HIGEN: HIERARCHICAL MULTI-RESOLUTION GRAPH GENERATIVE NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

In real world domains, most graphs naturally exhibit a hierarchical structure. However, data-driven graph generation is yet to effectively capture such structures. To address this, we propose a novel approach that recursively generates community structures at multiple resolutions, with the generated structures conforming to training data distribution at each level of the hierarchy. The graphs generation is designed as a sequence of coarse-to-fine generative models allowing for parallel generation of all sub-structures, resulting in a high degree of scalability. Furthermore, we model the output distribution of edges with a more expressive multinomial distribution and derive a recursive factorization for this distribution, making it a suitable choice for graph generative models. This allows for the generation of graphs with integer-valued edge weights. Our method achieves state-of-the-art performance in both accuracy and efficiency on multiple graph datasets.

1 INTRODUCTION

Data-driven approaches to graph generation is both challenging and and highly valuable for various applications [3], including discovering new molecular and chemical structures, generation of realistic data networks, knowledge graph synthesizing scene graphs in computer vision and virtual reality [15; 19]. There are natural hierarchical community structures in these domains mentioned above, such as paragraphs, sentences, and words of a document, communities in user-item graphs, room of an apartment on a floor, columns of cars and groups of pedestrians on a city block. Higher level relations reflect long range and high-level interactions between communities, while low-level relations reflect the local structures. Realistic graph generation models must learn both of these interactions and be able to capture the cross-level relations. While hierarchical, multi-resolution generative models were developed for specific data types such as voice [18], image [20] and molecular motifs [8], these methods rely on domain-specific priors that are not be suitable for general graphs. To the best of our knowledge, there exists no generation models suitable for generic graphs that are both learned from data and handle interacting semantic hierarchies.

Graph generative models has been studied extensively. Classical methods from [5] and [1] are based on random graph theory that can only capture a set of hand engineered graph statistics. Leskovec et al. [10] proposed a scalable generative model based on the Kronecker product of matrices that can learn some graph properties such as degree distribution, but is very limited in modeling the underlying distributions. These models fail to capture important graph properties such as community structure in large family of graphs. Motivated by recent advances in recurrent neural networks (RNN) and graph neural networks (GNN), various neural network based generative models has been proposed [25; 11; 12]. These methods, which belong to the family of autoregressive algorithms, generate graphs as a sequence of edges or nodes so they highly rely on a appropriate node ordering and they don't take into account the community structures present within graphs. Moreover, due to their recursive nature they are computationally expensive for moderately large graphs, with generation process of $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ steps for GraphRNN and GRAN, respectively.

Herein, we propose an efficient hierarchical Multi-Resolution Generative (MRG) model to address the limitations of existing generative models by capturing community structures and cross-level interactions. The proposed model captures the hierarchical relations by allowing the representation of a node at each level to depend not only on its community but also on its corresponding super-node at the higher level. This approach allows the generation process at the lower level to be independent

of the specific ordering of the nodes at the higher levels, reducing the overall sensitivity to initial random permutations. The graphs generation is designed as a sequence of coarse-to-fine generative models where given a higher level (lower resolution) graph, the generation of the communities in the lower level can be performed in parallel, resulting in a high degree of scalability through parallelism. The output distribution of edges are parameterized by a multinomial distribution and a recursive factorization is derived for this distribution which enable the use of existing autoregressive methods for generating communities. This results an expressive distribution that can additionally models the graphs with integer-valued edge weights.

2 PROBLEM FORMULATION

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as a set of nodes (vertices) $\mathcal{V}$ and edges $\mathcal{E}$ with sizes $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ and adjacency matrix $\mathbf{A}^{\pi}$ for the node ordering π. A graph can be decomposed to *partition graphs* (a.k.a. community or cluster), denoted by $\mathcal{C}_i = (\mathcal{V}(\mathcal{C}_i), \mathcal{E}(\mathcal{C}_i))$ with adjacency matrix $\mathbf{A}_i$, and *bipartite graphs*, denoted by $\mathcal{B}_{ij} = (\mathcal{V}(\mathcal{C}_i), \mathcal{V}(\mathcal{C}_j), \mathcal{E}(\mathcal{B}_{ij}))$ with adjacency matrix $\mathbf{A}_{ij}$. Coarsening the graph results a graph at the higher level. Formally, each partition graph at level l, $\mathcal{C}_i^l$, is mapped to a node 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 of these parent nodes and edges are determined by the sum of the weights of the edges within the partition graph and bipartite, *i.e.* $w_{ii}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{C}_i^l)} w_e$ and $w_{ij}^{l-1} = \sum_{e \in \mathcal{E}(\mathcal{B}_{ij}^l)} w_e$, respectively. This process continues recursively in a bottom-up manner until a single node graph $\mathcal{G}^0$ is obtained which results in a hierarchical graph (hyper-graph) $\mathcal{HG} := \{\mathcal{G}^0, ..., \mathcal{G}^{L-1}, \mathcal{G}^L\}$.

Community detection Different community detection algorithms have been proposed that try to identify communities based on specific metrics. or cluster nodes with similar features. The Louvain algorithm [2] is a popular method for graph coarsening, which is a process of reducing the resolution of a graph by grouping similar nodes together. It is a community detection algorithm that iteratively detects communities by maximizing a modularity function. The algorithm starts with each node as its own community and then repeatedly merges communities based on the highest increase in modularity until no further improvement can be made. The resulting communities form a coarser graph with fewer nodes, where each node represents a group of nodes from the original graph. This heuristic algorithm is computationally efficient and scalable to large graphs for community detection based on graph topology, making it a suitable choice for our graph coarsening step.

Hierarchical Multi-Resolution Graph Generation This work aims to establish a multi-resolution framework that generates graph in a coarse-to-fine approach. We can show that:

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

That is, given a higher level graph, the graph at its subsequent (child) level can be specified by a conditional probability, and this process can be repeated until the lowest level, or leaf level, is attained.

Community-based Graph Generation Based on the community structure of a hyper-graph, conditional probability of a graph at level l, $p(\mathcal{G}^l \mid \mathcal{G}^{l-1})$, can be factorized according to its partition graphs and bipartites:

$$p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) \approx \prod_{i \in \mathcal{V}_{\mathcal{G}^{l-1}}} p(\mathcal{C}_i^l \mid \mathcal{G}^{l-1}) \prod_{<i,j> \in \mathcal{E}_{\mathcal{G}^{l-1}}} p(\mathcal{B}_{ij}^l \mid \mathcal{G}^{l-1}, \mathcal{C}_i^l, \mathcal{C}_j^l)$$

Here, we assume that the partition graph $\mathcal{C}_i^l$ is independent of all other components in its level given the parent graph $\mathcal{G}^{l-1}$. Additionally, the bipartite graphs $\mathcal{B}_{ij}^l$ are assumed to be independent of the rest of components given the parent graph $\mathcal{G}^{l-1}$ and their corresponding pairs of parts $(\mathcal{C}_i^l, \mathcal{C}_j^l)$. As shown in lemma 4, this conditional independence property is met when the edge weights in a level

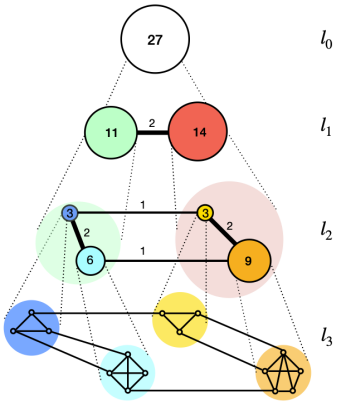


Figure 1: (a) A sample hierarchical graph with 3 levels is shown. Communities are shown in different colors and the weight of a node and the weight of an edge in a higher level, represent the sum of the in-community connections in the corresponding community and the total weight of corresponding bipartite, 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, partition graphs are in shown different colors and bipartites are colored in gray.

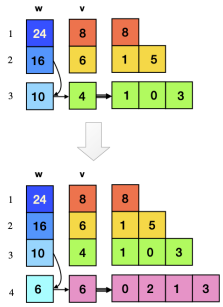


Figure 2: Decomposition of multinomial distribution as a recursive *stick-breaking* process where at each iteration, first a fraction of the remaining weights w_m is allocated to the m-th row (the m-th node in the sub-graph) and then this fraction v_m is distributed among that row of lower triangular adjacency matrix, $\hat{A}$.

is modeled by multinomial distribution, that is given the weights of parent edges of partition graphs and bipartites, the distribution of group of edges in components are independent and are also multinomial. Therefore, given the graph at a higher level, the generation of graph at its following level is reduced to generation of its partition and bipartite sub-graphs. As illustrated in figure 1, each of these sub-graphs corresponds to a block in the adjacency matrix, so the proposed hierarchical model generates adjacency matrix in a blocks-wise fashion and constructs the final graph topology. As a result, the generation of the partitions in each level can be performed in parallel and subsequently, the generation decisions of all bipartites in each level may occur at one pass. In the following, we specify the final generative probabilities, $p(\mathcal{C}_i^l \mid \mathcal{G}^{l-1})$, to include the state of the parent graph and also to model the non-negative integer valued weights of the edges in a recursive form.

2.1 Probability Distribution of Candidate Edges

In a hierarchical graph, the edges has non-negative integer valued weights while the sum of all the edges in partition graph $\mathcal{C}_i^l$ 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. Therefore, the edge weights in each subgraph can be modeled as a multinomial distribution. So, let's denote the set 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

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

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 $\mathcal{B}_{ij}^l$.

Likewise, the probability distribution of the set of candidate edges for each partition graph can be modeled by a multinomial distribution but since the generation decisions happens as a sequential process, we are interested in factorizing this probability distribution accordingly.

Lemma 1 *A random vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution can be recursively decomposed to a sequence of binomial distributions:*

$$\begin{aligned} &\textit{Mu}(\mathrm{w}_1,\ ..., \mathrm{w}_E \mid w,\ [\theta_1,\ ..., \theta_E]) \\ &\qquad = \prod_{e=1}^{E} \textit{Bi}(\mathrm{w}_e \mid w - \sum\nolimits_{i<e} \mathrm{w}_i, \hat{\theta}_e), \\ &\textit{where: } \hat{\theta}_e = \frac{\theta_e}{1 - \sum_{i<e} \theta_i} \end{aligned} \tag{2}$$

This decomposition is 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 [13].

This lemma offers modeling the generation of a partition graph as edge-by-edge generation sequence hence is analogous to autoregressive algorithms such as GraphRNN [25] with $\mathcal{O}(|\mathcal{V}_\mathcal{C}|^2)$ generation steps. As a more efficient alternative, we are interested in generating a partition graph one node at a time which entails factorizing the edges probability in a group-wise form where the candidate edges between the t-th node and already generated graph are grouped together. The following theorem present such factorization.

Theorem 1 *For a random counting vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution $\textit{Mu}(\mathbf{w} \mid w, \boldsymbol{\theta})$, 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 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 modeled as a chain of binomials and multinomials:*

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

$$\textit{where: } \eta_{\mathrm{v}_m} = \frac{\mathbf{1}^T\, \boldsymbol{\theta}_m}{1 - \sum_{i<m} \mathbf{1}^T\, \boldsymbol{\theta}_i},\ \ \boldsymbol{\lambda}_m = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T\, \boldsymbol{\theta}_m}. \tag{3}$$

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 probability 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 fraction of the remaining probability mass is allocated to a group by a binomial distribution and then this fraction is distributed among that group's members by a multinomial distribution.*

Proof: *Refer to appendix B for the proof.* ■

Based on theorem 1, at the t-th step of generating a partition graph, one can characterize the generative probability of the group of candidate edges corresponding to node $v_t(\mathcal{C}_i^l)$, denoted as $\mathbf{u}_t := \mathcal{E}_t(\mathcal{C}_i^l)$ (the t-th row of the lower triangle of adjacency matrix $\hat{\mathbf{A}}_i^l$), by the product of a binomial and a multinomial distribution. This process is visualized in figure 2. In order increase model's expressiveness, we further extend this probability to a mixture model with K mixtures:

$$p(\mathbf{u}_t) = \sum_{k=1}^{K} \boldsymbol{\beta}_k^l \mathrm{Bi}(\mathrm{v}_t \mid w_{ii}^{l-1} - \sum_{i<t} \mathrm{v}_i, \eta_{t,k})\ \mathrm{Mu}(\mathbf{u}_t \mid \mathrm{v}_t, \boldsymbol{\lambda}_{t,k}) \tag{4}$$

$$\boldsymbol{\lambda}_{t,k} = \mathrm{softmax}\left(\{\mathrm{MLP}_{\boldsymbol{\theta}}^l(\left[\Delta \boldsymbol{h}_{\mathcal{E}(\mathcal{C}_{i,t}^l)};\ h_{Pa(\mathcal{C}_i^l)}\right])\}\right)[k,:] \tag{5}$$

$$\eta_{t,k} = \mathrm{sigmoid}\left(\mathrm{MLP}_{\eta}^l(\left[\mathrm{pool}(\boldsymbol{h}_{\mathcal{C}_{i,t}^l});\ h_{Pa(\mathcal{C}_i^l)}\right])\right)[k]$$

$$\boldsymbol{\beta}^l = \mathrm{softmax}\left(\mathrm{MLP}_{\beta}^l(\left[\mathrm{pool}(\boldsymbol{h}_{\mathcal{C}_{i,t}^l});\ h_{Pa(\mathcal{C}_i^l)}\right])\right)$$

Where $\Delta \boldsymbol{h}_{\mathcal{E}(\mathcal{C}_{i,t}^l)}$ is a $|\mathcal{E}_t(\mathcal{C}_i^l)| \times d_h$ dimensional matrix composed of the set of edge representations $\{\Delta h_{<t,s>} := h_t - h_s \mid \forall\ <t,s>\ \in \mathcal{E}_t(\mathcal{C}_i^l)\}$, and $\boldsymbol{h}_{\mathcal{E}(\mathcal{C}_{i,t}^l)}$ is a $t \times d_h$ matrix of node representations

of already generated nodes in the partition graph. The graph level representation is obtained by the $\mathrm{addpool}()$ aggregation function. Here, $\mathrm{MLP}^l_{\boldsymbol{\theta}}()$ acts at edge level and produce $K \times |\mathcal{E}_t(\mathcal{C}^l_i)|$ dimensional output, while both $\mathrm{MLP}^l_{\eta_v}()$ and $\mathrm{MLP}^l_{\beta}()$ produce K dimensional arrays for K mixture models. All of the MLP networks are build by two hidden layers with ReLU activation functions and the mixture weights are denoted by $\boldsymbol{\beta}^l$. For each partition graph $\mathcal{C}^l_i$, node representation of its parent node $h_{Pa(\mathcal{C}^l_i)}$, *i.e.* the node that represent $\mathcal{C}^l_i$ at the higher level, is used as the context and concatenated to the representation matrices. The operation $\left[\boldsymbol{x};\ y\right]$ denotes the concatenation of each row of matrix $\boldsymbol{x}$ with vector y. This context enriches the node/edge representations by capturing long range interactions and encoding the global structure of the graph while generating a local component.

The node representations are obtained by the attention based graph neural network (GNN) in [12], simply denoted as $h_i = \mathrm{GNN}^l(\mathcal{G}_{in}; \gamma^l)$ that is a GNN parameterized by a set of parameters γ^l.

On the other hand, the generation of edges in bipartite graph $\mathcal{B}^l_{ij}$ can be simply performed simultaneously, and we similarly use the mixture of multinomial distribution (2) to model the generative probability:

$$p(\mathbf{w} := \mathcal{E}(\mathcal{B}^l_{ij})) = \sum_{k=1}^{K} \boldsymbol{\beta}^l_k \mathrm{Mu}(\mathbf{w} \mid w^{l-1}_{ij}, \boldsymbol{\theta}^l_{ij,k}) \tag{6}$$

$$\boldsymbol{\theta}^l_{ij,k} = \mathrm{softmax}\left(\mathrm{MLP}^l_{\boldsymbol{\theta}}\big(\left[\Delta \boldsymbol{h}_{\mathcal{E}(\mathcal{B}^l_{ij})};\ \Delta h_{Pa(\mathcal{B}^l_{ij})}\right]\big)\right)[k,:] \tag{7}$$

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

where $\Delta_{Pa(\mathcal{B}^l_{ij})}$ is the edge representation of the parent edge of the bipartite graph, the edge that represent the $\mathcal{B}^l_{ij}$ at the higher level.

The probability of the integer-valued edges are modeled by $\mathrm{softmax}()$ function in (5) and (7), but since the final graphs in our experiments have binary edges weights, we instead use multi-hot activation function $\sigma : \mathbb{R}^K \to (K-1)$-simplex, defined as

$$\sigma(\mathbf{z})_i = \frac{\mathrm{sigmoid}(z_i)}{\sum_{j=1}^{K} \mathrm{sigmoid}(z_j)}.$$

for the leaf level while the upper levels still employ $\mathrm{softmax}$ activation. In our experiments, this function could better model the edge probability of the leaf level compared to standard $\mathrm{softmax}$ function. As an alternative, we also modeled the edges at the leaf level by the mixture of Bernoulli using $\mathrm{sigmoid}()$ activation for the output while higher levels use mixture of multinomials. A possible extension to this work could be using the cardinality potential model [7], derived to model the distribution over the set of binary random variables, for the last level.

3 Experiments

In our empirical studies, we compare the proposed method against some well-established baselines on two synthetics datasets and three real-world datasets.

First, we generated **R**elaxed **C**aveman **G**raphs (**RCG**) which starts with $7 \leq l < 25$ cliques of size $15 \leq k < 25$. Edges are then randomly rewired with probability $p = 1/l$ to different cliques. We also generated **P**lanted **P**artition **G**raphs (**PPG**). This model partitions a graph with n nodes in $20 \leq l < 30$ groups with $15 \leq k < 25$ nodes each. Nodes of the same group are linked with a probability $p_{in} = .75$, and nodes of different groups are linked with probability $p_{out} = 10/(kl^2)$. Both of these datasets that exhibit strong community structures are generated using NetworkX Python package [6].

The real-world datasets are (1) **Protein** dataset which contains 918 protein graphs, each of which has 100 to 500 nodes for amino acids and has edges for amino acid pairs closer than 6 Angstroms [4], (2) **Ego** dataset which contains 757 3-hop ego networks with 50 to 300 nodes extracted from the CiteSeer dataset, with nodes representing documents and edges representing citation relationships [21], and (3) **Point Cloud** with 41 simulated 3D point clouds of household objects. This dataset has

about 1.4k nodes on average with maximum of over 5k nodes. Each point is mapped to a node and edges connecting the k-nearest neighbors in Euclidean distance in 3D space are added to the graphs [16].

To partition graphs and obtain hierarchical graph structures, we applied Louvain algorithm on all of these datasets. This resulted in hierarchical graphs of depth $L = 2$ for the synthetic datasets, while for the real world graphs it produced at least 3 levels so we spliced out the intermediate levels so that all have equal depth of $L = 3$.[1] Before training the models, we follow the protocol in [12] to randomly create a 80%-20% training-testing split, with 20% of the training data reserved as the validation set.

Experimental setup: To provide a fair comparison, we closely follow the experimental setup of You et al. [25] and Liao et al. [12]. We compared the proposed model against the baseline methods including Erdos-Renyi [5], GraphVAE [23], GraphRNN & GraphRNN-S [25], and GRAN [12]. The results of the baselines are extracted from [12] for the real-world graphs while we retrained them for Ego and synthetic datasets. The neural network based methods have the following structures. GraphVAE model used a 3-layer GCN encoder and an MLP decoder with 2 hidden layers where all hidden dimensions are set to 128 for all experiments. For GraphRNN and GraphRNN-S, the best settings reported in the original paper were used. GRAN enjoyed 7 layers of GNNs with one round of message passing. Hidden dimensions are set to 128 for [Ego, RCG], 256 for Point Cloud and 512 for [Protein, PPG] for GRAN, while we used smaller hidden dimensions of 64 for [Ego, RCG, Point Cloud, Protein] and 128 for PPG.

We tested our proposed multi-resolution model (MRG) model with two variants: 1) the model that uses mixture of multinomial distribution (4) to describe the output distribution for all levels is simply denoted by **MRG**, 2) the model that replace the output distribution of the leaf level with mixture of Bernoulli distribution while higher levels use mixture of multinomials is indicated by **MRG-B**. To obtain node and edge representation, each level has its own GNN and output models, which are indexed by the level number of our model definition in section 2. We use the same GNN architecture as GRAN, with 7 layers of GNNs with one round of message passing, but we choose smaller hidden dimensions, setting it to 64 for [Ego, RCG, Point Cloud], and 128 for [Protein, PPG]. For both GRAN and MRG, the number of mixtures is set $K = 20$ and block size and stride are both set to 1. In general, MRG models uses less parameters compared to GRAN. The comparison of total number of parameters of MRG and GRAN are listed in appendix C. MRG models are training by the Adam optimizer [9] with learning rate of 5e-4.

For evaluation of the graph generative models, we follow the approach in [14; 12] which compare the following distributions of 4 different graph statistics between ground truth and generated graphs: (1) degree distributions, (2) clustering coefficient distributions, (3) the number of occurrence of all orbits with 4 nodes, and (4) the spectra of the graphs by computing the eigenvalues of the normalized graph Laplacian. The first 3 metrics characterize local graph statistics while the spectra represents global structure. After computing these statistics, the maximum mean discrepancy MMD score is computed over these statistics. MMD score in [14] depends on Gaussian kernels with the first Wassertein distance, (the earth mover's distance (EMD)). However, evaluating this kernel is computationally expensive for moderately large graphs, so we follow Liao et al. [12] in using total variation (TV) distance as an alternative measure which is very faster while still consistent with EMD. Most recently, O'Bray et al. [17] suggested using other efficient kernels such as an RBF kernel, or a Laplacian kernel, or a linear kernel. Also, Thompson et al. [24] proposed new evaluation metrics for comparing graph sets by leveraging a random-GNN where GNNs are employed to extract meaningful graph representations. Here we choose to comply with the experimental setup and evaluation metrics by GRAN.

The performance of the proposed graph generative models, evaluated using the maximum mean discrepancy (MMD) metric, are reported in Table 1. Additionally, samples of the generated graphs are presented in Figure 3. The results indicate that the proposed models outperform the existing graph generative models in most cases while it is on par with the best baseline in remaining cases. This performance gap is particularly noticeable when the graph datasets has community structures. These findings suggest that the proposed models are effective at generating graphs, particularly

[1]The proposed architecture can be trained on HGs with uneven heights by adding empty graphs at the root levels of those HGs with lower height so that they are not sampled during the training.

Table 1: In this table the quality of generated graphs are compared in terms of the MMD of graph degree distributions (*Deg.*), clustering coefficient (*Clus.*), 4-node orbits (*Orbit*), and the spectra of the graph Laplacian (*Spec.*). For all the metrics, the smaller the better. Graph sizes, $(|V|_{max}, |V|_{avg}, |E|_{max}, |E|_{avg})$, are listed for each dataset.

	Protein (500, 258, 1575, 646)				**3D Point Cloud** (5.03k, 1.4k, 10.9k, 3k)				**Ego** (399, 144, 1062, 332)				**PPG** (696, 477, 7.5k, 4.4k)				**RCG** (576, 261, 6.6k, 2.2k)			
	Deg.	Clus.	Orbit	Spec.	Deg.	Clus.	Orbit	Spec.	Deg.	Clus.	Orbit	Spec.	Deg.	Clus.	Orbit	Spec.	Deg.	Clus.	Orbit	Spec.
Erdos-Renyi	$5.64e^{-2}$	1	1.54	$9.13e^{-2}$	$3.1e^{-1}$	1.22	1.27	$4.26e^{-2}$	$1.6e^{-1}$	$9.4e^{-1}$	$8.5e^{-1}$	$1.8e^{-1}$	$2.83e^{-1}$	1.04	$1.94e^{-1}$	$2.01e^{-1}$	$1.7e^{-1}$	$8.0e^{-1}$	$1.2e^{-1}$	$2.0e^{-1}$
GraphVAE	$4.8e^{-1}$	$7.14e^{-2}$	$7.4e^{-1}$	$1.1e^{-1}$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
GraphRNN-S	$4.02e^{-2}$	**$4.79e^{-2}$**	$2.3e^{-1}$	$2.1e^{-1}$	-	-	-	-	$6.51e^{-3}$	$2.24e^{-1}$	$6.35e^{-2}$	$7.30e^{-2}$	$4.34e^{-2}$	$3.01e^{-1}$	$3.32e^{-2}$	$1.70e^{-2}$	$7.02e^{-2}$	$2.47e^{-2}$	$3.41e^{-2}$	$4.91e^{-2}$
GraphRNN	$1.06e^{-2}$	$1.4e^{-1}$	$8.8e^{-1}$	$1.88e^{-2}$	-	-	-	-	$2.44e^{-2}$	$3.46e^{-1}$	$1.35e^{-1}$	$8.91e^{-2}$	$9.65e^{-2}$	$3.12e^{-1}$	$2.97e^{-2}$	$4.90e^{-2}$	$6.74e^{-2}$	$1.82e^{-2}$	$3.00e^{-2}$	$4.93e^{-2}$
GRAN	**$1.98e^{-3}$**	$4.86e^{-2}$	$1.3e^{-1}$	**$5.13e^{-3}$**	**$1.75e^{-2}$**	$5.1e^{-1}$	$2.1e^{-1}$	**$7.45e^{-3}$**	$3.2e^{-2}$	$1.7e^{-1}$	$2.6e^{-2}$	$4.6e^{-2}$	$5.67e^{-2}$	$2.3e^{-1}$	$2.82e^{-1}$	$1.71e^{-2}$	$7.50e^{-2}$	$1.34e^{-2}$	$9.95e^{-2}$	$5.70e^{-2}$
MRG-B	$5.1e^{-3}$	$6.27e^{-2}$	$1.08e^{-1}$	$8.0e^{-3}$	$1.29e^{-1}$	**$3.4e^{-1}$**	$5.9e^{-2}$	$8.9e^{-3}$	**$4.1e^{-3}$**	**$6.2e^{-2}$**	$1.8e^{-2}$	**$1.42e^{-2}$**	**$4.79e^{-3}$**	**$8.79e^{-2}$**	$4.8e^{-2}$	**$1.85e^{-3}$**	**$1.45e^{-2}$**	**$1.29e^{-2}$**	$2.75e^{-2}$	**$4.1e^{-3}$**
MRG	$6.49e^{-3}$	$2.24e^{-1}$	**$5.78e^{-2}$**	$1.31e^{-2}$	$2.07e^{-1}$	$8.06e^{-1}$	**$2.75e^{-2}$**	$2.24e^{-2}$	$1.78e^{-2}$	$2.24e^{-1}$	**$1.16e^{-2}$**	$2.07e^{-2}$	$1.51e^{-1}$	$3.68e^{-1}$	**$7.75e^{-3}$**	$1.93e^{-2}$	$4.45e^{-2}$	$2.60e^{-2}$	**$1.15e^{-2}$**	$5.76e^{-2}$

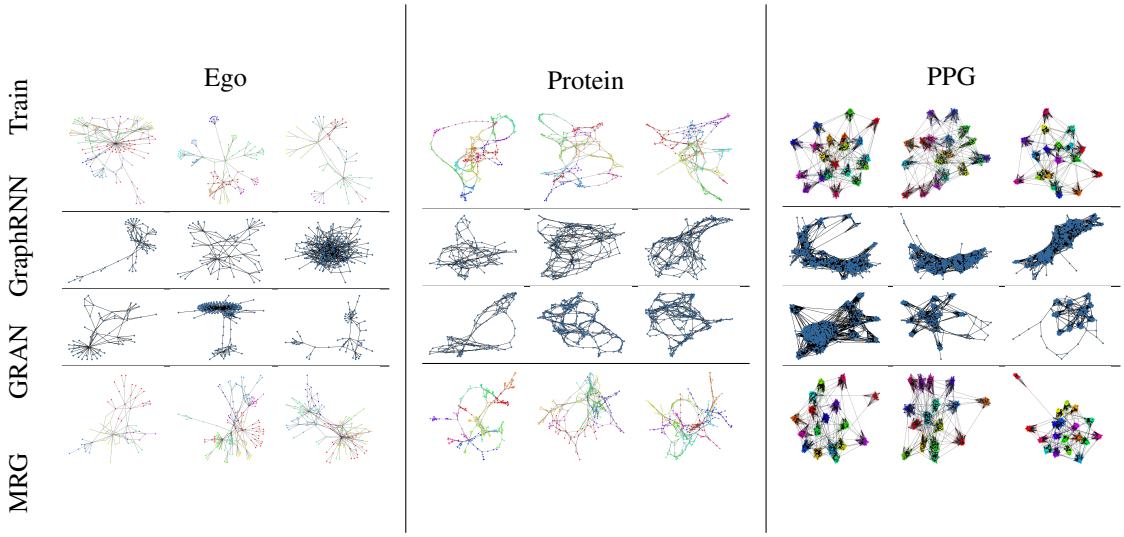


Figure 3: Sample graphs generated by different models are compared to training samples at the top. Communities are distinguished with different colors in training and MRG samples.

those with community structures, and demonstrate the potential of the proposed models in a variety of applications. More graph samples, including their hierarchical structures, generated by the MRG models are presented in appendix B.

3.1 Ablation Studies

In this section, two ablation studies were conducted to evaluate more compact forms of the MRG model.

The first study evaluated the performance of MRG with fewer hierarchical levels by splicing out the middle level of the Ego dataset, resulting in hierarchical graphs (HGs) with only 2 levels after the root, *i.e.* $L = 2$. The results, presented in Table 3, show that the generation quality of the models drops slightly when the number of levels is decreased, indicating that having more hierarchical levels improves the expressiveness of the model.

Moreover, we train the MRG with shared model parameters across levels such that all levels use similar GNN and output models. The performance comparison in Table 3 show that using individual models for each level offers better results. This can can be explained by the fact that graph at different levels exhibits different characteristics such as graph sparsity that may require tailored models for optimal performance.

4 Discussion

In contrast to the proposed method, GRAN can generates the graph topology in a block-wise fashion with a fixed block size, where nodes are partitioned into blocks according to an ordering but their

Table 2: Comparison of models with different number of levels and shared model parameters across the levels (MRG-B shared).

	Ego			
	Deg.	Clus.	Orbit	Spec.
MRG-B 3-level	$4.1e^{-3}$	$6.2e^{-2}$	$1.8e^{-2}$	$1.42e^{-2}$
MRG-B 2-level	$4.73e^{-3}$	$5.43e^{-2}$	$1.41e^{-2}$	$1.9e^{-2}$
MRG-B shared	$1.87e^{-2}$	$3.68e^{-1}$	$3.20e^{-2}$	$3.16e^{-2}$

intra-block connections are not modeled separately. Moreover, the performance of GRAN degrades with increasing the block size since two adjacent nodes in an ordering do not necessary have relevancy and might be far different clusters. On the other hand, the proposed method first generates the block of each community of nodes that has strong relations to each other and then connects the blocks. This approach allows for capturing the the relationships between nodes within a community, leading to an improved performance.

In comparison, GRAN can generate the graph topology in a block-wise fashion with fixed block size where the nodes are split into blocks according to an ordering and intra-block connections are not modeled separately. Moreover, the performance of GRAN degrades with increasing the block size since two adjacent nodes in an ordering do not necessary have relevancy and might be far different clusters, but the proposed method first generates the block of each community of nodes that has strong relations to each other and then connect the blocks.

Node ordering sensitivity The graph generation process is reduced to generation of multiple small partitions and is performed sequentially across the levels, therefore, given an ordering for the parent level, the graph generation depends only on the permutation of the node within the components rather than the node ordering of the entire graph. In other words, the proposed method is invariant to big portion of the possible node permutations and therefore, the set of distinctive adjacency matrices is very smaller in this hierarchical generative model. Therefore, it is significantly less sensitive to node ordering compared to the available models. As an example, node ordering $\pi_1 = [v_1, v_2, v_3, v_4]$ with clusters $[v_1, v_2], [v_3, v_4]$ has similar HG as $\pi_2 = [v_1, v_3, v_2, v_4]$ since node ordering inside communities are preserved at all levels.

5 Conclusion

We proposed a novel data-drive generative model for generic hierarchical graphs. This model does not rely on domain-specific priors and can be used widely. Our method also supports maximally parallelized implementations insofar as the graph is amenable to balanced recursive tree decomposition. We demonstrated the effectiveness and efficiency of our method on 2 synthetic and 3 real datasets. While the Louvain algorithm we depend on for community detection is rule-based, still the proposed method is proven to be effective.

For future work, developing a fully end-to-end algorithm for encoding and decoding with joint learning of community structures, instead of depending on an external algorithm for community detection, will be both challenging and desirable. Moreover, both for the current method using various community-detection algorithms and for the future end-to-end solution, validation on datasets that are orders of magnitude bigger than what we used in this work to introduce the new method will be an informative and worthy undertaking.

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[2] 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.

[3] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International Conference on Machine Learning*, pp. 2302–2312. PMLR, 2020.

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

[5] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[6] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[7] Hossein Hajimirsadeghi, Wang Yan, Arash Vahdat, and Greg Mori. Visual recognition by counting instances: A multi-instance cardinality potential kernel. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2596–2605, 2015.

[8] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020.

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

[10] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11(Feb):985–1042, 2010.

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

[12] 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.

[13] 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.

[14] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows, 2019.

[15] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[16] 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.

[17] 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.

[18] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[19] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[20] Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel multiscale autoregressive density estimation. In *International Conference on Machine Learning*, pp. 2912–2921. PMLR, 2017.

[21] 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.

[22] Kyle Siegrist. *Probability, Mathematical Statistics, Stochastic Processes*. LibreTexts, 2017.

[23] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

[24] 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.

[25] 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 Appendix

A PROOF OF THEOREM 1

Lemma 2 *A random vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution can be recursively decomposed to a sequence of binomial distributions:*

$$\begin{aligned} &\boldsymbol{Mu}(\mathrm{w}_1,\ ..., \mathrm{w}_E \mid w,\ [\theta_1,\ ..., \theta_E]) \\ &\qquad = \prod_{e=1}^{E} \boldsymbol{Bi}(\mathrm{w}_e \mid w - \sum\nolimits_{i<e} \mathrm{w}_i, \hat{\theta}_e), \\ &\textit{where: } \hat{\theta}_e = \frac{\theta_e}{1 - \sum_{i<e} \theta_i} \end{aligned} \tag{8}$$

This decomposition is 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 [13].

This lemma offers modeling the generation of a partition graph as edge-by-edge generation sequence hence is analogous to autoregressive algorithms such as GraphRNN [25] with $\mathcal{O}(|V_C|^2)$ generation steps. As a more efficient alternative, we are interested in generating a partition graph one node at a time which entails factorizing the edges probability in a group-wise form where the candidate edges between the t-th node and already generated graph are grouped together. The theorem 1 present such factorization.

Now, for a random counting vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution $\mathrm{Mu}(w, \boldsymbol{\theta})$, 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 $\mathrm{v}_m := \sum_{e=1}^{E_m} \mathrm{u}_{m,e}$.

Lemma 3 *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:*

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

In the other words, the multinomial distribution is preserved when its counting variables are combined [22].

Lemma 4 *Given the sum of counting variables in the groups, the groups are independent and each of them has multinomial distribution:*

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

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

Proof:

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

■

Theorem 2 *Given the aforementioned grouping of counts variables, the multinomial distribution can be modeled as a chain of binomials and multinomials:*

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

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

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

According to lemma 3, sum of the weights of the groups is a multinomial and by lemma 2, it can be decomposed to a sequence of binomials:

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

Also based on lemma 4, given the sum of the wights of all groups, the groups are independent and has multinomial distribution:

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

■

B GENERATED SAMPLES

Generated hierarchical graphs sampled MRG models are presented in this section.

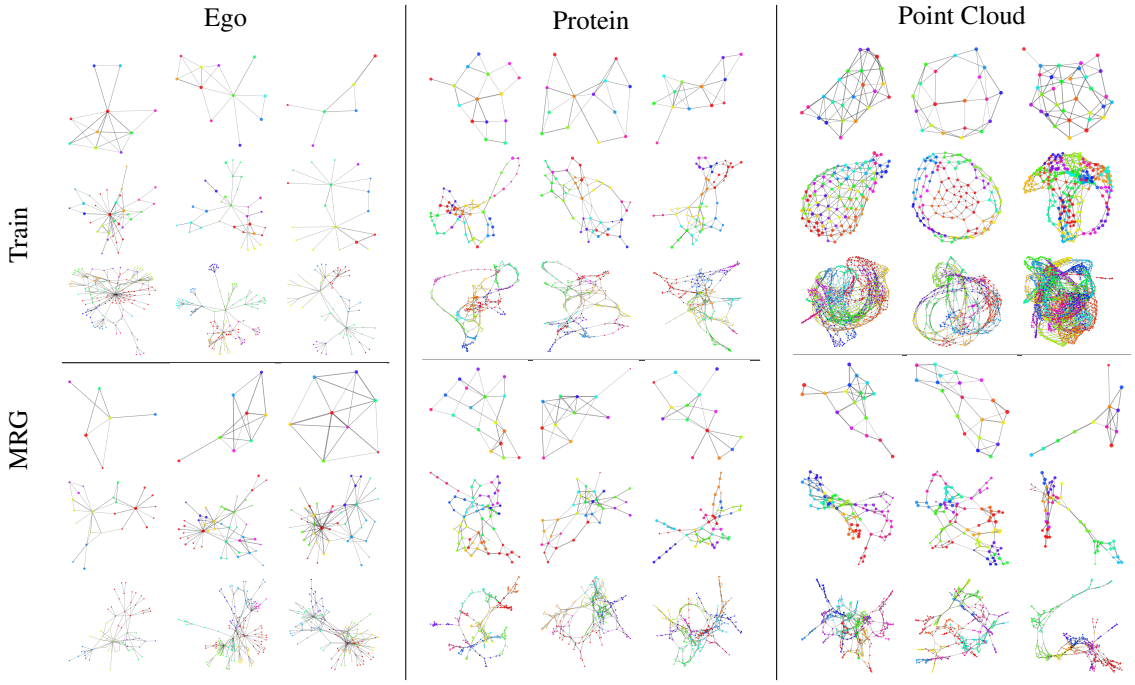
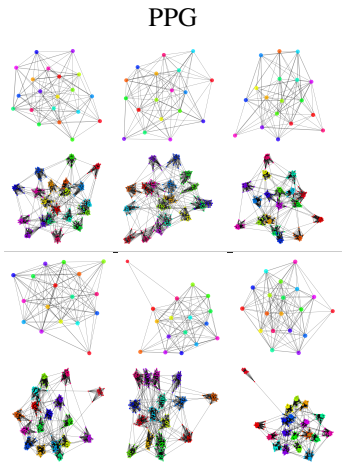


Figure 4: Sample hyper-graphs at 3 levels generated by different models shown at the bottom with training samples at the top.



C EXPERIMENTAL DETAILS:

Datasets First, we generated **R**elaxed **C**aveman **G**raphs (**RCG**) which starts with $7 \leq l < 25$ cliques of size $15 \leq k < 25$. Edges are then randomly rewired with probability $p = 1/l$ to different cliques. We also generated **P**lanted **P**artition **G**raphs (**PPG**). This model partitions a graph with n nodes in $20 \leq l < 30$ groups with $15 \leq k < 25$ nodes each. Nodes of the same group are linked with a probability $p_{in} = .75$, and nodes of different groups are linked with probability $p_{out} = 10/(kl^2)$. Both of these datasets that exhibit strong community structures are generated using NETWORKX Python package [6]. The real-world datasets are (1) **Protein** dataset which contains 918 protein graphs, each of which has 100 to 500 nodes [4], (2) **Ego** dataset which contains 757 3-hop ego networks with 50 to 300 nodes extracted from the CiteSeer dataset, with nodes representing documents and edges representing citation relationships [21], and (3) **Point Cloud** with 41 simulated 3D point clouds of household objects. This dataset has about 1.4k nodes on average with

maximum of over 5k nodes. Each point is mapped to a node and edges connecting the k-nearest neighbors in Euclidean distance in 3D space are added to the graphs [16].

To partition graphs and obtain hierarchical graph structures, we applied Louvain algorithm on all of these datasets. This resulted in hierarchical graphs of depth $L = 2$ for the synthetic datasets, while for the real world graphs it produced at least 3 levels so we spliced out the intermediate levels so that all have equal depth of $L = 3$.[2] Before training the models, we follow the protocol in [12] to randomly create a 80%-20% training-testing split, with 20% of the training data reserved as the validation set.

Experimental setup: To provide a fair comparison, we closely follow the experimental setup of You et al. [25] and Liao et al. [12]. We compared the proposed model against the baseline methods including Erdos-Renyi [5], GraphVAE [23], GraphRNN & GraphRNN-S [25], and GRAN [12]. The results of the baselines are extracted from [12] for the real-world graphs while we retrained GRAN for synthetic datasets. The neural network based methods have the following structures. GraphVAE model used a 3-layer GCN encoder and an MLP decoder with 2 hidden layers where all hidden dimensions are set to 128 for all experiments. For GraphRNN and GraphRNN-S, the best settings reported in the original paper were used. GRAN enjoyed 7 layers of GNNs with one round of message passing. Hidden dimensions are set to 128 for [Ego, RCG], 256 for Point Cloud and 512 for [Protein, PPG] for GRAN, while we used smaller hidden dimensions of 64 for [Ego, RCG, Point Cloud, Protein] and 128 for PPG.

We use the same GNN architecture as GRAN, with 7 layers of GNNs with one round of message passing, but we choose smaller hidden dimensions, setting it to 64 for [Ego, RCG, Point Cloud], and 128 for [Protein, PPG]. For both GRAN and MRG, the number of mixtures is set $K = 20$ and block size and stride are both set to 1. In general, MRG models uses less parameters compared to GRAN. The comparison of total number of parameters of MRG and GRAN are listed in appendix C. MRG models are training by the Adam optimizer [9] with learning rate of 5e-4.

Table 3: Number of trainable parameters of GRAN vs MRG models.

	Protein	**3D Point Cloud**	**Ego**	**PPG**	**RCG**
GRAN	$1.75e^7$	$5.7e^6$	$1.5e^7$	$1.77e^7$	$1.??e^7$
MRG-B	$7.36e^6$	$8.09e^6$	$7.31e^6$	$1.12e^7$	$4.17e^6$
MRG	$9.06e^6$	$1.20e^7$	$8.96e^6$	$1.47e^7$	$5.94e^6$

.

[2]The proposed architecture can be trained on HGs with uneven heights by adding empty graphs at the root levels of those HGs with lower height so that they are not sampled during the training.