LGDC: Latent Graph Diffusion via Spectrum-Preserving Coarsening

Nagham Osman*
University College London
London, UK
nagham.osman.21@ucl.ac.uk

Keyue Jiang*
University College London
London, UK

Davide Buffelli
MediaTek Research
London, UK

Xiaowen Dong
University of Oxford
Oxford, UK

Laura Toni
University College London
London, UK

Abstract

Graph generation is a critical task across scientific domains. Existing methods fall broadly into two categories: autoregressive models, which iteratively expand graphs, and one-shot models, such as diffusion, which generate the full graph at once. In this work, we provide an analysis of these two paradigms and reveal a key trade-off: autoregressive models stand out in capturing fine-grained local structures, such as degree and clustering properties, whereas one-shot models excel at modeling global patterns, such as spectral distributions. Building on this, we propose LGDC (latent graph diffusion via spectrum-preserving coarsening), a hybrid framework that combines strengths of both approaches. LGDC employs a spectrum-preserving coarsening-decoarsening to bidirectionally map between graphs and a latent space, where diffusion efficiently generates latent graphs before expansion restores detail. This design captures both local and global properties with improved efficiency. Empirically, LGDC matches autoregressive models on locally structured datasets (Tree) and diffusion models on globally structured ones (Planar, Community-20), validating the benefits of hybrid generation.

1 Introduction

Graph generation underpins applications in drug discovery, molecular design, and social networks. Most approaches fall into two classes: (i) *autoregressive models*, which construct graphs through iterative local expansion, and (ii) *one-shot models*, which fix the graph size and generate the full structure in a single pass.

Early work on graph generation largely followed the autoregressive paradigm. GraphRNN You et al. [2018] pioneered this direction by modeling node interactions as a sequence of connection events. Advances in sequential modeling, particularly transformers, led to the emergence of a family of autoregressive graph generators, including GRAN [Liao et al., 2019], BiGG [Dai et al., 2020a], and GraphGen [Goyal et al., 2020a]. The most advanced system in this line of research is HSpectre [Bergmeister et al., 2024], which frames generation as an iterative expansion process that grows a subgraph into a full graph; although architecturally distinct, each prediction step remains a local expansion, keeping it within the autoregressive family.

More recently, one-shot generative models [Jo et al., 2022, Haefeli et al., 2022a, Vignac et al., 2023] have gained prominence. Key approaches include diffusion models [Niu et al., 2020, Vignac et al.,

*Core contributors

NeurIPS 2025 New Perspectives in Advancing Graph Machine Learning Workshop

2023, Haefeli et al., 2022b, Xu et al., 2024, Siraudin et al., 2024] and flow-based models [Eijkelboom et al., 2024, Qin et al., 2024, Hou et al., 2024, Jiang et al., 2025], which generate entire graphs in a single shot, unlike autoregressive expansion. These models learn progressive transformations between a reference distribution and the data distribution by parameterizing the reverse denoising process with a neural network to enable sampling.

While both autoregressive and one-shot approaches have achieved considerable success in graph generation, a systematic understanding of their principles and trade-offs remains limited. This motivates the central question of this work:

How do autoregressive and one-shot models compare, and can we combine their strengths?

We hypothesize a trade-off between the two paradigms: autoregressive models, which generate graphs through sequential local expansions, excel at fine-grained local dependencies (e.g., parent–child links in Tree graphs) but often lose long-range coherence. Conversely, one-shot models learn holistic transformations of the adjacency structure, capturing global patterns, such as community structure or planarity, though often at the expense of local detail. This intuition guides our evaluation on datasets that stress different aspects: Tree graphs emphasize local attachment rules, while Planar and SBM graphs stress global organization. This intuition guides our evaluation on datasets that stress different aspects: Tree graphs emphasize local attachment rules, while Planar and SBM graphs stress global organization. This intuition is supported by Table 1, which summarizes training dataset characteristics across local and global metrics. Tree graphs are dominated by local dependencies, reflected in their stable local statistics and minimal global variation, whereas Planar and SBM graphs exhibit pronounced global organization in spectral and connectivity metrics, aligning with our hypothesis of local–global structural contrast. As shown in Table 2, autoregressive models perform best on Tree graphs, while one-shot models dominate on Planar and SBM graphs, confirming this trade-off.

Table 1: Training Dataset Statistics across local and global metrics.

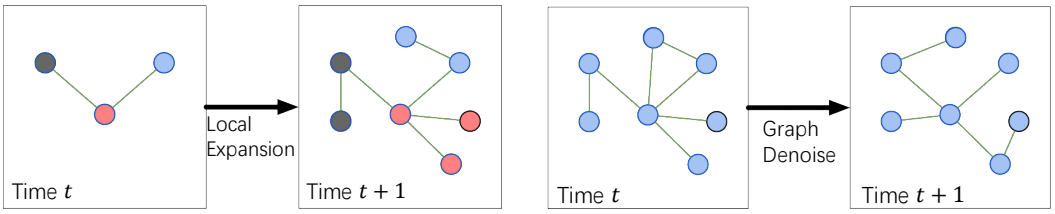
Local Metrics				**Global Metrics**			
Metric	**Tree**	**Planar**	**SBM**	**Metric**	**Tree**	**Planar**	**SBM**
Degree	0.0002	0.0000	0.0003	Spectre	0.0091	0.0076	0.0060
Orbit	0.0000	0.0001	0.0310	Components	0.0000	0.0000	0.0004
Motif	0.0000	0.0007	0.0346	Edge Conn.	0.0000	0.0039	0.0205
Clustering	0.0000	0.0165	0.0331	ASPL	0.0065	0.0000	0.0009
				Diameter	0.0241	0.0002	0.0001

Table 2: Graph Generation Comparison. Autoregression models v.s. One-Shot generation. For a fair comparison, we disabled the training designs that are unrelated to the fundamental generation path mechanism, such as the target guidance DeFoG and the predictor-corrector in Cometh. Details regarding the evaluation metrics can be found in Section 3.1.

		Planar		Tree		SBM	
Model	Class	V.U.N.↑	A.Ratio↓	V.U.N.↑	A.Ratio↓	V.U.N.↑	A.Ratio↓
Train set	—	100	1.0	100	1.0	85.9	1.0
BiGG [Dai et al., 2020b]	Autoregressive	5.0	16.0	75.0	5.2	10.0	11.9
GraphGen [Goyal et al., 2020b]	Autoregressive	7.5	210.3	95.0	33.2	5.0	48.8
HSpectre [Bergmeister et al., 2024]	Autoregression	62.5	2.9	82.5	2.1	45.0	10.2
GruM [Jo et al., 2024]	One-shot	74.4±5.15	3.2±0.4	/	/	73.5±6.7	2.6±0.6
Cometh [Siraudin et al., 2024]	One-shot	75.5± 7.37	3.0± 5.6	69.5± 3.6	1.40± 0.4	65.5±4.5	4.7±0.6
DeFoG [Qin et al., 2024]	One-shot	77.5±8.37	3.5±1.7	73.1±11.4	1.50±0.3	85.0±7.1	3.7±0.9

Motivated by this complementarity, we propose a hybrid framework, *latent graph diffusion via spectrum-preserving coarsening (LGDC)*[2]. The intuition is as follows: one-shot diffusion captures global structure, but can be expensive on the original graph and may blur fine-grained connectivity. We instead run diffusion on a coarsened latent graph for efficient global modeling. To ensure global faithfulness, we adopt spectrum-preserving coarsening under restricted spectral similarity, which keeps the principal Laplacian eigenvalues and eigenspaces of the coarse and original graphs

[2]In this work, 'latent space' refers to the coarsened graph domain, not a Euclidean embedding.



(a) Graph generation with local expansion. The color represents the child nodes expanded from a same ancestral node.

(b) Graph Generation with iterative denoising. The number of nodes are fixed, where every individual node and edge have the possibility to change.

Figure 1: Two categories of graph generation methods.

close [Loukas and Vandergheynst, 2018], allowing coarse eigenvectors to substitute for the originals, yielding a compact yet globally coherent surrogate. A single autoregressive-inspired expansion-and-refinement step during decoding then restores local connectivity. While Table 2 does not directly evaluate our hybrid, it illustrates the motivating local–global trade-off: diffusion supplies global coherence, while autoregressive refinement sharpens local structure.

Concretely, LGDC follows a two-stage framework: a diffusion model first samples a small fixed-size latent graph $\mathcal{G}_c$, which is then expanded back to a graph $\mathcal{G}$ in the original space via a decoarsening model. Training leverages graph–coarsened graph pairs $(\mathcal{G}, \mathcal{G}_c)$ from spectrum-preserving coarsening, enabling joint optimization of the latent diffusion model $p_\theta(\mathcal{G}_c)$ and expansion model $p_\phi(\mathcal{G} \mid \mathcal{G}_c)$.

Our contributions are summarized as follows:

- We propose LGDC, a unified graph generation framework combining spectrum-preserving coarsening, latent diffusion, and a single expansion–refinement step for scalable generation.
- LGDC captures both local (e.g., neighborhoods) and global (e.g., spectral) structures in one generative process. Empirically, it achieves results comparable to diffusion models on globally structured datasets (Planar, Community-20) and to autoregressive models on locally structured ones (Tree), demonstrating LGDC's strong balance between fine-grained accuracy and global coherence while maintaining high sample validity and diversity.
- LGDC achieves high efficiency and lower computational complexity compared to pure diffusion or autoregressive models (e.g., HSpectre). For graph size n, latent size n_c, and sampling steps T, its complexity is $O(n^2+Tn_c^2)$ versus $O(Tn^2)$ for one-shot and $O(Tn^2/3)$ for autoregressive generation assuming a same architecture for backbone models.

2 Methodology

In this section, we introduce *LGDC*, a hybrid graph generation model that first samples a graph in the latent space and expands it back to the original space. An overview is shown in Fig. 2 and Eq. 1.

$$\mathcal{G} \xrightarrow{\text{coarsen (3)}} \mathcal{G}_c \xrightarrow{\text{diffuse/denoise (9)–(8)}} \hat{\mathcal{G}}_c \xrightarrow{\text{one-shot expand/refine (10),(11)}} \hat{\mathcal{G}}. \tag{1}$$

The subsequent sections present the core components of our approach: Subsection 2.1 introduces the problem formulation, Subsection 2.2 details the spectrum-preserving graph coarsening module, Subsection 2.3 describes the latent-space diffusion process, Subsection 2.4 outlines the parameterization of the expansion model, and Subsection 2.5 summarizes the sampling pipeline of the framework.

2.1 Problem Formulation

We start by formalizing the problem. Let $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ denote a graph with n nodes, node features $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, and edge types encoded in $\boldsymbol{A} \in \{0,1\}^{n\times n}$. The graph generation problem aims to approximate the data distribution $p(\mathcal{G})$ and sample novel graphs from it. Instead of modeling $p(\mathcal{G})$ directly in the original space, we introduce a coarsened latent graph representation $\mathcal{G}_c = (\boldsymbol{X}_c, \boldsymbol{A}_c)$ of fixed size $n_c \ll n$. The generative process is parameterized by neural networks and expressed as:

$$p(\mathcal{G}) = \mathbb{E}_{p_\theta(\mathcal{G}_c)} p_\phi(\mathcal{G} \mid \mathcal{G}_c), \tag{2}$$

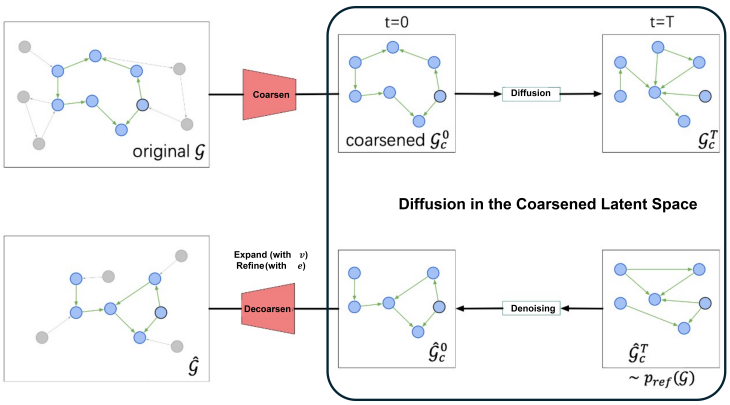


Figure 2: The architecture of LGDC framework (Latent Graph Diffusion via Coarsening)

where $p_\theta(\mathcal{G}_c)$ defines a distribution over latent graphs, and $p_\phi(\mathcal{G} \mid \mathcal{G}_c)$ maps them back to the original space.

2.2 Spectrum-Preserving Graph Coarsening

Our approach leverages latent diffusion to model a graph's global structure by operating on coarsened graphs that preserve key global properties of the original. We adopt spectrum-preserving coarsening under restricted spectral similarity [Loukas and Vandergheynst, 2018], which guarantees that the principal Laplacian eigenvalues and eigenspaces of the coarse and original graphs remain close. Since these spectral quantities encode fundamental topological characteristics such as connectivity, community structure, and planarity, the resulting coarse graph serves as a compact yet globally faithful surrogate [Bergmeister et al., 2024].

Formally, given a graph $\mathcal{G}$ with Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ and a projection matrix $\boldsymbol{C} \in \{0,1\}^{n_c \times n}$ that maps each fine node to a coarse one, the coarsened graph $\mathcal{G}_c = (\boldsymbol{X}_c, \boldsymbol{A}_c)$ is defined as

$$\boldsymbol{X}_c = \boldsymbol{C}\,\boldsymbol{X}, \quad \boldsymbol{A}_c = \boldsymbol{D}_c - \boldsymbol{L}_c, \text{ with } \boldsymbol{L}_c \;=\; \boldsymbol{C}\,\boldsymbol{L}\,\boldsymbol{C}^\top. \tag{3}$$

The coarse graph is spectrally ϵ-approximate to the original if

$$(1-\epsilon)\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}) \;\le\; \mathrm{Tr}(\boldsymbol{X}_c^\top \boldsymbol{L}_c \boldsymbol{X}_c) \;\le\; (1+\epsilon)\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}). \tag{4}$$

This restricted similarity criterion ensures the coarse graph preserves the spectral energy of the original, enabling it to mimic global structural patterns. To satisfy this condition, we adopt the REC algorithm [Loukas and Vandergheynst, 2018], a non-learning coarsening method in Algorithm 1.

2.3 Discrete Diffusion in the Latent Space

After coarsening the graphs, we lean on a discrete diffusion similar to Vignac et al. [2023] for parameterizing $p_\theta(\mathcal{G}_c)$. Specifically, our goal is to build a diffusion model that transforms the samples from an easy-to-sample source distribution, $\mathcal{G}_c^T \sim p_T(\mathcal{G}_c)$, to samples from the target distribution, $\mathcal{G}_c^0 \sim p(\mathcal{G}_c)$. This is done through defining an iterative sampling process, where

$$p_\theta(\mathcal{G}_c^0) = \int_{\mathcal{G}_c^{[1:T-1]}} p(\mathcal{G}_c^0 \mid \mathcal{G}_c^1) \cdot \ldots p(\mathcal{G}_c^{t-1} \mid \mathcal{G}_c^t) \cdot \ldots p(\mathcal{G}_c^{T-1} \mid \mathcal{G}_c^T) p(\mathcal{G}_c^T) d\mathcal{G}_c^{[1:T-1]} \tag{5}$$

where we denote $d\mathcal{G}_c^{[1:T-1]} \triangleq d\mathcal{G}_c^{T-1}, \ldots d\mathcal{G}_c^1$. Thus, the model design translates to fitting a transition model, $p_\theta(\mathcal{G}_c^{t-1} \mid \mathcal{G}_c^t)$. To this end, we follow the permutation-equivariant graph transformer of Vignac et al. [2023] and decompose the transition model as,

$$p_\theta(\mathcal{G}_c^{t-1} \mid \mathcal{G}_c^t) = p_\theta(\boldsymbol{X}_c^{t-1} \mid \mathcal{G}_c^t) \cdot p_\theta(\boldsymbol{A}_c^{t-1} \mid \mathcal{G}_c^t) \tag{6}$$

$$\text{with } p_\theta(\boldsymbol{X}_c^{t-1} \mid \mathcal{G}_c^t) = \sum_{\boldsymbol{X}_c^0} p\big(\boldsymbol{X}_c^{t-1} \mid \boldsymbol{X}_c^0, \boldsymbol{X}_c^t\big)\; p_\theta(\boldsymbol{X}_c^0 \mid \mathcal{G}_c^t), \tag{7}$$

$$\text{and } p_\theta(\boldsymbol{A}_c^{(t-1)} \mid \mathcal{G}_c^{(t)}) = \sum_{\boldsymbol{A}_c^0} p\big(\boldsymbol{A}_c^{t-1} \mid \boldsymbol{A}_c^0, \boldsymbol{A}_c^t\big)\; p_\theta(\boldsymbol{A}_c^0 \mid \mathcal{G}_c^t). \tag{8}$$

where we parameterize the joint reconstruction probability by factorizing the node and edge predictions: $p_\theta(\mathcal{G}_c^0 \mid \mathcal{G}_c^t) = p_\theta(\boldsymbol{X}_c^0 \mid \mathcal{G}_c^t) \cdot p_\theta(\boldsymbol{A}_c^0 \mid \mathcal{G}_c^t)$. The model is trained to reconstruct the original graph $\mathcal{G}_c^0$ from its noisy version $\mathcal{G}_c^t$. We follow Vignac et al. [2023] and construct the noisy graph iteratively perturbing the clean graph through,

$$p(\mathcal{G}_c^t \mid \mathcal{G}_c^{t-1}) \;=\; \big(X^{t-1}Q_X^t,\, E^{t-1}Q_E^t\big), \text{such that } p(\mathcal{G}_c^{(t)} \mid \mathcal{G}_c^0) \;=\; \big(X\,Q_X^{(1:t)},\, E\,Q_E^{(1:t)}\big), \tag{9}$$

where per–step transitions are defined by $Q_X^{(t)} \in [0,1]^{a\times a}$ and $Q_E^{(t)} \in [0,1]^{b\times b}$,which we refer to Vignac et al. [2023] for a detailed introduction.

Training objective. The parameters θ are optimized to reconstruct the clean graph $\mathcal{G}_c^0$ from its noisy counterpart $\mathcal{G}_c^t$ at any timestep t. This is achieved by minimizing the combined cross-entropy loss for both node feature and adjacency matrix reconstruction.

2.4 Expansion and Refinement

To reconstruct a full-resolution graph in a single decoding pass, we adopt the *expansion–refinement* formalism of Bergmeister et al. [2024]. Let $\boldsymbol{v} \in \mathbb{N}^{n_c}$ denote the *node expansion vector*, where $\boldsymbol{v}_i$ specifies how many fine nodes are generated from each coarse node i in $\mathcal{G}_c$, such that $\sum_{i=1}^{n_c} \boldsymbol{v}_i = n$. Let $\boldsymbol{e} \in \{0,1\}^{|\tilde{\mathcal{E}}|}$ be an *edge-selection mask* over a set of candidate edges $\tilde{\mathcal{E}}$.

Expansion. The expansion operator $\tilde{\mathcal{G}} = \mathrm{Expand}(\mathcal{G}_c; \boldsymbol{v})$ replicates each coarse node i into $\boldsymbol{v}_i$ fine nodes. Candidate edges are then constructed in two groups: (i) **intra-cluster edges**, connecting fine nodes that originate from the same coarse node (modeling local substructure within a cluster); and (ii) **inter-cluster edges**, connecting nodes that originate from different coarse nodes i and j whenever (i,j) is an edge in the coarse graph $\mathcal{G}_c$. In other words, inter-cluster connectivity is inferred from the coarse-level adjacency, serving as a structural scaffold for generating fine-level links between clusters. This process yields a dense candidate edge set $\tilde{\mathcal{E}}$ used in the subsequent refinement step.

Refinement. The refinement step then applies the binary mask $\boldsymbol{e}$ to prune edges and form the final reconstructed graph $\hat{\mathcal{G}}$:

$$\mathcal{G}_c \;\xrightarrow{\;\boldsymbol{v}\;}\; \tilde{\mathcal{G}} \;\xrightarrow{\;\boldsymbol{e}\;}\; \hat{\mathcal{G}}. \tag{10}$$

Hence, $\boldsymbol{e}_k = 1$ indicates that the k-th candidate edge in $\tilde{\mathcal{E}}$ is retained in $\hat{\mathcal{G}}$.

Parameterization and Training. Since $\mathcal{G}$ is uniquely determined by $(\boldsymbol{v}, \boldsymbol{e})$, the conditional model is factorized as:

$$p_\phi(\mathcal{G} \mid \mathcal{G}_c) = p_\phi(\boldsymbol{v}, \boldsymbol{e} \mid \mathcal{G}_c) = p_\phi(\boldsymbol{v} \mid \mathcal{G}_c)\, p_\phi(\boldsymbol{e} \mid \mathrm{Expand}(\mathcal{G}_c; \boldsymbol{v}), \mathcal{G}_c). \tag{11}$$

The supervision pairs $(\boldsymbol{v}^\star, \boldsymbol{e}^\star)$ are obtained by inverting one spectrum-preserving coarsening step. Model parameters ϕ are optimized to maximize the reconstruction likelihood (or equivalently the ELBO) following Bergmeister et al. [2024], applied here to a single coarse-to-fine transition.

2.5 Sampling Pipeline

At test time, LGDC generates in two stages: (i) Latent diffusion starting from the prior at step T, we apply the DiGress-style reverse transitions (Eqs. 9–8) to obtain a sampled latent graph $\hat{\mathcal{G}}_c$, and (ii) One–shot decoding where we decode in a single coarse to fine pass by drawing $(\boldsymbol{v}, \boldsymbol{e}) \sim p_\phi(\cdot \mid \hat{\mathcal{G}}_c)$ via Eq. 11 and applying Eq. 10 to produce $\hat{\mathcal{G}}$. Unlike Bergmeister et al. [2024], which uses multiple expansion–refinement cycles (e.g., $1 \to 3 \to 8 \to \cdots \to n$), LGDC performs a single coarse to fine inversion from n_c to n at training and testing that significantly reduce the computation complexity.

End-to-end Complexity. LGDC's inference cost decomposes into (i) latent denoising on $\mathcal{G}_c$, which is $O(T\,n_c^2)$, and (ii) one-shot expand/refine, which is $O(n^2)$ in the dense worst case. Thus the overall complexity is $O\big(n^2 + T\,n_c^2\big)$. The sampling complexity calculation is in Appendix A.

Table 3: Graph Generation Results. We compare the most advanced autoregressive and one-shot methods and disable target guidance for fair comparison. The A.Ratio. result on Planar excludes Orbit ratios. The HSepctre results are taken from Bergmeister et al. [2024] with one round of expansion.

		Planar		Tree		Community-20		
Model	Class	V.U.N. ↑	A.Ratio ↓	V.U.N. ↑	A.Ratio ↓	Deg. ↓	Clus. ↓	Orb. ↓
HSpectre [Bergmeister et al., 2024]	Autoregression	62.5	**2.90**	82.5	2.10	/	/	/
DeFoG Qin et al. [2024]	One-shot	77.5±8.37	4.07	73.1±11.4	**1.50**	0.071	0.115	0.037
LGDC (Ours)	Hybrid	**82.5±2.7**	3.06	**86.0±2.0**	1.70	**0.037**	**0.027**	**0.007**

3 Experimental Results

3.1 Datasets and Evaluation Protocol

Datasets. We evaluate LGDC on three synthetic benchmarks that probe complementary structural regimes: (i) *Tree* graphs [Bergmeister et al., 2024], which are deterministic and acyclic, testing local structural fidelity; (ii) *Planar* graphs [Martinkus et al., 2022], which impose global planarity while still requiring consistent local layouts; and (iii) *Community-20* graphs [Vignac et al., 2023], a small SBM dataset with connected clusters, challenging models to capture global community structure.

Setup. We fix the coarsening ratio to approximately $n_c \approx n/5$ to balance fidelity and cost under a fixed compute budget. This ensures the latent graph remains large enough to preserve global structure while reducing the quadratic cost of diffusion in that space. Concretely, we set $M = 16$ for Tree and Planar, and $M = 4$ for Community-20. A broader sweep of coarsening ratios is left to future work.

Evaluation. We assess generated graphs on sample quality and structural fidelity. Table 3 reports validity, uniqueness, and novelty (V.U.N.), and the average ratio (A. Ratio), summarizing deviations of standard statistics from the reference distribution. To probe fidelity in detail, Table 4 separates *local* metrics (degree, clustering, orbit/motif), *global* metrics (spectral distance, connectivity, diameter/ASPL), and *cross-scale* wavelet statistics based on Laplacian eigen-decomposition, which combine both aspects. For compactness, we also report Local and Global Ratios, normalizing deviations against reference sets.

3.2 Performance and Complexity Analysis

Generation Performance. The results in Table 3 show that LGDC achieves the strongest performance on the locally-focused Tree dataset, reaching a V.U.N. score of 86.0 that surpasses even the specialized autoregressive HSpectre. On the globally oriented Planar and Community-20 graphs, LGDC also delivers competitive results: it improves V.U.N. on Planar to 82.5, outperforming both HSpectre and DeFoG, and attains consistently low discrepancy scores on Community-20 across statistics. These outcomes highlight LGDC's strength as a hybrid model that unifies the complementary advantages of autoregressive and one-shot paradigms across both local and global generation tasks.

Generation Complexity. Standard one-shot graph generative models require $O(Tn^2)$ FLOPs, while autoregressive methods average $O(Tn^2/3)$. In contrast, LGDC operates much more efficiently, requiring only $O\big(n^2 + T\,n_c^2\big)$ FLOPs, when $n_c \ll n$ (see Appendix A.2). Since T typically exceeds n in graph generation tasks, the savings are substantial, and with small compression ratio n_c/n, LGDC achieves significant computational gains over both paradigms.

Capturing Local and Global Patterns. Table 4 provides a quantitative analysis of LGDC's performance before and after the expansion step. Each subtable corresponds to one dataset, while the two columns (**Diffusion** vs. **Expansion**) represent the coarsened latent graphs ($\mathcal{G}_c$) and the reconstructed full graphs ($\hat{\mathcal{G}}$), respectively. Metrics are grouped into **Local**, **Global**, and **Local+Global** categories. Local metrics (Degree, Clustering, Motif/Orbit) measure fine-grained connectivity; global metrics (Spectral distance, Edge Connectivity, Components, Diameter, ASPL) reflect large-scale structure; and joint metrics (Wavelet distance and V.U.N.) assess multi-scale consistency between both levels.

Numerically, the *Local* metrics in all datasets show clear improvement after expansion, for instance, degree and motif errors typically decrease by one order of magnitude (e.g., from 0.0016 to 0.0006

or 0.0697 to 0.0079), confirming that the refinement step successfully restores node-level structure and motif statistics. The *Global* metrics remain of the same scale before and after expansion, with small variations in spectral or diameter scores (e.g., spectral distance changes from 0.0085 to 0.0078 or 0.0126 to 0.0090), showing that the overall topological organization learned in the latent space is largely preserved. Meanwhile, the *Wavelet* metric, which combines both local and global information, typically decreases after expansion (e.g., $0.0078 \rightarrow 0.0041$ or $0.0089 \rightarrow 0.0063$), indicating improved multi-scale coherence in the reconstructed graphs. V.U.N. scores also remain high (above 80 in all cases), verifying that sample validity and diversity are retained.

Overall, these numerical patterns demonstrate that the latent diffusion step captures coherent global structure, while the expansion–refinement step recovers fine-scale connectivity without sacrificing global consistency. The complementary trends across all metric groups confirm LGDC's ability to produce graphs that are simultaneously globally organized and locally accurate within a single decoding stage.

Table 4: Evaluation of LGDC across datasets. "Diffusion" refers to graphs in the coarsened latent space, and "Expansion" shows the same metrics after decoding to the original size.

(a) Tree

Metric	Diffusion	Expansion
Local		
Degree	0.0005 ± 0.0003	0.0002 ± 0.0002
Orbit	0.0001 ± 0.0000	0.0000 ± 0.0000
Clustering	0.0000 ± 0.0000	0.0000 ± 0.0000
L. Ratio	4.22 ± 3.11	0.86 ± 0.52
Global		
Spectre	0.0085 ± 0.0012	0.0078 ± 0.0009
Components	0.0003 ± 0.0002	0.0088 ± 0.0053
Diameter	0.0249 ± 0.0120	0.0789 ± 0.0202
G. Ratio	1.79 ± 0.63	2.21 ± 0.31
Local + Global		
Wavelet	0.0078 ± 0.0004	0.0041 ± 0.0003
VUN	97.5 ± 1.58	86.0 ± 2.0

(b) Planar

Metric	Diffusion	Expansion
Local		
Degree	0.0016 ± 0.0005	0.0006 ± 0.0001
Clustering	0.0553 ± 0.0020	0.0509 ± 0.0133
Motif	0.0697 ± 0.0131	0.0079 ± 0.0030
L. Ratio	4.81 ± 1.52	19.88 ± 3.51
Global		
Spectre	0.0126 ± 0.0021	0.0090 ± 0.0011
Edge Conn.	0.0162 ± 0.0036	0.0099 ± 0.0030
Diameter	0.0047 ± 0.0046	0.0106 ± 0.0086
G. Ratio	9.03 ± 6.81	19.46 ± 7.34
Local + Global		
Wavelet	0.0089 ± 0.0008	0.0063 ± 0.0005
VUN	100.0 ± 0.0	82.5 ± 2.74

(c) Community-20

Metric	Diffusion	Expansion
Local		
Degree	0.0060 ± 0.0030	0.0369 ± 0.0028
Clustering	0.0322 ± 0.0193	0.0266 ± 0.0023
Motif	0.0030 ± 0.0012	0.0317 ± 0.0059
L. Ratio	0.45 ± 0.24	1.92 ± 0.69
Global		
Spectre	0.0457 ± 0.0175	0.1431 ± 0.0084
Components	0.0021 ± 0.0016	0.0029 ± 0.0017
ASPL	0.0022 ± 0.0020	0.0014 ± 0.0012
G. Ratio	0.53 ± 0.32	1.18 ± 0.14
Local + Global		
Wavelet	0.0410 ± 0.0166	0.1175 ± 0.0061
VUN	—	—

4 Conclusion

We presented *LGDC*, a hybrid graph generator that combines spectrum-preserving coarsening with latent discrete diffusion and a single expansion–refinement step. By design, this approach unifies the complementary strengths of one-shot and autoregressive paradigms: diffusion in a compact latent space imposes global organization at low cost, while local-expansion based decoding restores fine local structure. Empirically, LGDC achieves strong V.U.N. and competitive discrepancy scores with an end-to-end sampling complexity of $O(n^2+Tn_c^2)$, offering practical efficiency. Nonetheless, several limitations remain. The one-shot expand–refine decoding can be error-prone: small inaccuracies in the edge mask $\boldsymbol{e}$ may propagate when the candidate edge set $\tilde{\mathcal{E}}$ is large, occasionally disturbing coarse-level consistency. Although latent diffusion is efficient, the expansion step still scales as $O(n^2)$ in the dense case, making very large graphs challenging to handle. The model's performance also depends on the chosen coarsening procedure and compression ratio n_c/n, as different projection matrices can yield variable cluster structures and reconstruction difficulty. Finally, the current refinement does not explicitly enforce global constraints such as planarity or connectivity, and our experiments focus on synthetic benchmarks rather than large, real-world graphs. Future work will explore iterative or uncertainty-aware refinement to mitigate decoding errors, sparse or structured expansion to improve scalability, and adaptive or learned spectrum-preserving coarsening for greater robustness. Extending LGDC to richer node and edge attributes and larger heterogeneous graphs offers a promising path toward scalable, structure-aware generative modeling across domains.

References

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR, 2018.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. *Efficient graph generation with graph recurrent attention networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2302–2312. PMLR, 2020a.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *WWW*, pages 1253–1263. ACM / IW3C2, 2020a.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *ICLR*. OpenReview.net, 2024.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 10362–10383. PMLR, 2022.

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

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

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 2020.

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. In *The First Learning on Graphs Conference*, 2022b. URL `https://openreview.net/forum?id=CtsKBwhTMKg`.

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. *arXiv preprint arXiv:2405.11416*, 2024.

Antoine Siraudin, Fragkiskos D. Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model, 2024. URL `https://arxiv.org/abs/2406.06449`.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *NeurIPS*, 2024.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *CoRR*, abs/2410.04263, 2024.

Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving molecular graph generation with flow matching and optimal transport. *CoRR*, abs/2411.05676, 2024.

Keyue Jiang, Jiahao Cui, Xiaowen Dong, and Laura Toni. Bures-wasserstein flow matching for graph generation. In *ICML 2025 Generative AI and Biology (GenBio) Workshop*, 2025. URL `https://openreview.net/forum?id=qjAkn9mW4C`.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2302–2312. PMLR, 2020b.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *WWW*, pages 1253–1263. ACM / IW3C2, 2020b.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *ICML*. OpenReview.net, 2024.

Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3243–3252. PMLR, 2018.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedingsof the 39th International Conferenceon Machine Learning*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685. IEEE, 2022.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In *NeurIPS*, pages 11287–11302, 2021.

Yinglong Guo, Dongmian Zou, and Gilad Lerman. An unpooling layer for graph generation. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 3179–3209. PMLR, 2023.

Truong Son Hy and Risi Kondor. Multiresolution equivariant graph variational autoencoder. *Mach. Learn. Sci. Technol.*, 4(1):15031, 2023.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4805–4815, 2018.

Ling Yang, Zhilin Huang, Zhilong Zhang, Zhongyi Liu, Shenda Hong, Wentao Zhang, Wenming Yang, Bin Cui, and Luxia Zhang. Graphusion: Latent diffusion for graph generation. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–12, 2024. doi: 10.1109/TKDE.2024.3389783.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Latent graph diffusion: A unified framework for generation and prediction on graphs. *CoRR*, abs/2402.02518, 2024.

A Complexity Comparison

A.1 Scalability in Graph Generation and the Computational Complexity Issue

Even though both autoregressive and one-shot models have achieve satisfactory results in graph generation, the scalability has long been a problem due to the heavy computation budget. In this section, we will illustrate the computational complexity of both families of graph generation methods.

The time complexity, consisting of *Training Complexity* and *Inference Complexity*, is related to the number of the floating point operations (FLOPs) (e.g. plus/multiplication) in the forward-propagation and back-propagation (where the gradient is calculated). The inference complexity consists of making predictions through the model in each step. The space complexity mainly concerns the memory usage of a model (RAM or GPU memory). The complexity of time and space is highly intertwined and is measured through the model architecture. As a brief solution, calculating each step of propagation with a specific model architecture is good enough for measuring the complexity.

The computational complexity of one-shot diffusion/flow models. The cost mainly comes from the fact that one-shot graph generation models uses a dense representation of graphs for computation, which results in a quadratic complexity w.r.t #nodes (n) and #dimension (d). The complexity of one-shot graph generation models, including Digress Vignac et al. [2023], DeFogQin et al. [2024] and BWFlowJiang et al. [2025], are generally similar as the recent development of this family of models mainly focusing on accelerating the sampling and reducing the training cost. The sampling steps T is a parameter that we cannot easily quantify. Though we assume it to be fixed here for simplicity but we also wish to emphasize its importance in graph generation.

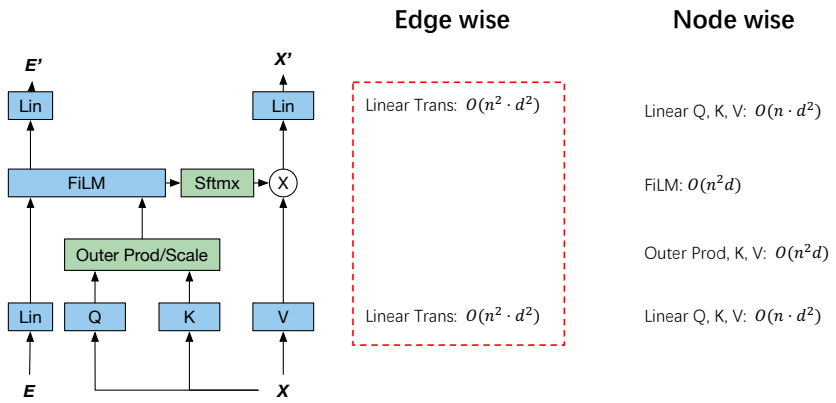


Figure 3: The time complexity of One-shot graph generation. Given the number of nodes n, dimension size d, number of layers L, and number of samples S, the overall model complexity for training is given as $O(S \cdot L \cdot (n^2 \cdot d^2 + n^2 \cdot d + n \cdot d^2))$. The inference complexity $O(T \cdot L \cdot (n^2 \cdot d^2 + n^2 \cdot d + n \cdot d^2))$ for a single sample. The leading term is $n^2 \cdot d^2$. which comes from the edge-wise prediction.

We consider the graph transformer Vignac et al. [2023] as the backbone model, which has a model architecture that is shown in 3 (simplified by removing all the conditional generation component). DiGress handles graphs with categorical node and edge attributes, represented by the spaces $\mathcal{X}$ and $\mathcal{E}$, respectively, with cardinalities a and b. We use x_i to denote the attribute of node i and $\boldsymbol{x}_i \in \mathcal{R}^a$ to denote its one-hot encoding. These encodings are organized in a matrix $\boldsymbol{X} \in \mathcal{R}^{n \times a}$ where n is the number of nodes. A tensor $\boldsymbol{E} \in \mathbb{R}^{n \times n \times b}$ groups the one-hot encoding $\boldsymbol{r}_{ij}$ of each edge, treating the absence of edge as a particular edge type. Each step of diffusion can be thought to be a joint node regression and link prediction process, where one would reverse the noising process of $q(\boldsymbol{E}_t, \boldsymbol{X}_t \mid \boldsymbol{E}_{t-1}, \boldsymbol{X}_{t-1})$, as $p_\theta(\boldsymbol{E}_{t-1}, \boldsymbol{X}_{t-1} \mid \boldsymbol{E}_t, \boldsymbol{X}_t)$. The forward path progressively corrupts the data points and formalizes a sequence, namely $(\boldsymbol{X}_0, \ldots, \boldsymbol{X}_T)$ and $(\boldsymbol{E}_0, \ldots, \boldsymbol{E}_T)$.

In the reverse path, a backbone model parameterized by θ is used to learn the process, where we denote as $p_\theta : \mathcal{X} \times \mathcal{E} \to \mathcal{X} \times \mathcal{E}$. As an example, the continuous graph generation models require sample n^2, which yields a complexity of $O(n^2 T)$. The discrete graph generation algorithm asks for a transition matrix that is capable of modeling a transition space of $\mathcal{R}^{n^2} \times \mathcal{R}^{n^2}$. Omitting the embedding size, the complexity is $O(n^2)$.

The computational complexity of auto-regressive models. Assuming the backbone model to be the same graph generation, a single step of autoregressive computation follows a similar pipeline. However, the difference comes from the fact that each expansion steps the graph transformer operates on graphs with different size. We consider T step expansion and each step expands $(n-1)/T$ nodes. Omitting the hidden dimension and number of layers, we now aim to calculate

$$S = \ 1^2 + \Big(1 + \frac{n-1}{T}\Big)^2 + \Big(1 + 2 \cdot \frac{n-1}{T}\Big)^2 + \cdots + n^2.$$

Let $d = \frac{n-1}{T}$., we simplify to$S = \sum_{t=0}^{T} \big(1 + td\big)^2$.. Expanding, we get $(1+td)^2 = 1 + 2td + t^2d^2$, so

$$\begin{aligned} S &= \sum_{t=0}^{T} \big(1 + 2td + t^2d^2\big) = \sum_{t=0}^{T} 1 \ + \ 2d \sum_{t=0}^{T} t \ + \ d^2 \sum_{t=0}^{T} j^2 \\ &= \frac{T+1}{6T} \Big(6Tn + (2T+1)(n-1)^2\Big). \end{aligned}$$

Thus, the complexity is $O(\frac{Tn^2}{3})$.

A.2 The computational complexity of latent graph generation

In the image generation task, a solution to generation is latent diffusion Rombach et al. [2022], Vahdat et al. [2021], where one first compares a high-resolution image into a low-rank latent space, and then diffuses over the latent space to achieve efficient sampling.

For this scenario, lets consider first compressing the graph into a latent space, with size n_c, we then operates on the latent space for $T/2$ step of generation, which yields $O(T/2 \cdot L \cdot (n_c^2 d^2))$ inference complexity. We then expand with one-step graph generation, which again has $O(T/2 \cdot L \cdot (n_c^2 d^2))$ FLOPs as the model only need to output the node expansion operator. Then, the edge predictor, which is a sparse graph neural network that takes in the expanded graph, which has worst $O(L \cdot (n^2 d^2))$ complexity. Thus, omitting the embedding size and layer complexity, the total computation complexity is $O(n^2 + Tn_c^2)$. Comparing to the full one-shot graph genertaion model ($O(Tn^2)$) and autoregressive model($O(Tn^2/3)$), the latent graph generation model has a significant gain in the computational efficiency as long as $\frac{n_c}{n}$ is small sufficiently.

B Related Work

Hierarchical structures for graph generation have been explored in several prior works Guo et al. [2023], Hy and Kondor [2023], Bergmeister et al. [2024]. Hy and Kondor [2023] proposed a hierarchical VAE that progressively reduces graph size from N_t to N_{t+1} through soft cluster assignments, similar to DiffPool Ying et al. [2018]. While effective for capturing multiscale structure, this approach assumes a fixed number of nodes per layer, is computationally expensive, and lacks theoretical guarantees on information preservation. On the other hand, Bergmeister et al. [2024] interpret the forward process of diffusion as graph coarsening and the reverse process as graph expansion. Their method iteratively adds and removes nodes and edges, bridging between a singleton graph and the full target distribution.

To the best of our knowledge, only two works explicitly consider latent graph diffusion Yang et al. [2024], Zhou et al. [2024]. These frameworks apply diffusion in a latent space but do not compress graphs by reducing the number of nodes (from n to n_c with $n \gg n_c$). Instead, they embed graphs into a Euclidean vector space, which limits scalability and prevents fully leveraging the efficiency benefits of latent diffusion. In contrast, our model combines spectrum-preserving graph coarsening with latent diffusion in the non-Euclidean space of graphs and a single expansion–refinement stage. This design enables efficient sampling in a substantially smaller space while maintaining a balance between global structural organization and high-fidelity local reconstruction.

C Randomized edge contraction (REC) algorithm

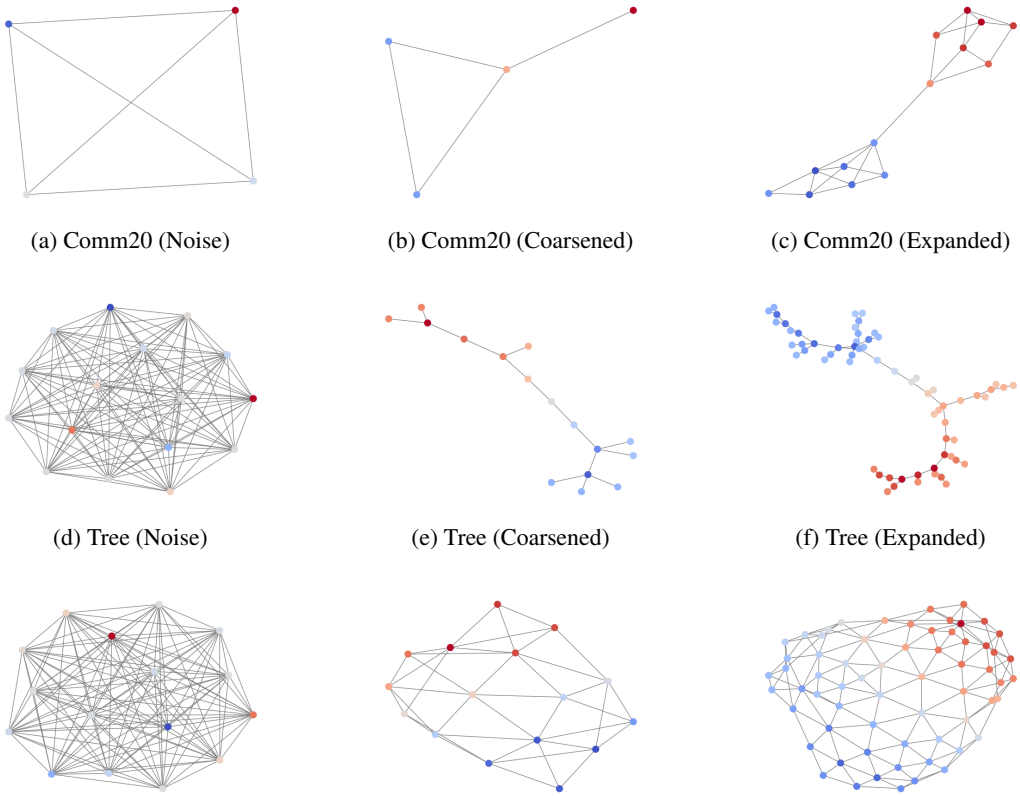
We introduce the REC algorithm in 1.

Algorithm 1: Randomized Edge Contraction (REC)

Input: Graph $G = (\boldsymbol{X}, \boldsymbol{A})$, iteration limit T.
Output: Coarsened graph $G_{\mathrm{c}} = (\boldsymbol{X}_c, \boldsymbol{A}_c)$.

$\mathcal{C} \leftarrow \mathcal{E}$, where $\mathcal{E}$ is the edge set where $\boldsymbol{A}_{ij} \neq 0$;
$G_{\mathrm{c}} \leftarrow G$;
$\Phi \leftarrow \sum_{e_{ij} \in \mathcal{E}} \boldsymbol{A}_{ij}$;
$t \leftarrow 0$;
$p_{\text{null}} \leftarrow 0$;
while $|\mathcal{C}| > 0$ *and* $t < T$ **do**
  $t \leftarrow t + 1$;
  Sample an outcome from $\mathcal{C} \cup \{\text{null}\}$ with probabilities $p_{ij} = \boldsymbol{A}_{ij}/\Phi$ and p_{null};
  if *an edge* $e_{ij} \in \mathcal{C}$ *was sampled* **then**
    $G_{\mathrm{c}} \leftarrow \mathrm{contract}(G_{\mathrm{c}}, e_{ij})$; `// as in Eq. (2)`
    Let $\mathcal{N}_{ij}$ be the set of edges incident to nodes i or j;
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{N}_{ij}$; `// Remove neighbors`
    $p_{\text{null}} \leftarrow p_{\text{null}} + \sum_{e_{pq} \in \mathcal{N}_{ij}} p_{pq}$;
  end
end
return G_{c};

D Visualizations



(a) Comm20 (Noise) (b) Comm20 (Coarsened) (c) Comm20 (Expanded)

(d) Tree (Noise) (e) Tree (Coarsened) (f) Tree (Expanded)

(g) Planar (Noise) (h) Planar (Coarsened) (i) Planar (Expanded)

Figure 4: Visualization of graph generation stages across three datasets. Rows correspond to datasets (Comm20, Tree, Planar), and columns correspond to stages (Noise, Coarsened, Expanded).