| = O\big(\sum_v d_v^2\big) = O(N\mu_2)$ in total. This term reduces to $O(m)$ exactly when the condition

$$\mu_2 \;=\; O(\bar{d}), \qquad \bar{d} := \frac{2m}{N}, \tag{29}$$

holds, i.e., when the second moment of the degree distribution is of the same order as the mean degree. A sufficient condition for Eq. 29 to hold is a bounded maximum degree, $\Delta = O(1)$, since then $\mu_2 = \frac{1}{N}\sum_v d_v^2 \le \Delta\bar{d} = O(\bar{d})$. This holds for many sparse real-world graphs but not all. A star graph, for instance, has $m = N-1 = O(N)$ edges, yet $|\mathcal{W}(G)| = \binom{N-1}{2} = O(N^2)$, since

Table 4: Per-step complexity, diffusion-based graph generative models only. Assumes constant per-call network cost, so candidate-pair count and per-step training complexity coincide. N is number of nodes and m is number of edges. Bold denotes complexity improved by our method relative to the diffusion baselines.

Method	Training (per step)	Sampling (per step)
DiGress	$O(N^2)$	$O(N^2)$
SparseDiff	$O(m)$[a]	$O(N^2)$[b]
EDGE	$O(\max(m, K^2))$[c]	$O(\max(m, K^2))$[c]
Ours	$\boldsymbol{O(N\mu_2)}$[d]	$O(N^2)$

[a] By explicit design choice $\lambda = m/N^2$ (Qin et al., 2023).
[b] Chunked for memory only; total compute per step still touches all $\binom{N}{2}$ pairs.
[c] Chen et al. (2023): the per-step term shown here is the summand already implicit in their stated total $O(T\max(K^2, m))$ over T steps (i.e., total $= T\times$ per-step). K = number of "active" nodes.
[d] $\mu_2 = \frac{1}{N}\sum_v d_v^2$, the mean squared degree; by the explicit choice of λ_S (Eq. 30).

Table 5: Comparison of autoregressive (AR) graph generation methods. Complexity values are adopted from the original papers for consistency. N is number of nodes and m is number of edges. Note that "per-step" here denotes one autoregressive generation step (row/node/token), whose number scales with N or m, which is not directly comparable to the fixed-T diffusion timestep used as "per-step" in Table 4.

Method	Paradigm	Generation Process	Overall Complexity	Per-step Complexity
GraphRNN	Flat AR	Row-by-row	$O(N^2)$[a]	$O(N)$
BiGG	Flat AR (tree)	Tree traversal	$O((N+m)\log N)$	$O(1)$
HiGen	Hierarchical AR	Cluster-wise	$O(n_c \log^2 N)$[b]	$O(N^2)$[b]
HGGT (K^2-trees)	Compressed AR	Sequence	$O(L_{\text{seq}}^2)$[c]	$O(L_{\text{seq}}^2)$[c]

[a] N row-emission steps $\times$ $O(N)$ per step.
[b] Karami (2024). n_c = number of clusters. $O(N^2)$ full attention or $O(N+m)$ linear attention for larger graphs; Per-step and Overall coincide with the full-attention variant here.
[c] $L_{\text{seq}} = O(m\log(N^2/m))$ is the flattened K^2-tree sequence length. Per-step and Overall coincide since generation is a single-pass sequence operation rather than one cost repeated over N or m steps.

every leaf pair shares the hub as a common neighbor. This shows that sparsity in m alone does not guarantee $O(m)$ cost under high degree heterogeneity. Accordingly, since $2m \le N\mu_2$ always holds (as $d_v^2 \ge d_v$ for integer degrees), we report our complexity as $O(N\mu_2)$ throughout (Table 4): this is the exact asymptotic order in all cases – including those where it coincides with $O(m)$ – and one that scales with the structural complexity that drives the cost of deterministic wedge enumeration, rather than with m alone.

Choice of λ_S. To guarantee the coverage sample does not reintroduce quadratic cost, and is not an arbitrary hyperparameter, we fix

$$\lambda_S \;:=\; \frac{c\,N\mu_2}{\binom{N}{2} - m - |\mathcal{W}(G)|}, \qquad \mu_2 = \frac{1}{N}\sum_v d_v^2, \;\; c > 0 \text{ a constant}, \tag{30}$$

so that $|S_t| = \left\lceil \lambda_S\left(\binom{N}{2} - m - |\mathcal{W}(G)|\right)\right\rceil = O(N\mu_2)$ by construction — matching the scale of $|\mathcal{Q}| = O(N\mu_2)$ rather than being an independent hyperparameter. This mirrors SparseDiff's choice $\lambda = m/N^2$ (Qin et al., 2023), which analogously pins their query-edge count to $O(m)$. Under Eq. 30, training's per-step complexity (Table 4) is $O(N\mu_2)$, not merely bounded in expectation but by construction. In practice, we select c per dataset to hold the realized per-step tensor footprint, batch $\times$ dim $\times$ $(|\mathcal{Q}| + |S_t|)$, within a fixed memory budget: c decreases as μ_2 (and hence $|\mathcal{Q}| = m + |\mathcal{W}(G)|$) grows, since a larger $\mathcal{Q}$ leaves less headroom for S_t before training reintroduces quadratic-scale cost or exhausts GPU memory. This is a complexity/memory safety margin, not a quality-tuning knob — c is chosen to be the largest value that keeps $|\mathcal{Q}| + |S_t|$ within budget, not selected by any generation-quality criterion.

Lemma B.1 (Wedge Count Bound) *Let $G = (V, E)$ be a graph on N nodes with edge count $m = |E|$, degree sequence $\{d_v\}_{v \in V}$, and second moment $\mu_2 = \frac{1}{N}\sum_v d_v^2$. Then*

$$|\mathcal{W}(G)| \;\le\; \frac{1}{2}\big(N\mu_2 - 2m\big). \tag{31}$$

For a fixed node $k \in V$, the number of unordered node pairs (i, j) with k as a common neighbour is $\binom{d_k}{2}$, since this counts the ways to choose two distinct neighbours of k from its neighbourhood of size d_k. Summing over all nodes k counts each pair (i, j) once for every common neighbour it has, giving the double-counting identity

$$\sum_{k=1}^{N} \binom{d_k}{2} \;=\; \sum_{(i,j)} \big|\mathcal{N}(i) \cap \mathcal{N}(j)\big|. \tag{32}$$

Restricting the right-hand sum in equation 32 to non-edges only removes non-negative terms, and replacing each remaining count $|\mathcal{N}(i) \cap \mathcal{N}(j)| \ge 1$ by the indicator $\mathbf{1}[\cdot \ge 1]$ only decreases each term further. Hence

$$\sum_{k=1}^{N} \binom{d_k}{2} \;\ge\; \sum_{(i,j)\notin E} \mathbf{1}\big[|\mathcal{N}(i) \cap \mathcal{N}(j)| \ge 1\big] \;=\; |\mathcal{W}(G)|, \tag{33}$$

where the final equality is Definition 4.1 of $\mathcal{W}(G)$. Expanding the left-hand side of equation 33,

$$\sum_{k=1}^{N} \binom{d_k}{2} \;=\; \frac{1}{2}\sum_{k=1}^{N} d_k(d_k - 1) \;=\; \frac{1}{2}\Big(\sum_{k=1}^{N} d_k^2 - \sum_{k=1}^{N} d_k\Big) \;=\; \frac{1}{2}\big(N\mu_2 - 2m\big), \tag{34}$$

using $\sum_k d_k^2 = N\mu_2$ and the handshake identity $\sum_k d_k = 2m$. Combining equation 33 and equation 34 gives equation 31.

Proposition B.2 (Coverage Bound) *Let G be a graph on N nodes with m edges and degree second moment $\mu_2 = \frac{1}{N}\sum_v d_v^2$, and let $\mathcal{Q}_t = E \cup \mathcal{W}(G) \cup S_t$ be the training-time candidate set of Eq. 4, with $|S_t| = \lceil cN\mu_2 \rceil$ fixed by the choice of λ_S (Eq. 30) for a constant $c > 0$. Then*

$$|\mathcal{Q}_t| \;\le\; \Big(\tfrac{1}{2} + c\Big)N\mu_2 + 1 \;=\; O(N\mu_2), \tag{35}$$

and consequently the fraction of all $\binom{N}{2}$ node pairs evaluated at each training step satisfies

$$\frac{|\mathcal{Q}_t|}{\binom{N}{2}} \;=\; O\Big(\frac{\mu_2}{N}\Big), \tag{36}$$

which vanishes as $N \to \infty$ whenever $\mu_2 = O(N)$.

By Definition 4.1, $\mathcal{Q} = E \cup \mathcal{W}(G)$ with $E \cap \mathcal{W}(G) = \emptyset$, so $|\mathcal{Q}| = m + |\mathcal{W}(G)|$. Applying Lemma B.1,

$$|\mathcal{Q}| \;=\; m + |\mathcal{W}(G)| \;\le\; m + \frac{1}{2}\big(N\mu_2 - 2m\big) \;=\; \frac{1}{2}N\mu_2. \tag{37}$$

By construction, $|S_t| = \lceil cN\mu_2 \rceil \le cN\mu_2 + 1$ holds deterministically for every realization of S_t, since λ_S fixes the sample *size*, not merely its expectation. Since $\mathcal{Q}_t = \mathcal{Q} \cup S_t$ with $S_t \subseteq \bar{E} \setminus \mathcal{W}(G)$ disjoint from $\mathcal{Q}$ (Eq. 4),

$$|\mathcal{Q}_t| \;=\; |\mathcal{Q}| + |S_t| \;\le\; \frac{1}{2}N\mu_2 + cN\mu_2 + 1 \;=\; \Big(\tfrac{1}{2} + c\Big)N\mu_2 + 1, \tag{38}$$

which is equation 35. Dividing by $\binom{N}{2} = \Theta(N^2)$ gives

$$\frac{|\mathcal{Q}_t|}{\binom{N}{2}} \;=\; O\Big(\frac{N\mu_2}{N^2}\Big) \;=\; O\Big(\frac{\mu_2}{N}\Big), \tag{39}$$

which is equation 36. This ratio tends to 0 as $N \to \infty$ precisely when $\mu_2 = O(N)$.

Proposition B.3 (Unbiasedness of the Reweighted ELBO) *Fix a diffusion step $t \in \{1, \dots, T-1\}$ and a clean graph $\mathbf{e}_0$. Let*

$$\hat{L}(t) \;=\; \underbrace{\sum_{(i,j)\in E} \ell_{ij}(t)}_{\textit{edge term}} \;+\; \underbrace{\sum_{(i,j)\in \mathcal{W}(G)} \ell_{ij}(t)}_{\textit{wedge term}} \;+\; \underbrace{\frac{1}{\lambda_S} \sum_{(i,j)\in S_t} \ell_{ij}(t)}_{\textit{coverage term}}, \tag{40}$$

where $\ell_{ij}(t) := w^{ij}_{\mathrm{pos}}(t)\, r^{ij}_t\, \mathbf{1}[e^{(i,j)}_t = \mathbb{M}]\ \mathrm{CE}\big(e^{(i,j)}_0, \hat{p}_{ij}\big)$, S_t is drawn uniformly without replacement from $\bar{E} \setminus \mathcal{W}(G)$ with $|S_t| = \lambda_S\big(\binom{N}{2} - m - |\mathcal{W}(G)|\big)$ (Eq. 23), and E, $\mathcal{W}(G)$ are deterministic given G. Let

$$L_{\mathrm{full}}(t) \;:=\; \sum_{(i,j)} \ell_{ij}(t) \tag{41}$$

denote the same loss summed over all $\binom{N}{2}$ node pairs. Then $\hat{L}(t)$ is an unbiased estimator of $L_{\mathrm{full}}(t)$ with respect to the randomness in S_t:

$$\mathbb{E}_{S_t}\big[\hat{L}(t)\big] \;=\; L_{\mathrm{full}}(t). \tag{42}$$

The set of all $\binom{N}{2}$ node pairs decomposes into three disjoint parts:

$$\{(i,j)\} \;=\; E \;\cup\; \mathcal{W}(G) \;\cup\; \big(\bar{E} \setminus \mathcal{W}(G)\big), \tag{43}$$

so $L_{\mathrm{full}}(t)$ splits additively into the corresponding three sums, and by linearity it suffices to match each term of $\hat{L}(t)$ in equation 40 to the corresponding part of $L_{\mathrm{full}}(t)$ in expectation.

Edge term. E is fixed given G and does not depend on S_t, so

$$\mathbb{E}_{S_t}\Big[\sum_{E} \ell_{ij}(t)\Big] \;=\; \sum_{E} \ell_{ij}(t), \tag{44}$$

which equals the edge part of $L_{\mathrm{full}}(t)$ exactly.

Wedge term. Likewise, $\mathcal{W}(G)$ is fixed given G (Definition 4.1) and does not depend on S_t, so

$$\mathbb{E}_{S_t}\Big[\sum_{\mathcal{W}(G)} \ell_{ij}(t)\Big] \;=\; \sum_{\mathcal{W}(G)} \ell_{ij}(t), \tag{45}$$

which equals the wedge part of $L_{\mathrm{full}}(t)$ exactly.

Coverage term. S_t is drawn uniformly without replacement from the population $\bar{E} \setminus \mathcal{W}(G)$, so every pair in this population has identical inclusion probability:

$$\Pr\big[(i,j) \in S_t\big] \;=\; \frac{|S_t|}{|\bar{E} \setminus \mathcal{W}(G)|} \;=\; \lambda_S, \qquad \forall\, (i,j) \in \bar{E} \setminus \mathcal{W}(G). \tag{46}$$

Writing the coverage sum via inclusion indicators $\mathbf{1}[(i,j) \in S_t]$ and applying linearity of expectation together with equation 46,

$$\mathbb{E}_{S_t}\Big[\sum_{S_t} \ell_{ij}(t)\Big] = \mathbb{E}_{S_t}\Big[\sum_{(i,j)\in \bar{E}\setminus\mathcal{W}(G)} \mathbf{1}[(i,j) \in S_t]\, \ell_{ij}(t)\Big] \tag{47}$$

$$= \sum_{(i,j)\in \bar{E}\setminus\mathcal{W}(G)} \Pr\big[(i,j) \in S_t\big]\, \ell_{ij}(t) \;=\; \lambda_S \sum_{\bar{E}\setminus\mathcal{W}(G)} \ell_{ij}(t). \tag{48}$$

Hence the Horvitz–Thompson-corrected coverage term satisfies

$$\mathbb{E}_{S_t}\Big[\frac{1}{\lambda_S} \sum_{S_t} \ell_{ij}(t)\Big] \;=\; \frac{1}{\lambda_S} \cdot \lambda_S \sum_{\bar{E}\setminus\mathcal{W}(G)} \ell_{ij}(t) \;=\; \sum_{\bar{E}\setminus\mathcal{W}(G)} \ell_{ij}(t), \tag{49}$$

which equals the remaining part of $L_{\mathrm{full}}(t)$ exactly.

Summing the three matched terms and using the partition equation 43 gives $\mathbb{E}_{S_t}[\hat{L}(t)] = L_{\mathrm{full}}(t)$, proving equation 42.

Proposition B.4 (Bias of the λ-Weighted Wedge Term) *Suppose the wedge term of the ELBO is weighted by $1/\lambda$, i.e.*

$$\hat{L}_\lambda(t) \;=\; \sum_{(i,j)\in E} \ell_{ij}(t) \;+\; \frac{1}{\lambda} \sum_{(i,j)\in \mathcal{W}(G)} \ell_{ij}(t) \;+\; \frac{1}{\lambda_S} \sum_{(i,j)\in S_t} \ell_{ij}(t), \tag{50}$$

with $\lambda = |\mathcal{W}(G)|/(\binom{N}{2} - m)$ as in Eq. 23, and let $L_{\text{full}}(t)$ denote the full-graph loss as in Proposition B.3. Then, with respect to the randomness in S_t,

$$\mathbb{E}_{S_t}\big[\hat{L}_\lambda(t)\big] \;=\; L_{\text{full}}(t) \;+\; \text{Bias}(t), \qquad \text{Bias}(t) \;=\; \frac{1-\lambda}{\lambda} \sum_{(i,j)\in \mathcal{W}(G)} \ell_{ij}(t). \tag{51}$$

The edge and coverage terms of $\hat{L}_\lambda(t)$ are identical to $\hat{L}(t)$ in Proposition B.3, so their expectations match the corresponding parts of $L_{\text{full}}(t)$ exactly, as shown there. Only the wedge term differs, by the constant factor $1/\lambda$:

$$\mathbb{E}_{S_t}\Big[\frac{1}{\lambda} \sum_{\mathcal{W}(G)} \ell_{ij}(t)\Big] \;=\; \frac{1}{\lambda} \sum_{\mathcal{W}(G)} \ell_{ij}(t) \;=\; \sum_{\mathcal{W}(G)} \ell_{ij}(t) \;+\; \Big(\frac{1}{\lambda} - 1\Big) \sum_{\mathcal{W}(G)} \ell_{ij}(t), \tag{52}$$

since $\mathcal{W}(G)$ is deterministic and does not depend on S_t. The first term on the right matches the wedge part of $L_{\text{full}}(t)$ exactly; the second term is the excess, giving $\text{Bias}(t) = \big(\frac{1}{\lambda} - 1\big) \sum_{\mathcal{W}(G)} \ell_{ij}(t) = \frac{1-\lambda}{\lambda} \sum_{\mathcal{W}(G)} \ell_{ij}(t)$. Summing all three terms and comparing to $L_{\text{full}}(t)$ gives equation 51.

Remark B.5 (Order of the Bias) *Write $\bar{\ell}_{\mathcal{W}}(t) := \frac{1}{|\mathcal{W}(G)|} \sum_{\mathcal{W}(G)} \ell_{ij}(t)$ for the average per-pair wedge loss at step t (assuming $\mathcal{W}(G) \neq \emptyset$). By definition of λ,*

$$\frac{1}{\lambda}\,|\mathcal{W}(G)| \;=\; \binom{N}{2} - m, \tag{53}$$

so the bias in equation 51 simplifies exactly to

$$\text{Bias}(t) \;=\; (1-\lambda)\Big(\binom{N}{2} - m\Big)\,\bar{\ell}_{\mathcal{W}}(t). \tag{54}$$

By Proposition B.2, $\lambda = O(\mu_2/N) \to 0$ as $N \to \infty$ whenever $\mu_2 = o(N)$ (the same regime in which coverage stays sub-quadratic), so $(1-\lambda) \to 1$ and

$$\text{Bias}(t) \;=\; \Theta\big(N^2\big) \cdot \bar{\ell}_{\mathcal{W}}(t) \qquad \text{as } N \to \infty. \tag{55}$$

This is the *same asymptotic order*, $\Theta(N^2)$, as $L_{\text{full}}(t)$'s own coverage-population part $\sum_{\bar{E}\setminus\mathcal{W}(G)} \ell_{ij}(t)$, which sums over $\binom{N}{2} - m - |\mathcal{W}(G)| = \Theta(N^2)$ terms. Consequently, the bias is negligible relative to $L_{\text{full}}(t)$ *only if* $\bar{\ell}_{\mathcal{W}}(t)$ is asymptotically smaller than the typical non-wedge, non-edge loss — a condition not implied by anything already established, and implausible in practice: wedge pairs are, by construction, the structurally closest non-edges to becoming edges, so a well-trained denoising network is expected to assign them *comparable or larger* loss than generic non-edges, not smaller. Absent such a separate, justified assumption, $\text{Bias}(t)$ is $\Theta(N^2)$ and therefore **not negligible** — it is of the same order as the full-graph loss itself, not a vanishing correction to it.

Definition B.6 (Class-Mass SNR) *For a training step t and pool weighting w^{ij}, define the* class-mass signal-to-noise ratio *as the ratio of total weighted mass assigned to true edges versus non-edges within the current masked pool (Eq. equation 25):*

$$\text{SNR}_w(t) \;:=\; \frac{\displaystyle\sum_{(i,j)\in\mathcal{Q}_t} w^{ij}\, e_0^{(i,j)}\, Q_{ij}(t)}{\displaystyle\sum_{(i,j)\in\mathcal{Q}_t} w^{ij}\,\big(1 - e_0^{(i,j)}\big)\, Q_{ij}(t)}. \tag{56}$$

For the unweighted baseline ELBO ($w^{ij} \equiv 1$ for all pairs), equation 56 reduces to $\text{SNR}_1(t) = n_t^+/n_t^-$, using n_t^+, n_t^- as defined in Eq. 26.

Proposition B.7 (Exact SNR Balancing) *Let $w_{\text{pos}}^{ij}(t)$ be defined as in Eq. 27, and assume $n_t^+ \geq 1$. Then the reweighted pool is exactly class-balanced at every step:*

$$\text{SNR}_{w_{\text{pos}}}(t) \;=\; 1, \tag{57}$$

and the improvement over the unweighted baseline is exactly

$$\frac{\text{SNR}_{w_{\text{pos}}}(t)}{\text{SNR}_1(t)} \;=\; \frac{n_t^-}{n_t^+} \;=\; w_b^+(t). \tag{58}$$

By Eq. 27, $w_{\text{pos}}^{ij}(t) = w_b^+(t)$ when $e_0^{(i,j)} = 1$ and $w_{\text{pos}}^{ij}(t) = 1$ when $e_0^{(i,j)} = 0$. Substituting into the numerator of equation 56:

$$\sum_{\mathcal{Q}_t} w_{\text{pos}}^{ij}(t)\, e_0^{(i,j)}\, Q_{ij}(t) \;=\; w_b^+(t) \sum_{\mathcal{Q}_t} e_0^{(i,j)}\, Q_{ij}(t) \;=\; w_b^+(t)\, n_t^+, \tag{59}$$

using the definition of n_t^+ (Eq. 26). Substituting the definition $w_b^+(t) = n_t^- / \max(n_t^+, 1) = n_t^- / n_t^+$ (since $n_t^+ \geq 1$ by assumption) gives

$$w_b^+(t)\, n_t^+ \;=\; \frac{n_t^-}{n_t^+} \cdot n_t^+ \;=\; n_t^-. \tag{60}$$

For the denominator, since $w_{\text{pos}}^{ij}(t) = 1$ whenever $e_0^{(i,j)} = 0$,

$$\sum_{\mathcal{Q}_t} w_{\text{pos}}^{ij}(t)\big(1 - e_0^{(i,j)}\big)\, Q_{ij}(t) \;=\; \sum_{\mathcal{Q}_t} \big(1 - e_0^{(i,j)}\big)\, Q_{ij}(t) \;=\; n_t^-. \tag{61}$$

Hence $\text{SNR}_{w_{\text{pos}}}(t) = n_t^- / n_t^- = 1$, proving equation 57. Since $\text{SNR}_1(t) = n_t^+ / n_t^-$ (Definition B.6), the ratio $\text{SNR}_{w_{\text{pos}}}(t) / \text{SNR}_1(t) = n_t^- / n_t^+ = w_b^+(t)$ follows immediately, proving equation 58.

Remark B.8 (Gain Relative to a Dense-Pool Baseline) *Let $n_t^{-,\text{dense}}$ denote the negative-class pool mass of a dense baseline ELBO that pools over all $\binom{N}{2}$ pairs rather than $\mathcal{Q}_t$, under an identical per-pair masking rule. Since $\mathcal{Q}_t \subset \{1, \ldots, \binom{N}{2}\}$ and both poolings apply the same masking indicator $Q_{ij}(t)$ to their respective (larger vs. smaller) non-edge populations,*

$$n_t^{-,\text{dense}} \;=\; \Theta\!\left(\binom{N}{2} - m\right), \qquad n_t^- \;=\; \Theta\big(|\mathcal{Q}_t| - m\big) \;=\; \Theta(N\mu_2) \tag{62}$$

by Proposition B.2, at any fixed masking fraction. Since $n_t^+ = \Theta(m)$ under either pooling (all true edges are candidates in both), the dense baseline's unweighted SNR is

$$\text{SNR}_1^{\text{dense}}(t) \;=\; \Theta\!\left(\frac{m}{N^2}\right), \tag{63}$$

while our reweighted SNR is exactly 1 (Proposition B.7). The total SNR gain of the reweighted, restricted-pool objective over an unweighted dense-pool ELBO is therefore

$$\frac{\text{SNR}_{w_{\text{pos}}}(t)}{\text{SNR}_1^{\text{dense}}(t)} \;=\; \Theta\!\left(\frac{N^2}{m}\right) \;=\; \Theta\!\left(\frac{1}{\rho}\right), \qquad \rho := \frac{m}{\binom{N}{2}} \tag{64}$$

the inverse graph density — i.e., the sparser the graph, the larger the SNR gain, which is precisely the regime motivating this work.

B.1 Practical analysis

Degree-aware masking behaves as designed. Figures 3- 8 validate the degree-aware retention schedule $\beta^{ij} = \min(\beta_{\max}, (d^{i*} d^{j*} / \bar{d}^2)^{\gamma/2})$ against a uniform baseline ($\beta^{ij} \equiv 1$), using closed-form expectations over each dataset's true graph structure – no model training or sampling is involved, so every curve here is an exact property of the schedule itself, not an empirical estimate.

The two schedules differ in one specific way, worth stating precisely: under uniform masking, the retention curve $\bar{\alpha}_t^{\beta^{ij}} = \bar{\alpha}_t$ (the probability an edge's true state has not yet been replaced by the mask

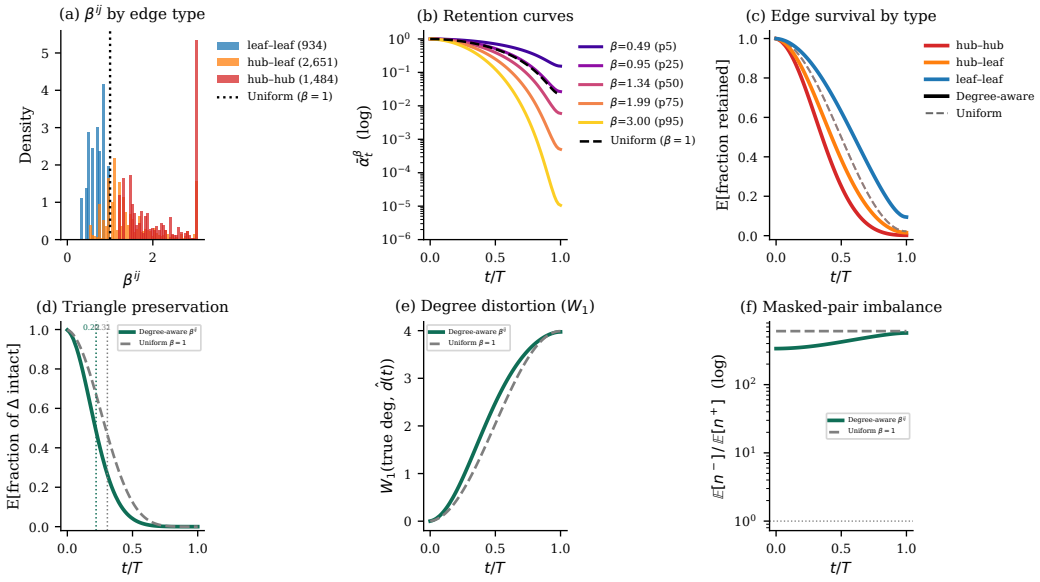


Figure 3: Cora: degree-aware masking analysis

token) is identical for every edge, regardless of what it connects. Degree-aware masking removes this blindness: because β^{ij} depends on (d^{i*}, d^{j*}), the *rate* of noise injection is itself informed by an edge's structural role, giving the denoiser an explicit curriculum built into the schedule rather than one it must discover on its own.

To make this concrete, we call a node a *hub* if its degree falls in the top quartile of the dataset's degree distribution, and classify an edge as hub–hub, hub–leaf, or leaf–leaf by how many endpoints qualify ("leaf" here means any non-hub node – the bottom three quartiles – not a degree-1 node specifically). Panels (a)–(c) show the intended effect is measurable, not just designed: β^{ij} spans nearly an order of magnitude across edge types (Cora: hub–hub mean 2.14 vs. leaf–leaf mean 0.68; Polblogs, the most degree-heterogeneous dataset we test, $d_{\max}/\bar{d} \approx 12.8$: hub–hub mean 2.68 vs. leaf–leaf mean 0.64). This separation carries through to the retention curves in panel (b), which visibly split by percentile instead of collapsing onto one line, and resolves into a genuine staged curriculum in panel (c): by the schedule's midpoint, hub–hub edges have already lost most of their signal (survival 0.18 for Polblogs, 0.27 for Cora), while leaf–leaf edges remain mostly intact (survival 0.64–0.67) – against a flat 0.517 for every edge type under uniform masking, which cannot distinguish them.

The trade-off is real. Front-loading hub structure is not free. Panel (d) measures the expected fraction of each graph's true triangles whose three constituent edges are *all* still unmasked at time t – how quickly triangle-completing structure is consumed by the forward process, and the component of graph structure most relevant to clustering coefficient, $CC = 3|\triangle|/|\text{wedges}|$ (panel (d) tracks the numerator's survival only, since it does not separately model wedge survival).

The *triangle-intact half-life* – the value of t/T at which this fraction first drops below one half – falls from 0.31 under uniform masking (identical across datasets by construction) to 0.19–0.22 under degree-aware masking, with the intact fraction at the midpoint 3–16$\times$ lower, worst for Polblogs. Since hub-adjacent edges dominate real triangles, this is the same front-loading from panel (c) acting on triangles specifically. This early half-life offers a plausible explanation for a separate finding in our step-reduction ablation (§C.3): sampling with only 50% of the trained step count produced almost no quality loss. Because $\bar{\alpha}_t^{\beta^{ij}}$ is a fixed function of t/T independent of how many discrete steps are used to traverse it, triangle-relevant structure settles into its final form at a consistent point along the schedule regardless of step count. If that point falls within the region step-reduction leaves densely

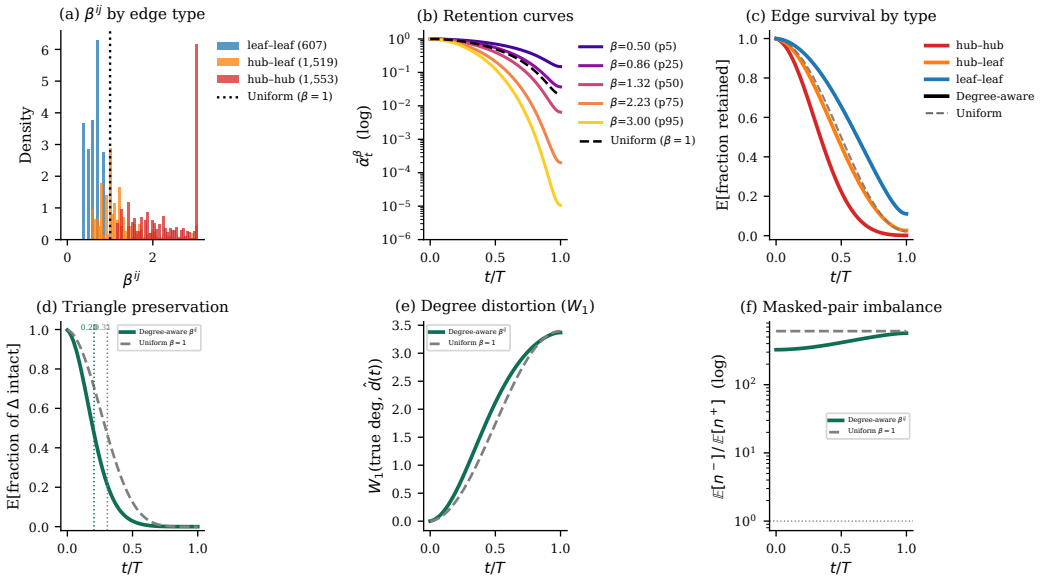


Figure 4: Citeseer: degree-aware masking analysis

1

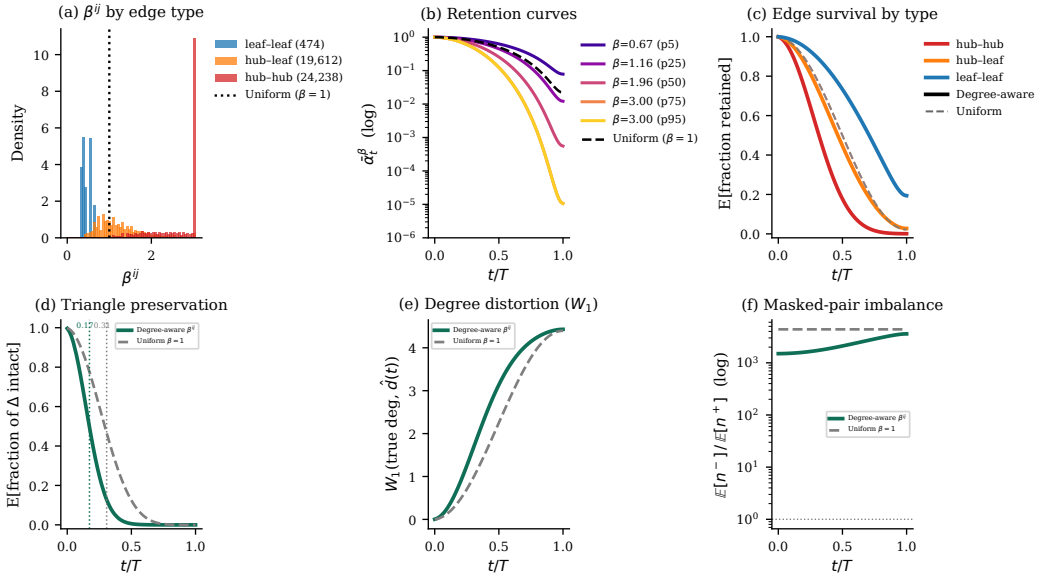


Figure 5: Pubmed: degree-aware masking analysis

sampled, this could explain why coarser step counts preserve quality; we flag this as a hypothesis connecting two separate observations, not a claim we test directly here.

Panel (e) shows a related cost in the degree distribution itself. At each t, we compute each node's *expected partially-revealed degree*, $\tilde{d}^i(t) = \sum_{j \in \mathcal{N}(i)} \bar{\alpha}_t^{\beta^{i,j}}$ under degree-aware masking (equivalently $d^{i*}\bar{\alpha}_t$ under uniform) – the number of true edges at i still unmasked, in expectation, at that point in the schedule – and compare the resulting distribution across all nodes to the true degree distribution via Wasserstein-1 distance. We use a distributional distance rather than a per-node error because what matters is whether the *shape* of the degree distribution is preserved at that checkpoint, not whether any single node's partial degree matches its final value exactly.

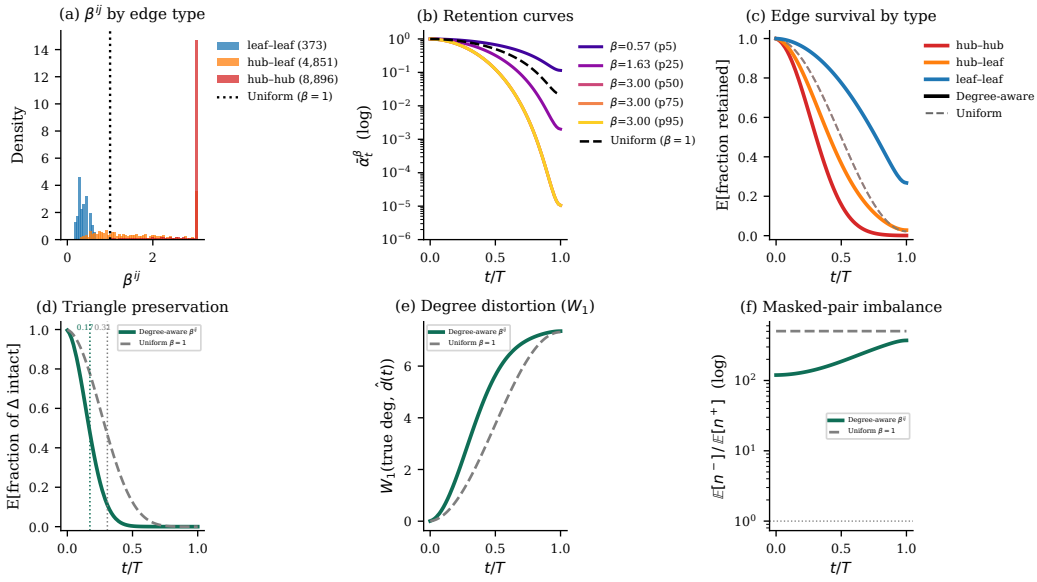


Figure 6: Alpha: degree-aware masking analysis

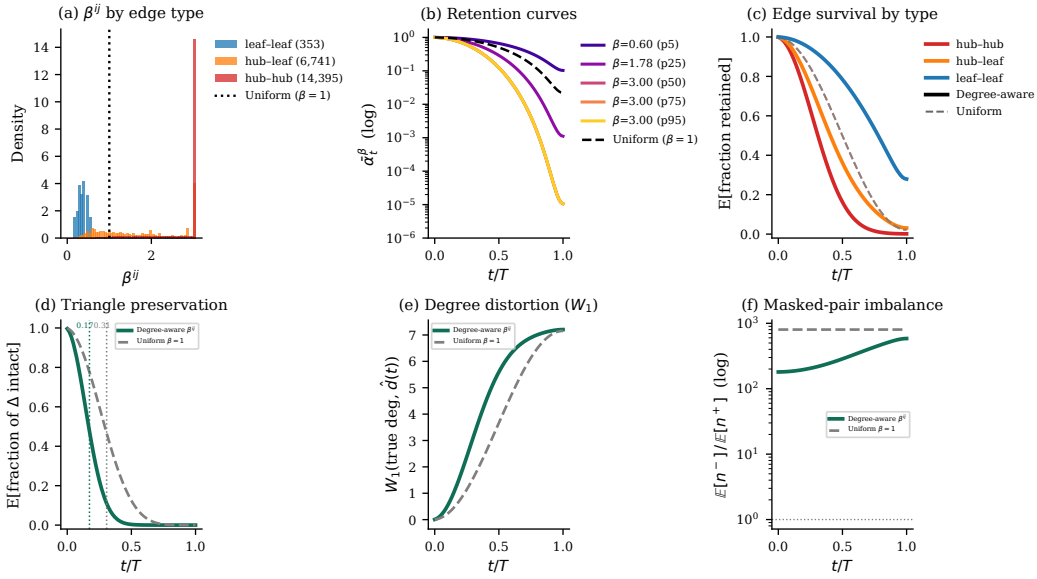


Figure 7: OTC: degree-aware masking analysis

By this measure, degree-aware masking is consistently *more* distorted than uniform at the midpoint ($1.23\times$ for Cora, $1.46\times$ for Polblogs): because hub and leaf edges resolve on different timescales, a node's partially-revealed degree becomes a skewed mixture of early-resolved leaf edges and late-resolved hub edges, rather than the simple rescaling of true degree that uniform masking's single shared rate produces by default.

Reweighting is what makes this trade-off pay off on real data. Panel (f) is where degree-aware masking earns its keep in practice. At each t, a true edge not yet revealed contributes to n_t^+ (masked edges); a true non-edge not yet revealed contributes to n_t^- (masked non-edges); we report their expected ratio $\mathbb{E}[n_t^-]/\mathbb{E}[n_t^+]$ – the same quantity the class-balance weight $w_b^+(t)$ (Eq. 27) is designed to correct. Note that n_t^+, n_t^- here are computed as closed-form expectations over the *full* graph, not

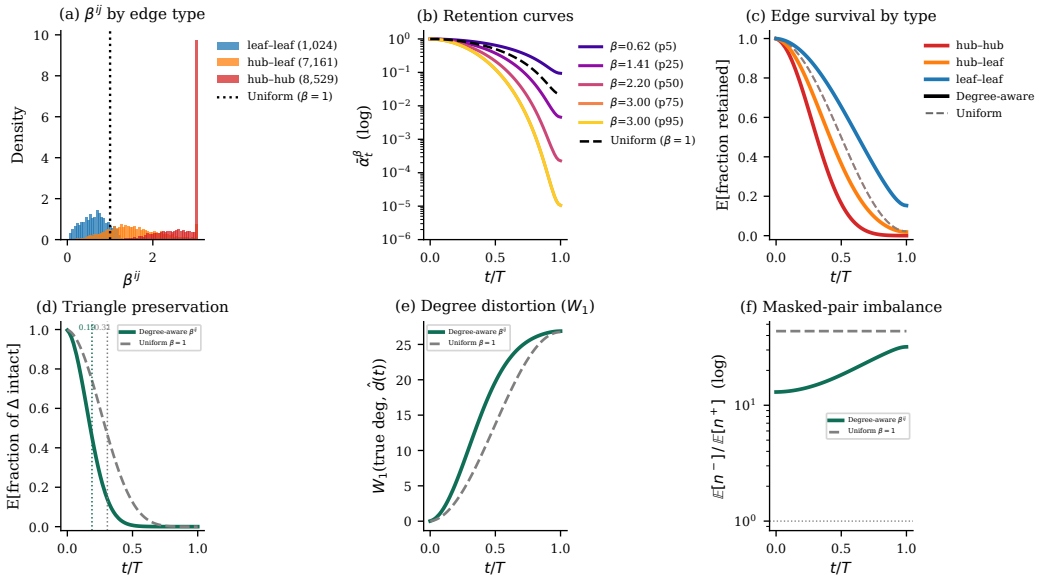


Figure 8: Polblogs: degree-aware masking analysis

over the restricted training-time candidate pool $\mathcal{Q}_t = \mathcal{Q} \cup S_t$; the two are related but not identical, since this analysis is model-free and makes no reference to the training-time restriction to $\mathcal{Q}_t$.

Graphs at the scale we target are extremely sparse (Cora $\rho \approx 0.0016$; Pubmed $\rho \approx 0.0002$), so even a moderate masking rate leaves an overwhelming excess of masked non-edges relative to masked edges. Degree-aware masking roughly halves this imbalance relative to uniform at every dataset we test: $0.69\times$ for Cora, $0.67\times$ for Citeseer, and $0.41\times$ for Polblogs – the reduction growing with degree heterogeneity, the same way the panel (e) cost does. The two mechanisms are therefore not independent design choices: degree-aware masking does not eliminate the sparsity-driven imbalance inherent to real graph data, but it substantially lightens the correction $w_b^+(t)$ has to make, letting the two work together rather than asking reweighting alone to compensate for a uniform schedule's worst-case imbalance.

Why this matters. Giving the schedule a sense of which edges matter structurally lets it teach the model something a uniform schedule simply cannot: hub connectivity first, periphery later. That choice costs something – triangles and degree shape both take a hit early in the trajectory, and that cost shows up later as a real, measurable gap in clustering coefficient, even in our best runs. But it also buys something that matters more for training on graphs this sparse: it substantially reduces the pool-level class imbalance that the reweighting term would otherwise have to correct on its own, doing part of that work for free before reweighting is even applied. Neither mechanism was designed to fully cover for the other, but together they leave the model in a measurably better-balanced training pool than either alone would produce – and that pattern held everywhere we looked, scaling with exactly the property, degree heterogeneity, that a degree-*aware* schedule ought to respond to.

C Extended Results

We provide a comprehensive evaluation across a diverse range of datasets, from small multi-graph collections to large single graphs, spanning citation and trust networks, and report sampling time alongside generation quality.

Figures 9 and 10 show the degree distribution of generated nodes for the large single-graph datasets. Our model closely tracks the training distribution across most of the degree range – the violin plots in Figure 9 show node density per degree value – but, as Figure 10 shows, it underrepresents the long-tail (high-degree) region on our larger datasets. Despite this, Table 1 shows our model achieves the best

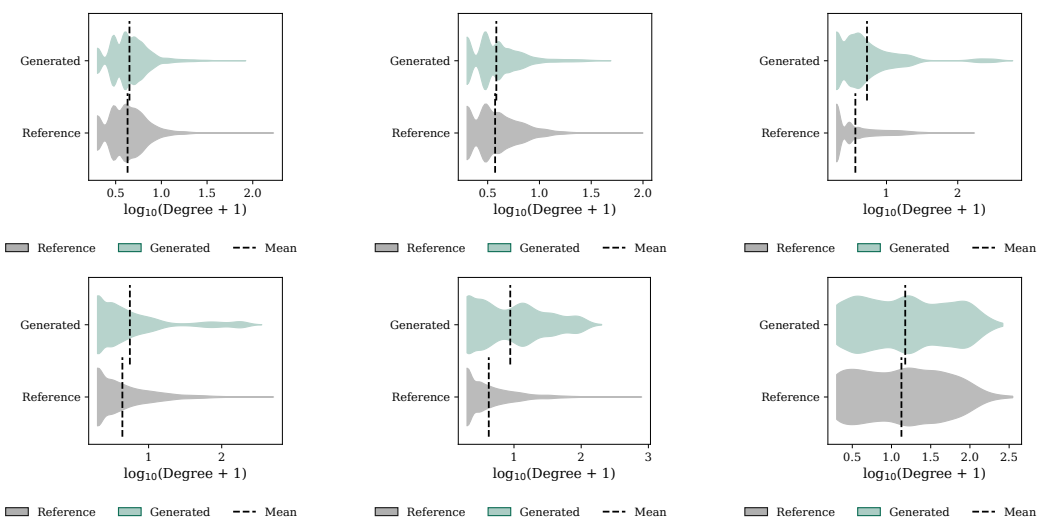


Figure 9: Degree distribution comparison across six datasets; plot width indicates node density.

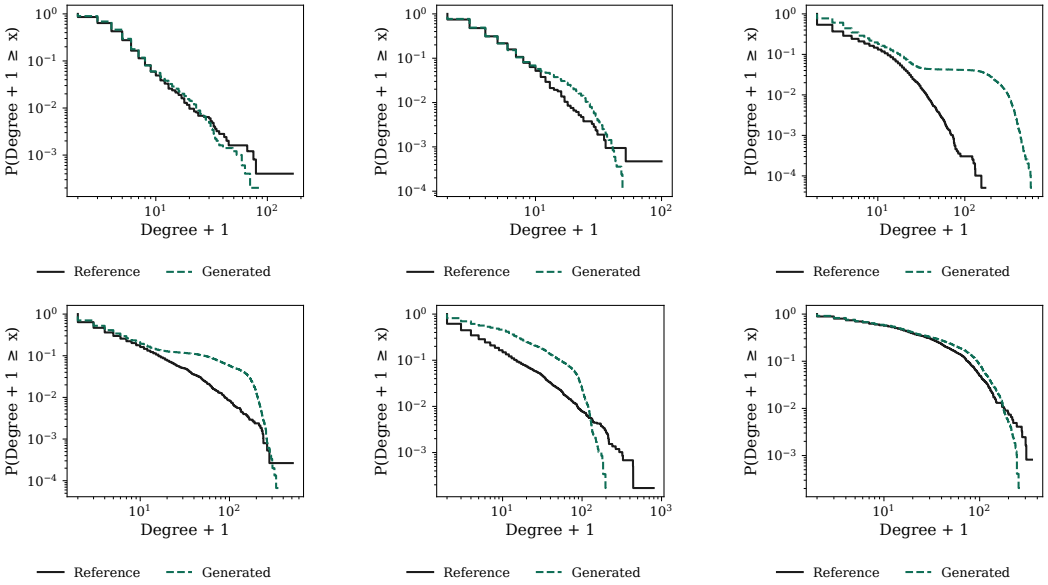


Figure 10: Degree distribution comparison across six datasets; plot width indicates node density.

clustering coefficient recovery among all methods compared, indicating strong local structural fidelity even where the global degree tail is harder to match.

C.1 Results on Small Graphs

Although our method targets large-graph generation, Table 6 shows it achieves performance comparable to methods specifically designed for and limited to small-scale graphs, most of which do not scale to the large-graph setting that is our primary focus.

On the ZINC molecular dataset, our model is competitive with dedicated molecule generation methods. As shown in Table 7, it achieves the best performance on all reported metrics except orbits and validity. We note that validity (§ E.1) here refers to *structural* validity – well-formedness of the generated graph – rather than chemical validity (e.g., valence-consistent bonding); our model's strong performance on other metrics should be read with this scope in mind.

C.2 Empirical sampling time

As discussed in Appendix B, our model achieves sub-quadratic *training* complexity, but sampling remains $O(N^2)$ per graph, the same asymptotic order as the diffusion baselines we compare against. Here we examine whether this translates into a practical bottleneck. We measure sampling (generation) time across graphs of varying size and compare it to other methods. As shown in Figure 11, our model generates graphs with more than 10k nodes in under 110 seconds, while EDGE and GraphMaker require approximately 200 seconds at the same graph size. ARROW-Diff achieves comparable speed to our model at this scale, but this comes at a different cost: it performs multiple rounds of random-walk sampling per graph, and, as shown in Table 1, produces substantially higher edge overlap (EO) than any other method we evaluate – a strong indicator of edge memorization rather than genuine structural generalization. Our model achieves comparable sampling time without this trade-off, generating graphs with low edge overlap alongside strong recovery of other structural statistics.

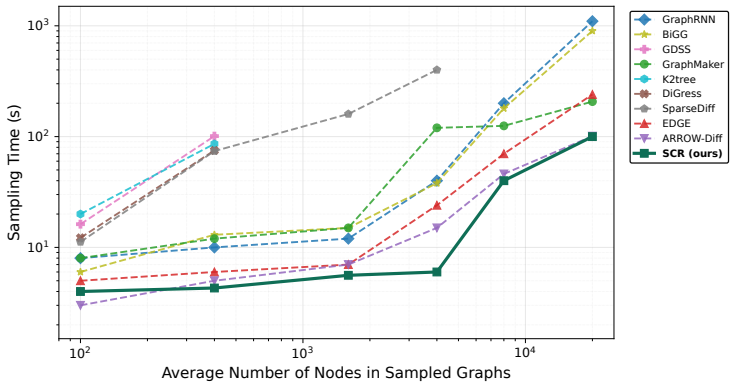


Figure 11: Sampling time across various graph sizes, averaged over 16 runs per model under identical conditions.

C.3 Efficiency of sampling: reduced diffusion steps

Motivation. While our method achieves sub-quadratic training cost, sampling remains $O(N^2)$ per graph, the same asymptotic order as the dense baselines we compare against. For a method whose central goal is scaling to large graphs, this asymmetry is a real practical concern: the training-time savings are of limited use if generating a single large graph still takes disproportionately long. Reducing the number of reverse diffusion steps is the most direct lever available to lower this per-graph cost without further architectural changes, so it is important to establish how much quality is actually lost when doing so – and whether the resulting time savings are worth the trade-off in practice.

Using a single model trained with T_{train} steps, we evaluate generation quality when sampling with $T \in \{10\%, 30\%, 50\%, 100\%\}$ of T_{train} reverse steps, skipping directly between the sampled t values rather than truncating a prefix.

Why step-skipping is exact. This introduces no approximation in the reverse update rule itself: $r^{ij}_{t \to t_{\text{prev}}}$ (Eq. 14) is derived for arbitrary $t_{\text{prev}} \leq t$, not only single-step transitions. What *can* change at coarser T is the distribution of intermediate states G_t the encoder is queried on – fewer, more widely spaced checkpoints than the trajectory seen during training. This experiment therefore tests the model's robustness to that intermediate-state shift, not an error in the sampling procedure.

Results. We illustrate the pattern using Cora, Citeseer, and Polblogs as a representative example (Figure 12 and Table 9); the same qualitative behavior is seen for the other datasets and settings we tested. Sampling time decreases roughly in proportion to T, while generation quality – measured via power-law exponent (PLE), characteristic path length (CPL), and clustering coefficient (CC) – is largely preserved down to $T \approx 50\%$. Degradation becomes pronounced only at $T = 10\%$, and is more

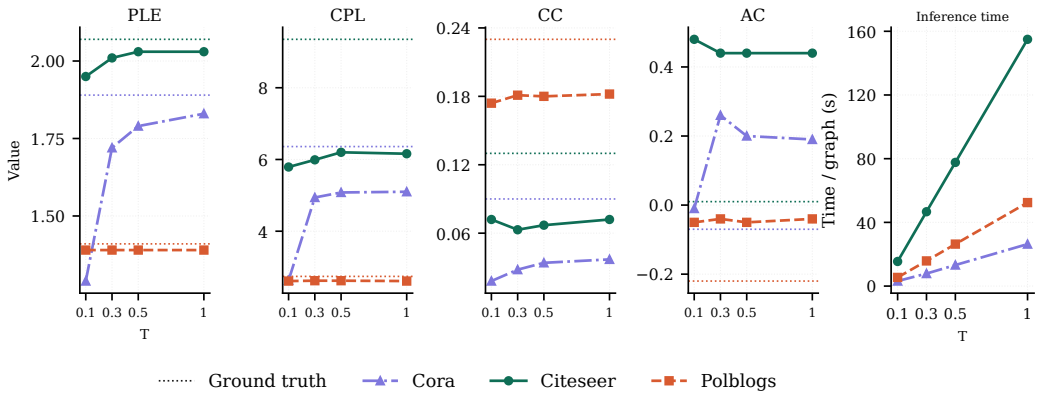


Figure 12: Structural statistics of generated graphs vs. inference-time diffusion steps.

severe for datasets trained with a smaller T_{train} (e.g. Cora, T_{train}=64) than for those trained with a larger one (Citeseer and Polblogs, T_{train}=512), even though both are being reduced to the same *fraction*.

Outcome: reduced inference steps and degree-aware masking. Across all six ablation datasets, truncating the number of reverse sampling steps to a fraction of T preserves generation quality substantially better under the degree-adaptive schedule β^{ij} than under uniform masking ($\beta^{ij} = 1$). Since $\beta^{ij} = \min(\beta_{\max}, (d^{i*}d^{j*}/\bar{d}^2)^{\gamma/2})$ increases with the degree product, $\bar{\alpha}_t^{\beta^{ij}}$ decays *faster* for hub–hub pairs, so they cross any fixed retention threshold at *smaller* t in the forward process and are consequently the pairs masked *longest* – and therefore resolved *last* – in the reverse chain, while leaf–leaf pairs ($\beta^{ij} \ll 1$) are masked latest in the forward direction and resolved first in reverse. This extends the forward-process analysis of § B.1 to the reverse sampling trajectory, and the two are consistent: the same pairs shown there to survive longest in the forward direction are shown here to be resolved last in reverse. We confirm this directly with a Monte Carlo simulation of the exact per-edge unmask hazard used at sampling time, $r^{ij}_{t\to t_{\text{prev}}} = (\bar{\alpha}_{t_{\text{prev}}}^{\beta^{ij}} - \bar{\alpha}_t^{\beta^{ij}})/(1 - \bar{\alpha}_t^{\beta^{ij}})$, run over the true $T-1, \ldots, 0$ reverse step sequence with 50,000 independent trials per edge class. Leaf–leaf edges (5th percentile of each dataset's true per-edge β^{ij} distribution) resolve at 44.7–49.0% of the way through the reverse trajectory on average; hub–hub edges (95th percentile) resolve at 50.6–57.7% – later in every one of the six datasets (Cora, CiteSeer, PubMed, Polblogs, Bitcoin Alpha, Bitcoin OTC), with no exceptions.

What reduced-step truncation actually preserves, then, is not privileged attention to hubs, but the fact that degree-aware masking is the only one of the schedules considered that assigns different edge classes to *different, degree-correlated* points in the trajectory at all. A K-step, uniformly-spaced skip schedule over $[0, T-1]$ always includes points near both $t \approx T$ and $t \approx 0$ regardless of K; under degree-aware masking this means even a heavily truncated schedule still separately captures an early, mostly-leaf-resolved scaffold and a late, hub-correcting refinement. Under uniform masking every edge shares the same marginal $\bar{\alpha}_t$, so there is no such temporal separation for truncation to exploit – coarsening the step count degrades every edge class identically rather than trading resolution on a subset that has already received comparatively more of the trajectory to condition on. We present this as the most parsimonious explanation consistent with the confirmed reveal-order data, not as an independently verified mechanism; isolating it from competing explanations would require comparing generation quality broken down by edge degree class at fixed truncated K, which we leave to future work.

The strength of this effect is also dataset-dependent in a way that should be stated explicitly rather than averaged away. The six datasets split into two regimes governed by how much headroom $\beta_{\max}$ leaves above the typical degree product: Cora, CiteSeer, and PubMed (all $\beta_{\max} = 1.5$, 41–63% of edges at the cap) show a consistent 10.2–13.0 percentage-point gap between leaf and hub reveal time; Polblogs, Bitcoin Alpha, and Bitcoin OTC show only 3.9–4.7 points. For the two Bitcoin trust networks this is not merely "more saturation" but a degenerate special case: $\beta_{\max} = 1.0$ is

numerically identical to the uniform baseline $\beta^{ij} = 1$, so the 86.7–88.1% of edges sitting at the cap are, by construction, scheduled *exactly as if degree-awareness were switched off* – only the uncapped low-degree tail ($\beta^{ij}_{p5} \approx 0.57$–$0.60$) is scheduled any differently from uniform at all, pulled earlier rather than the hub tier being pulled later. Polblogs uses the same $\beta_{\max} = 1.5$ as the citation networks yet still falls in the weak-separation group, because its exceptionally high mean degree ($\bar{d} = 27.36$, an order of magnitude above the other datasets) pushes even its *median* edge to the cap, not only its tail.

C.4 Effect of the prior scaffold.

We compare three terminal-distribution priors for the visible pairs at $t = T$, holding the degree-aware masking exponent β^{ij} (Eq. 6) identical across all three, so that only the *value* distribution of revealed pairs differs:

- **Erdős–Rényi (ER):** $\rho^{ij} = p := m/\binom{N}{2}$ for all pairs — equivalently, every pair is an i.i.d. $\mathrm{Bernoulli}(p)$ draw with a single shared parameter, uniform over all $\binom{N}{2}$ pairs regardless of node degree. Since $p = \bar{d}/(N-1)$, where $\bar{d} = 2m/N$ is the mean degree, this is the degree-blind special case of Eq. 24 obtained by replacing d^{i*}, d^{j*} with $\bar{d}$ for every node.
- **Stochastic Block Model (SBM):** $\rho^{ij} = B_{z_i,z_j}$, with block assignments z_i taken from each dataset's ground-truth class labels and K set to the dataset's native number of classes (Cora: $K = 7$; Polblogs and Alpha: $K = 2$); B_{kl} is fit as the empirical block-pair edge density on the training graph.
- **Chung–Lu (ours, default):** $\rho^{ij} = d^{i*}d^{j*}/2m$ (Eq. 24), as used throughout the main results.

Table 3 holds β^{ij} fixed across all three arms, so the reveal *order* is identical and only the reveal *value* ρ^{ij} differs — isolating what the model is told about the hub pairs that degree-adaptive masking always exposes first. Structurally, only the Chung–Lu prior scales with per-node degree, so it is the only choice whose functional form is consistent with that ordering: it assigns high edge probability to exactly the pairs β^{ij} has already prioritized, while ER's flat p and SBM's block average B_{z_i,z_j} anchor those same pairs to an uninformative statistic. This structural advantage translates into an empirical one in most, though not all, cases: on Alpha, which has no natural block structure, SCR sweeps all four metrics while ER and SBM overestimate clustering by 7–8×. On Polblogs, whose near-binary ideological structure happens to align with the SBM's own inductive bias, SBM edges out SCR on three metrics — the one condition structurally favorable to it. What we want to emphasize here is that graph-generation priors can be anchored to a well-known reference distribution — as with ER, SBM, or Chung–Lu — but depending on the behavior of the noise schedule governing the trajectory between the original and noised graph, some priors are better suited than others and lead to higher-quality generation. In the majority of cases, Chung–Lu is best suited to our degree-aware forward process, since both are governed by the same underlying quantity: node degree.

C.5 Effect of the regularizers.

Table 3 ablates the two structural regularizers, $\mathcal{L}_{deg}$ and $\mathcal{L}_{tri}$, against default SCR on Cora, Polblogs, and Bitcoin-Alpha. The full model is best or tied-best in 10 of 12 dataset-metric cells, with two explainable exceptions discussed below. Removing $\mathcal{L}_{deg}$ consistently and substantially degrades clustering-coefficient recovery on all three datasets ($|\Delta_{CC}|$ increases 2–4×), which is worth stating precisely because $CC = 2{\cdot}\text{triangles}/(d(d-1))$ is degree-normalized: an incorrect degree distribution distorts CC even when triangle-matching pressure from $\mathcal{L}_{tri}$ and the HT-reweighted candidate-set gradient remain active, making $\mathcal{L}_{deg}$ a precondition for correct clustering recovery rather than a term that merely competes with it. Conversely, removing $\mathcal{L}_{tri}$ consistently degrades assortativity on all three datasets ($|\Delta_{AC}|$ nearly doubles on Cora and Polblogs) — evidence that $\mathcal{L}_{tri}$'s practical role is shaping degree-correlation and mixing structure rather than literal triangle-count matching. The one dataset-specific exception is Bitcoin-Alpha, where $\mathcal{L}_{deg}$ improves every other metric but degrades AC ($|\Delta|$ $0.12 \to 0.27$); given this dataset's extreme degree heterogeneity (max degree 511 on N=3,775, versus $\bar{d} \approx 4$–27 elsewhere), we attribute this to tension between the degree-matching gradient and the mixing structure induced by such an extreme tail, and report it as an observed effect rather than a fully diagnosed mechanism. The power-law exponent is essentially unaffected by either ablation on Cora and Polblogs ($|\Delta_{PLE}| \leq 0.02$ in every row), consistent with the degree-adaptive

Table 6: Comparison of metrics for small graphs. MMD values are reported as $\times 10^{-2}$ (lower is better); FID↓ is based on a neural feature space; *Uniq*↑ indicates the percentage of non-isomorphic generated graphs; and *Nov*↑ measures the percentage of generated graphs not present in the training set (both higher are better).

	Community						Ego					
	MMD ($\times 10^{-2}$)↓			FID↓	Uniq↑	Nov↑	MMD ($\times 10^{-2}$)↓			FID↓	Uniq↑	Nov↑
Method	Deg.	Clus.	Orb.		(%)	(%)	Deg.	Clus.	Orb.		(%)	(%)
GraphRNN 2018	15.23 ±0.41	9.55 ±0.23	2.97 ±0.18	8.38 ±0.22	72.4 ±1.8	68.9 ±2.1	7.88 ±0.37	101.45 ±2.4	14.87 ±0.53	87.76 ±0.74	65.3 ±2.1	62.4 ±2.3
BiGG 2020	14.91 ±0.17	8.53 ±0.29	9.03 ±0.13	7.35 ±0.32	71.4 ±0.2	68.9 ±0.1	9.21 ±0.2	27.52 ±0.3	13.90 ±0.2	34.23 ±0.1	67.8 ±0.1	65.5 ±0.3
GDSS 2022	5.33 ±0.1	10.11 ±0.09	4.73 ±0.07	11.17 ±0.1	94.5 ±0.01	88.6 ±0.03	6.82 ±0.03	5.33 ±0.01	12.71 ±0.02	16.83 ±0.1	89.3 ±0.01	88.9 ±0.1
DiGress 2023	3.07 ±0.11	**1.69** ±0.06	2.89 ±0.08	4.07 ±0.09	98.6 ±0.3	98.1 ±0.5	6.09 ±0.15	**0.97** ±0.05	11.09 ±0.22	16.76 ±0.31	99.1 ±0.1	97.7 ±0.2
EDGE 2023	**1.63** ±0.05	7.01 ±0.27	**2.10** ±0.04	**2.24** ±0.06	**99.9** ±0.0	**98.8** ±0.1	5.59 ±0.09	16.61 ±0.48	**6.21** ±0.05	14.59 ±0.12	99.5 ±0.1	98.6 ±0.1
SparsDiff 2023	4.05 ±0.13	1.78 ±0.03	2.99 ±0.03	3.97 ±0.09	97.6 ±0.4	97.1 ±0.3	7.02 ±0.13	5.3 ±0.03	11.10±0.22	15.76 ±0.11	98.1 ±0.1	97.8 ±0.1
K2tree 2024	3.02 ±0.1	2.30 ±0.13	2.93 ±0.31	5.17±0.02	95.2 ±0.01	48.2 ±0.02	7.13 ±0.07	4.31 ±0.01	11.70±0.05	16.04 ±0.02	55.2 ±0.01	61.2 ±0.01
GraphMaker 2024	27.88 ±0.95	32.33 ±1.07	19.78 ±0.66	56.11 ±1.42	45.8 ±1.3	43.1 ±1.6	35.78 ±1.22	33.60 ±0.91	67.54 ±1.84	189.95 ±2.3	38.4 ±0.8	35.9 ±1.1
ARROW-Diff 2025	12.38 ±0.46	86.49 ±2.5	16.46 ±0.67	13.43 ±0.41	41.2 ±1.9	39.8 ±1.7	16.13 ±0.38	12.88 ±0.34	15.23 ±0.72	22.45 ±0.59	48.7 ±1.2	45.5 ±1.4
SCR (ours)	13.01 ±0.03	78.84 ±0.18	3.50±0.14	14.85 ±0.02	**99.9** ±0.0	98.3 ±0.1	**3.06** ±0.91	84.29 ±0.09	11.05 ±0.31	**2.84** ±0.05	**100.0** ±0.0	**100.0** ±0.0

Table 7: Comparison of metrics on ZINC. MMD values are reported as $\times 10^{-2}$ (lower is better); FID↓ is based on a neural feature space; *Uniq*↑, *Val*↑, *Nov*↑ report the percentage of generated molecules that are unique, chemically valid, and not present in the training set, respectively (all higher is better; Uniq/Nov computed among valid molecules).

	MMD ($\times 10^{-2}$)↓					FID↓	UVN (%) ↑		
Method	Deg.	Clus.	Orb.	Spec.	NSPDK		Uniq	Val	Nov
GraphRNN 2018	3.5±0.1	7.5±0.2	6.5±0.3	9.5±0.1	12.5±0.1	42.2±0.5	66.1±0.5	58.5±0.4	63.8±0.6
BiGG 2020	4.3±0.2	7.1±0.1	6.9±0.2	13.4±0.2	14.5±0.1	41.5±0.1	61.1±0.1	59.9 ±0.2	66.5±0.1
GDSS 2022	5.8 ±0.2	6.3±0.1	6.2±0.2	13.2±0.1	12.7±0.1	38.7±0.5	63.5±0.1	62.1±0.1	63.7±0.1
DiGress 2023	0.9±0.1	2.7±0.3	**3.5**±0.1	7.5±0.2	5.7±0.1	15.6±0.4	**100.0**±0.0	**98.1**±0.1	**100.0**±0.0
EDGE 2023	2.8±0.2	4.1±0.1	7.3±0.1	9.1±0.2	8.7±0.3	14.8±0.7	99.0±0.3	92.3±0.1	96.8±0.2
SparsDiff 2023	6.8±0.5	6.7±0.2	16.5±0.3	10.2±0.1	18.7±0.6	17.8±0.1	89.5±0.1	88.5±0.1	95.6±0.2
K2tree 2024	1.7±0.1	2.3±0.1	5.6±0.2	4.2±0.1	6.0±0.1	12.5±0.1	99.0±0.1	88.5±0.3	99.0±0.1
ARROW-Diff 2025	10.2±0.5	9.5±0.1	23.5±0.7	11.5±0.2	17.4±0.5	69.8±0.5	58.7±0.6	61.3±0.1	60.1±0.1
SCR (ours)	**0.42**±0.1	**0.4**±0.1	17.31±0.5	**1.70**±0.1	**5.0**±0.1	**7.77**±0.3	**100.0**±0.1	90.6±0.2	**100.0**±0.1

masking schedule β^{ij} — held fixed across all three arms — being the primary driver of overall degree-distribution shape, with $\mathcal{L}_{deg}$ and $\mathcal{L}_{tri}$ acting as finer, dataset-dependent corrections on top of it.

D Implementation Details

Backbone. Message passing uses a GIN-style sum-aggregation convolution: neighbor messages are weighted by the *current diffusion step's own edge probabilities*, not by a learned attention score, so there is no attention anywhere in the message-passing stack itself. Each node's initial embedding is the sum of two 3-layer MLPs: one encoding its permanent target degree d^*, the other its current (noisy-state) degree d_t – i.e. the degree deficit the node still needs to satisfy – both log-scale normalized by $\log(1+d)/\log(1+d_{\max})$. Activations are SiLU throughout, except the diffusion-timestep embedding module, which uses Mish. Parameters use PyTorch's default initialization; no custom initialization scheme is applied. Depth and embedding width are tuned per dataset, ranging from 3 to 6 message-passing layers and embedding dimension 32 or 64 (Table 8).

Triangle-conditioned edge decoding. For each candidate edge (i, j), a triangle feature $\hat{\tau}_t^{ij}$ – the number of common *currently-revealed* neighbors of i, j at noise level t – is log-scaled and projected by a 3-layer MLP, then concatenated with the two endpoint embeddings before decoding. Edge logits are produced by a multi-head self-attention block (4 heads, fixed across all datasets) over the resulting 3-token sequence (source embedding, target embedding, edge context), followed by FiLM (Feature-wise Linear Modulation) conditioning on the raw, un-attended edge context – a stabilizing bypass path for early, high-noise timesteps when node embeddings carry little differentiated signal. A single-layer linear head additionally regresses each node's target degree and triangle count during training only (gated off at sampling time), providing a direct node-level gradient toward both quantities.

Optimization. All models are trained with Adam, gradient clipping at global norm 1.0, and dropout 0.1, uniformly across every dataset. Random seed defaults to 0 and is applied to Python, NumPy, and PyTorch's CPU/CUDA generators; we do not enable CUDA-deterministic algorithms, so individual runs are statistically, but not bit-exactly, reproducible across re-execution.

Table 8: Per-dataset implementation hyperparameters, verified against the exact command-line flags used for training. L = number of message-passing layers; **non-$\mathcal{Q}$ count** $= |S_t|$, the size of the coverage sample.

Dataset	dim	L	T	$\beta_{\max}$	γ	LR	Epochs	Batch	λ_{deg}	λ_{tri}	λ_{LT}	non-Q frac.
Cora	64	5	128	1.5	1.0	8e-5	50,000	4	0.5	0.1	0.01	0.1
CiteSeer	64	5	512	1.5	1.0	1e-4	50,000	4	0.5	0.05	0.01	0.1
Polblogs	64	5	256	1.5	1.0	8e-5	50,000	4	0.5	1.0	0.1	0.1
Bitcoin Alpha	32	6	128	1.0	1.0	8e-5	50,000	1	0.1	0.01	0.01	0.001
Bitcoin OTC	64	6	128	1.0	1.0	8e-5	50,000	1	0.1	0.05	0.01	0.005
PubMed	64	6	128	1.5	1.0	1e-4	50,000	1	0.1	0.05	0.01	0.002
Community	64	3	256	1.5	1.0	8e-5	10,000	16	0.01	0.05	0.01	0.1
Ego	64	5	128	1.0	1.0	8e-5	12,000	64	0.5	0.5	0.05	0.1
ZINC	64	5	256	1.5	1.0	5e-5	10,000	32	0.01	0.01	0.001	0.1

Table 9: Effect of reduced inference-time diffusion steps, reported as raw signed values against ground truth. Absolute step count is shown alongside each fraction, since T_{train} differs across datasets (64 vs. 512) and quality depends on absolute steps, not fraction alone. Time/graph is computed by dividing each run's total generation time by its number of generated graphs. Training convergence and T_{train} selection are detailed in Appendix D.

Dataset	Condition	Steps	EO	PLE	CPL	CC	AC	Time/graph (s)
Cora ($T_{\text{train}} = 64$)	ground truth	–	100	1.89	6.36	0.090	−0.07	–
	100%	64	0.20	1.83	5.10	0.037	0.19	26.35
	50%	32	0.16	1.79	5.08	0.034	0.20	13.20
	30%	19	0.15	1.72	4.94	0.028	0.26	7.85
	10%	6	0.16	1.29	2.63	0.018	−0.01	3.15
Citeseer ($T_{\text{train}} = 512$)	ground truth	–	100	2.07	9.35	0.130	0.01	–
	100%	512	0.13	2.03	6.16	0.072	0.44	154.95
	50%	256	0.12	2.03	6.20	0.067	0.44	77.65
	30%	153	0.09	2.01	5.99	0.063	0.44	46.68
	10%	51	0.07	1.95	5.79	0.072	0.48	15.45
Polblogs ($T_{\text{train}} = 512$)	ground truth	–	100	1.41	2.74	0.230	−0.22	–
	100%	512	1.45	1.39	2.61	0.182	−0.04	52.44
	50%	256	1.22	1.39	2.62	0.180	−0.05	26.36
	30%	153	1.16	1.39	2.62	0.181	−0.04	15.80
	10%	51	2.62	1.39	2.61	0.174	−0.05	5.43

Reproducibility Table 8 reflects the exact command-line flags in our released training scripts at the time of writing. We report the values actually used to train the checkpoints in this paper; where a table entry disagrees with a script's own comment, the table entry is authoritative.

E DATASETS

Here we describe all datasets used for evaluation. For all single-graph datasets, the raw directed graphs are converted to undirected, self-loops are removed, and the largest connected component (LCC) is retained. Node and edge counts below reflect this preprocessing.

Polblogs The Political Blogs dataset is a directed network of hyperlinks between political blogs collected around the 2004 US Presidential election Adamic & Glance (2005). Nodes are blogs and edges are hyperlinks. After converting to an undirected LCC and removing self-loops, the resulting graph has $N = 1{,}222$ nodes and $M = 16{,}714$ edges ($\rho = 0.022$). The network exhibits a strong community structure aligned with political leaning, high clustering ($\text{CC} = 0.226$), short characteristic path length ($\text{CPL} = 2.74$), and disassortative degree mixing ($r = -0.22$), consistent with a hub-and-spoke blog ecosystem.

Citation Networks Cora, Citeseer, and Pubmed are directed citation graphs in which nodes represent documents with sparse bag-of-words features and edges represent citation links Sen et al.

(2008). We discard node features and treat each as a structural generation benchmark. After undirected LCC extraction: Cora yields $N = 2{,}485$, $M = 5{,}069$ (retaining 92% of original nodes); Citeseer yields $N = 2{,}120$, $M = 3{,}679$ (retaining 64% of original nodes, as the raw graph contains a large fraction of isolated or weakly connected documents); and Pubmed yields $N = 19{,}717$, $M = 44{,}324$ (all nodes in one component). All three are sparse ($\rho < 0.002$) with low clustering and moderate path lengths.

Signed Bitcoin Datasets: Alpha and OTC The Bitcoin-Alpha and Bitcoin-OTC datasets are signed, directed trust networks in which each edge carries an integer rating from -10 (total distrust) to $+10$ (total trust), reflecting user-level trust on two Bitcoin trading platforms Kumar et al. (2016; 2018). We discard edge signs and weights, convert to undirected, and retain the LCC. Bitcoin-Alpha: $N = 3{,}775$, $M = 14{,}120$ ($\rho = 0.002$, CC $= 0.078$, CPL $= 3.58$). Bitcoin-OTC: $N = 5{,}875$, $M = 21{,}489$ ($\rho = 0.001$, CC $= 0.059$, CPL $= 3.60$). Both exhibit heavy-tailed degree distributions ($\alpha \approx 1.9$–2.0) and disassortative mixing, consistent with trust networks dominated by a small number of high-reputation hubs.

Community This dataset consists of 510 synthetic two-community graphs with $60 \leq |V| \leq 160$ and $231 \leq |E| \leq 1{,}956$. Each graph contains two communities generated independently by the Erdős–Rényi model Erdos et al. (1960) with $n = |V|/2$ nodes and intra-community connection probability $p = 0.3$. An additional $0.05|V|$ inter-community edges are added uniformly at random between the two communities You et al. (2018). The graphs are dense relative to real-world networks ($\bar{\rho} = 0.149$) with high clustering ($\overline{\text{CC}} = 0.294$) and low disassortativity ($\bar{r} = -0.021$).

Ego This dataset contains 757 three-hop ego networks extracted from the Citeseer citation graph Sen et al. (2008), with $50 \leq |V| \leq 399$ and $56 \leq |E| \leq 1{,}062$. Each ego network is the subgraph induced by the neighbours within three hops of a sampled seed node, following the construction of You et al. (2018). Compared to Community, Ego networks are sparser ($\bar{\rho} = 0.036$), more heterogeneous in size ($\sigma_N = 85$), and have a heavier-tailed degree distribution ($\bar{\alpha} = 2.07$).

ZINC ZINC12k is a molecular graph benchmark drawn from the ZINC database of commercially available drug-like compounds Irwin et al. (2012). We use the standard 12-graph split provided by PyTorch Geometric Fey & Lenssen (2019): 10,000 training / 1,000 validation / 1,000 test graphs. Nodes represent heavy atoms and edges represent covalent bonds; bond types (single, double) are collapsed to binary edges, and node atom-type features are discarded to focus purely on structural generation. The resulting graphs satisfy $9 \leq |V| \leq 37$ (mean 23.2) and $8 \leq |E| \leq 41$ (mean 24.9), with mean degree $\bar{d} \approx 2.15$ and maximum degree $d_{\max} = 4$ imposed by carbon's valence. The graph topology is essentially acyclic: mean triangle count is 0.06 and mean clustering coefficient is 0.006, reflecting the absence of three-membered rings in drug-like molecules under standard valence rules.

E.1 Evaluation Metrics

We evaluate structural fidelity across two dimensions: several key graph statistics and distributional distance metrics, including Maximum Mean Discrepancy (MMD).

Edge Overlap (EO). For single-graph datasets the generated graph $\hat{G}$ is compared edge-by-edge to the ground-truth graph $G^* = (V, E^*)$. Edge overlap is the fraction of ground-truth edges recovered:

$$\text{EO}(G^*, \hat{G}) \;=\; \frac{|\,E^* \cap \hat{E}\,|}{|E^*|} \times 100 \;=\; \frac{(\mathbf{A}^* \odot \hat{\mathbf{A}})_+}{2|E^*|} \times 100\,, \tag{65}$$

where $\mathbf{A}^*$ and $\hat{\mathbf{A}}$ are the respective binary adjacency matrices and $\odot$ denotes elementwise product. By construction $\text{EO}(G^*, G^*) = 100$, so the ground-truth row serves as a self-overlap reference. For generation, EO$\downarrow$ is preferred: a model that memorises the training graph achieves EO $= 100$, while a model that generates a structurally valid but novel graph achieves EO ≈ 0.

Power-Law Exponent (PLE). The degree distribution of many real-world graphs follows a power law $P(k) \propto k^{-\alpha}$ for $k \geq x_{\min}$. We estimate α by maximum likelihood with fixed $x_{\min} = 1$ using

the continuous-approximation MLE of Clauset et al. (2009):

$$\hat{\alpha} \;=\; 1 + n\left[\sum_{i=1}^{n} \ln \frac{k_i}{x_{\min}}\right]^{-1}, \tag{66}$$

where the sum runs over all n nodes with $k_i \geq 1$. Both the ground-truth and generated values are computed identically, so the comparison is internally consistent.

Characteristic Path Length (CPL). CPL is the mean shortest-path distance over all pairs of reachable nodes:

$$\mathrm{CPL}(G) \;=\; \frac{\displaystyle\sum_{i\neq j,\; d(i,j)<\infty} d(i,j)}{|\{(i,j) : d(i,j) < \infty,\; i \neq j\}|}, \tag{67}$$

where $d(i,j)$ is the geodesic distance between i and j. For $N \leq 2000$ we compute exact all-pairs shortest paths via Dijkstra in $O(N(M + N\log N))$ time. For $N > 2000$ we estimate CPL by breadth-first search from $k = 500$ uniformly sampled source nodes and average the resulting distances, giving an unbiased estimator at $O(k \cdot M)$ cost instead of $O(N \cdot M)$.

Clustering Coefficient (CC). We use the *global* clustering coefficient (network transitivity), defined as the ratio of closed paths of length two to all paths of length two:

$$\mathrm{CC}(G) \;=\; \frac{3\,|\mathcal{T}|}{|\mathcal{W}|} \;=\; \frac{3\,|\mathcal{T}|}{\frac{1}{2}\sum_{v\in V} d_v(d_v-1)}, \tag{68}$$

where $|\mathcal{T}|$ is the number of triangles (each counted once) and $|\mathcal{W}| = \frac{1}{2}\sum_v d_v(d_v-1) = \sum_v \binom{d_v}{2}$ is the number of wedges (paths of length two). The factor of 3 arises because each triangle contains exactly three wedges. $\mathrm{CC} \in [0,1]$: $\mathrm{CC} = 1$ means every pair of neighbours of every node are themselves connected. Note that the MMD-based clustering metric (§ C.1) uses *per-node* local clustering coefficients $c_v = 2|\{(u,w) \in E : (v,u),(v,w) \in E\}|/(d_v(d_v-1))$, whereas CC in the comparison table is the global transitivity.

Assortativity Coefficient (AC). Degree assortativity measures the tendency of high-degree nodes to connect to other high-degree nodes. Following Newman (2003), it is the Pearson correlation coefficient of degrees across all edges:

$$r \;=\; \frac{M^{-1}\displaystyle\sum_{(i,j)\in E} d_i d_j \;-\; \left[M^{-1}\sum_{(i,j)\in E}\frac{d_i+d_j}{2}\right]^2}{M^{-1}\displaystyle\sum_{(i,j)\in E}\frac{d_i^2+d_j^2}{2} \;-\; \left[M^{-1}\sum_{(i,j)\in E}\frac{d_i+d_j}{2}\right]^2}, \tag{69}$$

where $M = |E|$ and d_i, d_j are the degrees of the two endpoints of each edge. $r \in [-1,1]$: $r > 0$ indicates assortative mixing (hubs connect to hubs), $r < 0$ disassortative mixing (hubs connect to low-degree nodes, typical of biological and online social networks), and $r = 0$ random attachment.

Validity. As implemented, this metric certifies *structural* validity, not chemical validity. A generated molecule is accepted iff it is connected and every atom's degree does not exceed an empirically-derived per-type maximum valence (the observed maximum degree of that atom type across the training set). This is a necessary but not sufficient condition for a chemically realizable molecule: because the model generates an atom type per node and a binary edge topology but never bond order, a structure can satisfy every degree constraint under single-bond assumptions while remaining chemically unrealizable once the required bond multiplicities are assigned, and no RDKit sanitization is performed. We therefore report validity as certifying valence-plausibility of the generated skeleton, not synthesizability.

Uniqueness. Uniqueness is the fraction of generated molecules with a distinct Weisfeiler–Lehman graph hash (computed over topology plus integer atom-type node labels),

$$\text{uniqueness} = \frac{|\{\text{unique WL-hashes}\}|}{n_{\text{total}}}.$$

Because the WL hash omits bond-order information, two generated graphs identical in topology and atom typing but differing only in a hypothetical bond-order assignment collapse to the same hash. This is consistent with, rather than a deflation relative to, the model's actual generative space, which never produces bond order.

Novelty. Novelty is defined relative to the *unique* generated set rather than the total generated count,

$$\text{novelty} = \frac{|\{\text{unique generated}\} \setminus \{\text{train hashes}\}|}{|\{\text{unique generated}\}|},$$

matching the convention used in GDSS Jo et al. (2022), and DiGress Vignac et al. (2023) and making the value directly comparable to those baselines. Conditioning on uniqueness this way ensures a model that repeatedly duplicates a small set of novel structures is not credited more novelty than one that generates the same novel structures without duplication; duplication is fully absorbed by the uniqueness metric. The statistic is undefined (reported as NaN) when zero unique molecules are produced, the correct convention for a ratio with a zero denominator.

Maximum Mean Discrepancy (MMD) For multi-graph datasets, we generate a set of graphs $\hat{\mathcal{G}} = \{\hat{G}_i\}$ and compare its structural statistics against a reference set $\mathcal{G}^* = \{G_j^*\}$ using the biased (V-statistic) MMD estimator. Given a positive-definite kernel κ:

$$\widehat{\text{MMD}}^2(\hat{\mathcal{G}}, \mathcal{G}^*) = \frac{1}{|\hat{\mathcal{G}}|^2} \sum_{i,i'} \kappa(\hat{G}_i, \hat{G}_{i'}) - \frac{2}{|\hat{\mathcal{G}}||\mathcal{G}^*|} \sum_{i,j} \kappa(\hat{G}_i, G_j^*) + \frac{1}{|\mathcal{G}^*|^2} \sum_{j,j'} \kappa(G_j^*, G_{j'}^*). \quad (70)$$

We report four variants, differing only in how each graph is embedded and which kernel κ is applied in Eq. 70.

Histogram-based variants. The first three represent each graph G by a feature histogram h_G and use a Gaussian kernel on the Earth Mover's Distance (EMD) between histograms, $\kappa_\sigma(h, h') = \exp(-d_{\text{EMD}}(h, h')^2/2\sigma^2)$:

- **degree_mmd**: h_G is the normalised degree histogram; $\sigma = 1$.
- **clustering_mmd**: h_G is the histogram of per-node local clustering coefficients c_v, over 100 bins on $[0, 1]$; $\sigma = 0.1 \times \text{bins} = 10$.
- **orbits_mmd**: h_G is the vector of mean 4-node orbit counts per node, computed via ORCA (Hočevar & Demšar, 2014), restricted to the two 3-path orbits and the 4-cycle orbit; $\sigma = 30$.

NSPDK. The fourth variant instead embeds each graph explicitly rather than as a histogram. The Neighborhood Subgraph Pairwise Distance Kernel Costa & De Grave (2010) is computed with `complexity=4` (setting both the neighborhood radius r and the pairwise distance bound d to 4) and discrete node/edge labels, decomposing each graph into pairs of bounded-radius subgraphs at bounded shortest-path distance and hashing each pair into an explicit, fixed-dimension feature vector $\phi(G)$. On ZINC, node labels for this decomposition are the one-hot atom-type index rather than the degree-string fallback used for the non-molecular datasets, so the resulting kernel is sensitive to local chemical composition jointly with topology, not topology alone. We plug the induced kernel $\kappa_{\text{NSPDK}}(G, G') = \langle \phi(G), \phi(G') \rangle$ directly into Eq. 70 in place of κ_σ, giving

$$\widehat{\text{MMD}}^2_{\text{NSPDK}} = K_{\hat{\mathcal{G}}\hat{\mathcal{G}}} + K_{\mathcal{G}^*\mathcal{G}^*} - 2K_{\hat{\mathcal{G}}\mathcal{G}^*},$$

where each $K_{\cdot\cdot}$ denotes the corresponding averaged double sum in Eq. 70 evaluated under κ_{NSPDK}. Unlike the Gaussian-kernel variants above, this is a *linear* kernel on explicit features — an unbandwidthed inner-product statistic with no smoothing parameter to tune or misreport.

FID. Fréchet Inception Distance is adapted to graphs by replacing the image-domain Inception-v3 activations with embeddings from a Graph Isomorphism Network (GIN) (Xu et al., 2018), following the random-GNN evaluation protocol of Thompson et al. (2022) rather than a downstream-task-pretrained network.

Architecture. The GIN has 3 layers total, of which the final $L-1=2$ are graph-convolutional layers, each with hidden dimension $H=35$ and a 2-layer MLP. Neighbourhood aggregation and graph-level readout both use sum pooling. The learnable ε in

$$h_v^{(l+1)} = f_\Theta\left((1+\varepsilon)h_v^{(l)} + \sum_{u\in\mathcal{N}(v)} h_u^{(l)}\right)$$

is fixed at $\varepsilon = 0$ (not learned). All linear layers use orthogonal initialization; batch normalization and ReLU follow each MLP. By default (`use_pretrained=False`), these weights are set once and never updated — no pre-training on any graph dataset, no fine-tuning at the start of each run — relying on the WL-equivalent expressivity of GIN aggregation to produce structure-sensitive embeddings without a labeled pretraining task analogous to ImageNet classification. We adopt this frozen-GIN design consistent with the GDSS evaluation protocol on which our codebase is based (Jo et al., 2022).

Node features. As with NSPDK, node inputs are the dataset's own attribute when present (atom type, for ZINC) and scalar node degree $h_v^{(0)} = d_v \in \mathbb{R}^1$ otherwise; on ZINC this again folds chemical composition into the embedding jointly with topology, not topology alone.

Graph embedding. The graph-level embedding concatenates the sum-pooled node representations from both convolutional layers,

$$\mathbf{z}_G = \left[\,\mathrm{SumPool}(H^{(1)}) \,\|\, \mathrm{SumPool}(H^{(2)})\,\right] \in \mathbb{R}^{35+35} = \mathbb{R}^{70}, \tag{71}$$

where $H^{(l)} \in \mathbb{R}^{N\times 35}$ is the matrix of node representations after layer l. Activations for both sets are standardized with a `StandardScaler` *fit on the reference set only* and then applied to both, avoiding calibration leakage from the generated distribution.

Fréchet distance. Each standardized activation set is summarized by a mean μ and covariance Σ, and the two Gaussians are compared via the closed-form Fréchet distance

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2\,(\Sigma_r\Sigma_g)^{1/2}\Big)\,,$$

computed with Dougal Sutherland's numerically stable matrix square root (adapted from `mseitzer/pytorch-fid`), with a diagonal ϵ-offset fallback when $\Sigma_r\Sigma_g$ is near-singular.

This closed form is exact only under a multivariate-Gaussian assumption on the embeddings, so FID here — as in the image domain — is a moment-matching approximation to distributional distance, not an exact one. in addition, since feature extractor is untrained and randomly initialized rather than a fixed, universally shared network, the embedding space (and thus the absolute FID scale) is tied to this GIN's specific initialization — closer in spirit to a random-kitchen-sink kernel (Rahimi & Recht, 2007) than to a semantically learned feature space. FID values here are therefore meaningful for comparisons *within* a consistent evaluation setup, but not as an absolute, cross-paper-comparable quantity unless the same feature extractor and seed are reproduced exactly.