# GRAPHSCALE: SCALING CONTROLLED GRAPH GENERATION THROUGH HIERARCHICAL CONSTRAINT COMPOSITION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Controlled graph generation aims to generate graphs that satisfy specified structural constraints, yet existing methods largely trade off controllability and scalability: models that support fine-grained structural conditioning typically operate on small graphs, while scalable generators provide limited control over topology. We study whether global structural attributes can be factorized into local constraints and recovered through hierarchical composition. We introduce `GraphScale`, a model-agnostic framework that recursively decomposes target graph-level attributes into a hierarchy of subgraph constraints, generates leaf subgraphs independently, and composes them bottom-up using cross-subgraph connectivity conditioned on the remaining global structural attributes. This hierarchical factorization allows the underlying generator to operate only on bounded-size subgraphs while enabling the composed graph to scale to orders of magnitude larger sizes. Across real and synthetic datasets, `GraphScale` extends existing conditional graph generators to graphs with thousands to tens of thousands of nodes while retaining control over multiple structural properties and preserving global topology. These results demonstrate that compositional structural control offers a promising approach to scalable controlled graph generation.

## 1 INTRODUCTION

Graph generation is fundamental to modeling complex structures across domains, including social networks, biological interactions, communication infrastructures, and knowledge graphs (Newman, 2003). Real-world graphs often contain thousands to millions of nodes and edges and exhibit rich structural regularities that are critical to downstream applications. This has driven growing interest in graph generative models that learn structural patterns from real-world networks while providing explicit control over generated graph properties (Chen et al., 2023; Zahirnia et al., 2024; Vignac et al.; Karami, 2024).

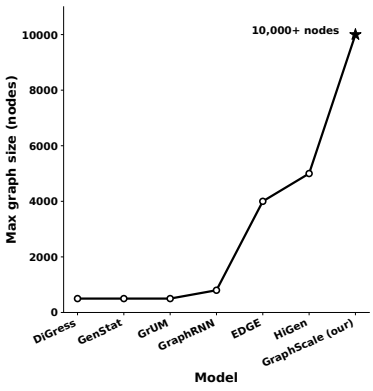



Figure 1: Scalability comparison of graph generation models reported or implied by experimental evaluations. Our `GraphScale` supports graph generation beyond 10,000+ nodes.

Recent advances in graph generation have enabled models to incorporate structural guidance into the generation process. In particular, graph generators can be conditioned on graph-level statistics, such as degree distributions, edge counts, and subgraph statistics (e.g., triangle counts), to steer the structure of generated graphs (Chen et al., 2023; Zahirnia et al., 2024).

Despite this progress, neural graph generation faces a fundamental trade-off between **controllability** and **scalability**. Existing methods typically control coarse attributes, such as node degrees and edge counts, but offer limited control over fine-grained structural attributes. These attributes capture higher-order patterns that coarse statistics miss and can substantially affect graph behavior and downstream learning (Newman, 2003; Hamilton et al., 2017; Wu et al., 2020). Control-



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.

ling them is essential for faithfully reproducing real-world structure and enabling targeted graph augmentation, simulation, and benchmarking (Zahirnia et al., 2024; Chen et al., 2023).

Moreover, existing attribute-conditioned generation methods remain difficult to scale. Most are evaluated on graphs with only a few hundred nodes (You et al., 2018; Vignac et al.), while dense adjacency representations, fixed-size latent spaces, and iterative generation often make computation and memory costs grow rapidly with graph size. Thus, methods that provide structural control at small scales do not readily extend to graphs with thousands or millions of nodes. Figure 1 highlights this gap across existing graph generation approaches. Recent diffusion- and statistics-based methods provide increasingly sophisticated attribute conditioning (Zahirnia et al., 2024; Chen et al., 2023; Vignac et al.; Jo et al., 2024), but remain limited in scalability.

**We address this gap with `GraphScale`, a model-agnostic hierarchical framework for controllable large-graph generation that decouples generator scale from graph scale.** Given target structural attributes, `GraphScale` recursively factorizes global attributes specifications into local constraints, generates bounded-size subgraphs with existing graph generators, and composes them through structure-aware cross-subgraph connectivity. It constructs the attribute hierarchy top-down, then merges subgraphs bottom-up while conditioning connectivity on the remaining global specification. This enables fine-grained structural control beyond the scale of the underlying generator. `GraphScale` offers two key advantages. First, its model-agnostic hierarchy decouples graph scale from the underlying generator, allowing existing small-graph models to generate substantially larger graphs without architectural or procedural changes. Second, it treats structural attributes as soft constraints and propagates them across hierarchy levels to guide both local generation and cross-subgraph connectivity. This provides control over diverse structural attributes while scaling beyond the limits of existing methods. Code and data will be release at GitHub.

We make the following contributions:

- `GraphScale`, a controlled graph generation framework that factorizes global attributes specification into local constraints and recovers target graphs through hierarchical composition.
- A *model-agnostic hierarchical composition principle* that generates bounded-size subgraphs independently and composes them bottom-up via connectivity conditioned on remaining global requirements, decoupling generator scale from graph scale.
- We show that hierarchical composition preserves global structural "*patterns*" beyond local statistics like singular-vector and degree distributions, and singular-value spectra .
- Across real and synthetic datasets, we scale controlled graph generation to thousands of nodes while preserving structural attributes beyond node degrees and edge counts.

## 2 Related Work

**Graph Generation and Scalability.** Graph generation spans non-deep-learning methods (Chakrabarti & Faloutsos, 2006; Akoglu & Faloutsos, 2009; Akoglu et al., 2008) and deep generative models. Deep graph generation has progressed from sequential autoregressive models such as GraphRNN (You et al., 2018) to approaches that exploit sparsity, structured representations, and hierarchical decompositions. BiGG (Dai et al., 2020) reduces autoregressive costs through parallelization, while HGGT (Jang et al., 2024) and HiGen (Karami, 2024) use hierarchical representations for large sparse graphs. Graphusion (Yang et al., 2024b) instead generates graphs in a lower-dimensional, topology-aware latent space. Despite these advances, these methods generally lack fine-grained control over multiple global structural requirements.

**Controlled Graph Generation.** Another line of work conditions generation on structural properties. DiGress (Vignac et al.), EDGE (Chen et al., 2023), GenStat (Zahirnia et al., 2024), TOPOGEN (Vakil & Amiri, 2026), and GraphMaker (Li et al., 2024) control graph- or node-level attributes, while ConStruct (Madeira et al., 2024), Arrow-Diff (Bernecker et al., 2025), and DeFoG (Qin et al., 2025) impose structural constraints or guidance. Other methods includes iterative local expansion Bergmeister et al. (2024),Sparse-Diff QIN et al. (2024),SBGD Su & shan Wu (2025),SPECTRE Martinkus et al. (2022),CDGS Huang et al. (2023), CatFlow Eijkelboom et al. (2024). However, these methods typically generate or diffuse over the full graph, making fine-grained multi-constraint control harder as graph size increases.

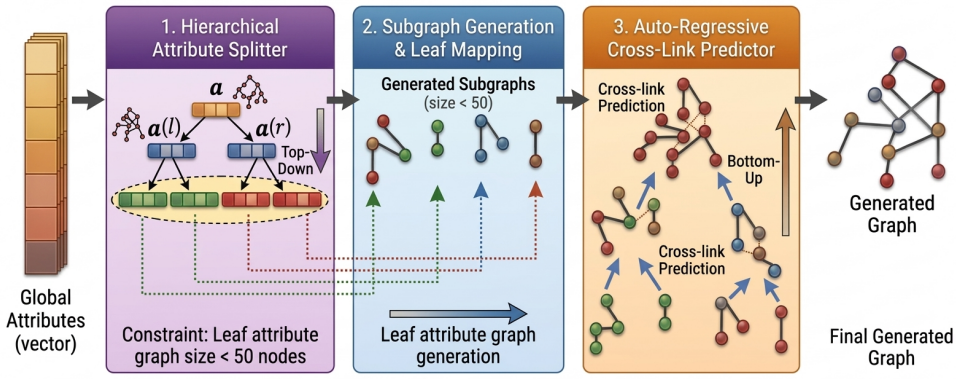



Figure 2: Architecture of `GraphScale` for scalable controlled graph generation from a global attributes specification. `GraphScale` factorizes the global specification into a binary hierarchy, generates bounded-size subgraphs satisfying leaf attributes, and recursively composes them bottom-up. Specifically, (1) an attribute splitter recursively decomposes the target attributes until each leaf specifies at most $n$ nodes; (2) a controlled subgraph generator independently generates each leaf subgraph; and (3) an autoregressive cross-link predictor connects subgraphs conditioned on the parent attribute $a_p$, decoupling generator scale from graph scale. Dotted lines denote predicted cross-links and solid lines denote existing links.

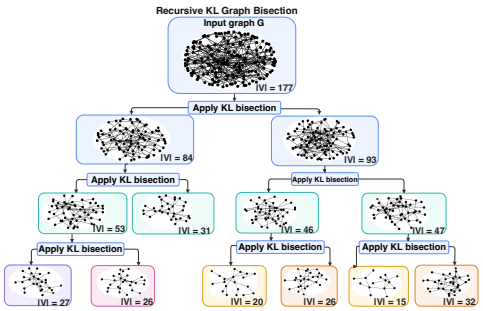
**Scalable Controlled Generation.** Thus, scalability and structural controllability remain largely separate: scalable generators focus on tractable graph modeling, whereas controlled generators generally operate at smaller scales. `GraphScale` targets their intersection through a *model-agnostic hierarchical composition principle*. We factorize global attributes specification into local constraints, generate bounded-size subgraphs independently, and recursively compose them while conditioning connectivity on remaining global attributes. This decouples generator scale from target graph scale while preserving global structural patterns beyond local statistics, including degree distributions, singular-value spectra, and singular-vector distributions. Consequently, `GraphScale` extends existing graph generators to large graphs while retaining fine-grained control over global structure.

## 3 Multiple Graph Attribute For Controled Graph Generation

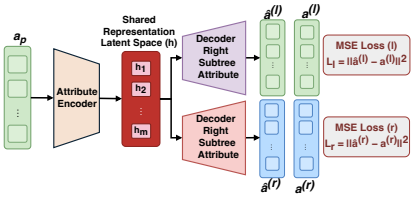
We consider the following graph topological properties for large scale controlled graph generation. For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges, the **number of nodes** is $|\mathcal{V}|$, and the **number of edges** is $|\mathcal{E}|$. The **density** measures how close the graph is to being fully connected, and for an undirected graph it is defined as $\text{Density}(\mathcal{G}) = \frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)}$. The **average clustering coefficient** measures the tendency of nodes to form tightly connected neighborhoods. For a node $v \in \mathcal{V}$ with degree $k_v$ and $e_v$ edges among its neighbors, the local clustering coefficient is $C_v = \frac{2e_v}{k_v(k_v-1)}$ when $k_v > 1$, and 0 otherwise; the graph-level average clustering coefficient is $\bar{C} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} C_v$. The **number of local bridges** counts edges $(u, v) \in \mathcal{E}$ such that the endpoints do not share any common neighbor, that is, $\mathcal{N}(u) \cap \mathcal{N}(v) = \emptyset$, and is given by $|\{(u, v) \in \mathcal{E} : \mathcal{N}(u) \cap \mathcal{N}(v) = \emptyset\}|$. Finally, the **number of cliques** counts complete subgraphs, where a clique is a subset of nodes $\mathcal{S} \subseteq \mathcal{V}$ such that for all distinct $u, v \in \mathcal{S}$, $(u, v) \in \mathcal{E}$. This graph attributes are computationally efficienct, however other graph attirbutes can also be used.

## 4 Method

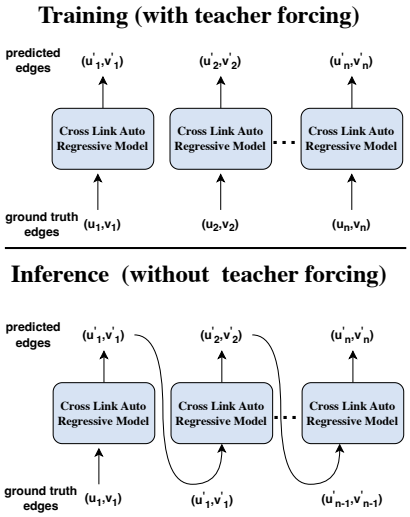
**Problem Statement.** Given a global attributes specification vector $\mathbf{a}$, the goal is to learn a generative model that produces large graphs $\mathcal{G}$ satisfying the specified attributes.




(a) Recursive application of the KL bisection algorithm to create subgraphs until each leaf contains at most $n$ nodes. Each partition contains non-overlapping nodes and edges.




(b) Architecture of attribute splitter.




(c) Cross Link Predictor with and without teacher forcing. During training, to avoid training drift, as the ground truth is available, we enable teacher forcing; during inference, we disable teacher forcing.

Figure 3: Components of `GraphScale` - attribute splitter to generate hirarhcy tree and cross-graph link predictor to aggregate sub-graphs, along with data generation method.

**Solution Overview.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $|\mathcal{V}| = N$ and $|\mathcal{E}| = M$, and let $\mathbf{a} \in \mathbb{R}^K$ denote its structural attribute vector. Given a global attributes specification $\mathbf{a}$, `GraphScale` decomposes large-graph generation into a hierarchy of bounded-size subgraph generation and composition problems. As illustrated in Figure 2, `GraphScale` (1) recursively factorizes $\mathbf{a}$ into a hierarchical attribute tree, with each leaf specifying a subgraph (Section 4.2); (2) independently generates leaf subgraphs using an existing graph generator; and (3) recursively composes them bottom-up using a constraint-aware cross-link predictor that generates inter-subgraph edges while satisfying the parent-level structural requirements (Sections 4.3–4.4). This hierarchical formulation decouples generator scale from graph scale while propagating structural constraints throughout the generation process.

### 4.1 Hierarchical Dataset Generation

Graphs often exhibit heterogeneous edge distributions and hierarchical organization, with communities capturing local interactions and inter-community edges capturing higher-level structure (Karami, 2024). Such hierarchy also arises naturally in graph growth processes such as preferential attachment (Chakrabarti & Faloutsos, 2006). However, hierarchy alone does not provide explicit control over structural properties across resolutions. `GraphScale` therefore exploits this organization by recursively decomposing global structural attributes into subgraph-level constraints and modeling the cross-subgraph interactions needed to recover the global structure.

Training this hierarchical generator requires supervision at multiple resolutions: how a graph decomposes into connected subgraphs, the attributes of each subgraph, and the inter-subgraph edges to recover the parent graph. Standard graph datasets provide graph-level supervision but lack these multi-resolution annotations. Thus, we construct a hierarchical graph-generation dataset that explicitly supervises attribute decomposition, subgraph generation, and cross-subgraph link prediction.

Naive approach like random node partitioning produce disconnected subgraphs and fail to capture meaningful parent–child structural relationships, while embedding-based clustering does not explicitly enforce a recursive graph hierarchy. We instead recursively apply the Kernighan–Lin (KL) graph bisection algorithm (Kernighan & Lin, 1970), partitioning each graph into connected *left* and

*right* children until every leaf contains at most $n = 50$ nodes. For every graph in the hierarchy, we compute the structural attributes defined in Section 3 using NetworkX (Hagberg et al., 2008). Each parent is thus associated with its attributes, child subgraphs and attributes, and inter-subgraph edges needed for reconstruction. Importantly, parent attributes need not decompose equally across children; their allocation depends on the underlying graph structure. We therefore treat child attributes as *guiding signals* rather than strict targets, providing the subgraph generator with *soft constraints*.

### 4.2 Hierarchical Structural Attribute Decomposition

Given a global structural attribute vector $\mathbf{a}$ for a large graph, `GraphScale` constructs a hierarchical attribute tree $\mathcal{T}(\mathbf{a})$ by recursively predicting attribute specifications for its child subgraphs. Specifically, given a parent attribute vector $\mathbf{a}^p$, the Attribute Splitter predicts attribute vectors for the left and right child subgraphs, $(\hat{\mathbf{a}}^{(l)}, \hat{\mathbf{a}}^{(r)})$. Since structural attributes do not generally decompose uniformly across subgraphs, the child attributes are not obtained by simply dividing the parent attributes by a fixed proportion. Instead, the splitter learns their allocation from the structural relationships observed in the training graphs.

**Attribute Splitter.** For computation advantage, we use a lightweight MLP with a shared backbone and two child-specific output heads (Figure 3b). Given a parent attribute vector $\mathbf{a}^p \in \mathbb{R}^d$, the shared backbone computes $\mathbf{h} = \phi\left(\mathbf{W}_2\phi(\mathbf{W}_1\mathbf{a}^p + \mathbf{b}_1) + \mathbf{b}_2\right)$ where $\phi(\cdot)$ denotes the ReLU activation. Two child-specific heads then transform the shared representation $\mathbf{h}$ into $\hat{\mathbf{a}}^{(l)}$ and $\hat{\mathbf{a}}^{(r)}$. The shared backbone captures dependencies common to the parent attributes, while the child-specific heads allow the model to learn asymmetric attribute allocations between the two subgraphs. The splitter is intentionally lightweight, however it is possible to use more expressive encoders .

**Training Objective.** For each parent graph, the training data provides the structural attributes of its two child subgraphs, denoted by $\mathbf{a}^{(l)}$ and $\mathbf{a}^{(r)}$. We train the splitter using the sum of squared prediction errors: $\mathcal{L}_{\text{split}} = \left|\hat{\mathbf{a}}^{(l)} - \mathbf{a}^{(l)}\right|_2^2 + \left|\hat{\mathbf{a}}^{(r)} - \mathbf{a}^{(r)}\right|_2^2$. Given a dataset $\mathcal{D}$ of parent–child attribute tuples, the training objective is $\mathcal{L} = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{a}^p,\mathbf{a}^{(l)},\mathbf{a}^{(r)})\in\mathcal{D}} \mathcal{L}_{\text{split}}$

**Recursive Inference.** At inference time, the splitter receives the global attribute vector $\mathbf{a}$ and predicts the corresponding child attributes. Each predicted child attribute vector is processed recursively to obtain the next level of the hierarchy until the associated subgraph reaches the predefined size threshold $n$. The resulting tree $\mathcal{T}(\mathbf{a})$ provides a hierarchy of soft attribute specifications that guides subgraph generation at different scales.

### 4.3 Controlled Local Subgraph Generation

As a model-agnostic framework, `GraphScale` adopt existing graph generative models as leaf-level subgraph generators, including autoregressive (You et al., 2018), adversarial (Bojchevski et al., 2018), latent-variable (Zahirnia et al., 2024), and attribute-conditioned models such as TOPOGEN (Vakil & Amiri, 2026). These models capture local graph structure, while `GraphScale` determines how generated subgraphs be organized and connected to satisfy the target global structure. Given a leaf-level attribute specification $\hat{\mathbf{a}}^{(i)}$ obtained from the hierarchical decomposition, the subgraph generator produces $\hat{\mathcal{G}}^{(i)}$ conditioned on $\hat{\mathbf{a}}^{(i)}$. As soft constraints, these attributes guide the generated subgraph toward the desired local structure without requiring exact satisfaction. Thus, the subgraph generator models *local structure*, while hierarchical decomposition and cross-subgraph connectivity coordinate these local structures to recover the desired *global structure*.

### 4.4 Auto-regressive Cross-Graph Link Predictor

The hierarchical composition process constructs a larger graph by recursively merging pairs of generated subgraphs. At each merge, the cross-subgraph edges must satisfy the structural attributes specified for the parent graph. We therefore formulate cross-subgraph connectivity as an attribute-conditioned link generation problem. Prior approaches to cross-graph link prediction include graph alignment methods for identifying node correspondences (Singh et al., 2008; Bayati et al., 2009), embedding-based methods that learn shared latent spaces across graphs (Shen & Chung, 2019; Shen

et al., 2020; Yang et al., 2024a), and community-aware models for inter-community connectivity and diffusion (Chen et al., 2020; Kumar et al., 2024). These methods provide useful alternatives for modeling cross-graph connectivity, but differ in their granularity and ability to incorporate global structural constraints. We instead use an attribute-conditioned autoregressive model to generate the cross-subgraph edges required by the parent attributes. Given two subgraphs $\mathcal{G}_l$ and $\mathcal{G}_r$ and their parent structural attribute $\mathbf{a}_p$, the model generates a sequence of cross-links $\mathcal{E}^{\times} = \{(u_t, v_t)\}_{t=1}^{T}$, where $u_t \in \mathcal{V}_l$ and $v_t \in \mathcal{V}_r$. At each step, the prediction is conditioned on the two subgraphs, the parent attribute, and the cross-links generated so far. This allows the connectivity between subgraphs to adapt to the remaining structural requirements during bottom-up composition.

***Graph & Attribute Encoder.*** Given child subgraphs $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ and $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$, we apply GraphSAGE to obtain node representations $\mathbf{H}_l$ and $\mathbf{H}_r$, followed by permutation-invariant readouts to obtain graph-level representations $\mathbf{g}_l$ and $\mathbf{g}_r$. We encode the parent structural attribute $\mathbf{a}_p$ with an MLP, $\mathbf{z}_p = \phi(\mathbf{a}_p)$, and initialize the decoder state as $\mathbf{s}_0 = \psi\left([\mathbf{g}_l \,|\, \mathbf{g}_r \,|\, \mathbf{z}_p]\right)$, where $\psi(\cdot)$ is a one-layer MLP. This joint representation conditions cross-subgraph connectivity on both child-subgraph structure and the parent-level structural attributes.

***Two-stage node selection.*** At decoding step $t$, the model autoregressively generates a cross-link $(u_t, v_t)$. We first select the source node $u_t \in \mathcal{V}_l$ and then select the target node $v_t \in \mathcal{V}_r$ conditioned on $u_t$. Let $\mathcal{E}^{\times}_{<t} = \{(u_i, v_i)\}_{i=1}^{t-1}$ denote the previously generated cross-links. The joint probability is factorized as

$$p_\theta\left(u_t, v_t \mid \mathcal{E}^{\times}_{<t}, \mathcal{G}_l, \mathcal{G}_r, \mathbf{a}_p\right) = p_\theta\left(u_t \mid \mathbf{s}_t, \mathbf{H}_l, \mathbf{H}_r\right) p_\theta\left(v_t \mid u_t, \mathbf{s}_t, \mathbf{H}_l, \mathbf{H}_r\right). \tag{1}$$

This factorization avoids explicitly scoring all $|\mathcal{V}_l||\mathcal{V}_r|$ possible node pairs and provides a scalable mechanism for generating cross-subgraph connectivity.

***Training objective.*** We train the model using teacher forcing, where the ground-truth cross-link history is provided to the decoder at each step. This prevents errors in early predictions from accumulating and causing distributional drift during training, allowing the model to learn each link conditioned on a consistent and correct connectivity history. Given the ground-truth cross-link sequence $\mathcal{E}^{\times}_{1:T} = \{(u^*_t, v^*_t)\}_{t=1}^{T}$, the negative log-likelihood is

$$\mathcal{L}_{\mathrm{NLL}}(\theta) = -\sum_{t=1}^{T}\left[\log p_\theta\left(u^*_t \mid \mathbf{s}_t, \mathbf{H}_l, \mathbf{H}_r\right) + \log p_\theta\left(v^*_t \mid u^*_t, \mathbf{s}_t, \mathbf{H}_l, \mathbf{H}_r\right)\right]. \tag{2}$$

***Inference.*** During inference, the decoder conditions on its own generated cross-link history. Because exact decoding is computationally expensive, we approximate this objective using beam search with width $K$ and select the highest-scoring sequence.

### 4.5 Large Graph Generation

Given a global structural attributes, `GraphScale` recursively factorizes the global specification into local subgraph structural attributes as described above, forming a binary decomposition tree whose depth adapts to the specified graph size and structure. Each leaf specifies a bounded-size subgraph generated independently by the underlying graph generator. `GraphScale` then merges these subgraphs bottom-up, predicting cross-subgraph edges at each internal node conditioned on the parent-level structural attributes. Subgraphs are merged until reaching the root. This separation of bounded-size generation from hierarchical connectivity decouples generator scale from graph scale while preserving global structural patterns through recursive composition.

## 5 Experiments

**Baselines.** Large-scale graph generation remains challenging and has recently started to get attention. We compare `GraphScale` with recent approach HiGen (Karami, 2024). Other graph generators, including DiGress (Vignac et al.), GDSS (Jo et al., 2022), GraphARM (Kong et al., 2023), EDGE (Chen et al., 2023), GraphRNN (You et al., 2018), SPECTRE (Martinkus et al., 2022), and

| Model | Control | Scalable | Spectral Difference (SD) ↓ Ogbn-Arxiv | Enzymes | SBM | Avg. | Network Similarity Distance (NSD) ↓ Ogbn-Arxiv | Enzymes | SBM | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| Erdos-Renyi | | ✓ | 7.53 | 1.68 | 3.48 | 4.23 | – | 23.36 | 25.21 | 24.28 |
| HiGen | | ✓ | – | 0.45 | 0.43 | 0.44 | – | 19.37 | 15.46 | 19.41 |
| TOPOGEN | ✓ | | OOM | OOM | OOM | – | OOM | OOM | OOM | – |
| DiGress | ✓ | | OOM | OOM | OOM | – | OOM | OOM | OOM | – |
| GruM | ✓ | | OOM | OOM | OOM | – | OOM | OOM | OOM | – |
| EDGE | ✓ | | OOM | OOM | OOM | – | OOM | OOM | OOM | – |
| GENSTAT | ✓ | | OOM | OOM | OOM | – | OOM | OOM | OOM | – |
| GraphScale | | | | | | | | | | |
| with TOPOGEN | ✓ | ✓ | **0.42** | 0.45 | **0.18** | **0.35** | 15.05 | 17.36 | **14.73** | 15.71 |
| with DiGress | ✓ | ✓ | – | 1.47 | 1.61 | 1.54 | – | 27.98 | 29.92 | 28.95 |
| with GruM | ✓ | ✓ | – | 0.35 | 0.28 | **0.32** | – | 14.87 | 15.51 | **15.19** |
| with HGGT | ✓ | ✓ | 0.53 | **0.29** | 0.74 | 0.52 | 22.44 | **14.28** | 20.46 | 19.06 |

Table 1: Large graph generation results. SD and NSD are structural metric; lower is better. "–" indicates substantially longer training time, and "OOM" indicates out-of-memory.

GenStat (Zahirnia et al., 2023), do not support large-graph generation and are thus not used as large-scale baselines. Instead, GraphScale can use existing small-graph generators, such as GruM (Jo et al., 2024) and DiGressVignac et al., TOPOGENVakil & Amiri (2026), and HGGT Jang et al. (2024) as leaf-level subgraph generators.

**Datasets.** We test all the methods on real-world datasets Enzymes Schomburg et al. (2004) and Ogbn-Arxiv Hu et al. (2020) as well as synthetic dataset Stochastic Block Model (SBM) Holland et al. (1983). We processed each dataset as described in Section 4.1. We divide the train-val-test split by triplets (parent graph, right sub-graph, left sub-graph). Enzymes, SBM, and Ogbn-Arxiv contain 37K, 273K, and 169K nodes, respectively, with 195K, 1.25M, and 1.15M edges, respectively.

**Evaluation Metrics.** While MMD You et al. (2018) is commonly used to compare distributions of generated and real graphs Qin et al. (2025); Eijkelboom et al. (2024); Vignac et al.; Huang et al. (2023), it is less informative for controlled generation because it does not measure fidelity to individual target attributes. We therefore directly measure difference of generated graphs with their target graphs using Spectral Difference (SD) and Network Similarity Distance (NSD). **Spectral Difference (SD↓)** (Jo et al., 2024) compares the sorted Laplacian eigenvalues, capturing global structural properties. To compare graphs of different sizes, we zero-pad the smaller eigenvalue vector and compute $SD = 1/n \times ||\lambda_{groundtruth} - \lambda_{pred}||_2$ where $n$ is the size of the larger graph. **Network Similarity Distance (NSD↓)** (Berlingerio et al., 2012) compares size-independent graph signatures based on seven local and egonet-level structural attributes, summarized using statistics including mean, median, standard deviation, skewness, and kurtosis. We use the resulting signature distance to quantify structural similarity between generated and target graphs. For the Attribute Splitter, we measure the mean squared error (MSE) between predicted and ground-truth attributes for both left and right children and average the two errors. All metrics are averaged over the test set.

### 5.1 Results

***Large Graph Generation.*** Table 1 summarizes large-graph generation under specified structural attributes. GraphScale achieves substantially lower SD than the baselines. With TOPOGEN, GraphScale attains an average SD of 0.39 on Enzymes and SBM, compared with 0.44 for HiGen and 4.23 for Erdős–R'enyi. Similar trends hold while comparing network signatures using NSD. These results demonstrate the benefit of hierarchical conditioning for controlled generation at scale. By decomposing global specifications into an hierarchal attribute tree and conditioning both subgraph generation and cross-subgraph connectivity, GraphScale enables bounded-size generators to produce substantially larger graphs while preserving global structural requirements. The choice of subgraph generator further affects the overall quality of the generated graph, with GraphScale with GruM achieving an average SD of 0.32 and NSD of 15.19.

***Attribute Splitter.*** Table 2 compares attribute splitters in GraphScale. Besides the MLP, we evaluate *Constrained* and *Variational* variants, which respectively regularize node counts to match the parent graph and learn a latent split representation. The MLP performs comparably to Con-

| Model | Arxiv | | | Enzymes | | | SBM | | |
|---|---|---|---|---|---|---|---|---|---|
| | Avg.(↓) | Left(↓) | Right(↓) | Avg.(↓) | Left(↓) | Right(↓) | Avg.(↓) | Left(↓) | Right(↓) |
| **Constrained Variational MLP** | 0.53 | 0.69 | 0.38 | 0.83 | 1.04 | 0.62 | 0.87 | 0.88 | 0.86 |
| **Variational MLP** | 0.65 | 0.83 | 0.46 | 0.86 | 1.12 | 0.60 | 0.95 | 1.01 | 0.89 |
| **Constrained MLP** | 0.44 | 0.53 | 0.34 | 1.09 | 1.33 | 0.86 | 0.98 | 1.06 | 0.89 |
| **MLP** | **0.42** | **0.51** | **0.32** | **0.74** | **0.88** | **0.61** | **0.71** | **0.79** | **0.64** |

Table 2: Attribute splitter Mean Square Error on Left, Right and the Average Attribute.

| | Model | SD (↓) | | | | NSD (↓) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Arxiv | Enzymes | SBM | Avg. | Arxiv | Enzymes | SBM | Avg. |
| **Generating Subgraphs** | TOPOGEN-GIN | 0.49 | 0.59 | 0.51 | 0.53 | 16.75 | **16.76** | 17.73 | 17.08 |
| | TOPOGEN-GCN | 0.48 | 0.53 | 0.60 | 0.54 | 16.63 | 25.74 | 16.33 | 19.57 |
| | TOPOGEN-GraphSAGE | 0.50 | 0.57 | 0.57 | 0.55 | 26.06 | 23.66 | 20.07 | 23.26 |
| | TOPOGEN-CNN | **0.33** | **0.46** | **0.19** | **0.33** | 23.32 | 18.94 | 15.03 | 19.10 |
| | DiGress | - | 0.61 | 0.43 | 0.52 | - | 22.56 | 19.29 | 20.93 |
| | GruM | - | 0.63 | 0.23 | 0.43 | - | 18.14 | **13.55** | **15.85** |
| | HGGT | 0.45 | 0.60 | 0.27 | 0.44 | **15.86** | 17.24 | 16.37 | 16.49 |

Table 3: Result of smaller subgraph generator and cross link predictor. SD denotes spectral difference between the generated subgraph and the ground-truth graph. Lower is better. Average values are computed across available datasets, ignoring missing entries.

strained MLP, indicating limited benefit from node-count regularization, while Variational MLP performs worse, likely due to uncertainty from latent sampling.

***Small Sub-Graph Generator.*** From Table 3 overall, the results indicate that TOPOGEN-CNN exhibits the lowest average SD (0.33), suggesting more consistent spectral similarity to the ground-truth subgraphs across the evaluated datasets. In contrast, TOPOGEN-GIN achieves the lowest average NSD (17.08) among the TOPOGEN variants, with particularly strong performance on Enzymes. Among the non-TOPOGEN baselines, HGGT obtains the lowest average NSD (16.49) while maintaining a competitive average SD (0.44), demonstrating relatively consistent performance across datasets.

***Cross-Graph Link Predictor.*** Table 4 evaluates the autoregressive cross-graph link predictor. The *random* variant adds edges by uniformly sampling node pairs across the two subgraphs. Random linking increases Spectral Difference across all the datasets, demonstrating the benefit of teacher forcing and conditioning on previously predicted edges and the desired attributes when constructing cross-subgraph connectivity.

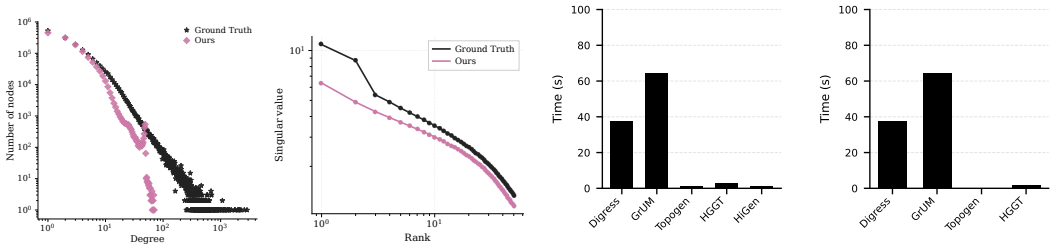
| Model | Arxiv SD ↓ | Enzymes SD ↓ | SBM SD ↓ |
|---|---|---|---|
| Random | 2.12 | 1.00 | 2.57 |
| Cross-link predictor | 0.04 | 0.09 | 0.12 |

Table 4: Cross-link prediction results. SD denotes spectral difference; lower is better.

### 5.2 GRAPHSCALE PRESERVES LARGE GRAPH STRUCTURAL FIDELITY

Figure 4a shows that `GraphScale` recovers the heavy-tailed degree distribution of the large-scale arXiv graph, including both the concentration of low-degree nodes and the tail of highly connected nodes. This shows that recursive composition of bounded-size subgraphs recovers global connectivity statistics at scales far beyond the underlying generator. More importantly, `GraphScale` preserves global structural properties beyond degree statistics. As shown in Figure 4b, the generated graph closely matches the real graph's singular-value spectrum, including its dominant components and overall decay, indicating that hierarchical composition preserves structure beyond degree distributions. The left and right singular-vector distributions show similar agreement (Figure 9). Together, these results show that `GraphScale` preserves graph structure across multiple levels, from degree distributions to global spectral attributes and node-level spectral representations, while generating graphs from bounded-size, recursively composed subgraphs. Hierarchical composition thus decouples generator scale from graph scale, enabling controlled generation of large graphs while preserving global structural attributes.

**Time Analysis:** Figure 4c reports the runtime of `GraphScale` and its three components across subgraph generators. Subgraph generation (Figure 4d) dominates, while attribute-hierarchy construction and cross-subgraph link prediction add minimal overhead.

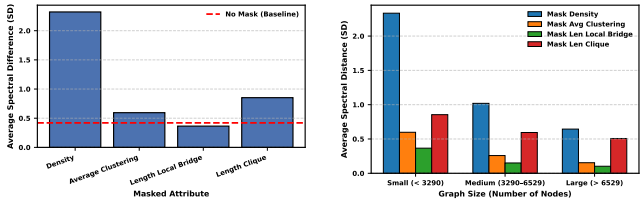



(a) Degree distribution (b) Singular values (c) Total time (d) Subgraph-generation

Figure 4: Structural and computational properties of the ground-truth ArXiv graph and the ArXiv graph generated using `GraphScale`.

Among the evaluated generators, TopoGen achieves the lowest runtime, by avoiding the iterative denoising required by diffusion-based generators such as DiGress and GruM.

## 6 Discussion




(a) Influence of each attribute while generating large graphs.

(b) Influence of each attribute while generating graphs of different sizes.

Figure 5: Introspection of attributes, decoding behavior, and generation time in `GraphScale`.

**Attribute Ablation Study.** We quantify the contribution of individual structural attributes by independently masking density, average clustering, local-bridge length, and clique length while keeping node and edge counts fixed. Figure 5a reports the resulting spectral difference (↓), with the red dashed line denoting the unmasked setting. Removing density makes largest increase in spectral difference, demonstrating that density provides a critical signal for recovering the target global topology. Masking clique length yields a similar, though smaller, degradation, while removing average clustering produces a modest increase in error. These results show that complementary structural attributes provide distinct signals for hierarchical graph composition rather than serving as redundant constraints. Figure 5b shows consistent trends across graph sizes. In particular, density remains a dominant structural signal as the number of nodes increases, indicating that the contribution of structural attributes is preserved under hierarchical scaling. Overall, the ablation results demonstrate that `GraphScale` can exploit multiple structural specifications to guide large-scale graph generation and preserve global topology beyond node and edge counts alone.

## 7 Conclusion

In this work, we introduced `GraphScale`, a framework for controlled large-scale graph generation that factorizes global structural attributes into local constraints and recovers the target graph through recursive hierarchical composition. `GraphScale` follows model-agnostic composition principle which decouples the scale of the underlying graph generator from that of the generated graph, enabling conventional small-scale generators, typically limited to a few hundred nodes, to generate graphs with thousands to tens of thousands of nodes without modifying their architecture or generation process. Across real-world and synthetic datasets, `GraphScale` consistently preserves global structural attributes beyond node and edge counts, including degree distributions and global spectral characteristics. These results demonstrate that bounded-size subgraph generation, combined with structure-aware cross-subgraph connectivity, help scale existing graph generators to large graphs while leaving the underlying generators unchanged. Future work will investigate more expressive attribute decomposition and cross-subgraph connectivity models to further improve structural fidelity and controllability.

## 8 AI USE STATEMENT

In this work, generative AI tools were used for grammar checking and language refinement. All AI-assisted content was carefully reviewed by the us; for example, we verified the text produced following grammar corrections and does not change the message. We take full responsibility for the final content of this work, including all text, claims, and other artifacts produced with the assistance of generative AI.

## 9 ETHICS STATEMENT

This work studies controlled graph generation using citation, molecular-topology, and synthetic benchmarks that are publicly available. The molecular experiments concern unlabeled topology and do not establish chemical validity. Potential downstream risks include reproducing structural biases. We make no claims of fairness or suitability for safety-critical applications. Downstream use should include validation of requested constraints and application-specific requirements.

## 10 REPRODUCIBILITY STATEMENT

We will release the code and data to support reproduction and further investigation of the proposed method.

## REFERENCES

Leman Akoglu and Christos Faloutsos. Rtg: A recursive realistic graph generator using random typing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 13–28. Springer, 2009. URL https://link.springer.com/article/10.1007/s10618-009-0140-7.

Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Rtm: Laws and a recursive generator for weighted time-evolving graphs. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 701–706, 2008. doi: 10.1109/ICDM.2008.123. URL https://ieeexplore.ieee.org/abstract/document/4781165.

Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 705–710. IEEE, 2009. URL https://ieeexplore.ieee.org/document/5360298.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2XkTz7gdpc.

Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Netsimile: A scalable approach to size-independent network similarity. *arXiv preprint arXiv:1209.2684*, 2012. URL https://arxiv.org/abs/1209.2684.

Tobias Bernecker, Ghalia Rehawi, Francesco Paolo Casale, Janine Knauer-Arloth, and Annalisa Marsico. Random walk diffusion for efficient large-scale graph generation. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. 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. URL https://proceedings.mlr.press/v80/bojchevski18a/bojchevski18a.pdf.

Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM computing surveys (CSUR)*, 38(1):2–es, 2006. URL https:

//www.cs.cmu.edu/~christos/courses/826-resources/PAPERS+BOOK/CMU-ONLY/graph-survey.pdf.

Dexiong Chen, Markus Krimmel, and Karsten Borgwardt. Flatten graphs as sequences: Transformers are scalable graph generators. *Advances in Neural Information Processing Systems*, 38: 69071–69109, 2026. URL https://neurips.cc/virtual/2025/loc/san-diego/poster/116880.

Ke-Jia Chen, Pei Zhang, Zinong Yang, and Yun Li. ibridge: Inferring bridge links that diffuse information across communities. *Knowledge-Based Systems*, 192:105249, 2020. URL https://www.sciencedirect.com/science/article/pii/S0950705119305623.

Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4585–4610. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chen23k.html.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pp. 2302–2312. PMLR, 2020. URL https://proceedings.mlr.press/v119/dai20b/dai20b.pdf.

Floor Eijkelboom, Grigory Bartosh, Christian A Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/15b780350b302a1bf9a3bd273f5c15a4-Abstract-Conference.html.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. URL https://www.researchgate.net/publication/236407765_Exploring_Network_Structure_Dynamics_and_Function_Using_NetworkX.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017. URL http://sites.computer.org/debull/A17sept/p52.pdf.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983. URL https://www.sciencedirect.com/science/article/abs/pii/0378873383900217.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Minkai Huang, Xiao Sun, Jingjing Wang, Ge Wang, Rong Xia, Li Yu, and Liang Zhao. Conditional diffusion based on discrete graph structures for molecular graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4773–4781, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/25549.

Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Graph generation with kˆ2-trees. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27621–27644, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/74d056577c164d45f713ce297a4872df-Paper-Conference.pdf.

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. URL https://proceedings.mlr.press/v162/jo22a.html.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=cZTFxktg23.

Mahdi Karami. Higen: Hierarchical graph generative networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KNvubydSB5.

Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970. URL https://ieeexplore.ieee.org/document/6771089.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In *International conference on machine learning*, pp. 17391–17408. PMLR, 2023. URL https://openreview.net/forum?id=98J48HZXxd5.

Mukesh Kumar, Shivansh Mishra, Shashank Sheshar Singh, and Bhaskar Biswas. Community-enhanced link prediction in dynamic networks. *ACM Transactions on the Web*, 18(2):1–32, 2024. URL https://dl.acm.org/doi/10.1145/3580513.

Mufei Li, Eleonora Kreačić, Vamsi K. Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs? *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ledQ1BCrwc.

Manuel Madeira, Clement Vignac, Dorina Thanou, and Pascal Frossard. Generative modelling of structurally constrained graphs. *Advances in Neural Information Processing Systems*, 37:137218–137262, 2024. URL https://neurips.cc/virtual/2024/poster/96282.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15159–15179. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/martinkus22a.html.

Mark E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2): 167–256, 2003. doi: 10.1137/S003614450342480. URL https://doi.org/10.1137/S003614450342480.

Yiming QIN, Clement Vignac, and Pascal Frossard. Sparse training of discrete diffusion models for graph generation, 2024. URL https://openreview.net/forum?id=oTRekADULK.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=KPRIwWhqAZ.

Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004. URL https://pubmed.ncbi.nlm.nih.gov/14681450/.

Xiao Shen and Fu-Lai Chung. Network embedding for cross-network node classification. *CoRR*, abs/1901.07264, 2019. URL http://arxiv.org/abs/1901.07264.

Xiao Shen, Quanyu Dai, Fu-Lai Chung, Wei Lu, and Kup-Sze Choi. Adversarial deep network embedding for cross-network node classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 2991–2999. AAAI Press, 2020. doi: 10.1609/AAAI.V34I03.5692. URL https://doi.org/10.1609/aaai.v34i03.5692.

Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, 2008. URL https://www.pnas.org/doi/10.1073/pnas.0806627105.

Junwei Su and shan Wu. SBGD: Improving graph diffusion generative model via stochastic block diffusion. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=FMg9mEKH17.

Nidhi Vakil and Hadi Amiri. Fine-grained graph generation through latent mixture scheduling, 2026. URL https://arxiv.org/abs/2605.02780.

Clement 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*. URL https://openreview.net/pdf?id=UaAD-Nu86WX.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020. URL https://ieeexplore.ieee.org/document/9046288.

Hongwei Yang, Hui He, Weizhe Zhang, Yan Wang, and Lin Jing. Multi-source and multi-modal deep network embedding for cross-network node classification. *ACM Trans. Knowl. Discov. Data*, 18(6):149:1–149:26, 2024a. doi: 10.1145/3653304. URL https://doi.org/10.1145/3653304.

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*, 36(11):6358–6369, 2024b. doi: 10.1109/TKDE.2024.3389783. URL https://ieeexplore.ieee.org/document/10508504.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/you18a.html.

Kiarash Zahirnia, Yaochen Hu, Mark Coates, and Oliver Schulte. Neural graph generation from graph statistics. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/72153267883fbcafdb6e4662382696c5-Abstract-Conference.html.

Kiarash Zahirnia, Yaochen Hu, Mark Coates, and Oliver Schulte. Neural graph generation from graph statistics. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/72153267883fbcafdb6e4662382696c5-Paper-Conference.pdf.

# 11 Appendix

## 11.1 Quality of Generated Large Graph

Figure 6 compares the quality of the generated graphs against the ground-truth graphs for graph of more than 2000 and 10000 nodes from ogbn-Arxiv dataset. GraphScale is able to generate graphs with node and edge structures similar to the ground truth while satisfying the desired density, particularly for large graphs. Similarly,Figure 7 shows the comparison between the GraphScale and HiGen for enzymes and SBM dataset. GraphScale is able to generate graph with attributes close to the ground truth graph while HiGen is able to generate small graphs more accurately compared to large graphs.

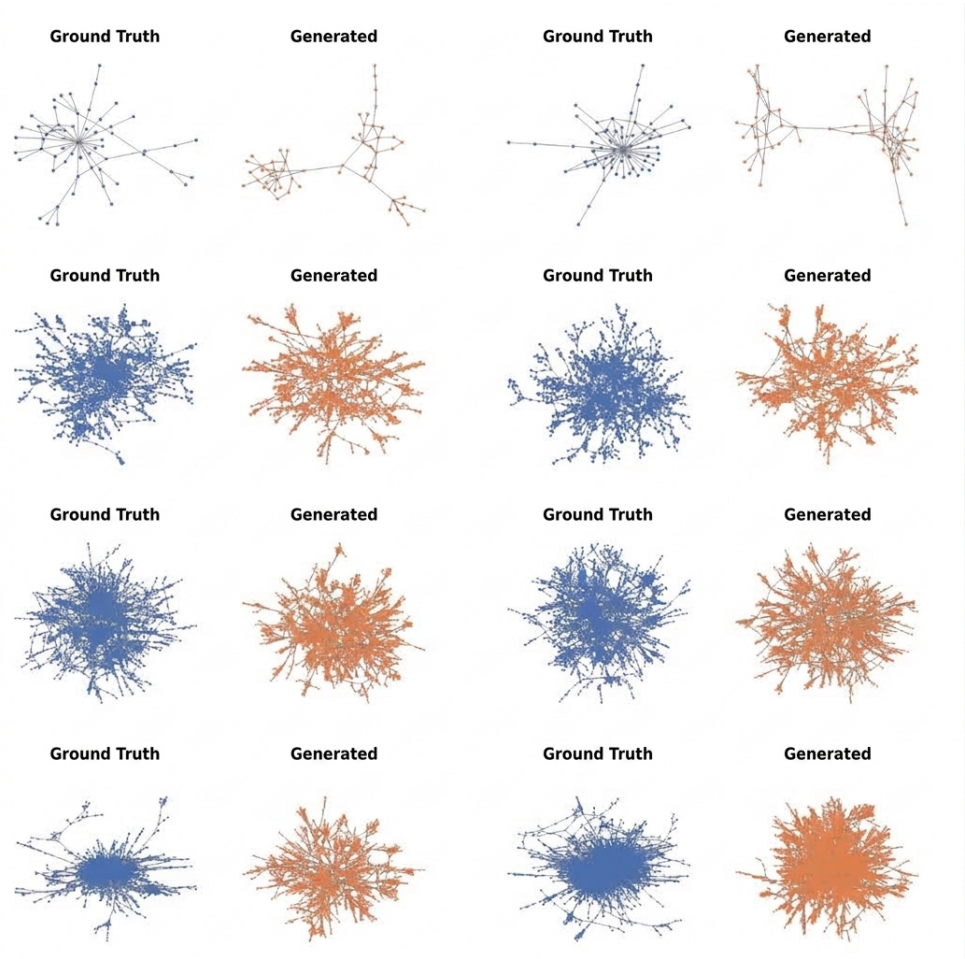



Figure 6: Graph Generated using `GraphScale` with thousands of nodes as targed graph.

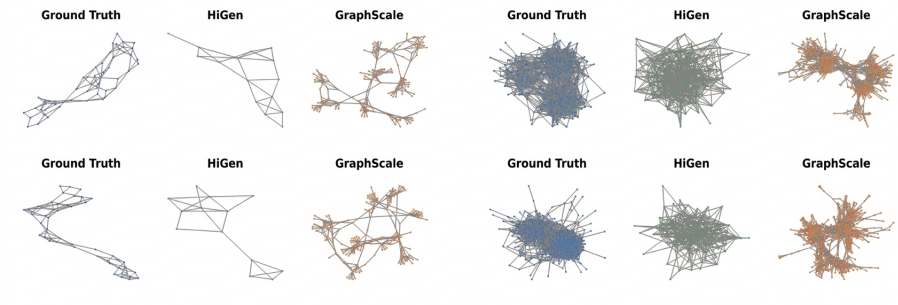



Figure 7: Graph Generated using HiGen and `GraphScale`

## 12 Detailed Related Work

Graph generation has received substantial attention in recent years, with prior work spanning a broad range of graph sizes and application domains. Existing methods can be broadly divided into approaches targeting small graphs, such as molecular and drug graphs, and those designed for large-scale graphs, such as citation, social, and infrastructure networks. While considerable progress has been made on small-graph generation, generating large graphs remains substantially more challenging due to the combinatorial size of the graph space and the cost of explicitly modeling graph struc-

ture. Moreover, existing large-graph generators primarily aim to reproduce the structural statistics of real-world graphs, with limited attention to generation under multiple fine-grained topological constraints. In contrast, `GraphScale` targets large-scale graph generation while explicitly supporting such constraints.

A major line of work in graph generation employs autoregressive models, which construct a graph incrementally by generating nodes, edges, or graph components conditioned on previously generated structure. For example, BiGG Dai et al. (2020) is an autoregressive framework for large graph generation that exploits graph sparsity to avoid explicitly generating the full adjacency matrix. BiGG further develops a parallel implementation of autoregressive generation to mitigate the sequential bottleneck inherent in conventional autoregressive models. Our approach is complementary to this line of work: `GraphScale` likewise leverages models that are effective at generating smaller graph structures, but extends their use to large graphs while incorporating multiple fine-grained topological constraints into the generation process.

Another prominent direction formulates graph generation as a diffusion process, in which a graph is progressively perturbed and a learned reverse process reconstructs a graph from noisy intermediate states. GraphMakerLi et al. (2024), for instance, adopts a diffusion-based formulation and incorporates graph-level and node-level labels to condition generation. In particular, node labels are sampled and fixed during the initial stages of generation, providing structural and semantic information that guides subsequent generation of node attributes and graph connectivity. Such conditional generation is particularly relevant when generated graphs are used as training data for downstream tasks, where controlling the generated node-label distribution can improve the utility of the resulting graphs.

Several recent methods instead focus on developing more compact or structured graph representations to make generation tractable at larger scales. HGGTJang et al. (2024) represents graphs using a K2-tree, an ordered tree representation that recursively partitions the adjacency matrix into $K \times K$ submatrices. This hierarchical representation provides a compact and lossless encoding of sparse graph structure and enables graph generation without relying on domain-specific rules. By recursively decomposing the adjacency matrix, HGGT avoids directly operating on the full dense representation, making hierarchical graph generation more computationally feasible.

GraphusionYang et al. (2024b) similarly explores latent-space graph generation. It employs a variational graph autoencoder to map graphs from a high-dimensional discrete space into a lower-dimensional topology-aware latent space, followed by latent-to-latent and latent-to-graph generation stages. The additional latent-to-latent transformation allows the model to perform generation in a smoother latent space before decoding the resulting representation into a graph. This approach highlights the broader trend of moving graph generation away from the original discrete graph space and into representations that are more amenable to generative modeling.

Other diffusion-based approaches explicitly incorporate structural information into the generation process. Arrow-DiffBernecker et al. (2025), for example, decomposes generation into two stages. First, a random-walk-based diffusion model generates candidate random walks based on the original graph topology. A GNN then evaluates the proposed edges and filters them according to the structural information learned from the original graph. The resulting graph is therefore guided by changes in node degrees and other topology-dependent signals. This design demonstrates the potential of incorporating explicit structural guidance into diffusion-based graph generation, although the structural guidance is centered primarily on degree-related changes.

DeFoG**?** formulates graph generation as a discrete diffusion process in which a graph transformer predicts the clean graph from an intermediate noisy graph. These predictions are subsequently converted into a continuous-time Markov chain (CTMC) rate matrix that determines the evolution of the graph during sampling. A key property of DeFoG is the decoupling between training and sampling: the model learns to predict clean graphs independently of the particular sampling trajectory, allowing the sampling schedule or guidance mechanism to be modified without retraining the underlying model. This flexibility distinguishes DeFoG from diffusion formulations in which the learned denoising process is more tightly coupled to a predefined forward and reverse trajectory.

A related direction explicitly incorporates structural constraints into the generative process. ConStructMadeira et al. (2024) employs diffusion to generate graphs satisfying properties such as planarity, acyclicity, and maximum degree. Rather than representing these properties solely as con-

ditioning features, ConStruct enforces them as hard constraints through a projection mechanism during reverse diffusion. The generative model proposes candidate edges, after which a projector determines whether the proposed edge can be added while preserving the desired graph property. This approach demonstrates that graph generation can be guided by explicit structural constraints rather than relying exclusively on learned distributions.

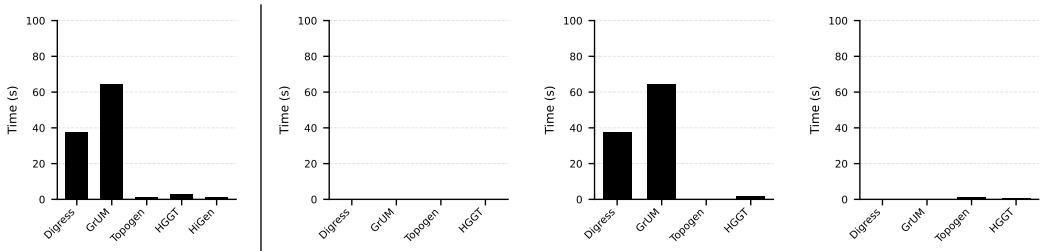
Constraint-guided diffusion has also been explored for molecular graph generation. incorporates discrete graph structures into the reverse diffusion process as conditioning information, allowing generation to be guided by a predefined structural representation. Other work **?** considers efficient graph sampling by adapting ODE solvers to exploit the semi-linear structure of the probability-flow ODE, improving the efficiency of diffusion-based sampling.

Beyond autoregressive and diffusion formulations, recent work has explored alternative graph traversal representations for generative modeling. Chen et al. (2026) uses transformers to generate Eulerian traversals of graphs, where each edge is visited exactly once. When a complete Euler path does not exist, the method decomposes the graph into segmented Euler paths and generates these segments instead. Such representations provide an alternative to directly modeling the adjacency matrix by transforming graph generation into a structured sequence-generation problem.

**Positioning of `GraphScale`**. Existing approaches have therefore explored several complementary strategies for graph generation: autoregressive generation for exploiting sparsity, diffusion for learning flexible graph distributions, hierarchical representations for compactly modeling large adjacency structures, latent-space generation for reducing the dimensionality of the discrete graph space, and projection or conditioning mechanisms for incorporating structural information. However, these methods generally focus either on improving the scalability of graph generation or on enforcing a limited set of structural properties. `GraphScale` instead focuses on scalable generation of large graphs subject to multiple fine-grained topological constraints. In particular, `GraphScale` adopts is model agnostic autoregressive generation framework and leverages existing models that are effective for smaller graphs, while introducing mechanisms that enable these models to generate large graphs under explicit structural requirements. This combination distinguishes `GraphScale` from prior large-graph generators that primarily target unconstrained graph distributions and from constraint-aware generators whose constraints are typically limited to a global graph properties such as planar or acyclic.

## 13 TIME ANALYSIS

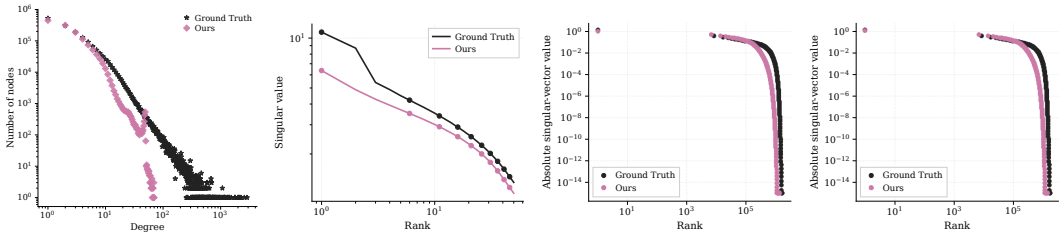
Figure 8 reports the overall runtime of `GraphScale` and its three components across the evaluated subgraph generators. Subgraph generation dominates the computational cost, while attribute-hierarchy construction and cross-subgraph link prediction incur substantially lower overhead. Among the evaluated generators, Topogen achieves the lowest overall runtime, primarily by avoiding the iterative denoising process required by diffusion-based generators such as Digress and GrUM.




(a) Total Time (b) Attribute Hierarchy (c) Subgraph Generation (d) Cross Link Prediction

Figure 8: Time distribution of `GraphScale` total and across three components over three subgraph generator. Overall generating attribute hierarchy and predicting cross link between the graph are less time consuming compared to generating subgraph. Topogen is faster compared to Digress and GrUM as it is not diffusion based process. Attribute hierarchy is lightweight and takes least time; hence lower bar values on similar scale are not visible.

## 14 GRAPHSCALE PRESERVES STRUCTURAL FIDALITY




(a) Degree distribution (b) Singular values (c) Left singular vector (d) Right singular vector

Figure 9: Structural properties of the ground truth ArXiv graph vs ArXiv graph generated using `GraphScale`.

Figure 9c and Figure 9d show similar singular-vector distributions, demonstrating that the generated graph retains the organization of nodes within these global structural components.