HiGen: Hierarchical Graph Generative Networks

Mahdi Karami [1]

Abstract

Most real-world graphs exhibit a hierarchical structure, which is often overlooked by existing graph generation methods. In this work, we introduce HiGen, a **Hi**erarchical **G**raph G**en**erative Network to address the limitations of existing generative models by incorporating community structures and cross-level interactions. This approach involves generating graphs in a coarse-to-fine manner, where graph generation at each level is conditioned on a higher level (lower resolution) graph. The generation of communities at lower levels is performed in parallel, followed by the prediction of cross-edges between communities using a separate model. This parallelized approach enables high scalability. To capture hierarchical relations, our model allows each node at a given level to depend not only on its neighbouring nodes but also on its corresponding super-node at the higher level. Furthermore, we address the generation of integer-valued edge weights of the hierarchical structure by modeling the output distribution of edges using a multinomial distribution. We show that multinomial distribution can be factorized successively, enabling the autoregressive generation of each community. This property makes the proposed architecture well-suited for generating graphs with integer-valued edge weights. Furthermore, by breaking down the graph generation process into the generation of multiple small partitions that are conditionally independent of each other, HiGen reduces its sensitivity to a predefined initial ordering of nodes. Empirical studies demonstrate that the proposed generative model captures both local and global properties of graphs and achieves state-of-the-art performance in terms of graph quality on various benchmark graph datasets.

1. Background

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a collection of nodes (vertices) $\mathcal{V}$ and edges $\mathcal{E}$ with corresponding sizes $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ and an adjacency matrix $\mathbf{A}^{\pi}$ for the node ordering π. The node set can be partitioned into c communities (a.k.a. cluster or modules) using a graph partitioning function $\mathcal{F} : \mathcal{V} \rightarrow \{1, ..., c\}$, where each cluster of nodes forms a sub-graph denoted by $\mathcal{C}_i = (\mathcal{V}(\mathcal{C}_i), \mathcal{E}(\mathcal{C}_i))$ with adjacency matrix $\mathbf{A}_i$. The cross-links between neighboring communities form a *bipartite graph*, denoted by $\mathcal{B}_{ij} = (\mathcal{V}(\mathcal{C}_i), \mathcal{V}(\mathcal{C}_j), \mathcal{E}(\mathcal{B}_{ij}))$ with adjacency matrix $\mathbf{A}_{ij}$. Each community is aggregated to a super-node and each bipartite corresponds to a super-edge linking neighboring communities, which induces a coarser graph at the higher (a.k.a. parent) level. Herein, the levels are indexed by superscripts. Formally, each community at level l, $\mathcal{C}_i^l$, is mapped to a node at the higher level graph, also called its parent node, $v_i^{l-1} := Pa(\mathcal{C}_i^l)$ and each bipartite at level l is represented by an edge in the higher level, also called its parent edge, $e_i^{l-1} = Pa(\mathcal{B}_{ij}^l) = (v_i^{l-1}, v_j^{l-1})$. The weights of the self edges and the weights of the cross-edges in the parent level are determined by the sum of the weights of

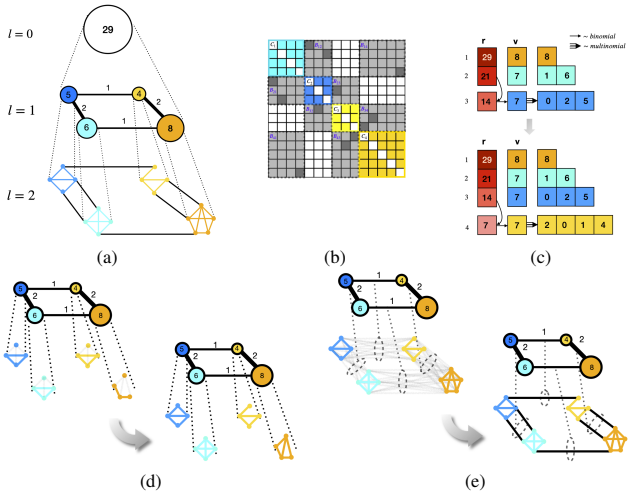

Figure 1: (a) A sample hierarchical graph with 2 levels is shown. Communities are shown in different colors and the weight of a node and the weight of an edge in a higher level, represent the sum of the edges in the corresponding community and bipartite, respectively. Node size and edge width indicate their weights. (b) The matrix shows corresponding adjacency of the graph $\mathcal{G}^2$ matrix where each of its sub-graphs corresponds to a block in the adjacency matrix, communities are shown in different colors and bipartites are colored in gray. (c) Decomposition of multinomial distribution as a recursive *stick-breaking* process where at each iteration, first a fraction of the remaining weights w_m is allocated to the m-th row (the m-th node in the sub-graph) and then this fraction v_m is distributed among that row of lower triangular adjacency matrix, $\hat{A}$. (d) Parallel generation of communities. (e) Parallel prediction of bipartites. Shadowed lines are the *augmented edges* representing candidate edges at each step.

the edges within their corresponding community and bipartite, respectively. Therefore, the edges in the induced graphs at the higher levels have integer-valued weights: $w_{ii}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{C}_i^l)} w_e$ and $w_{ij}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{B}_{ij}^l)} w_e$, moreover sum of all edge weights remains constant in all levels so $w_0 := \sum_{e \in \mathcal{E}(\mathcal{G}^l)} w_e = |\mathcal{E}|, \ \forall \, l \in [0, ..., L]$.

This clustering process continues recursively in a bottom-up approach until a single node graph $\mathcal{G}^0$ is obtained, producing a *hierarchical graph*, defined by the set of graphs in all levels of abstractions, $\mathcal{HG} := \{\mathcal{G}^0, ..., \mathcal{G}^{L-1}, \mathcal{G}^L\}$. This forms a dendrogram tree with $\mathcal{G}^0$ being the root and $\mathcal{G}^L$ being the final graph that is generated at the leaf level. An $\mathcal{HG}$ is visualized in figure 1a.

2. Hierarchical Graph Generation

In graph generative networks, the objective is to learn a generative model, $p(\mathcal{G})$ given a set of training graphs. Given a particular node ordering π, and a hierarchical graph $\mathcal{HG} := \{\mathcal{G}^0, ..., \mathcal{G}^{L-1}, \mathcal{G}^L\}$, produced by recursively applying a graph partitioning function, $\mathcal{F}$, we can factorize the generative model using the chain rule of probability as:

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

In other words, the generative process involves specifying the probability of the graph at each level conditioned on its parent level graph in the hierarchy.

Based on the partitioned structure within each level of $\mathcal{HG}$, and since the integer-valued weights of the edges in each level can be modeled by a multinomial distribution, the conditional generative probability $p(\mathcal{G}^l \mid \mathcal{G}^{l-1})$ can be decomposed into the conditional probability of its communities and bipartite graphs as:

$$\begin{aligned} p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) &= \prod p(\mathcal{C}_i^l \mid \mathcal{G}^{l-1}) \times \prod p(\mathcal{B}_{ij}^l \mid \mathcal{G}^{l-1}) \\ &\sim \prod_{i \, \in \, \mathcal{V}(\mathcal{G}^{l-1})} \text{Mu}([w_e]_{e \, \in \, \mathcal{C}_i^l} \mid w_{ii}^{l-1}, \boldsymbol{\theta}_{ii}^l) \times \\ &\quad \prod_{(i,j) \in \, \mathcal{E}(\mathcal{G}^{l-1})} \text{Mu}([w_e]_{e \, \in \, \mathcal{B}_{ij}^l} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij}^l) \end{aligned}$$

where $\{\boldsymbol{\theta}_{ij}^l[e] \in [0, 1], \text{ s.t. } \mathbf{1}^T \boldsymbol{\theta}_{ij}^l = 1 \mid \forall \ (i, j) \in \mathcal{E}(\mathcal{G}^{l-1})\}$ are the multinomial model parameters. Refer to Appendix A for the proof.

Therefore, given the parent graph at a higher level, the generation of graph at its subsequent level can be reduced to generation of its partition and bipartite sub-graphs. As illustrated in figure, this decomposition enables parallel generation of the communities in each level which can be followed by predicting all bipartite sub-graphs in that level at one pass. Each of these sub-graphs corresponds to a block in the adjacency matrix, as visualized in figure 1b, so the proposed hierarchical model generates adjacency matrix in a blocks-wise fashion and constructs the final graph topology.

2.1. Community Generation

As derived in theorem A.3 (refer to appendix A), the edge weights within each community can be jointly modeled using a multinomial distribution. Our objective is to model the generative probability of communities in each level as an autoregressive process. To accomplish this, we need to factorize the multinomial distribution accordingly. Inspired by GRAN (Liao et al., 2019), a community can be generated efficiently by generating one node at a time. This requires decomposing the generative probability of edges in a group-wise form, where the candidate edges between the t-th node and the already generated graph are grouped together. In other words, this model completes the lower triangle adjacency matrix one row at a time conditioned on the already generated sub-graph and the parent-level graph. The following theorem formally derives this decomposition for multinomial distributions.

Theorem 2.1. *For a random counting vector $\mathbf{w} \in \mathbb{Z}_+^E$ with a multinomial distribution $Mu(\mathbf{w} \mid w, \boldsymbol{\theta})$, let's split it into M disjoint groups $\mathbf{w} = [\mathbf{u}_1, \ ..., \mathbf{u}_M]$ where $\mathbf{u}_m \in \mathbb{Z}_+^{E_m}$, $\sum_{m=1}^M E_m = E$, and also split the probability vector accordingly as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ ..., \boldsymbol{\theta}_M]$. Additionally, let's define sum of all variables in the m-th group by a random count variable $\mathrm{v}_m := \sum_{e=1}^{E_m} \mathrm{u}_{m,e}$. Then, the multinomial distribution can be factorized as a chain of binomial and multinomial distributions:*

$$\begin{aligned} &Mu(\mathbf{w} = [\mathbf{u}_1, \ ..., \mathbf{u}_M] \mid w, \boldsymbol{\theta} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M]) \\ &= \prod_{m=1}^{M} Bi(\mathrm{v}_m \mid w - \sum_{i<m} \mathrm{v}_i, \eta_{\mathrm{v}_m}) \, Mu(\mathbf{u}_m \mid \mathrm{v}_m, \boldsymbol{\lambda}_m), \\ &\textit{where: } \eta_{\mathrm{v}_m} = \frac{\mathbf{1}^T \boldsymbol{\theta}_m}{1 - \sum_{i<m} \mathbf{1}^T \boldsymbol{\theta}_i}, \ \ \boldsymbol{\lambda}_m = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T \boldsymbol{\theta}_m}. \end{aligned} \quad (1)$$

Here, the probability of binomial, η_{v_m}, is the fraction of the remaining probability mass that is allocated to v_m, i.e. *the sum of all weights in the m-th group. The vector parameter $\boldsymbol{\lambda}_m$ is the normalized multinomial probabilities of all count variables in the m-th group. Intuitively, this decomposition of multinomial distribution can be viewed as a recursive stick-breaking process where at each step, first a binomial distribution is used to determine how much probability mass to allocate to the current group, and a multinomial distribution is used to distribute that probability mass among the variables in the group. The resulting distribution is equivalent to the original multinomial distribution.*

Proof. Refer to appendix A.1 for the proof. □

Let $\hat{\mathcal{C}}^l_{i,t}$ denote an already generated sub-graph, at the t-th step, augmented with the set of candidate edges, from the new node, $v_t(\mathcal{C}^l_i)$, to its preceding node denoted by $\hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t}) := \{(t,j) \mid j < t\}$. We collect the weights of these edges in the random vector $\mathbf{u}_t := [w_e]_{e \in \hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t})}$ (that is the t-th row of the lower triangle of adjacency matrix $\hat{\mathbf{A}}^l_i$), where the sum of the candidate edge weights is v_t. Based on theorem 2.1, the probability of $\mathbf{u}_t$ can be characterized by the product of a binomial and a multinomial distribution. This process is illustrated in figure 1c. We further increase the expressiveness of the generative network by extending this probability to a mixture model with K mixtures:

$$p(\mathbf{u}_t) = \sum_{k=1}^{K} \boldsymbol{\beta}^l_k \mathrm{Bi}(\mathrm{v}_t | w^{l-1}_{ii} - \sum_{i<t} \mathrm{v}_i, \eta^l_{t,k}) \mathrm{Mu}(\mathbf{u}_t \mid \mathrm{v}_t, \boldsymbol{\lambda}^l_{t,k}) \tag{2}$$

$$\boldsymbol{\lambda}^l_{t,k} = \mathrm{softmax}\Big(\mathrm{MLP}^l_{\boldsymbol{\theta}}\big(\Big[\Delta \boldsymbol{h}_{\hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t})} \,||\, h_{Pa(\mathcal{C}^l_i)}\Big]\big)\Big)[k,:]$$

$$\eta^l_{t,k} = \mathrm{sigmoid}\Big(\mathrm{MLP}^l_{\eta}\big(\Big[\mathrm{pool}(\boldsymbol{h}_{\hat{\mathcal{C}}^l_{i,t}}) \,||\, h_{Pa(\mathcal{C}^l_i)}\Big]\big)\Big)[k]$$

$$\boldsymbol{\beta}^l = \mathrm{softmax}\Big(\mathrm{MLP}^l_{\beta}\big(\Big[\mathrm{pool}(\boldsymbol{h}_{\hat{\mathcal{C}}^l_{i,t}}) \,||\, h_{Pa(\mathcal{C}^l_i)}\Big]\big)\Big)$$

Where $\Delta \boldsymbol{h}_{\hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t})}$ is a $|\hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t})| \times d_h$ dimensional matrix, consisting of the set of edge features $\{\Delta h_{(t,s)} := h_t - h_s \mid \forall\ (t,s) \in \hat{\mathcal{E}}_t(\hat{\mathcal{C}}^l_{i,t})\}$, $\boldsymbol{h}_{\hat{\mathcal{C}}^l_{i,t}}$ is a $t \times d_h$ matrix of node features in the augmented community graph. The mixture weights are denoted by $\boldsymbol{\beta}^l$. Here, the node features are learned by GNN models and the graph level representation is obtained by the $\mathrm{addpool}()$ aggregation function. In order to produce $K \times |\mathcal{E}_t(\mathcal{C}^l_i)|$ dimensional matrix of multinomial probabilities, the $\mathrm{MLP}^l_{\boldsymbol{\theta}}()$ network acts at the edge level, while $\mathrm{MLP}^l_{\eta^{\mathrm{v}}}()$ and $\mathrm{MLP}^l_{\beta}()$ act at the graph level to produce the binomial probabilities and K dimensional arrays for K mixture models, respectively. All of these MLP networks are built by two hidden layers with $\mathrm{ReLU}()$ activation functions.

During the generation process of each community $\mathcal{C}^l_i$, the node features of its parent node $h_{Pa(\mathcal{C}^l_i)}$ are used as the context. This context is concatenated to the node and edge feature matrices using the operation $\big[\boldsymbol{x} \,||\, y\big]$, which concatenates vector y to each row of matrix $\boldsymbol{x}$. The purpose of this context is to enrich the node and edge features by capturing long-range interactions and encoding the global structure of the graph, which is important for generating local components.

2.2. Bipartite Generation

Once all the communities in level l are generated, the edges of all bipartite graphs at that level can be predicted simultaneously. An augmented graph $\hat{\mathcal{G}}^l$ composed of all the communities, $\{\mathcal{C}^l_i \ \forall i \in \mathcal{V}(\mathcal{G}^{l-1})\}$, and the candidate edges of all bipartites, $\{\mathcal{B}^l_{ij} \ \forall (i,j) \in \mathcal{E}(\mathcal{G}^{l-1})\}$, is used as the input of a GNN to obtain node and edge features. We similarly extend the multinomial distribution of a bipartite, in (7), using a mixture model to express its generative probability:

$$p(\mathbf{w} := \hat{\mathcal{E}}(\mathcal{B}^l_{ij})) = \sum_{k=1}^{K} \boldsymbol{\beta}^l_k \mathrm{Mu}(\mathbf{w} \mid w^{l-1}_{ij}, \boldsymbol{\theta}^l_{ij,k})$$

$$\boldsymbol{\theta}^l_{ij,k} = \mathrm{softmax}\Big(\mathrm{MLP}^l_{\boldsymbol{\theta}}\big(\Big[\Delta \boldsymbol{h}_{\hat{\mathcal{E}}(\mathcal{B}^l_{ij})} \,||\, \Delta h_{Pa(\mathcal{B}^l_{ij})}\Big]\big)\Big)[k,:]$$

$$\boldsymbol{\beta}^l = \mathrm{softmax}\Big(\mathrm{MLP}^l_{\beta}\big(\Big[\mathrm{pool}(\Delta \boldsymbol{h}_{\hat{\mathcal{E}}(\mathcal{B}^l_{ij})}) \,||\, \Delta h_{Pa(\mathcal{B}^l_{ij})}\Big]\big)\Big)$$

where the random vector $\mathbf{w} := [w_e]_{e \in \hat{\mathcal{E}}(\mathcal{B}^l_{ij})}$ is the set of weights of all candidate edges in bipartite $\mathcal{B}^l_{ij}$ and $\Delta \boldsymbol{h}_{Pa(\mathcal{B}^l_{ij})}$ are the parent edge features of the bipartite.

Node Feature Encoding: To encode node features, we extend GraphGPS proposed by Rampášek et al. (2022). GraphGPS combines local message-passing with global attention mechanism and uses positional and structural encoding for nodes and edges to construct a more expressive and a scalable graph transformer (GT) (Dwivedi & Bresson, 2020). To apply GraphGPS on augmented graphs, we use distinct initial edge features to distinguish augmented (candidate) edges from real edges. Furthermore, for bipartite generation, the attention scores in the Transformers of the augmented graph $\hat{\mathcal{G}}^l$ are masked to restrict attention only to connected communities. The details of model architecture are provided in appendix C.

3. Experiments

In our empirical studies, we compare the proposed hierarchical graph generative network against state-of-the-art autoregressive models: GRAN and GraphRNN models, diffusion models: DiGress (Vignac et al., 2022) and GDSS (Jo et al., 2022) and a GAN-based model: SPECTRE (Martinkus et al., 2022), on a range of synthetics and real datasets of various sizes.

Datasets: We used 4 different benchmark graph datasets: (1) the synthetic *Stochastic Block Model (SBM)* dataset consisting of 200 graphs with 2-5 communities each with 20-40 nodes, used in a previous work (Martinkus et al., 2022); (2) the *Protein* including 918 protein graphs, each has 100 to 500 nodes representing amino acids that are linked if they are closer than 6 Angstroms (Dobson & Doig, 2003), (3) the *Enzyme* that has 587 protein graphs of 10-125 nodes, representing protein tertiary structures of the enzymes from the BRENDA database (Schomburg et al., 2004) and (4) the *Ego* dataset containing 757 3-hop ego networks with 50-300 nodes extracted from the CiteSeer dataset, where nodes represent documents and edges represent citation relationships (Sen et al., 2008).

Table 1: Comparison of generation metrics on benchmark datasets. The baseline results for SBM and Protein graphs are obtained from (Martinkus et al., 2022; Vignac et al., 2022), and the results for enzyme graphs (except for GRAN, which we implemented) are obtained from (Jo et al., 2022), while we implemented them for Ego. "-": not applicable due to resource issue or not reported in the reference papers. **Metrics:** We follow (Liao et al., 2019) and compute maximum mean discrepancy (MMD) of four different graph statistics between the ground truth and generated graph. Here we adopt total variation (TV) distance as the kernel for computing evaluation metrics except for the enzyme dataset where we use a Gaussian EMD kernel to be consistent with the results reported in (Jo et al., 2022). Refer to Appendix D for detailed explanation.

	Stochastic block model				*Protein*			
Model	Deg. ↓	Clus. ↓	Orbit↓	Spec. ↓	Deg. ↓	Clus. ↓	Orbit↓	Spec. ↓
Training set	0.0008	0.0332	0.0255	0.0063	0.0003	0.0068	0.0032	0.0009
GraphRNN	0.0055	0.0584	0.0785	0.0065	0.0040	0.1475	0.5851	0.0152
GRAN	0.0113	0.0553	0.0540	0.0054	0.0479	0.1234	0.3458	0.0125
SPECTRE	0.0015	0.0521	0.0412	0.0056	0.0056	0.0843	0.0267	0.0052
DiGress	**0.0013**	**0.0498**	0.0433	-	-	-	-	-
HiGen-m	0.0017	0.0503	0.0604	0.0068	0.0041	0.109	0.0472	0.0061
HiGen	0.0019	**0.0498**	**0.0352**	**0.0046**	**0.0012**	**0.0435**	**0.0234**	**0.0025**

	Enzyme			*Ego*			
Model	Deg. ↓	Clus. ↓	Orbit ↓	Deg. ↓	Clus. ↓	Orbit ↓	Spec. ↓
Training set	0.0011	0.0025	3.7e-4	2.2e-4	0.010	0.012	1.4e-3
GraphRNN	0.017	0.062	0.046	0.024	0.34	0.14	0.089
GRAN	0.054	0.087	0.033	0.032	0.17	0.026	0.046
GDSS	0.026	0.061	0.009	-	-	-	-
HiGen-m	0.027	0.157	1.2e-3	0.011	0.063	**0.021**	0.013
HiGen	**0.012**	**0.038**	**7.2**e-4	**1.9**e-3	**0.049**	0.029	**0.004**

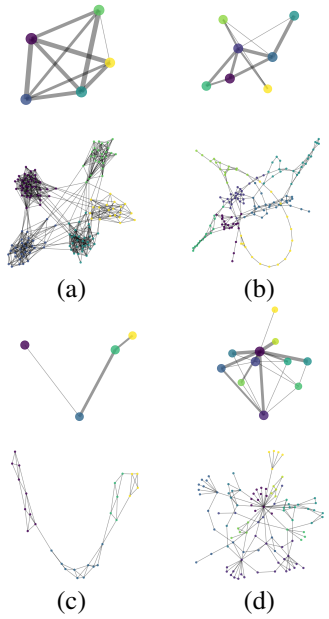

Figure 2: Samples from HiGen. a) SBM, b) Protein, c) Enzyme and d) Ego. Communities are distinguished with different colors and both levels are depicted.

Graph Partitioning Different algorithms approach the problem of graph partitioning (clustering) using various clustering quality functions. For example, the Louvain algorithm (Blondel et al., 2008) starts with each node as its community and then repeatedly merges communities based on the highest increase in modularity until no further improvement can be made. This heuristic algorithm is computationally efficient and scalable to large graphs for community detection. As the modularity metric is based on the graph structure, it is well-suited for our problem. Therefore, we employed the Louvain algorithm to hierarchically cluster the graph datasets in our experiments and then spliced out the intermediate levels to achieve $\mathcal{HG}$s with uniform depth of $L = 2$.

Model Architecture In the experiments, the GNN models consist of 8 layers of GraphGPS layers (Rampášek et al., 2022). The input node feature of GNNs is augmented with positional and structural encoding, where the first 8 eigenvectors corresponding to the smallest non-zero eigenvalues of the Laplacian and diagonal of the random-walk matrix up to 8-steps are used. Each level has its own GNN and output models. The details of the model architecture are presented in Appendix C and D.

We conducted experiments using the proposed hierarchical graph generative network (HiGen) model with two variants for the output distribution of the leaf edges: 1) **HiGen**: the probability of the community edges' weights at the leaf level are modeled by mixture of Bernoulli, using $\mathrm{sigmoid}()$ activation in equation 2, since the leaf levels in our experiments have binary edges weights, while higher levels use mixture of multinomials. 2)**HiGen-m**: the model uses a mixture of multinomial distributions (2) to describe the output distribution for all levels. In this case, we observed that modeling the probability parameters of edge weights of the leaf level, denoted as $\boldsymbol{\lambda}_{t,k}$ in (2), by a multi-hot activation function, defined as $\sigma(\mathbf{z})_i := {}^{\mathrm{sigmoid}(z_i)}/_{\sum_{j=1}^{K}\mathrm{sigmoid}(z_j)}$ where $\sigma : \mathbb{R}^K \to (K-1)$-simplex, provided slightly better performance than the standard $\mathrm{softmax}()$ function. However, for both HiGen and HiGen-m, the probabilities of the integer-valued edges at the higher levels are still modeled by the standard $\mathrm{softmax}()$ function.

For training, HiGen models used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 5e-4 and its default settings of $\beta_1 = 0.9,\ \beta_2 = 0.999$ and ϵ=1e-8.

The performance metrics of the proposed HiGen models are reported in Table 1, with generated graph samples presented in Figure 2. The results demonstrate that HiGen effectively captures graph statistics and achieves state-of-the-art on all the benchmarks graphs across various generation metrics. This improvement in both local and global properties of the generated graphs highlights the effectiveness of the hierarchical graph generation approach, which models communities and cross-community interactions separately. Graph samples generated by the HiGen models, as well as the experimental evaluation of different node ordering and partitioning functions, are presented in Appendix E.3.

References

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Xiaohui Chen, Xu Han, Jiajing Hu, Francisco JR Ruiz, and Liping Liu. Order matters: Probabilistic modeling of node sequence for graph generation. *arXiv preprint arXiv:2106.06189*, 2021.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

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.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

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.

Scott Linderman, Matthew J Johnson, and Ryan P Adams. Dependent multinomial models made easy: Stick-breaking with the pólya-gamma augmentation. *Advances in Neural Information Processing Systems*, 28, 2015.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows, 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.

Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In *International Workshop on Mining and Learning with Graphs at KDD*, 2013.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. *arXiv preprint arXiv:2106.01098*, 2021.

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, 2022.

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.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Kyle Siegrist. *Probability, Mathematical Statistics, Stochastic Processes*. LibreTexts, 2017. URL https://stats.libretexts.org/Bookshelves/Probability_Theory/Probability_Mathematical_Statistics_and_Stochastic_Processes_(Siegrist).

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. *arXiv preprint arXiv:2201.09871*, 2022.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, pp. 5694–5703, 2018.

A. Probability Distribution of Communities and Bipartites

Theorem A.1. *Given a graph $\mathcal{G}$ and an ordering π, assuming there is a deterministic function that provides the corresponding high-level graphs in a hierarchical order as $\{\mathcal{G}^L, \mathcal{G}^{L-1}, ..., \mathcal{G}^0\}$, then:*

$$p(\mathcal{G} = \mathcal{G}^L, \pi) = p(\{\mathcal{G}^L, \mathcal{G}^{L-1}, ..., \mathcal{G}^0\}, \pi) = p(\mathcal{G}^L, \pi \mid \{\mathcal{G}^{L-1}, ..., \mathcal{G}^0\}) \; ... \; p(\mathcal{G}^1, \pi \mid \mathcal{G}^0) \; p(\mathcal{G}^0)$$
$$= \prod_{l=0}^{L} p(\mathcal{G}^l, \pi \mid \mathcal{G}^{l-1}) \times p(\mathcal{G}^0) \tag{3}$$

Proof. The factorization is derived by applying the chain rule of probability and last equality holds as the graphs at the coarser levels are produced by a partitioning function acting on the finer level graphs. Overall, this hierarchical generative model exhibits a Markovian structure. □

Lemma A.2. *Given the sum of counting variables in the groups, the groups are independent and each of them has multinomial distribution:*

$$p(\mathbf{w} = [\mathbf{u}_1, \; ..., \mathbf{u}_M] | \{\mathrm{v}_1, ..., \mathrm{v}_M\}) = \prod_{m=1}^{M} \mathit{Mu}(\mathrm{v}_m, \; \boldsymbol{\lambda}_m)$$
$$\textit{where: } \boldsymbol{\lambda}_m = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T \, \boldsymbol{\theta}_m}$$

Here, probability vector (parameter) $\boldsymbol{\lambda}_m$ is the normalized multinomial probabilities of the counting variables in the m-th group.

Proof.

$$\begin{aligned}
p(\mathbf{w} | \{\mathrm{v}_1, ..., \mathrm{v}_M\}) &= \frac{p(\mathbf{w})}{p(\{\mathrm{v}_1, ..., \mathrm{v}_M\})} I(\mathrm{v}_1 = \mathbf{1}^T \, \mathbf{u}_1, \; ..., \mathrm{v}_M = \mathbf{1}^T \, \mathbf{u}_M) \\
&= \frac{\frac{w!}{\prod_{i=1}^{E} \mathrm{w}_i!} \prod_{i=1}^{E} {\boldsymbol{\theta}_i}^{\mathrm{w}_i}}{\frac{w!}{\prod_{i=1}^{M} \mathrm{v}_i!} \prod_{i=1}^{M} {\alpha_i}^{\mathrm{v}_i}} I(\mathrm{v}_1 = \mathbf{1}^T \, \mathbf{u}_1, \; ..., \mathrm{v}_M = \mathbf{1}^T \, \mathbf{u}_M) \\
&= \frac{\frac{w!}{\prod_{i=1}^{E} \mathrm{w}_i!} \boldsymbol{\theta}_1^{\mathrm{w}_1} ... \boldsymbol{\theta}_E^{\mathrm{w}_E}}{\frac{w!}{\prod_{i=1}^{M} \mathrm{v}_i!} (\mathbf{1}^T \, \boldsymbol{\theta}_1)^{\mathrm{v}_1} ... (\mathbf{1}^T \, \boldsymbol{\theta}_M)^{\mathrm{v}_M}} \\
&= \frac{\mathrm{v}_1!}{\prod_{i=1}^{E_1} \mathbf{u}_{1,i}!} \prod_{i=1}^{E_1} {\boldsymbol{\lambda}_{1,i}}^{\mathbf{u}_{1,i}} \times ... \times \frac{\mathrm{v}_M!}{\prod_{i=1}^{E_M} \mathbf{u}_{M,i}!} \prod_{i=1}^{E_1} {\boldsymbol{\lambda}_{M,i}}^{\mathbf{u}_{M,i}} \\
&= \mathrm{Mu}(\mathrm{v}_1, \; \boldsymbol{\lambda}_1) \times ... \times \mathrm{Mu}(\mathrm{v}_M, \; \boldsymbol{\lambda}_M)
\end{aligned}$$

□

Theorem A.3. *Let the random vector $\mathbf{w} := [w_e]_{e \, \in \, \mathcal{E}(\mathcal{G}^l)}$ denote the set of weights of all edges of $\mathcal{G}^l$ such that their sum is $w_0 = \mathbf{1}^T \, \mathbf{w}$. The joint probability of $\mathbf{w}$ can be described by a multinomial distribution: $\mathbf{w} \sim \mathit{Mu}(\mathbf{w} \mid w_0, \boldsymbol{\theta}^l)$. By observing that the sum of edge weights within each community $\mathcal{C}_i^l$ and bipartite graph $\mathcal{B}_{ij}^l$ are determined by the weights of their parent edges in the higher level, w_{ii}^{l-1} and w_{ij}^{l-1} respectively, we can establish that these components are conditionally independent and each of them follow a multinomial distribution:*

$$p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) \sim \prod_{i \, \in \, \mathcal{V}(\mathcal{G}^{l-1})} \mathit{Mu}([w_e]_{e \, \in \, \mathcal{C}_i^l} \mid w_{ii}^{l-1}, \boldsymbol{\theta}_{ii}^l) \times \prod_{(i,j) \in \, \mathcal{E}(\mathcal{G}^{l-1})} \mathit{Mu}([w_e]_{e \, \in \, \mathcal{B}_{ij}^l} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij}^l) \tag{4}$$

where $\{\boldsymbol{\theta}_{ij}^l[e] \in [0, 1], \; s.t. \; \mathbf{1}^T \boldsymbol{\theta}_{ij}^l = 1 \mid \forall \; (i, j) \in \; \mathcal{E}(\mathcal{G}^{l-1})\}$ are the multinomial model parameters.

Proof. In a hierarchical graph, the edges has non-negative integer valued weights while the sum of all the edges in community $\mathcal{C}_i^l$ and bipartite graph $\mathcal{B}_{ij}^l$ are determined by their corresponding edges in the parent graph, *i.e.* w_{ii}^{l-1} and w_{ij}^{l-1} respectively. Let the random vector $\mathbf{w} := [w_e]_{e \in \mathcal{E}(\mathcal{G}^l)}$ denote the set of weights of all edges of $\mathcal{G}^l$ such that $w_0 = \mathbf{1}^T \mathbf{w}$, its joint probability can be described as a multinomial distribution:

$$\mathbf{w} \sim \text{Mu}(\mathbf{w} \mid w_0, \boldsymbol{\theta}^l) = \frac{w_0!}{\prod_{e=1}^{|\mathcal{E}(\mathcal{G}^l)|} \mathbf{w}[e]!} \prod_{e=1}^{|\mathcal{E}(\mathcal{G}^l)|} (\boldsymbol{\theta}^l[e])^{\mathbf{w}[e]}, \tag{5}$$

where $\{\boldsymbol{\theta}^l[e] \in [0,1], \text{ s.t. } \mathbf{1}^T \boldsymbol{\theta}^l = 1\}$ are the parameters of the multinomial distribution.[1] Therefore, based on lemma A.2 these components are conditionally independent and each of them has a multinomial distribution:

$$p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) \sim \prod_{i \in \mathcal{V}(\mathcal{G}^{l-1})} \text{Mu}([w_e]_{e \in \mathcal{C}_i^l} \mid w_{ii}^{l-1}, \boldsymbol{\theta}_{ii}^l) \times \prod_{(i,j) \in \mathcal{E}(\mathcal{G}^{l-1})} \text{Mu}([w_e]_{e \in \mathcal{B}_{ij}^l} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij}^l)$$

where $\{\boldsymbol{\theta}_{ij}^l[e] \in [0,1], \text{ s.t. } \mathbf{1}^T \boldsymbol{\theta}_{ij}^l = 1 \mid \forall\, (i,j) \in \mathcal{E}(\mathcal{G}^{l-1})\}$ are the parameters of the model.

Therefore, the log-likelihood of $\mathcal{G}^l$ can be decomposed as the log-likelihood of its sub-structures:

$$\log p_{\phi^l}(\mathcal{G}^l \mid \mathcal{G}^{l-1}) = \sum_{i \in \mathcal{V}_{\mathcal{G}^{l-1}}} \log p_{\phi^l}(\mathcal{C}_i^l \mid \mathcal{G}^{l-1}) + \sum_{(i,j) \in \mathcal{E}_{\mathcal{G}^{l-1}}} \log p_{\phi^l}(\mathcal{B}_{ij}^l \mid \mathcal{G}^{l-1}) \tag{6}$$

□

Bipartite distribution: Let's denote the set of weights of all candidate edges of the bipartite $\mathcal{B}_{ij}^l$ by a random vector $\mathbf{w} := [w_e]_{e \in \mathcal{E}(\mathcal{B}_{ij}^l)}$, its probability can be described as

$$\mathbf{w} \sim \text{Mu}(\mathbf{w} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij}^l) = \frac{w_{ij}^{l-1}!}{\prod_{e=1}^{|\mathcal{E}(\mathcal{B}_{ij}^l)|} \mathbf{w}[e]!} \prod_{e=1}^{|\mathcal{E}(\mathcal{B}_{ij}^l)|} (\boldsymbol{\theta}_{ij}^l[e])^{\mathbf{w}[e]} \tag{7}$$

where $\{\boldsymbol{\theta}_{ij}^l[e] \mid \boldsymbol{\theta}_{ij}^l[e] \geq 0,\ \sum \boldsymbol{\theta}_{ij}^l[e] = 1\}$ are the parameter of the distribution, and the multinomial coefficient $\frac{n!}{\prod \mathbf{w}[e]!}$ is the number of ways to distribute the total weight $w_{ij}^{l-1} = \sum_{e=1}^{|\mathcal{E}(\mathcal{B}_{ij}^l)|} \mathbf{w}[e]$ into all candidate edges of $\mathcal{B}_{ij}^l$.

Community distribution: Similarly, the probability distribution of the set of candidate edges for each community can be modeled jointly by a multinomial distribution but as our objective is to model the generative probability of communities in each level as an autoregressive process we are interested to decomposed this probability distribution accordingly.

Lemma A.4. *A random counting vector $\mathbf{w} \in \mathbb{Z}_+^E$ with a multinomial distribution can be recursively decomposed into a sequence of binomial distributions as follows:*

$$\textit{Mu}(\mathrm{w}_1,\ ...,\mathrm{w}_E \mid w,\ [\theta_1,\ ...,\theta_E]) = \prod_{e=1}^{E} \textit{Bi}(\mathrm{w}_e \mid w - \textstyle\sum_{i<e} \mathrm{w}_i, \hat{\theta}_e), \tag{8}$$

$$\textit{where: } \hat{\theta}_e = \frac{\theta_e}{1 - \sum_{i<e} \theta_i}$$

This decomposition is known as a stick-breaking process, where $\hat{\theta}_e$ is the fraction of the remaining probabilities we take away every time and allocate to the e-th component (Linderman et al., 2015).

This lemma enable us to model the generation of a community as an edge-by-edge autoregressive process, similar to existing algorithms such as GraphRNN (You et al., 2018) or DeepGMG (Li et al., 2018) with $\mathcal{O}(|\mathcal{V}_{\mathcal{C}}|^2)$ generation steps.

[1] It is analogous to the random trial of putting n balls into k boxes, where the joint probability of the number of balls in all the boxes follows the multinomial distribution.

A.1. Proof of Theorem 2.1

For a random counting vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution $\text{Mu}(w, \boldsymbol{\theta})$, let's split it into M disjoint groups $\mathbf{w} = [\mathbf{u}_1, \ ..., \mathbf{u}_M]$ where $\mathbf{u}_m \in \mathbb{Z}_+^{E_m}$, $\sum_{m=1}^M E_m = E$, and also split the probability vector as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ ..., \boldsymbol{\theta}_M]$. Additionally, let's define sum of all weights in m-th group by a random variable $\text{v}_m := \sum_{e=1}^{E_m} \text{u}_{m,e}$.

Lemma A.5. *Sum of the weights in the groups, $\mathbf{u}_m \in \mathbb{Z}_+^{E_m}$, $\sum_{m=1}^M E_m = E$ has multinomial distribution:*

$$p(\{\text{v}_1, ..., \text{v}_M\}) = \textit{Mu}(w, \ [\alpha_1, ..., \alpha_M])$$
$$\textit{where: } \alpha_m = \sum \boldsymbol{\theta}_m[i]. \tag{9}$$

In the other words, the multinomial distribution is preserved when its counting variables are combined (Siegrist, 2017).

Theorem A.6. *Given the aforementioned grouping of counts variables, the multinomial distribution can be modeled as a chain of binomials and multinomials:*

$$\textit{Mu}(w, \boldsymbol{\theta} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M]) = \prod_{m=1}^{M} \textit{Bi}(w - \sum_{i<m} \text{v}_i, \ \eta_{\text{v}_m}) \, \textit{Mu}(\text{v}_m, \ \boldsymbol{\lambda}_m), \tag{10}$$
$$\textit{where: } \eta_{\text{v}_m} = \frac{\mathbf{1}^T \boldsymbol{\theta}_m}{1 - \sum_{i<m} \mathbf{1}^T \boldsymbol{\theta}_i}, \ \ \boldsymbol{\lambda}_m = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T \boldsymbol{\theta}_m}$$

Proof. Since sum of the weights of the groups, v_m, are functions of the weights in the group:

$$p(\mathbf{w}) = p(\mathbf{w}, \{\text{v}_1, ..., \text{v}_M\}) = p(\mathbf{w}|\{\text{v}_1, ..., \text{v}_M\}) p(\{\text{v}_1, ..., \text{v}_M\})$$

According to lemma A.5, sum of the weights of the groups is a multinomial and by lemma A.4, it can be decomposed to a sequence of binomials:

$$p(\{\text{v}_1, ..., \text{v}_M\}) = \text{Mu}(w, \ [\alpha_1, ..., \alpha_M]) = \prod_{m=1}^{M} \text{Bi}(w - \sum\nolimits_{i<m} \text{v}_i, \hat{\eta}_m),$$
$$\text{where: } \alpha_m = \mathbf{1}^T \boldsymbol{\theta}_m, \ \hat{\eta}_e = \frac{\alpha_e}{1 - \sum_{i<e} \alpha_m}$$

Also based on lemma A.2, given the sum of the wights of all groups, the groups are independent and has multinomial distribution:

$$p(\mathbf{w}|\{\text{v}_1, ..., \text{v}_M\}) = \prod_{m=1}^{M} \text{Mu}(\text{v}_m, \ \boldsymbol{\lambda}_m)$$
$$\text{where: } \boldsymbol{\lambda}_m = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T \boldsymbol{\theta}_m}$$

□

B. Discussion

The GRAN model can generate graphs one block of nodes at a time in an autoregressive fashion where the block size is fixed and nodes are assigned to blocks based on an ordering. However, the model's performance deteriorates as the block size increases, since adjacent nodes in an ordering may not be relevant and may belong to different clusters. Additionally, intra-block connections are not modeled separately. In contrast, our proposed method generates blocks of nodes within each community that have strong relationships and then predicts the cross-links between communities using a separate model. As a result, this approach enables the model to capture both local relationships between nodes within a community and global relationships across communities, resulting in improved expressiveness of the graph generative model.

The proposed hierarchical model allows for highly parallelizable training and generation. Specifically, let n_c be the size of the largest graph cluster, then, it only requires $\mathcal{O}(n_c \log n)$ sequential steps to generate a graph of size n.

Node ordering sensitivity The predefined ordering of dimensions can be crucial for training autoregressive (AR) models (Vinyals et al., 2015), and this sensitivity to node orderings is particularly pronounced in autoregressive graph generative model (Liao et al., 2019; Chen et al., 2021). However, in the proposed approach, the graph generation process is divided into the generation of multiple small partitions, performed sequentially across the levels, rather than generating the entire graph by a single AR model. Therefore, given an ordering for the parent level, the graph generation depends only on the permutation of the nodes within the graph communities rather than the node ordering of the entire graph. In other words, the proposed method is invariant to a large portion of possible node permutations, and therefore the set of distinctive adjacency matrices is much smaller in HiGen. For example, the node ordering $\pi_1 = [v_1, v_2, v_3, v_4]$ with clusters $\mathcal{V}_{\mathcal{G}_1} = \{v_1, v_2\}$ and $\mathcal{V}_{\mathcal{G}_2} = \{v_3, v_4\}$ has a similar hierarchical graph as $\pi_2 = [v_1, v_3, v_2, v_4]$, since the node ordering within the communities is preserved at all levels. Formally, let $\{\mathcal{C}_i^l \ \forall i \in \mathcal{V}_{\mathcal{G}^{l-1}}\}$ be the set of communities at level l produced by a deterministic partitioning function, where $n_i^l = |\mathcal{V}(\mathcal{C}_i^l)|$ denotes the size of each partition. The upper bound on the number of distinct node orderings in an HG generated by the proposed process is then reduced to $\prod_{l=1}^{L} \prod_i n_i^l!$. [2]

C. Graph Neural Network (GNN) architectures

To overcome limitations in the sparse message passing mechanism, Graph Transformers (GTs) (Dwivedi & Bresson, 2020) have emerged as a recent solution. One key advantage of GTs is the ability for nodes to attend to all other nodes in a graph, known as global attention, which addresses issues such as over-smoothing, over-squashing, and expressiveness bounds Rampášek et al. (2022). GraphGPS provide a recipe for creating a more expressive and scalable graph transformer by making a hybrid message-passing graph neural networks (MPNN)+Transformer architecture. Additionally, recent GNN models propose to address the limitation of standard MPNNs in detecting simple substructures by adding features that they cannot capture on their own, such as the number of cycles. A framework for selecting and categorizing different types of positional and structural encodings, including *local, global, and relative* is provided in (Rampášek et al., 2022). Positional encodings, such as eigenvectors of the adjacency or Laplacian matrices, aim to indicate the spatial position of a node within a graph, so nodes that are close to each other within a graph or subgraph should have similar positional encodings. On the other hand, structural encodings, such as degree of a node, number of k-cycles a node belong to or the diagonal of the m-steps random-walk matrix, aim to represent the structure of graphs or subgraphs, so nodes that share similar subgraphs or similar graphs should have similar structural encodings.

In order to encode the node features of the augmented graphs in our model, we customized GraphGPS in various ways. We incorporated distinct initial edge features to distinguish augmented (candidate) edges from real edges. Furthermore, for bipartite generation, we apply a mask on the attention scores of the transformers of the augmented graph $\hat{\mathcal{G}}^l$ to restrict attention only to connected communities. Specifically, the i-th row of the attention mask matrix is equal to 1 only for the index of the nodes that belong to the same community or the nodes of the neighboring communities that are linked by a bipartite, and 0 (i.e., no attention to those positions) otherwise.

The time and memory complexity of GraphGPS can be reduced to $\mathcal{O}(n + m)$ per layer by using *linear Transformers* such as Performer (Choromanski et al., 2020) for global graph attention, while they can be as high quadratic in the number of nodes if the original Transformer architecture is employed. Since our datasets are composed of graphs smaller than 1000 nodes, we leverage the original Transformer architecture.

D. Experimental details

Metrics To evaluate the graph generative models, we adopt the approach proposed in (Liu et al., 2019; Liao et al., 2019), which compares the distributions of four different graph statistics between the ground truth and generated graphs: (1) degree distributions, (2) clustering coefficient distributions, (3) the number of occurrences of all orbits with four nodes, and (4) the spectra of the graphs by computing the eigenvalues of the normalized graph Laplacian. The first three metrics capture local graph statistics, while the spectra represents global structure. The maximum mean discrepancy (MMD) score over these statistics are used as the metrics. While Liu et al. (2019) computed MMD scores using the computationally expensive Gaussian earth mover's distance (EMD) kernel, Liao et al. (2019) proposed using the total variation (TV) distance as an alternative measure. TV distance is much faster and still consistent with the Gaussian EMD kernel. Most recently,

[2] It is worth noting that all node permutations do not result in distinctive adjacency matrices due to the automorphism property of graphs (Liao et al., 2019; Chen et al., 2021). Therefore, the number of node permutations provides an upper bound rather than an exact count.

O'Bray et al. (2021) suggested using other efficient kernels such as an RBF kernel, or a Laplacian kernel, or a linear kernel. Additionally, Thompson et al. (2022) proposed new evaluation metrics for comparing graph sets using a random-GNN approach where GNNs are employed to extract meaningful graph features. However, in this work, we follow the experimental setup and evaluation metrics of (Liao et al., 2019), except for the enzyme dataset where we use a Gaussian EMD kernel to be consistent with the results reported in (Jo et al., 2022). GNN-based performance metrics of HiGen model are also reported in appendix E.3.

Datasets: For the benchmark datasetst, graph sizes, denoted as $D_{dataset} = (|\mathcal{V}|_{max}, |\mathcal{V}|_{avg}, |\mathcal{E}|_{max}, |\mathcal{E}|_{avg})$, are: $D_{protein} = (500, 258, 1575, 646)$, $D_{Ego} = (399, 144, 1062, 332)$, $D_{Point-Cloud} = (5.03k, 1.4k, 10.9k, 3k)$,

Before training the models, we applied Louvain algorithm to obtain hierarchical graph structures for all of datasets and then trimmed out the intermediate levels to achieve uniform depth of $L = 2$. In case of $\mathcal{HG}$ s with varying heights, empty graphs can be added at the root levels of those $\mathcal{HG}$s with lower heights to avoid sampling them during training. An 80%-20% split was randomly created for training and testing and 20% of the training data was used for validation purposes.

Model Architecture: In our experiments, the GraphGPS models consisted of 8 layers, while each level of hierarchical model has its own GNN parameters. The input node features were augmented with positional and structural encodings, which included the first 8 eigenvectors corresponding to the smallest non-zero eigenvalues of the Laplacian matrices and the diagonal of the random-walk matrix up to 8 steps. We leverage the original Transformer architecture for all detests except Point Cloud dataset which use Performer. The hidden dimensions were set to 64 for the Protein, Ego, and Point Cloud datasets, and 128 for the Stochastic Block Model and Enzyme datasets. The number of mixtures was set to K=20.

In comparison, the GRAN models utilized 7 layers of GNNs with hidden dimensions of 128 for the Stochastic Block Model, Ego, and Enzyme datasets, 256 for the Point Cloud dataset, and 512 for the Protein dataset. Despite having smaller model sizes, HiGen achieved better performance than GRAN.

For training, the HiGen models used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 5e-4 and default settings for β_1 (0.9), β_2 (0.999), and ϵ (1e-8).

The experiments for the Enzyme and Stochastic Block Model datasets were conducted on a MacBook Air with an M2 processor and 16GB RAM, while the rest of the datasets were trained using an NVIDIA L4 Tensor Core GPU with 24GB RAM as an accelerator.

E. Additional Results

Table 2 presents the results of various metrics for HiGen models on all benchmark datasets. The structural statistics are evaluated using the Total Variation kernel as the Maximum Mean Discrepancy (MMD) metric.

In addition, the table includes the average of random-GNN-based metrics (Thompson et al., 2022) over 10 random Graph Isomorphism Network (GIN) initializations. The reported metrics are MMD with RBF kernel (GNN MMD), the harmonic mean of improved precision+recall (GNN F1 PR) and harmonic mean of density+coverage (GNN F1 PR).

E.1. Point Cloud

We also evaluated HiGen on the *Point Cloud* dataset, which consists of 41 simulated 3D point clouds of household objects.This dataset consists of large graphs of approximately 1.4k nodes on average with maximum of over 5k nodes. In this dataset, each point is mapped to a node in a graph, and edges are connecting the k-nearest neighbors based on Euclidean distance in 3D space (Neumann et al., 2013).

However, due to the quadratic growth of the number of candidate edges in the augmented graph $\hat{\mathcal{G}}^l$ – the graph composed of all the communities and the candidate edges of all bipartites used in section 2.2 for bipartite generation – memory limitations can arise when dealing with large graphs in the point cloud dataset. To address this issue, we can sample sub-graphs and generate one (or a subset of) bipartites at a time to fit the available memory. In our experimental study, we generated bipartites sequentially, sorting them based on the index of their parent edges in the parent level. In this case, the augmented graph $\hat{\mathcal{G}}^l$ used for obtaining the node features of $\mathcal{B}^l_{ij}$ consists of all the communities $\{\mathcal{C}^l_k \ \ \forall k \leq \max(i,j)\}$ and all the bipartites $\{\mathcal{B}^l_{mn} \ \ \forall (m,n) \leq (i,j)\}$, augmented with the candidate edges of $\mathcal{B}^l_{ij}$. This model is denoted by HiGen-s in table 3.

Table 2: Various graph generative perfoprmance metrics for HiGen models on all benchmark datasets.

Model	Deg. ↓	Clus. ↓	Orbit↓	Spec. ↓	GNN MMD ↓	GNN F1 PR ↑	GNN F1 DC ↑
Enzyme							
HiGen-m	6.61e-03	2.65e-02	2.15e-03	8.75e-03	2.15e-02	9.70e-01	8.97e-01
HiGen	2.31e-03	2.08e-02	1.51e-03	9.56e-03	1.80e-02	9.78e-01	9.83e-01
Protein							
HiGen-m	0.0041	0.109	0.0472	0.0061	6.71e-02	9.79e-01	9.85e-01
HiGen	0.0012	0.0435	0.0234	0.0025	6.71e-02	9.79e-01	9.85e-01
Stochastic block model							
HiGen-m	0.0017	0.0503	0.0604	0.0068	1.54e-01	9.12e-01	0.83
HiGen	0.0019	0.0498	0.0352	0.0046	4.32e-02	9.86e-01	1.07
Ego							
HiGen-m	0.011	0.063	0.021	0.013	4.20e-02	0.87	0.68
HiGen	1.9e-3	0.049	0.029	0.004	5.20e-02	0.88	0.69

The GraphGPS models that was used for this experiment have employed Performer (Choromanski et al., 2020) which offers linear time and memory complexity. The results in Table 3 highlights the performance improvement of HiGen-s in both local and global properties of the generated graphs.

Table 3: Comparison of generation metrics on benchmark 3D point cloud. The baseline results are obtained from (Liao et al., 2019).

	3D Point Cloud			
Model	Deg. ↓	Clus. ↓	Orbit↓	Spec.↓
Erdos-Renyi	3.1e-01	1.22	1.27	4.26e-02
GRAN	**1.75e-02**	5.1e-01	2.1e-01	7.45e-03
HiGen-s	3.48e-02	**2.82e-01**	**3.45e-02**	**5.46e-03**

An alternative approach is to sub-sample a large graph such that each augmented sub-graph consists of a bipartite $\mathcal{B}_{ij}^{l}$ and its corresponding pair of communities $\mathcal{C}_i^l$, $\mathcal{C}_j^l$. This approach allows for parallel generation of bipartite sub-graphs but does not consider the connectivity between neighboring bipartites.

E.2. Ablation studies

In this section, two ablation studies were conducted to evaluate the sensitivity of HiGen with different node orderings and graph partitioning functions.

Node Ordering In our experimental study, the nodes in the communities of all levels are ordered using breadth first search (BFS) node ordering while the BFS queue are sorted by the total weight of edges between a node in the queue and predecessor nodes plus its self-edge. To compare the sensitivity of the proposed generative model against GRAN, we trained the models with default node ordering and random node ordering. The performance results, presented in Table 4, confirm that the proposed model is significantly less sensitive to the node ordering whereas the performance of GRAN drops considerably with non-optimal orderings.

Different Graph Partitioning In this experimental study, we evaluated the performance of HiGen using different graph partitioning functions. Firstly, to assess the sensitivity of the hierarchical generative model to random initialization in the Louvain algorithm, we conducted the HiGen experiment three times with different random seeds on the Enzyme dataset. The average and standard deviation of performance metrics are reported in Table 5 which demonstrate that HiGen consistently achieves almost similar performance across different random initializations.

Additionally, we explored spectral clustering (SC), which is a relaxed formulation of k-min-cut partitioning (Shi & Malik,

Table 4: Ablation study on node ordering. Baseline HiGen used the BFS ordering and baseline GRAN used DFS ordering. π_1, π_2 and π_3 are default, random and π_3 node ordering, respectively. Total variation kernel is used as MMD metrics of structural statistics. Also, the average of random-GNN-based metrics aver 10 random GIN initialization are reported for MMD with RBF kernel (GNN MMD), the harmonic mean of improved precision+recall (GNN F1 PR) and harmonic mean of density+coverage (GNN F1 PR).

	Enzyme						
Model	Deg. ↓	Clus. ↓	Orbit↓	Spec. ↓	GNN MMD ↓	GNN F1 PR ↑	GNN F1 DC ↑
GRAN	8.45e-03	2.62e-02	3.46e-02	2.11e-02	6.63e-02	9.50e-01	8.32e-01
GRAN (π_1)	1.75e-02	2.89e-02	3.78e-02	2.03e-02	6.51e-02	8.24e-01	6.69e-01
GRAN (π_2)	3.90e-02	3.24e-02	3.81e-02	2.38e-02	1.26e-01	8.31e-01	6.72e-01
HiGen	2.31e-03	2.08e-02	1.51e-03	9.56e-03	1.80e-02	9.78e-01	9.83e-01
HiGen (π_1)	1.83e-03	2.21e-02	6.75e-04	7.08e-03	1.78e-02	9.84-01	9.77e-01
HiGen (π_2)	3.31e-03	2.34e-02	2.06e-03	9.10e-03	2.04e-02	9.47-01	8.81e-01
HiGen (π_3)	1.34e-03	2.13e-02	6.94e-04	6.56e-03	1.90e-02	9.61e-01	9.74e-01

2000), as an alternative partitioning method. To determine the number of clusters, we applied SC to partition the graphs over a range of $0.7\sqrt{n} \leq k \leq 1.3\sqrt{n}$, where n represents the number of nodes in the graph. We computed the modularity score of each partition and selected the value of k that yielded the maximum score.

The results presented in Table 5 demonstrate the robustness of HiGen against different graph partitioning functions.

Table 5: Multiple initialization of Louvain partitioning algorithm and also min-cut partitioning

	Enzyme						
Model	Deg. ↓	Clus. ↓	Orbit↓	Spec. ↓	GNN MMD ↓	GNN F1 PR ↑	GNN F1 DC ↑
HiGen	2.64e-03±4.7e-4	2.09e-02±4.0e-4	7.46e-04±4.4e-4	1.74e-02±1.5e-3	2.00e-02±3.1e-3	.98±4.6e-3	.96±1.0e-2
HiGen (SC)	2.24e-03	2.10e-02	5.59e-04	8.30e-03	2.00e-02	.98	.94 ,

E.3. Graph Samples

Generated hierarchical graphs sampled from HiGen models are presented in this section.

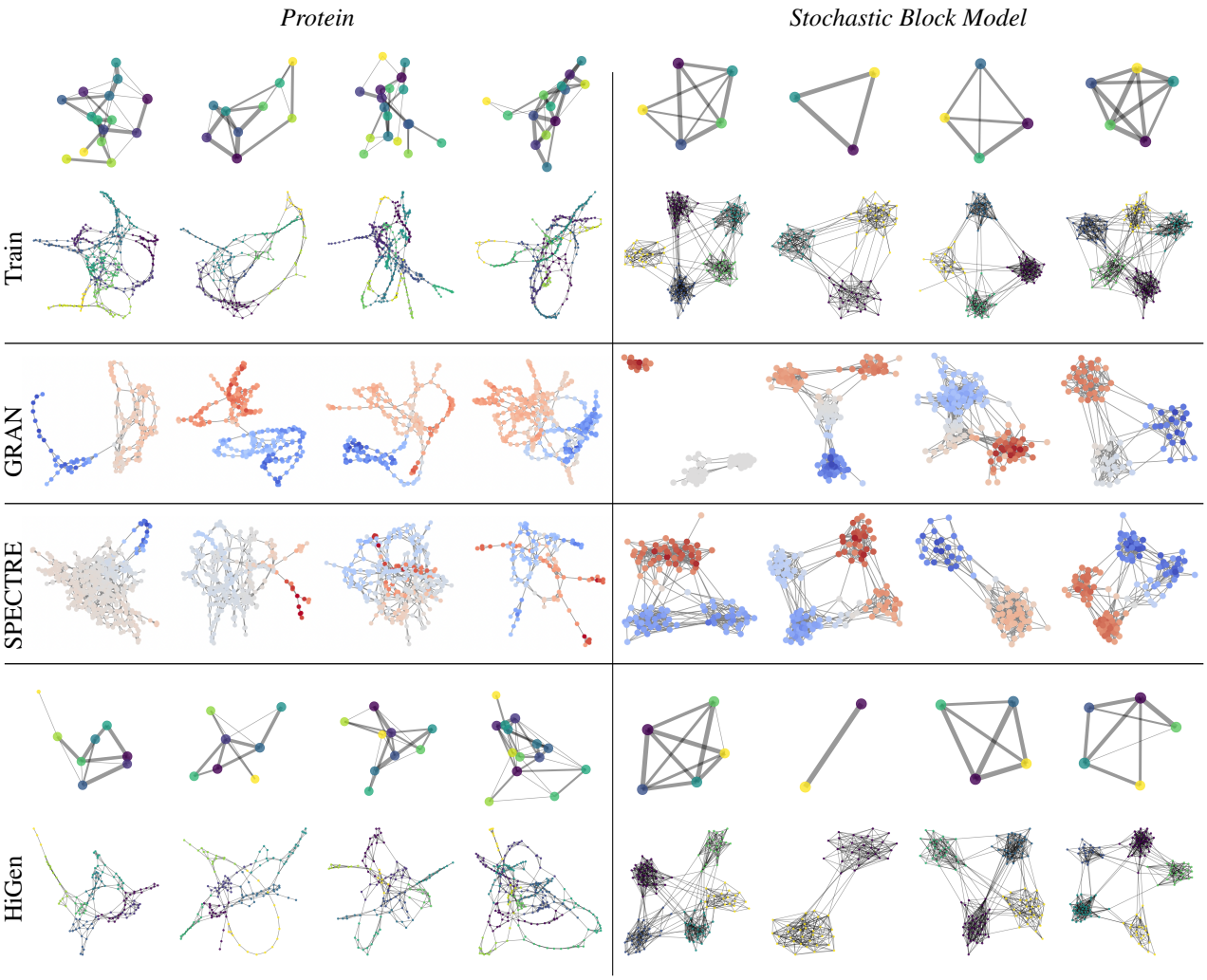

Figure 3: Samples from HiGen trained on *Protein* and *SBM*. Communities are distinguished with different colors and both levels are depicted. The samples for GRAN and SPECRE are obtained from (Martinkus et al., 2022).

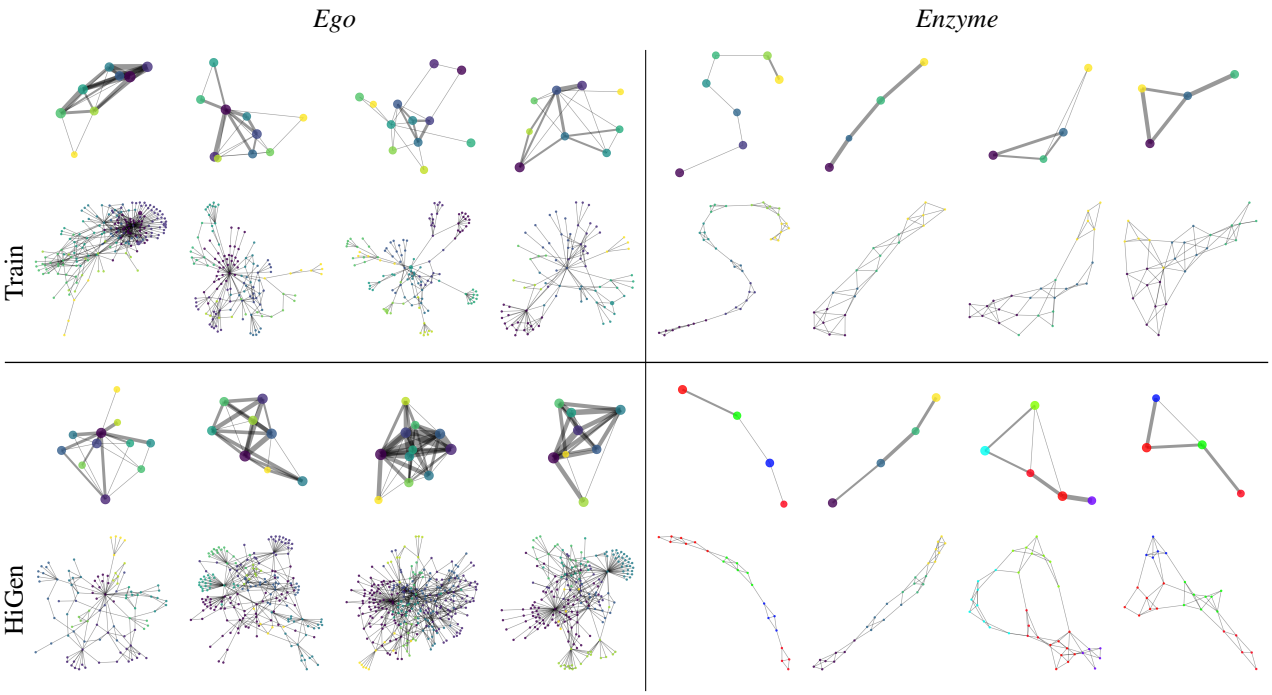

Figure 4: Samples from HiGen trained on *Protein* and *SBM*. Communities are distinguished with different colors and both levels are depicted.

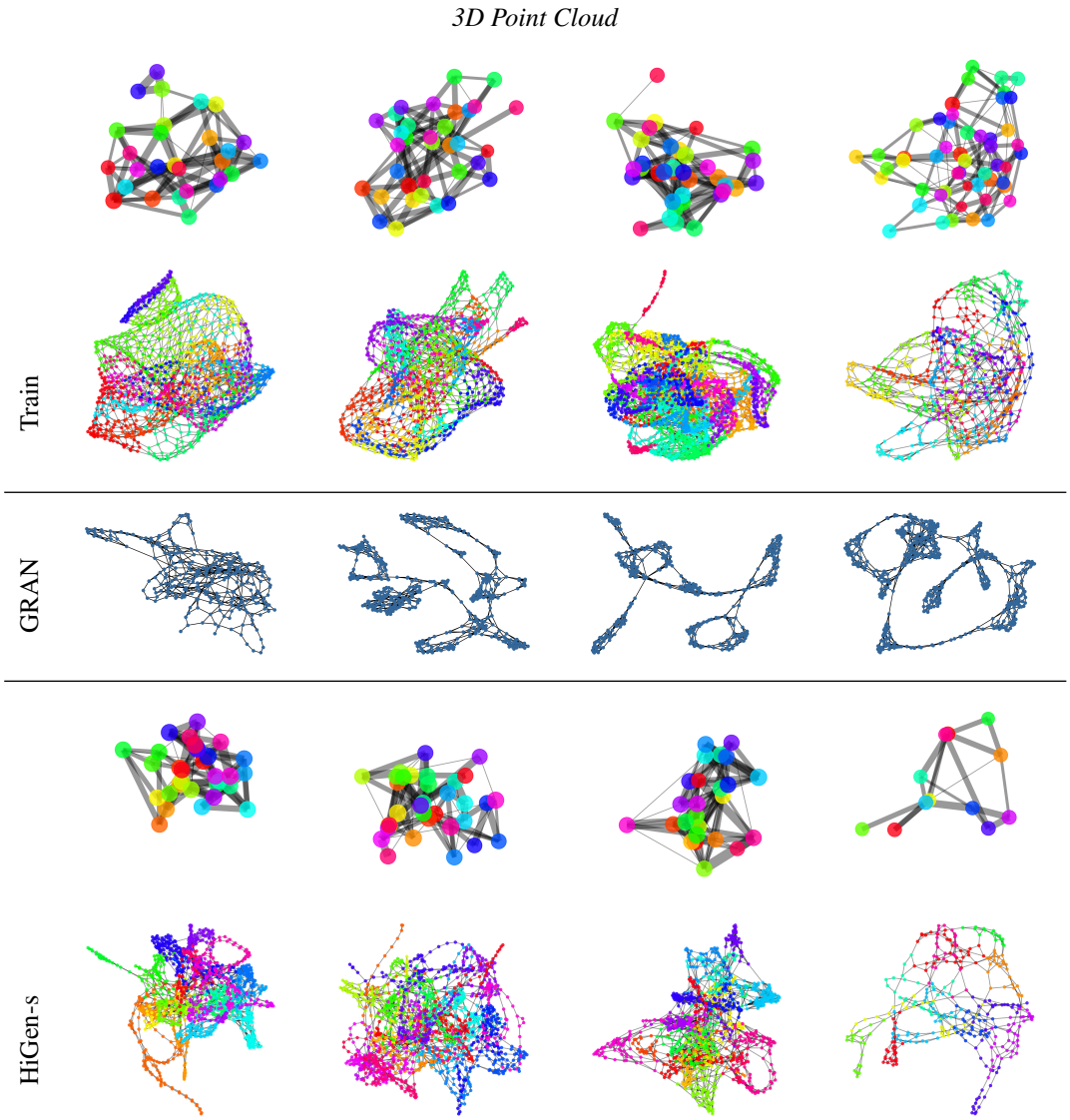

Figure 5: Samples from HiGen trained on *3D Point Cloud*. Communities are distinguished with different colors and both levels are depicted. The samples for GRAN are obtained from (Liao et al., 2019).