MULTI-RESOLUTION GRAPH DIFFUSION

Mahdi Karami[*]
School of Computer Science
University of Waterloo, ON, Canada
karami1@ualberta.ca

Igor Krawczuk [*]
LIONS EPFL,
Lausanne, Switzerland
igor@krawczuk.eu

Volkan Cevher
LIONS EPFL,
Lausanne, Switzerland

ABSTRACT

Recent graph denoising diffusion models achieved high fidelity in modeling small to medium sized graphs, however they often struggle from a complexity-expressivity tradeoff due the most high quality methods requiring architectures or features which scale quadratic in the number of nodes. This work proposes hierarchical diffusion, a novel approach that leverages the inherent hierarchical community structure found in real-world datasets in order to alleviate this issue. Our method decomposes the diffusion process into a sequence of conditional diffusion processes, where a parent graph representing community structure and edge distribution guides the generation of individual communities and their cross-connections. This process recursively refines the graph from coarse to fine resolutions until the final graph is generated. Importantly, our method preserves permutation equivariance by construction, and we further design our method to allow the diffusion process to take into account the global edge distribution when connecting partitions explicitly during the diffusion process, while retaining the ability to distribute training and inference across machines to scale horizontally.

1 INTRODUCTION

Despite remarkable achievements on small and medium-sized graphs (e.g., molecules, proteins), recent denoising diffusion models (Vignac et al., 2023; Huang et al., 2022; Jo et al., 2022) encounter scalability challenges when applied to larger, more complex structures like social networks, financial networks (Li et al., 2023), road networks (Rong et al., 2023), and electronic circuits Oliveira et al. (2021).

Real-world networks often exhibit locally heterogeneous patterns, where groups of nodes ("*communities*") tightly connect within themselves but there exist sparse inter-connections between groups. This naturally forms a *hierarchical multi-resolution* structure in graphs, with smaller communities nested within larger ones. In such structure, lower levels capture fine-grained local interactions within communities, while higher levels represent broader relationships between communities and the overall network structure. Therefore, to truly capture the complexity of such graphs, it is crucial for an expressive graph generative network to not only learn the these community structures but also to model how they interact across different levels. By incorporating this *inductive bias* into the model architecture, we can train an expressive generative model that can effectively capture hierarchical structure inherent in graphs (Karami, 2023).

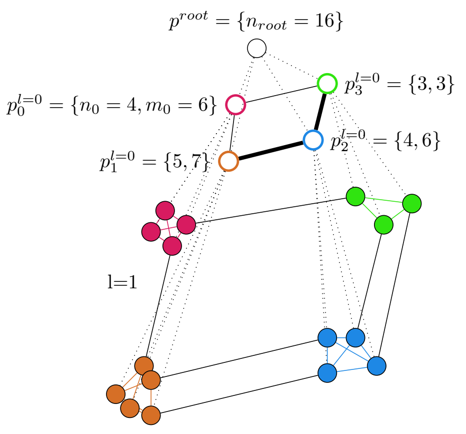
This work introduces **M**ulti-**R**esolution **Gra**ph **D**iffusion (**MRGraD**) which exploits the *community structure* and *multi-resolution hierarchy* inherent in many real world graph datasets to tackle this complexity while preserving expressiveness.

2 BACKGROUND

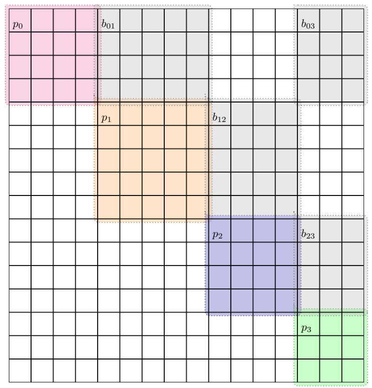
2.1 HIERARCHICAL MULTI-RESOLUTION GRAPH STRUCTURE

We denote a graph $\mathcal{G}$ as $(\mathcal{V}, \mathcal{E})$ with sizes $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Applying recursive graph partitioning (clustering), from fine to coarse resolution, we build a hierarchical graph , containing

[*]Equal Contribution.



(a) An example of hierarchical the hierarchical diffusion process. Each $\mathcal{G}_p^{(l)}$ is conditioned on a node embedding $Z[p]^{(l)}$ corresponding to it in the parent graph $\mathcal{G}^{(l-1)}$. $p_i^{l=0} = (n_i, m_i)$ refers to a parent node at layer $l = 0$ i representing n_i child nodes and m_0 internal child edges in the next layer $l = 1$.



(b) Leaf level adjacency matrix belonging to the graph in Figure 1a. Color shaded areas denote the partition subgraphs which are diffused independently and on which we compute structural features, gray shaded areas are the bipartites diffused together (mirrored on the lower triangular), unshaded areas are skipped during diffusion of the leaf level.

Figure 1: Summary of the core idea behind our method. While Karami (2023) and Davies et al. (2023) diffuse each bipartite region b_{ij} individually, we diffuse them jointly to preserve global edge awareness.

graphs at different levels of resolutions: $\mathcal{HG} = \{\mathcal{G}^0, ..., \mathcal{G}^{L-1}, \mathcal{G}^L\}$. Figure 1a depicts an $\mathcal{HG}$. Here, $\mathcal{G}^0 = (\{v_0\}, \{(v_0, v_0)\})$ is a singleton graph at the root of a dendrogram tree and $\mathcal{G}^L = \mathcal{G}$ is the original graph at the leaf level. A graph partitioning function $\mathcal{F} : \mathcal{V} \to \{1, ..., c\}$ partitions the nodes into c communities (a.k.a. clusters), where each cluster of nodes forms a sub-graph: $\mathcal{C}_i = (\mathcal{V}(\mathcal{C}_i), \mathcal{E}(\mathcal{C}_i))$. The cross-edges between adjacent communities form a *bipartite graph*: $\mathcal{B}_{ij} = (\mathcal{V}(\mathcal{C}_i), \mathcal{V}(\mathcal{C}_j), \mathcal{E}(\mathcal{B}_{ij}))$. In each level l, aggregating each community $\mathcal{C}_i^l$ to a super-node $v_i^{l-1} := Pa(\mathcal{C}_i^l)$ at the higher level (where $Pa(\cdot)$ maps a a partition to its representation in the parent graph) and mapping each bipartite to a super-edge (parent edge), $e_i^{l-1} = Pa(\mathcal{B}_{ij}^l) = (v_i^{l-1}, v_j^{l-1})$ induces a coarser graph, $\mathcal{G}^{l-1}$, at the parent level. Levels are indexed by superscripts. Self-loop and cross-edge weights in the parent level are the sum of weights within their corresponding community and bipartite, respectively. Thus, parent-graph edges have integer weights: $w_{ii}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{C}_i^l)} w_e$ and $w_{ij}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{B}_{ij}^l)} w_e$. Furthermore, the sum of all edge weights $w_0 := \sum_{e \in \mathcal{E}(\mathcal{G}^L)} w_e = |\mathcal{E}|$ is the weight of the root-level self loop, indicating the final graph size.. In the generation process, we sample the initial graph size, w_0, from the empirical distribution of edge counts of the graphs in the training set. Then we recursively generate (sample) the graph at each level given its parent level graph.

2.2 Diffusion Models

Diffusion models, introduced by Sohl-Dickstein et al. (2015) represent a distinct paradigm in generative modeling featuring two main components: a forward and a reverse Markov process. In the forward diffusion process, represented by $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^T q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, data $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ is gradually corrupted into a sequence of increasingly noisy latent variables $\boldsymbol{x}_{1:T} = \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$, while in the the reverse Markov process, a network $p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^T p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is trained to reverse it by progressively denoise the latent variables, ultimately recovering the original data $\boldsymbol{x}_0$. For effective forward diffusion, it's crucial that the transition probability $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ converges to a fixed distribution $\pi(\boldsymbol{x})$ as the number of diffusion steps T approaches infinity, regardless of the initial data point $\boldsymbol{x}_0$. Therefore, we can choose $\pi(\boldsymbol{x})$ as the prior distribution $p(\boldsymbol{x}_T)$ when the number of steps T is sufficiently large. Furthermore, for efficient training, the transition probability $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$

should have a closed-form expression or be pre-computed, allowing efficient sampling of $\boldsymbol{x}_t$ at each step.

3 Methodology

3.1 Hierarchical Generative Process

Our goal is to develop a *hierarchical multi-resolution denoising diffusion model* for generating graphs from coarse to fine resolutions. Given a hierarchical graph $\mathcal{HG} := \{\mathcal{G}^0, ..., \mathcal{G}^{L-1}, \mathcal{G}^L\}$, the generative distribution can be factorized using the chain rule as:

$$p(\mathcal{G}) = p(\{\mathcal{G}^L, ..., \mathcal{G}^0\}) = p(\mathcal{G}^L \mid \{\mathcal{G}^{L-1}, ..., \mathcal{G}^0\}) \; ... \; p(\mathcal{G}^1 \mid \mathcal{G}^0) \, p(\mathcal{G}^0) = \prod_{l=0}^{L} p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) \times p(\mathcal{G}^0) \tag{1}$$

Therefore, it is a Markov process. In simpler terms, we start from the root level (coarsest resolution), $\mathcal{G}^0$, then conditioned that information we predict the next level of detail $\mathcal{G}^1$, and progressively add details until we reach the final, full graph $\mathcal{G}^L$.

We leverage the hierarchical structure of $\mathcal{HG}$, to efficiently generate graphs. By assuming that community probabilities within a level given the parent graph, $p(\mathcal{C}_i^l \mid \mathcal{G}^{l-1})$, are independent, we can decompose the conditional generative probability as follows:

$$p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) = \prod_{i \,\in\, \mathcal{V}(\mathcal{G}^{l-1})} p\big(\mathcal{C}_i^l \mid \mathcal{G}^{l-1}\big) \times p\big(\{\mathcal{B}_{ij}^l\}_{(i,j) \in\, \mathcal{E}(\mathcal{G}^{l-1})} \mid \mathcal{G}^{l-1}, \{\mathcal{C}_k^l\}_{\mathcal{C}_k^l \in \mathcal{G}^l}\big) \tag{2}$$

This decomposition indicates that generating a graph at level l hinges upon two fundamental aspects:

1. **Independent Community Generation**: Each community $\mathcal{C}_i^l$ is independently generated, conditioned on the parent graph at a higher level. This enables parallel generation of small sub-graphs rather than generating a large graph, enhancing computational efficiency.
2. **Inter-Community Connection Modeling**: Considering both the parent graph and the generated communities, the collection of bipartite graphs $\{\mathcal{B}_{ij}^l\}$ connecting these communities is generated jointly. This approach captures the complex inter-dependencies between communities.[1]

3.2 Denoising Diffusion Model for Community and Cross-Community Generation

For community sub-graphs, we follow DiGress Vignac et al. (2023) to define a diffusion process in the discrete state-space for graphs. Each edge type e (including type: 0 for absent edges) and node type v are represented by categorical distributions $q_t(E_{ij}) \in S^{d_e}, q_t(X_i) \in S^{d_n}$ (where S^k is the k dimensional simplex) and independently diffusing them at step t with a discrete state space transition matrix $\boldsymbol{Q}^t \in \mathbb{R}^{d \times d}$, where the size of the matrix is determined by the number of edge types for edge features, $d = d_e$, and the number of node types for nodes, $d = d_v$. Here, Q_{ij}^t represents the probability of jumping from state i to state j at time step t: $q(z^t = j \mid z^{t-1} = i) = Q_{ij}^t$.

It was shown in Vignac et al. (2023), adopting the marginal distribution –$\boldsymbol{m}_X$ and $\boldsymbol{m}_E$ for the node and edge features, respectively– as the prior distribution and defining transition probabilities to asymptotically approach them (instead of simply using uniform distributions), the sparse structure of the original graph is maintained during diffusion, leading to faster convergence with fewer steps. To achieve this property, it is sufficient to use

$$\boldsymbol{Q}_X^t = \alpha^t \boldsymbol{I} + \beta^t \; \mathbf{1}_a \boldsymbol{m}_X' \quad \text{and} \quad \boldsymbol{Q}_E^t = \alpha^t \boldsymbol{I} + \beta^t \; \mathbf{1}_b \boldsymbol{m}_E' \tag{3}$$

, with α^t, β^t being noise schedule dependent weights. Using this approach, the transition probability from state i to state j reflects the marginal probability of category j in the training set, guaranteeing sparsity.

[1]Note that we choose to predict all bipartites together at once, but it could also be reduced to generating each of them independently.

Table 1: Structure exploiting graph generation models compared. Columns: PE=Permutation Equivariant, HS=Horizontal Scalability, C=Complexity, EA=Edge Aware, i.e. not edge independent. The † marks models with $\mathcal{O}\left(n^2\right)$ complexity via spectral features in the worst case $m \approx n^2$. A + marks methods which adaptively choose subsets of edges to diffuse at each timestep, * a possible partial EA via parent graph embeddings.

MODEL	PE	HS	C	EA
HIGEN (KARAMI, 2023)	NO, AR	NA,AR	$\mathcal{O}\left(L^2\right)^{\dagger}$	✓
DIGREES (VIGNAC ET AL., 2023)	✓	NO	$\mathcal{O}\left(Tn^2\right)$	✓
EDGE (CHEN ET AL., 2023)	✓	NO	$\mathcal{O}\left(T\max(K^2, m)\right)^{\dagger,+}$	✓
(BERGMEISTER ET AL., 2023)	✓	NO	$\mathcal{O}\left(n+m\right)^{\dagger,+}$	✓
(DAVIES ET AL., 2023)	✓	✓	$\mathcal{O}\left(\max_p\left(n_p\right)^2\right)^{\dagger}$	NO*
MRGRAD	✓	✓	$\mathcal{O}\left(n+m\right)$	✓*

Community Denoising: The denoising network for communities at level l predicts the node and edge features of a clean community graph, $\mathcal{C}^{l,0} = (\boldsymbol{X}^0, \mathbf{E}^0)$, given a noisy graph $\mathcal{C}^{l,t} = (\boldsymbol{X}^t, \mathbf{E}^t)$ as input and conditioned on features of the parent graph, formally:

$$p_\theta\big(\mathcal{C}^{l,0} = (\boldsymbol{X}^0, \mathbf{E}^0) \mid \mathcal{C}^{l,t} = (\boldsymbol{X}^t, \mathbf{E}^t), \mathcal{G}^{l-1}\big).$$

We characterize the denoising network by the graph transformer network proposed by Dwivedi & Bresson (2020) augmented with spectral features for efficiency. Although its complexity is quadratic in the number of nodes ($\mathcal{O}(n_{\mathcal{C}}^2)$), it operates on smaller community graphs, enabling parallelization and resulting in a scalable model.

Bipartite Graph Denoising: After generating the community graphs at level l, we only diffuse the bipartite graphs, and we predict the clean inter-community connections (bipartite graphs) using:

$$p_\theta\big(\{\mathcal{B}_{ij}^{l,0}\}_{(i,j)\in\,\mathcal{E}(\mathcal{G}^{l-1})} \mid \{\mathcal{B}_{ij}^{l,t}\}_{(i,j)\in\,\mathcal{E}(\mathcal{G}^{l-1})}, \{\mathcal{C}_k^l\}_{\mathcal{C}_k^l\in\mathcal{G}^l},\ \mathcal{G}^{l-1}\big)$$

where noisy bipartites and clean communities are given as the input to the model and it is conditioned on the parent graph features. The denoising network is parameterized with a customised version of GraphGPS neural network proposed by Rampášek et al. (2022a) which combines local message-passing for the local sparsity structure in the graph together with global attention mechanism. Positional and structural node features are used for higher expressiveness. Spectral features are computed based on the local adjacency matrix of each community for efficiency. Since community graphs are not diffused, these features are pre-computed once and reused during denoising process. Therefore, this network has a linear complexity of $\mathcal{O}(n + m)$. We further discuss the scalability of our approach in Appendix A.

4 RELATED WORK

Aside from Vignac et al. (2023); Karami (2023); Austin et al. (2021); Rampášek et al. (2022b) on which we build directly, the closest related work to ours is Davies et al. (2023) which concurrently extend DiGress in a way similar to us (coarse level parent graph modeling communities, independent diffusion of each partition, then connecting the partitions in a horizontally scalable way). However, unlike our method they do not take into account global structure directly during the inter-community generation and are not able to leverage structural node features (see Appendix A).

The next closest works are EDGE (Chen et al., 2023) and the concurrent (Bergmeister et al., 2023). Both choose a slightly different approach to scalability, namely that of a graph coarsening process, which incrementally adds nodes during the diffusion process. For this they derive graph-theoretically guided coarsening-diffusions (degree guided for Chen et al. (2023), based on the local spectral variability of Loukas Coarsening algorithm for Bergmeister et al. (2023)).

While this improves scalability and has the benefit of allowing extrapolation across graph sizes, it also makes it more complicated to split training and inference across multiple machines, as at least during the selection forward pass, the whole graph needs to be processed. The partition and bipartite

Table 2: Comparison between our proposed methods and the prior art on the Enzyme dataset using EMD following Karami (2023),see paper for details on metrics (lower is better). For each metric, we select the best checkpoint based on the validation value, then report the test metric. A - indicates missing values.

	ENZYME			
	DEG.	CLUS.	ORB.	SPCT.
DIGRESS	0.0040	0.0830	0.0020	-
HIGEN-M	0.0270	0.1570	0.0012	-
HIGEN	0.0120	0.0380	0.0007	-
S-MRGRAD-1K	0.0064	0.0653	0.0047	0.0590
S-MRGRAD-50	0.0134	0.0584	0.0091	0.0182
D-MRGRAD-50	0.0763	0.3457	0.0025	0.0183

diffusion models at each layer of our recursive hierarchical modeling can be trained completely independently and the inference can be distributed and pipelined across multiple nodes.

Finally, it is worth mentioning Yan et al. (2023) independently invented the sampling process also leveraged by Bergmeister et al. (2023) and argues that equivariance might actually be harmful in learning graph distributions. While we think the work makes an important point,we disagree and choose to retain permutation equivariance, and comment on why in-depth in Appendix D (briefly: the possibility of non-unique samples from isomorphism classes established by Ivanov et al. (2019), concerns about overfitting and a preference for explicit inductive biases for generalisation).

1. hierarchical, specialized structure seems to emerge naturally in the large scale data regime as more and more symmetries are learned from the data in the pursuit of generalizable features, even when given minimal inductive biases e.g. using massively scaled MLPs (Bachmann et al., 2023) or transformers (Zhang et al., 2023; Shen et al., 2023)
2. modular architectures and specialization has been shown to provide generalization benefits in Mittal et al. (2022).

Thus, we leave exploring module-free and hierarchy free generative graph models for future work.

5 EXPERIMENTS

For ease of comparison with previous work, here we will focus on the Enzyme, DD2, Ego2 and SBM datasets.

1. DD2 which is the protein dataset of Dobson & Doig (2003) comprised of 918 graphs with $n \in [100, 500]$ nodes representing physical structures, linked when they are closer in physical proximity than a threshold
2. Ego2 a dataset of 757 graphs representing the 3-hop ego networks extracted from the CiteSeer dataset, with $n \in [50 - 300]$ Sen et al. (2008)
3. Enzyme 587 graphs with $n \in [10, 125]$ representing proteins structures derived from Schomburg et al. (2004)
4. SBM, a 200 graph Synthetic Block Model dataset with 2-5 communities with $n \in [20, 40]$ taken from Martinkus et al. (2022).

We compare both the edge aware ordinal and a simplified categorical version of our model which ignores the weight of each edge (referred to as D-MRGraD for **D**iscrete Multi-Resolution Graph Diffusion and S-MRGraD for **S**implfied MRGraD respectively) to the DiGress and HiGen baselines.

We follow Liao et al. (2019) with their metrics (Degree Distribution, Clustering Coefficients, Orbits up to 4 and Spectral, i.e. eigenvalue distribution), in particular in reporting earth movers distance (EMD) on Enzyme instead of MMD. EMD and MMD are estimated on sampling the same number of graphs from the model as are in the validation/test set. Also for partitioning we use the Louvain algorithm for hierarchical segmentation as in Karami (2023).

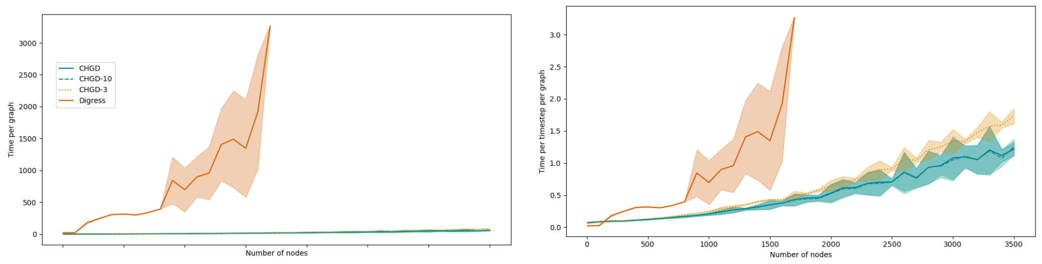


Figure 2: Comparison of the scaling behavior of DiGress and our proposed method (labeled CHGD here) under identical conditions. We sample $k = 10$ graphs at each n_{new} with $n_{step} = 100$ steps, using the sparsity pattern of a random graph from the `Enzyme` dataset (i.e. $m_{new} = \alpha m_{old} = {n^2_{new}}/{n^2_{old}} m_{old}$) with checkpoints trained `Enzyme` with $T_{digress} = 1e3$ and $T_{MRGraD} = 50$. We report up to the point where all methods crash OOM. Measured on a 80GB NVIDIA A100, on the same setting described in Appendix C. MRGraD indicates scaling the top level number of communities with $n_{top,new} = \min\left(\sqrt{n_{new}} \min\left(1, \alpha\right)\right)$, while MRGraD-3 and 10 respectively indicate additionally constraining the single partiton on level 1 to that many clusters in order to simulate reaching a resolution limit during clustering and failing to reduce n_p on the second layer.

We use rejection sampling during the last diffusion step to ensure that the generated community graph is connected, and also on the complete generation process on any remaining disconnected graphs.

We used `Enzyme` to tune each variant manually and then use these settings for all the others. We used hierarchical partitioning technique of Karami (2023) with identical settings for dataset pre-processing into $L = 2$ layers. Detailed architectures, hyper-parameters and experimental details can be found in Appendix C.

As can be seen in the metrics in Table 3,Table 2, the latency plot in Figure 2 and the visualisations in Figures 3 to 6, while clearly not yet state of the art in terms of fidelity, our method succeeds in combining the strengths of both DiGress and Higen. On `Enzyme`, for which the model was tuned,Table 2 shows that we are able to roughly match DiGress when using the same number of diffusion steps, and are able to massively reduce number of diffusion steps with tolerable or no loss of fidelity depending on the metric evaluated. We can also see that the inclusion of the edge weight continues to benefit modeling in terms of the Spectral and Orbit EMD metrics. On the other datasets shown in Table 3 results are similar albeit more mixed and we are hopeful that further tuning can improve the performance of our model and bring it on par or beyond DiGress to compete with the results reported in (Bergmeister et al., 2023) and (Chen et al., 2023).

In Figure 2, we show that not only can we reduce the number of diffusion steps, we also achieve massive scaling improvements. Comparing to Figure 6 of Bergmeister et al. (2023), at $n = 1500$ they achieve ≈ 2 seconds per diffusion step with $T = 256,\ t_{total} \approx 200s$ on a dataset of planar graphs, while our setup on enzyme-like sparsity structures achieves ≈ 0.5 seconds per diffusion step with $T = 50,\ t_{total} \approx 25s$ at that graph size. We can't assume perfect transferability due to the dataset and hardware being different, however our DiGress implementation appears to be about $1.5\times$ slower than their setup and implementation. Using this margin ($4\times$ faster with a reference $1.5\times$ slower), we are comfortable claiming competitiveness and are planning to replicate Bergmeister et al. (2023) and perform exact measurements in future work. Please refer to appendix C for more experiments and experimental details.

6 Conclusion

In this work, we introduced Multi-Resolution Graph Diffusion (MRGraD), a framework that leverages the inherent hierarchical modularity observed in real-world graph data. Our evaluations in Section 5 demonstrate its promising potential, but further work remains in tuning, detailed evaluation on more datasets and comparison with strong baselines like Bergmeister et al. (2023). Future work could also be the exploration of combining and extending ideas from Bergmeister et al. (2023); Yan et al. (2023) and Chen et al. (2023) with our framework. A shared limitation shared by structure exploiting models is the suitability of the prior (i.e., the community structure for our work, spectrally guided local expansion for Bergmeister et al. (2023) etc.). We are excited to continue investigation in these directions.

REFERENCES

Emmanuel Abbe and Enric Boix-Adsera. On the non-universality of deep learning: Quantifying the cost of symmetry. *Advances in Neural Information Processing Systems*, 35:17188–17201, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6d9aac9407bcb1a5957401fa0b8de693-Abstract-Conference.html.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling mlps: A tale of inductive bias. *arXiv preprint arXiv:2306.13575*, 2023.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and Scalable Graph Generation through Iterative Local Expansion, December 2023. URL http://arxiv.org/abs/2312.11529.

Sudhanshu Chanpuriya, Cameron N Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. On the power of edge independent graph models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=OrPraBRj45z.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Alex O. Davies, Nirav S. Ajmeri, and Telmo M. Silva Filho. Size Matters: Large Graph Generation with HiGGs, November 2023. URL http://arxiv.org/abs/2306.11412.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv:2309.03800*, 2023.

Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 31376–31390. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/cb2a4cc70db72ea779abd01107782c7b-Paper-Conference.pdf.

Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 201–210. IEEE, 2022.

Sergei Ivanov, Sergei Sviridov, and Evgeny Burnaev. Understanding isomorphism bias in graph data sets, 2019.

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

Mahdi Karami. HiGen: Hierarchical graph generative networks. *arXiv preprint arXiv:2305.19337*, 2023.

Bobak T. Kiani, Thien Le, Hannah Lawrence, Stefanie Jegelka, and Melanie Weber. On the hardness of learning under symmetries, January 2024. URL http://arxiv.org/abs/2401.01869.

Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. {GG}-{gan}: A geometric graph generative adversarial network, 2021. URL `https://openreview.net/forum?id=qiAxL3Xqx1o`.

Gautier Krings and Vincent D. Blondel. An upper bound on community size in scalable community detection, 2011.

Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. Diga: Guided diffusion model for graph recovery in anti-money laundering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4404–4413, 2023.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Frank McSherry, Michael Isard, and Derek G Murray. Scalability! but at what {COST}? In *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, 2015.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a Modular Architecture Enough? *Advances in Neural Information Processing Systems*, 35:28747–28760, December 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/b8d1d741f137d9b6ac4f3c1683791e4a-Abstract-Conference.html`.

Isadora Oliveira, Marcelo Danigno, Paulo F. Butzen, and Ricardo Reis. Benchmarking Open Access VLSI Partitioning Tools. In *2021 IEEE 12th Latin America Symposium on Circuits and System (LASCAS)*, pp. 1–4. IEEE, 2021. ISBN 978-1-72817-670-3. doi: 10.1109/LASCAS51355.2021.9459131. URL `https://ieeexplore.ieee.org/document/9459131/`.

Euan Ong and Petar Veličković. Learnable commutative monoids for graph neural networks. In Bastian Rieck and Razvan Pascanu (eds.), *Proceedings of the First Learning on Graphs Conference*, volume 198 of *Proceedings of Machine Learning Research*, pp. 43:1–43:22. PMLR, 09–12 Dec 2022. URL `https://proceedings.mlr.press/v198/ong22a.html`.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022a.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022b.

Can Rong, Jingtao Ding, Zhicheng Liu, and Yong Li. Complexity-aware large scale origin-destination network generation via diffusion model, 2023.

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.

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

Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. ModuleFormer: Modularity Emerges from Mixture-of-Experts, 2023. URL `http://arxiv.org/abs/2306.04640`.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265, 2015.

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*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Qi Yan, Zhengyang Liang, Yang Song, Renjie Liao, and Lele Wang. SwinGNN: Rethinking Permutation Invariance in Diffusion Models for Graph Generation, July 2023. URL http://arxiv.org/abs/2307.01646.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent Modularity in Pre-trained Transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4066–4083. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.250. URL https://aclanthology.org/2023.findings-acl.250.

A Discussion on the Scalability of the diffusion process

For very large graph processing, horizontal scaling, i.e. dividing processing a single instance load between multiple nodes, has become the industry standard - sometimes excessively so McSherry et al. (2015). Notwithstanding the important cautioning of this paper, when deep learning and massive graph processing meet, horizontal scaling remains an important area of research as the COST in the sense of McSherry et al. (2015) will usually be found relatively quickly. E.g. for the structural features which form an important part of digress, we incur a $\mathcal{O}\left(n^2\right)$ complexity every time they need to be recomputed.

While it might be tempting to simply diffuse each partition and bipartite graph independently, it is likely that this would hamper the models ability to enforce global structures, e.g. the specific ordering of protein segments or molecules, since it would require the model to encode a lot of information into each parent node embedding. Indeed, Chanpuriya et al. (2021) have proven that edge independent models have inherent limits in their ability to model triangle rich graphs, which are common in real world datasets. As noted in Chen et al. (2023) diffusion models like DiGress do not suffer from this theoretical limitation since every sample step depends on all previously sampled edges, but this property would be lost if we diffuse each section of the graph independently.

Since our core assumption is that the partitions do not strongly depend on the rest of the graph given some information about the global structure, we choose the following tradeoff:

1. We denoise each partition independently of any other section of the graph, conditioning only on the parent graph
2. We then denoise *all* bipartite edges *jointly*, meaning that they have full dependence on all regions of the graph
3. However, we retain the locally computed spectral node features of each partition, side stepping the full $\mathcal{O}\left(n^2\right)$ complexity for the feature computation and reducing it proportional to the size of largest community as $\mathcal{O}\left(\max_i n_{\mathcal{C}_i}^2\right)$. We use the (linear complexity) GraphGPS model (Rampášek et al., 2022b) with these locally computed features and use all edges (bipartite and partition edges) for in the massage passing, in order to compute node embeddings, then derive the edge features from these embeddings

With this, the total complexity of the bipartite edge computation on layer l is only quadratic in the largest community on that level $\mathcal{O}\left(\max_i n_{\mathcal{C}_i}^2\right)$ and the feature computation can be done according to the $\mathcal{O}\left(n\right)$ complexity of GraphGPS. Note that due to the resolution limit we can use the resolution parameter γ in the Louvain algorithm to impose an upper bound community sizes Krings & Blondel (2011), ensuring that no community with $n_{\mathcal{C}_i}$ will contain more than $\frac{n_{\mathcal{C}_i}}{(n_{\mathcal{C}_i}-1)\gamma}m$ edges,i.e. $n_{\mathcal{C}_i} \leq 1 + \frac{m}{\gamma d_{max}}$ for d_{max} the largest node degree in the graph, which allows us to choose gamma s.t. e.g. $\max n_{\mathcal{C}_i} \leq \sqrt{n}$.

The combined approach can then in theory achieve an overall (parallelized) complexity that scales linearly with the graph size, $\mathcal{O}(n)$, for the final graph generation (by diffusing $\sqrt{n}$ communities in parallel with $\mathcal{O}\left(n\right)$ total complexity each, then connecting them with the $\mathcal{O}\left(n+m\right)$ GraphGPS, making it efficient for large-scale graph generation.

B Detailed related work

Aside from Vignac et al. (2023); Karami (2023); Austin et al. (2021); Rampášek et al. (2022b) on which we build directly, the closest related work to ours is Davies et al. (2023) which concurrently extend DiGress in a way similar to us (coarse level parent graph modeling communities, independent diffusion of each partition, then connecting the partitions in a horizontally scalable way). However, unlike our method they do not take into account global structure directly during the inter-community generation and are not able to leverage structural node features (see Appendix A). The next closest works are EDGE (Chen et al., 2023) and the concurrent (Bergmeister et al., 2023). For additional discusison of related work see Appendix B .

1. hierarchical, specialized structure seems to emerge naturally in the large scale data regime as more and more symmetries are learned from the data in the pursuit of generalizable features,

even when given minimal inductive biases e.g. using massively scaled MLPs (Bachmann et al., 2023) or transformers (Zhang et al., 2023; Shen et al., 2023)

2. modular architectures and specialization has been shown to provide generalization benefits in Mittal et al. (2022).

For these reasons, we leave the exploration of module-free and hierarchy free generative graph models for future work.

C Experimental Details

C.1 Manual tuning

In order to tune the hyper parameters, we started with Digress SBM parameters and then performed an approximate grid (steps chosen based on experience) search across learning rate, batch size, weight decay parameters, then reduced the diffusion steps incrementally until we observed a breakdown in performance. The submitted version of the manuscript mentions rule of thumb adaptation but this was actually a leftover from a previous draft and is removed in the camera ready.

C.2 Additional Results

Table 3: Further comparison S-MRGraD-50 and the prior art. Following Karami (2023) from which we also take the baseline numbers, `Enzyme` is reported with EMD, the rest is reported with MMD. For each metric, we select the best checkpoint based on the validation value, then report the test metric. A - indicates missing values.

	ENZYME				EGO2			
	Deg.	Clus.	Orb.	Spct.	Deg.	Clus.	Orb.	Spct.
DiGress	0.0040	0.0830	0.0020	-	0.0708	0.0092	0.1205	-
HiGen-m	0.0270	0.1570	0.0012	-	0.1140	0.0378	0.0535	-
HiGen	0.0120	0.0380	0.0007	-	0.0472	0.0031	0.0387	-
S-MRGraD-50	0.0134	0.0584	0.0091	0.0182	0.2150	0.0410	0.0609	0.0425
	DD2				SBM			
	Deg.	Clus.	Orb.	Spct.	Deg.	Clus.	Orb.	Spct.
DiGress	-	-	-	-	0.0013	0.0498	0.0433	-
HiGen-m	0.0041	0.0109	0.0472	0.0061	0.0017	0.0503	0.0604	0.0068
HiGen	0.0012	0.0435	0.0234	0.0025	0.0019	0.0498	0.0352	0.0046
S-MRGraD-50	0.1250	1.0473	0.2771	0.1561	0.0381	0.0126	0.0446	0.0532

C.3 Input And Auxiliary Features

Following Vignac et al. (2023), across all variations we represent a graph $\mathcal{G}$ as a tuple of feature tensors $(\mathbf{X}, \mathbf{E}, \mathbf{y})$ with $\mathbf{X} \in \mathbb{R}^{n \times d_V}, \mathbf{E} \in \mathbb{R}^{n \times n \times d_E}, \mathbf{y} \in \mathbb{R}^{d_y}$ and the feature dimensions d_V, d_E, d_y as follows.

In each setting, on top of any features listed below, we compute auxiliary features $\mathbf{X}', \mathbf{E}', \mathbf{y}'$ with sizes d'_V, d'_E, d'_y as follows (any redundancies are due to code reuse)

1. $d'_V = (1 + 3 + 1 + k_{vec}) + (1 + 1 + 1) = 7 + k_{vec}$,corresponding to a constant dummy feature set to 1.0, the number of $3, 4$ and 5 cycles the node is involved in, an indicator on whether the node is part of a component which isn't the largest connected components (LCC), the node specific elements of the eigenvectors corresponding to the k_{vec} smallest non-zero eigenvalues, the size of the partition that the node belongs to (estimated in the parent level), the total number of nodes in the expected graph and finally, the normalized timestep t/t_{max}

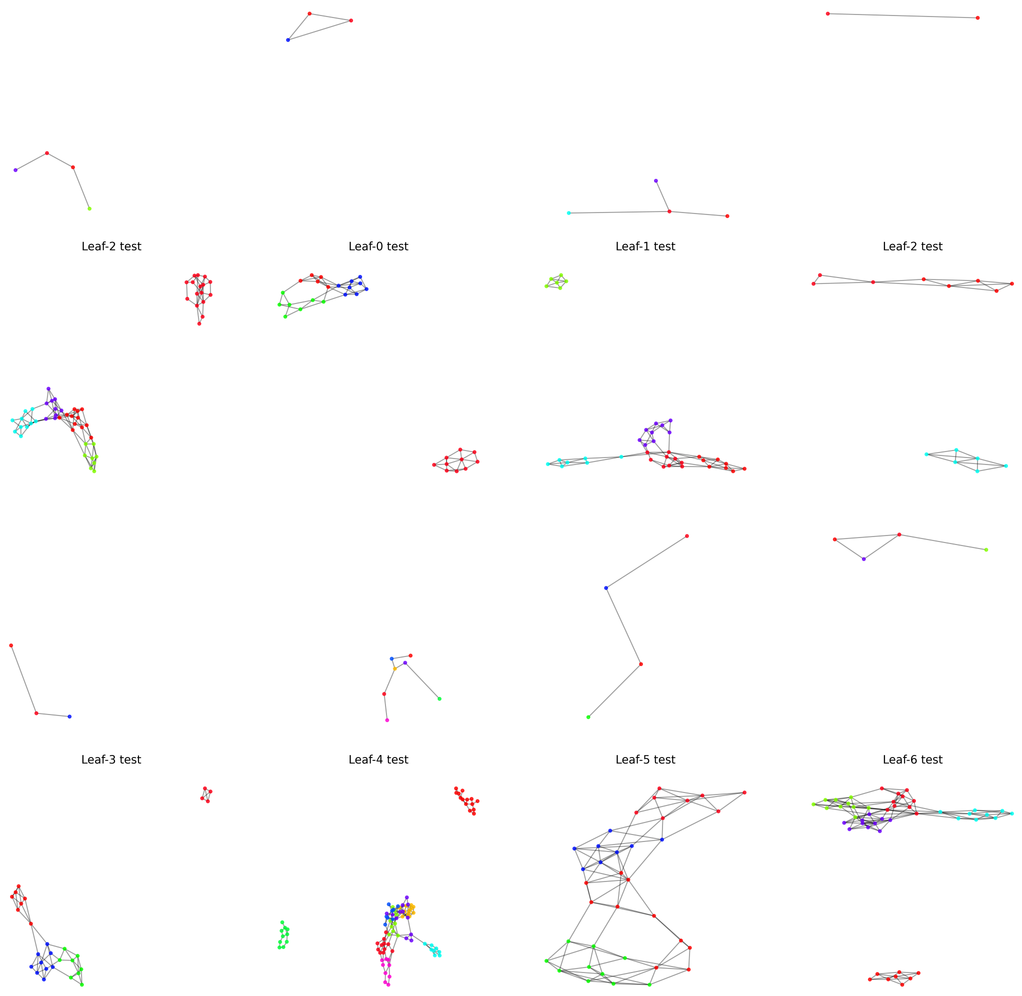


Figure 3: `Enzyme`-D-MRGraD-50

2. $d'_E = 0$ no auxiliary features on the edges
3. $d'_y = (1 + 4 + 1 + k_{val} + 1) + (1 + 1 + 1) = 10 + k_{val}$, the number of nodes n in the graph, the number of $3, 4, 5$ and 6 cycles within the graph, the number components in the graph, the k_{val} lowest non-zero eigenvalues, the normalized timestep t/t_{max}, the size of the partition that the node belongs to (estimated in the parent level), the total number of nodes in the expected graph and finally, the normalized timestep t/t_{max}

As soon as we have a parent graph, we also concatenate the node embedding corresponding $Z_{V,i}$ corresponding to the partition the i a given node belongs to with the node and global features, addimg dim_{parent} dimensions.

C.3.1 Simplified Discrete Model

In the simplified discrete setting (without edge weights,types or node types), we use a simple one hot encoding of a (weightless) edge and no other features for nodes and global features, i.e.

1. the final node feature size is $d_V = d'_V + dim_{parent}$
2. the final edge feature size is $d_E = 2 + d'_E = 2$
3. the final global feature size is $d_y = d'_y + dim_{parent}$

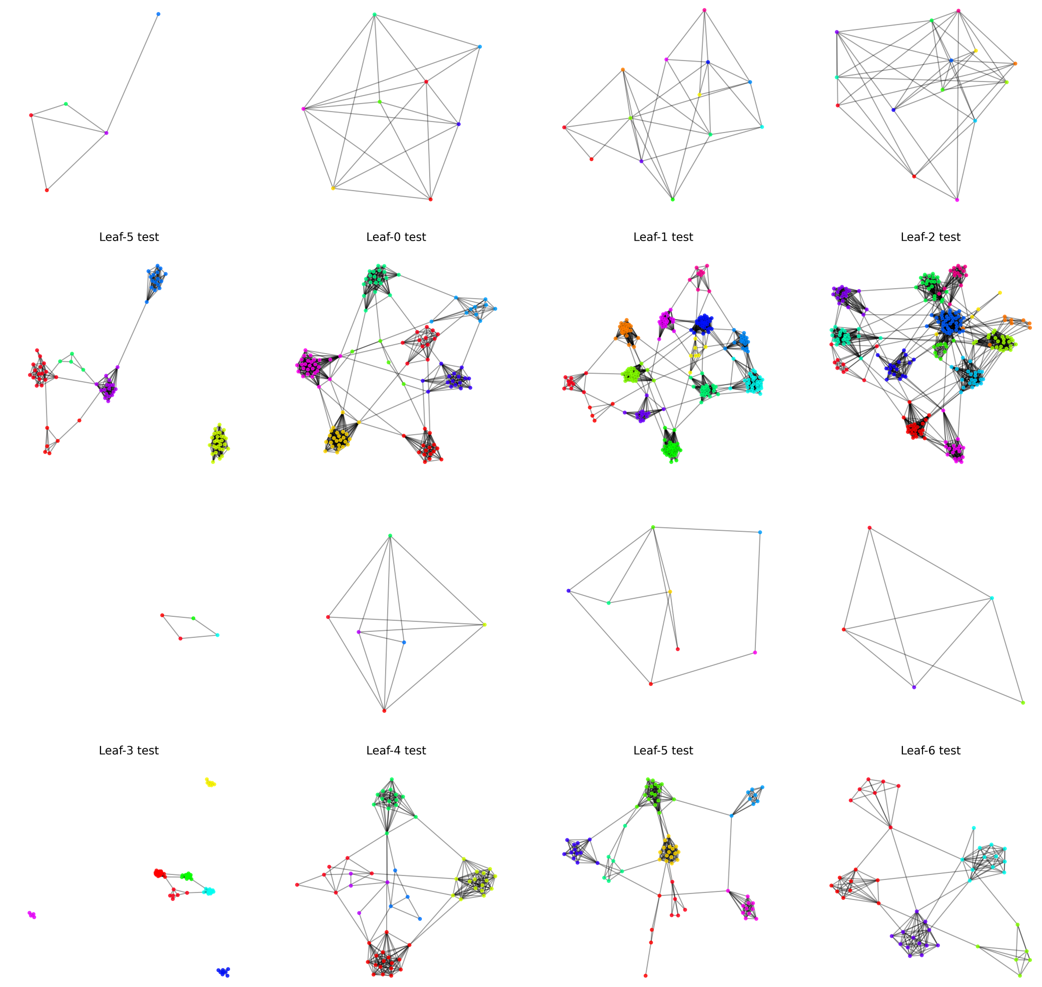


Figure 4: `Ego2`-D-MRGraD-50

C.3.2 Categorical Integer Model

We one-hot encode the modeled edge weights as well as the community sizes tracked as node features, leading to features increased by w_{max} and $n_{part,max}$ respectively

1. the final node feature size is $d_V = d'_V + dim_{parent} + n_{part,max}$
2. the final edge feature size is $d_E = d'_E + w_{max} = w_{max}$
3. the final global feature size is $d_y = d'_y + dim_{parent}$

C.4 Diffusion model parametrization

We use two types of GNNs for the diffusion process

1. the GraphTransformer used in Digress Rampášek et al. (2022b) (Digress GAT), used for generating the partitions
2. the GraphGPS Rampášek et al. (2022b) used for the bipart stitching

We build on the publically available code bases of both Karami (2023) and Vignac et al. (2023) and do not modify them except for hyperparameter tweaks and our modification for local propagation.

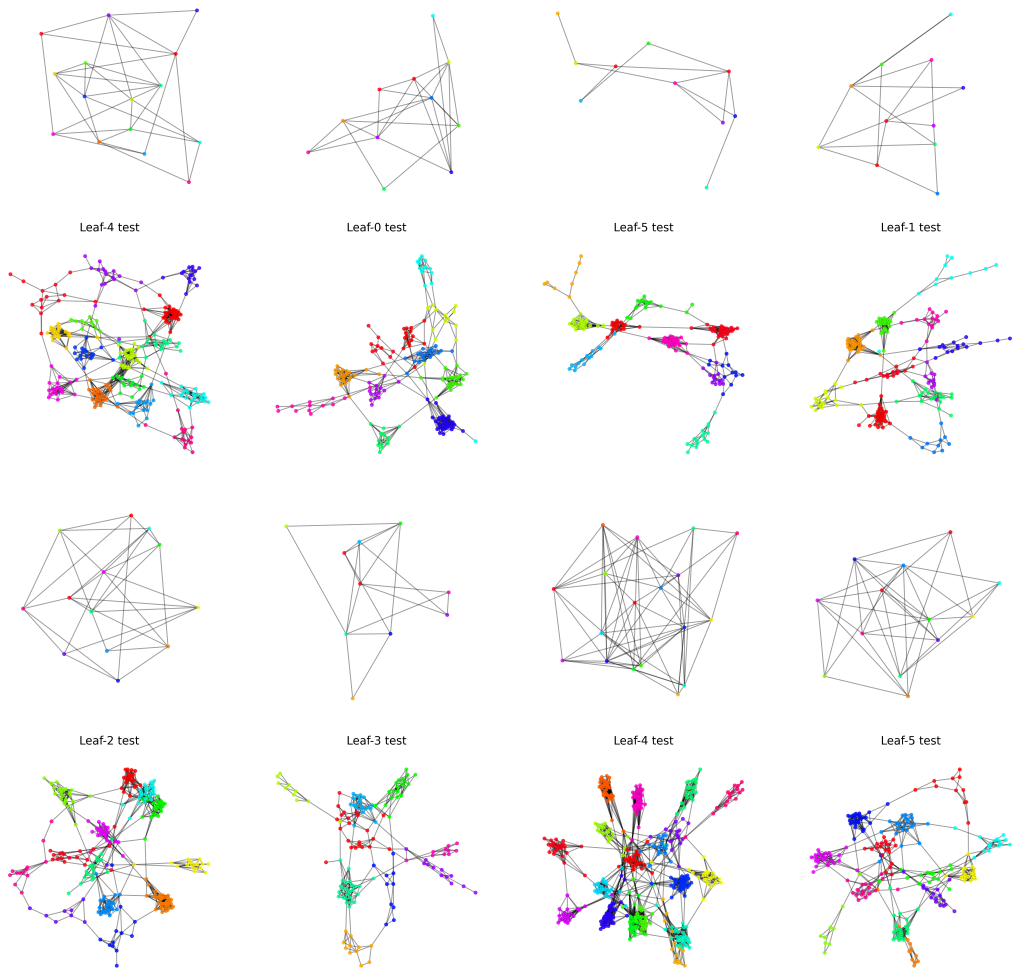


Figure 5: `DD2`-D-MRGraD-50

We also use a simple MLP to predict the size of each community.

Hyperparameters across all settings are given in Appendix C.5

C.5 HYPERPARAMETERS

Table 4: Hyperparamters shared across all experiments

lr	0.0002
n_{batch}	8
WEIGHT DECAY	$1e-12$
OPTIMIZER	ADAMW
k_{vec}	2
k_{val}	5

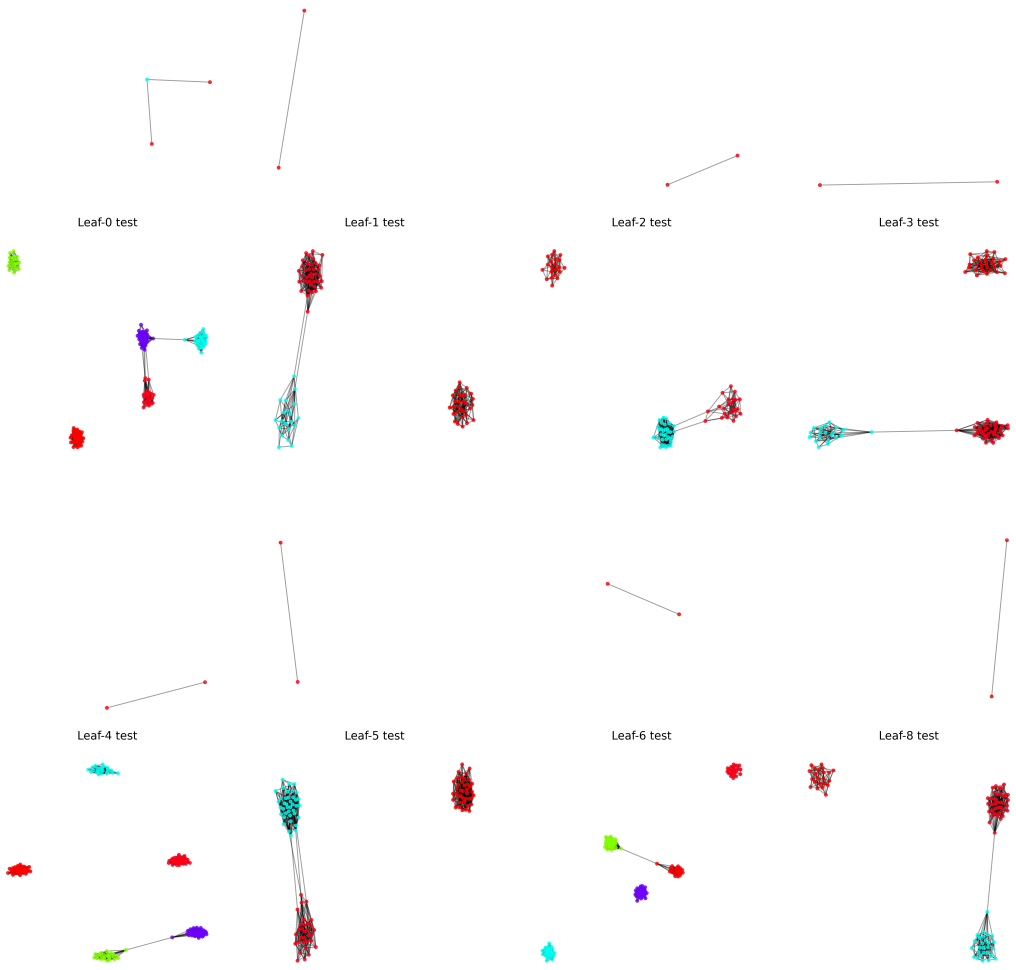


Figure 6: SBM-D-MRGraD-50

Table 5: Hyperparameters for S-MRGraD-1k (see also Table 3)

T_{BP}	1000
EPOCHS	5000
T_P	1000

D On Yan et al. 2023 and permutation equivariant models

While we think recent work like Kiani et al. (2024); Abbe & Boix-Adsera (2022) lends credence to the core of their argument with which we fully agree ("invariant target distributions are hard to learn"), we think that the reasoning in the paper is subtly flawed for three reasons

1. "recall" is not the right metric for distributional modeling since it is also a measure of memorization and overfitting
2. their arguments assumes unique samples of $\mathcal{G}$ in the training dataset, which was demonstrated to be wrong in common graph datasets in Ivanov et al. (2019).

Table 6: Hyperparameters for S-MRGraD-50 (see also Table 3)

T_{BP}	50
EPOCHS	1000
T_P	50

3. their lemmas are all directly concerned with $p_\theta(\mathbf{A}^*)$ the "effective" target distribution, but this is not what equivariant generators or diffusion models need to earn it is $p_\theta(\mathcal{G})$, the probability of a given graph, which we argue they achieve by implicitly learning $p_\theta(\mathbf{A}^{\pi_{latent}}|\pi_{latent})$, i.e. a single canonical instance from the permutation set

We argue based on the results already presented in Krawczuk et al. (2021)(the earliest permutation equivariant graph generative model we are ware of) that the difficulty of learning this function due to the size of target distribution but an optimisation and identification problem, i.e. training algorithm needs to identify and commit to the "canonical" π_{latent}). Already in Krawczuk et al. (2021) the authors noted that symmetry breaking using latent positional embeddings during training was crucial to help the network identify $A^{\pi_{latent}}$ solve what the called the "collision problem" of mapping specific sets in the latent states to specific embeddings or probability sets at the output. One can easily imagine a sequence of training examples that will lead the lead to the "reinterpretation" of these latent sets as belonging to a different graph at each gradient update, slowing down the progress in training to essentially random noise, which we think maps cleanly onto the problem of (Kiani et al., 2024, Sec. 4.3). It is also worth noting that they also argued that can regain the uniform distribution over permutations by randomly permuting the latent positional encoding at inference - exactly the strategy re-invented (and, to acknowledge their contribution, properly formalized) by Yan et al. (2023). However, due to their PE inductive bias, they do not need to do this during training.

The results of Abbe & Boix-Adsera (2022) support the idea of "learning under symmetries" having potential pathologies. *However*, we think that the impressive performance of Yan et al. (2023) is not due to the *avoidance* of permutation equivariance mainly due to 1) architectural and diffusion process improvements, 2) effectively working in the setting of Bachmann et al. (2023) and are reintroducing a learned approximate permutation equivariance through their permutation augmentation[2]. There has been an exciting convergence in the literature, with works like Frasca et al. (2022) carefully tracking and experimenting with the specific symmetries we leverage as inductive biases and works like Bachmann et al. (2023); Ong & Veličković (2022) and indeed Yan et al. (2023) instead leverage compute combined with data or augmentations to have the models learn these symmetries. This is ultimately an example of the pareto frontier identified by Edelman et al. (2023), with symmetry inductive biases representing a form of knowledge in the absence of exhaustive data or enough compute to leverage augmentations.

However, as the community navigates this pareto frontier, we think it is important to both correctly disentangle what makes our methods work and also use the correct definition for "work".

Note that in the case of an imbalanced distribution of isomorphism classes as observed in many real world datasets in Ivanov et al. (2019), a "magic" PE method which can navigate the optimisation landscape will still find the correct distribution, while the non-PE methods might now suffer a bias due to overfitting on the training dataset - maximising recall, but not actually learning the task at (distributional modelling). Even in the benign unique isomorphism class setting implicitly assumed by Yan et al. (2023) large graphs is also highly unlikely that an augmentation will be able to generate a meaningful fraction of all permutations, although a possible concentration analysis in future work might alleviate this concern.

Until then, it is for this reason (a focus on correct generalisation without reliance on benign datasets) we chose to focus this work on PE architectures.

[2]based on inspecting their github codebase