HIERARCHICAL MULTI-RESOLUTION GRAPH GENERATION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

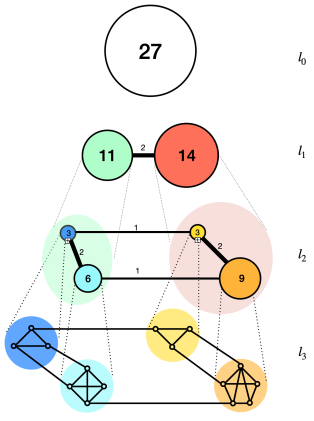
In real world domains, graphs often have natural hierarchies. However, data-driven graph generation is yet to effectively respect and exploit such structures. We propose a novel approach that recursively generates community structures at multiple resolutions, with the generated structures conforming to training data distribution at each level of the hierarchy. While the generation of a community at one level takes place sequentially, lower-level sub-structures of the community can be handled in parallel. This significantly improves the speed of both generation and learning, resulting in $\mathcal{O}(\log n)$ generative rounds. Our method is further supported by an expressive probability distribution for intermediate and leaf levels of this hierarchical model. Our method achieves the state of the art performance in graph generation in both accuracy and efficiency on many datasets.

1 INTRODUCTION

As powerful algebraic structures for representing relations, graphs are ubiquitously relevant and useful. Data-driven approaches to graph generation is both challenging and widely applicable (Dai et al., 2020). Document generation is a classical case in point (Blei et al.). For telecommunications R&D, generation of realistic and representative data network topologies is highly desirable (Onat & Stojmenovic, 2007). For recommender systems, the generation of graphs has received much attention (He et al., 2021) cite Yingxue team's work). In computer vision and virtual reality, the generation of scene graphs, e.g. for household arrangement (Manolis Savva et al., 2019; Ramakrishnan et al., 2021), is a key enabling technology (zhu). For autonomous driving R&D, generation of interactive scenarios is highly useful for simulation platforms such as CARLA (Dosovitskiy et al., 2017) and SMARTS (zho).

There are natural hierarchies in all the domains mentioned above: sections, paragraphs, sentences, and words of a document, communities in user-item graphs, areas in a room in an apartment, columns of cars and groups of pedestrians on a city block, etc. On the one hand, higher level relations, such as close two groups of pedestrians are, reflect high-level community structures. On the other hand, low-level relations and distributions, such as how dense a group of pedestrians is, also affect higher-level relations and alter structural boundaries, such as how likely the pedestrian groups are to mingle into each other. Realistic graph generation models must respect both the within-level distributions and the cross-level distributions governing hierarchical interaction. While hiearchical, multi-resolution generative models were developed for specific data types such as voice Oord et al. (2016) and molecular motifs Jin et al. (2020), these methods rely on domain-specific priors not true of 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 the interacting semantic hierarchy.

We propose to fill in this gap by capturing community structures with the cross-level hierarchical interactions. In our model, a node's representation at each level is not only a function of its community but also depends on its corresponding super-node at the higher level. This both captures the hierarchical relations and allows the generation process at the lower level to be independent of specific ordering of the super-nodes. This means that our proposed method has a high degree of scalability through parallelism. It enables a recursive tree-structured multi-level decomposition that improves over the row-by-row decomposition of GRAN (Liao et al., 2019) and allows the overall generation process for a graph of n nodes to be $\mathcal{O}(\log n)$ instead of the $\mathcal{O}(n)$ of GRAN.



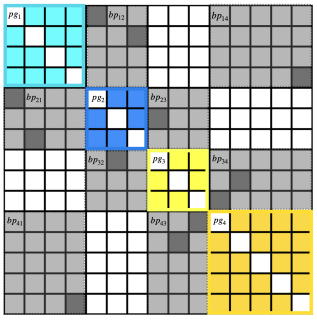


Figure 1: Hierarchical A sample hierarchical graph with 3 levels is shown in (a), communities are shown in different colors and the weight of a node and the weight of an edge in a higher level, represent the sum of the in-community connections in the corresponding community and the total weight cross-community (bipartite), respectively, in the lower level. The matrix shows its corresponding adjacency matrix where each of these sub-graphs corresponds to a block in the adjacency matrix, partition graphs are in shown different colors and bipartites are colored in gray.

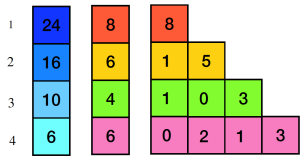


Figure 2: Decomposition of multinomial distribution as a recursive *stick-breaking* process where at each iteration, first a fraction of the remaining weights r_m is allocated to m-th row (m-th node in the sub-graph) and then this fraction v_m is distributed among that row of lower triangular adjacency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

In sum, we propose a novel graph generation method in which (1) community structures of *generic graphs* are encoded from training data and respected in generation and (2) *parallelization is maximized* through structural recursion, enabling significantly more efficient implementation in practice. Additionally, as will be explained below, (3) the generation of each new edge could utilize information about the entire graph through its relationship to a parent graph, (4) sensitivity to initial random permutation is minimized, and (5) generation of graphs with integer-valued edge weights is supported.

2 Problem Formulation

A graph $\mathcal{G} = (V,\ E)$ is defined as a set of nodes or vertices V and edges E with sizes $n = |V|$ and $m = |E|$ 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) and *bipartite graphs* that are composed of the cross link of two partition graphs. Each partition graph in level l, $pg_i^l = \big(V(pg_i^l),\ E(pg_i^l)\big)$, is aggregated to supernode that is represented by a node in the higher level, also called its parent node, $v_i^{l-1} = Pa(pg_i^l)$ with the self edge weight $w_{ii}^{l-1} = \sum_{e \in E(pg_i^l)} w_e$ that is the sum of all intra-community edges. Moreover, for each bipartite $bp_{ij}^l = \Big(V(pg_i^l),\ V(pg_j^l),\ E(bp_{ij}^l)\Big)$ an edge is added in the higher level, also called its parent edge, $e_i^{l-1} = Pa(bp_{ij}^l) =< v_i^{l-1}, v_j^{l-1} >$ with weight equal to the sum of the cross edges in the bipartite $w_{ij}^{l-1} = \sum_{e \in E(bp_{ij}^l)} w_e$. Bipartite adjacency matrix $\mathbf{A}_{XY}$ specifies cross links between partition X and Y , $|E(bp_{ij}^l)|$. This is how an abstract network is created in the higher level $\mathcal{G}^{l-1}$.

A hierarchical graph (hyper-graph) $\mathcal{HG}$ is defined by the set of graphs in all levels of abstractions , $\mathcal{HG} := \{\mathcal{G}^L, \mathcal{G}^{L-1}, ..., \mathcal{G}^0\}$, where leaf level $\mathcal{G}^l$ is the final graph that is being generated and root graph $\mathcal{G}^0$ is a single node graph. An $\mathcal{HG}$ is visualized in figure 1. This hierarchical tree structure helps modeling the short and long interactions among nodes, and also the flow of information between them, in multiple level of abstraction which is a key aspect of our generative model.

Community detection Different community detection algorithms has been proposed that try to find communities that have the most of the links within communities and a few cross communities or cluster nodes with similar features. Following our problem definition, we are interested in graph

topology-based coarsening algorithms. We use the well established Louvain algorithm (Blondel et al., 2008) that offers a fast converging heuristic algorithm for community detection and results in hierarchical communities with high modularity (a measure for overall quality of the partitioning of a graph into communities). Other community detection algorithms such as clustering technique that leverages the Laplacian spectrum to find strongly connected communities (Bruna et al., 2013) and min-cut are left for future consideration.

3 HIERARCHICAL MULTI-RESOLUTION GRAPH GENERATION

The goal is to generate a multi-resolution representation for a graph in a coarse-to-fine approach. Given a higher level graph, the graph at one level below can be specified by a conditional probability, and this may repeat until the leaf level is reached. We capture this intuition with the theorem below.

Theorem 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:*

$$\begin{aligned} p(\mathcal{G} = \mathcal{G}^L, \pi) &= p(\{\mathcal{G}^L, \mathcal{G}^{L-1}, ..., \mathcal{G}^0\}, \pi) \\ &= p(\mathcal{G}^L, \pi \mid \{\mathcal{G}^{L-1}, ..., \mathcal{G}^0\})\, p(\mathcal{G}^{L-1}, \pi \mid \{\mathcal{G}^{L-2}, ..., \mathcal{G}^0\}) \,...\, p(\mathcal{G}^1, \pi \mid \mathcal{G}^0)\, p(\mathcal{G}^0) \\ &= \prod_{l=0}^{L} p(\mathcal{G}^l, \pi \mid \mathcal{G}^{l-1}) \times p(\mathcal{G}^0) \end{aligned} \tag{1}$$

GRAPH GENERATION ACCORDING TO PARTITION

Now, the conditional of graph at level l, $p(\mathcal{G}^l \mid \mathcal{G}^{l-1})$, can be factorized according to its sub-structures of partition graphs and bipartite graphs:

$$\begin{aligned} p(\mathcal{G}^l \mid \mathcal{G}^{l-1}) &= p(\{pg_i^l \;\; \forall i \in V_{\mathcal{G}^{l-1}}\} \& \{bp_{ij}^l \;\; \forall < i, j > \in E_{\mathcal{G}^{l-1}}\} \mid \mathcal{G}^{l-1}) \\ &= \prod_{i \in V_{\mathcal{G}^{l-1}}} p(pg_i^l \mid \mathcal{G}^{l-1}) \prod_{<i,j> \in E_{\mathcal{G}^{l-1}}} p(bp_{ij}^l \mid \mathcal{G}^{l-1}, pg_i^l, pg_j^l) \end{aligned} \tag{2}$$

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

$$\log p_{\phi^l}(\mathcal{G}^l \mid \mathcal{G}^{l-1}) = \sum_{i \in V_{\mathcal{G}^{l-1}}} \log p_{\phi^l}(pg_i^l \mid \mathcal{G}^{l-1}) + \sum_{<i,j> \in E_{\mathcal{G}^{l-1}}} \log p_{\phi^l}(bp_{ij}^l \mid \mathcal{G}^{l-1}, pg_i^l, pg_j^l) \tag{3}$$

Here, we assume that the partition graphs pg_i^l are independent given the the parent graph $\mathcal{G}^{l-1}$, and also the bipartite graphs bp_{ij}^l are independent given the the parent graph $\mathcal{G}^{l-1}$ and their corresponding pair of parts (pg_i^l, pg_j^l). Therefore, given the graph at a higher level, generation of the graph at the next lower level is reduced to generation of part 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 MRG generates adjacency blocks in parallel and constitutes the final matrix topology.

Moreover, given the conditional independence of these parts, generation of the next lower level can be performed in parallel. Thus, in our proposed method, the generation decisions of the cross edges of each bipartite bp_{ij}^l may occur all at once given that its corresponding pair of parts (pg_i^l, pg_j^l) are already generated. On the other hand, similar to GRAN (Graph Recurrent Attention Network) proposed by Liao et al. (2019), generation of each partition graph is still performed one at a time; in other words, the lower triangle of adjacency matrix $\hat{\mathbf{A}}_i^l$ is completed row-by-row. This process continues until the desirable number of edges is reached. Similar to the link prediction problem in graphs, the decision function can be formulated in terms of the candidate edge representation defined as $\Delta h(< a, b >) := h_a - h_b$, which is why node representations are required in this process.

For *node representation*, we follow GRAN (Liao et al., 2019) to first obtain initial feature of the nodes and then compute node representation h_i using *GNN with attentive messages* model. We simply define it as $h_i = \mathrm{GNN}^l(\mathcal{G}_{in}; \gamma^l)$ that is a GNN parameterized by set of parameters γ^l. We choose only $R = 1$, for one round of message passing in our model.

PROBABILITY DISTRIBUTION OF CANDIDATE EDGES

The edges in a hierarchical graph has non-negative integer weights where w_{ii}^{l-1} and w_{ij}^{l-1} are the sum of all the edges in partition graph pg_i^l and bipartite graph bp_{ij}^l. Therefore, the probability of $\mathcal{E}(bp_{ij}^l)$, the set of all candidate edges of bipartite bp_{ij}^l, can be modeled as a multinomial distribution as

$$\mathbf{w} = \mathcal{E}(bp_{ij}^l) \sim \text{Mu}(\mathbf{w} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij}^l) = \frac{n!}{\prod_{e=1}^{|\mathcal{E}(bp_{ij}^l)|} \mathbf{w}_e!} \prod_{e=1}^{|\mathcal{E}(bp_{ij}^l)|} (\theta_{ij,e}^l)^{\mathbf{w}_e} \tag{4}$$

where the multinomial coefficient $\frac{n!}{\prod \mathbf{w}_e!}$ is the number of ways to divide the total weigh $w_{ij}^{l-1} = \sum_{e=1}^{|\mathcal{E}(bp_{ij}^l)|} \mathbf{w}_e$ into $|\mathcal{E}(bp_{ij}^l)|$ candidate edges of bp_{ij}^l with corresponding probability vector $\boldsymbol{\theta}_{ij}^l$ such that $\sum \boldsymbol{\theta}_{ij,e}^l = 1$.

Likewise, for each partition graph, the probability distribution of the set of all candidate edges can be modeled by multinomial distribution but as the partition graph generation happens in a node-by-node sequential manner we need to specify the 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:*

$$\textit{Mu}(\mathrm{w}_1, \ ..., \mathrm{w}_E \mid w, \ [\theta_1, \ ..., \theta_E]) = \prod_{e=1}^{E} \textit{Bi}(\mathrm{w}_e \mid w - \sum\nolimits_{i<e} \mathrm{w}_i, \hat{\theta}_e), \tag{5}$$

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

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 (Linderman et al., 2015).

The above lemma allows the generation of a partition graph to be implemented recursively as a sequence of edge-by-edge generation that is analogous to GraphRNN (You et al., 2018) with $\mathcal{O}(|V_{pg}|^2)$. However, as stated above, our goal is to generate the graph node-by-node but predict the edges in a group-wise manner. For that we need the following theorem.

Theorem 2 *For a random vector $\mathbf{w} \in \mathbb{Z}_+^E$ with multinomial distribution $\textit{Mu}(\mathbf{w} \mid w, \boldsymbol{\theta})$, we 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 we define sum of all weights in m-th group by a random variable $\mathrm{v}_m := \sum_{e=1}^{E_m} \mathbf{u}_{m,e}$ and also we split the probability vector as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ ..., \boldsymbol{\theta}_M]$. Then the multinomial distribution can be modeled as a chain of binomials and multinomials*

$$\textit{Mu}(\mathbf{w} = [\mathbf{u}_1, \ ..., \mathbf{u}_M] \mid w, \boldsymbol{\theta} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M]) = \prod_{m=1}^{M} \textit{Bi}(\mathrm{v}_m \mid w - \sum_{i<m} \mathrm{v}_i, \eta_{\mathrm{v}_m}) \ \textit{Mu}(\mathbf{u}_m \mid \mathrm{v}_m, \boldsymbol{\theta}_{\mathbf{u}_m}), \tag{6}$$

$$\eta_{\mathrm{v}_m} = \frac{\mathbf{1}^T \boldsymbol{\theta}_m}{1 - \sum_{i<m} \mathbf{1}^T \boldsymbol{\theta}_i}, \ \ \boldsymbol{\theta}_{\mathbf{u}_m} = \frac{\boldsymbol{\theta}_m}{\mathbf{1}^T \boldsymbol{\theta}_m}$$

where, the probability of binomial, η_{v_m}, is the fraction of the remaining probability mass that is allocated to v_m, the sum of all weights in the m-th group, and the probability vector $\boldsymbol{\theta}_{\mathbf{u}_m}$ is the normalized multinomial probabilities of each element in its own 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 and then this fraction is distributed among that group's members.

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

Therefore, at step m of generating the partition graph, we characterize the generative probability of the group of candidate edges $\mathcal{E}(pg_{i,m}^l)$ corresponding to node $v_m(pg_i^l)$, *i.e.* the m-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 illustrated figure 2. We further extend the generative probability to a mixture model to increase model expressiveness:

$$p(\mathbf{u}_m := \mathcal{E}(pg_{i,m}^l)) = \sum_{k=1}^{K} \beta_{i,m,k} \mathbf{Bi}(\mathrm{v}_m \mid w_{ii}^{l-1} - \sum_{i<m} \mathrm{v}_i, \eta_{\mathrm{v}_m,k}) \ \mathbf{Mu}(\mathbf{u}_m \mid \mathrm{v}_m, \boldsymbol{\theta}_{\mathbf{u}_m,k}) \tag{7}$$

$$\boldsymbol{\theta}_{\mathbf{u}_m,k} = \mathrm{softmax}\left(\{\mathrm{MLP}_{\boldsymbol{\theta}_\mathbf{u}}^l([\Delta h_e, h_{Pa(pg_i^l)}]) \mid \forall e \in \mathcal{E}(pg_{i,m}^l)\}\right)[k,:] \tag{8}$$

$$\eta_{\mathrm{v}_m,k} = \mathrm{sigmoid}\left(\mathrm{MLP}_{\eta_\mathrm{v}}^l([\mathrm{addpool}(\{h_v \mid \forall v \in V(pg_{i,m}^l)\}), h_{Pa(pg_i^l)}])\right)[k] \tag{9}$$

$$[\beta_{ij,1}^l, \ ..., \beta_{ij,K}^l] = \mathrm{softmax}\left(\mathrm{MLP}_\beta^l([\mathrm{addpool}(\{h_v \mid \forall v \in V(pg_{i,m}^l)\}), h_{Pa(pg_i^l)}])\right) \tag{10}$$

Where, $\mathrm{MLP}_{\boldsymbol{\theta}_\mathbf{u}}^l()$ acts at edge level and results in $K \times |\mathcal{E}(pg_{i,m}^l)|$ dimensional output, while graph level representations, obtained by an add pooling aggregation function, are fed into $\{\mathrm{MLP}_{\eta_\mathrm{v}}^l(), \ \mathrm{MLP}_\beta^l()\}$ that output K dimensional arrays. All of the MLP models, we have used two hidden layer with ReLU activation functions.

For bipartite graph bp_{ij}^l, we employ the mixture of multinomial distribution (4) to characterize the generative probability:

$$p(\mathbf{w} := \mathcal{E}(bp_{ij}^l)) = \sum_{k=1}^{K} \beta_{ij,k} \mathbf{Mu}(\mathbf{w} \mid w_{ij}^{l-1}, \boldsymbol{\theta}_{ij,k}^l) \tag{11}$$

$$\boldsymbol{\theta}_{ij,k}^l = \mathrm{softmax}\left(\left\{\mathrm{MLP}_{\boldsymbol{\theta}}^l([\Delta h_e, \Delta h_{Pa(bp_{ij}^l)}]) \mid \ \forall e \in \mathcal{E}(bp_{ij}^l)\right\}\right)[k,:] \tag{12}$$

$$[\beta_{ij,1}^l, \ ..., \beta_{ij,K}^l] = \mathrm{softmax}\left(\mathrm{MLP}_\beta^l\big(\mathrm{addpool}(\{\Delta h_e \mid \forall e \in \mathcal{E}(bp_{ij}^l)\}), \Delta h_{Pa(bp_{ij}^l)}]\big)\right) \tag{13}$$

In order to encode the global structure of the graph while generating a local component, we use the node representation of parent node $h_{Pa(pg_i^l)}$, the node that represent the pg_i^l at the parent level, for each partition graph pg_i^l and the edge representation of parent edge $\Delta h_{Pa(bp_{ij}^l)}$, the edge that represent the bp_{ij}^l at the parent level, for each bipartite graph bp_{ij}^l. Such extra information is concatenated to the candidate edge representation to provide richer edge state that captures long range interactions.

As in our experiments the graphs have binary edges weights, we use multi-hot activation function defined as

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

instead of the standard softmax function to obtain the probability of multinomials in the leaf level of the model. We also run experiments with the mixture of Bernoulli for the last layer using sigmoid activation for the output.

Remark Training and generation of the proposed hierarchical model is highly parallelizable and require $\mathcal{O}(c \log n)$ sequential steps where c the size of largest graph parts.

Remark: 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 ajcanent nodes are not necessary sharing similar properties, 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.

Remark 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 the set of distinctive adjacency matrices. Therefore, it is significantly less sensitive to node ordering compared to the available models.

4 RELATED WORK

Graph generative models has been studied extensively. Classical methods from Erdos & Rényi (1960) and Barabási & Albert (1999) are based on random graph theory that can only capture a set of hand engineered graph statistics. Leskovec et al. (2010) 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.

With recent progress in GNN (graph neural networks), several deep neural network models have been introduced (De Cao & Kipf, 2018; Simonovsky & Komodakis, 2018; Kipf & Welling, 2016; Ma et al., 2018; Liu et al., 2019) that are base on variational autoencoders (Kingma & Welling, 2013), But these methods are weak in capturing the complex dependencies in graph structures and thus quality of graph generation degrades as graphs become moderate or large in size.

Auto-regressive deep architectures, on the other hand, model graph generation as a sequential decision making process. Li et al. (2018) proposed generative model based on GNN but it has high complexity of $\mathcal{O}(mn^2)$. GraphRNN (You et al., 2018) models graph generation with a two-stage RNN architecture, with the first RNN generating new nodes and the second generating links of the new nodes. It thus has to traverses all elements of the adjacency matrix in a predefined order, resulting in $\mathcal{O}(n^2)$ and not scalable to large graphs. On the other hand, GRAN (Liao et al., 2019) uses graph attention networks and improves the complexity by generating the adjacency matrix in a row-by-row fashion, *i.e.* generating all the edges between the new node and existing graph in one step, resulting in $\mathcal{O}(n)$ recursive computation. In an attempt to improve the scalability of generative models for graph, Dai et al. (2020) proposed an algorithm for sparse graphs that reduce the training complexity to $\mathcal{O}(\log n)$ while its generation time is increased to $\mathcal{O}((n+m)\log n)$ complexity and its recursive generation process does not incorporate community structure of the graph.

In explicitly dealing with hierarchical structures, Jin et al. (2020) proposed a generation method for molecular graphs that recursively selects motifs, the basic building blocks, from a set and predicts the attachment of that motif to emerging molecule.This model require prior domain-specific knowledge and relies on molecule-specific graph motifs. Moreover, graphs are abstracted in only two levels and component generation cannot be performed in parallel. A hierarchical normalizing flow model for molecular graphs is proposed in De Cao & Kipf (2018) that generates new molecules from a single node by recursively dividing every node into two nodes. Merging and splitting of pair of nodes in this model is based on the the node's neighborhood so it does not include the diverse community structure of the graphs and hence its hierarchical generation is structurally limited.

Overall, our method belongs to the GNN family. As explained in Section 3, it extends GRAN (Liao et al., 2019) with a general treatment, supported by expressive distributions, of the hierarchical structures so far only exploited in an ad hoc (Jin et al., 2020) or limited De Cao & Kipf (2018) way.

5 EXPERIMENTS

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

Datasets First, we generated relaxed Caveman graph **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 link different cliques. We also generated planted partition graph **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 are generated with NETWORKX Python package.

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 (Dobson & Doig, 2003), (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 (Sen et al., 2008), and (3) **Point Cloud** with 41 simulated 3D point clouds of household objects (denoted as FirstMM-DB) with on average over 1k nodes and maximum of over

Table 1: In this table MMD of graph Deg.: degree distributions, Clus.: clustering coefficient , Orbit: 4-node orbits, and the Spec.:spectra of the graph Laplacian. For all the metrics, the smaller the better. Given $Inf := (|V|_{max}, |V|_{avg}, |E|_{max}, |E|_{avg})$, $Inf_{Protein} \approx (500,\ 258, 1575, 646)$, $Inf_{Ego} \approx (399,\ 144, 1062, 332)$, $Inf_{3D-point-cloud} \approx (5037,\ 1377, 10886, 3k)$, $Inf_{PPG} \approx (696,\ 477, 7484, 4.37k)$, $Inf_{RCG} \approx (576, 261, 6624, 2.2k)$.

	Protein				**3D Point Cloud**				**Ego**				**PPG**				**RCG**			
	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	$9.13e^{-2}$	0.31	1.22	1.27	$4.26e^{-2}$	0.16	0.94	0.85	0.18	0.283	1.04	0.194	0.201	0.17	0.8	0.12	0.2
GraphVAE*	0.48	$7.14e^{-2}$	0.74	0.11	-	-	-	-												
GraphRNN-S	$4.02e^{-2}$	$4.79e^{-2}$	0.23	0.21	-	-	-	-												
GraphRNN	$1.06e^{-2}$	0.14	0.88	$1.88e^{-2}$	-	-	-	-												
GRAN	$1.98e^{-3}$	$4.86e^{-2}$	0.13	$5.13e^{-3}$	$1.75e^{-2}$	0.51	0.21	$7.45e^{-3}$	$3.2e^{-2}$	0.17	$2.6e^{-2}$	$4.6e^{-2}$	0.0567	0.23	0.282	0.0171	0.075	0.0134	0.0995	0.057
MRG-B	$8.9e^{-3}$	$7.85e^{-2}$	0.0108	0.0102	0.129	0.345	0.059	0.0089	$4.1e^{-3}$	0.062	$1.8e^{-2}$	$1.42e^{-2}$	0.00479	0.0879	0.048	0.00185	0.0145	$1.29e^{-2}$	0.0275	0.00715
MRG	0.00649	0.224	0.0578	0.0131	0.207	0.806	0.0275	0.0224	$1.78e^{-2}$	0.224	$1.16e^{-2}$	$2.07e^{-2}$	$1.51e^{-1}$	0.368	0.00775	0.0193	0.0445	0.026	0.0115	0.0576

5k nodes for the points and with edges connecting the k-nearest neighbors in Euclidean distance in 3D space (Neumann et al., 2013).

The Louvain algorithm is applied on all of these dataset, which results in hierarchical graphs of depth $L = 2$ for the synthetic datasets, while for the real-world graphs it produces at least 3 levels so we spliced out the intermediate levels so that all have equal depth of $L = 3$.[1] In using each of these five datasets, we follow the protocol in Liao et al. (2019) 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. (2018) and Liao et al. (2019). We compare the graph generation quality of the proposed model against Erdos-Renyi (Erdos & Rényi, 1960), GraphVAE (Simonovsky & Komodakis, 2018), GraphRNN & GraphRNN-S (You et al., 2018), and GRAN Liao et al. (2019). The results of the baselines are grabbed from Liao et al. (2019) for the real-world graphs and the setting and we retrained GRAN for synthetic datasets. GraphVAE model had a 3-layer GCN encoder, 2 hidden layers decoder MLP decoder with 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 Block size and stride are both set to 1 and hidden dimensions are set to 128 for {Ego, RCG} 256 for {Point Cloud} and 512 for {Protein, PPG}. The number of Bernoulli mixtures is set to 20 for GRAN and MRG.

In the proposed model, each level has its own GNN and output models, which are indexed by the level index of our model definition in section 3. MRG and MRG-B denote the proposed multi-resolution model respective with mixture of multinomial distribution and with mixture of Bernouli distribution at the leaf level. We use the same GNN articteture as GRAN but we choose smaller hidden dimensions, 64 for {Ego, RCG, Point Cloud}, and 128 for {Protein, PPG}. Adam optimizer Kingma & Ba (2014) with learning rate of 5e-4 was used and for the generation evaluation the best model is chosen based on the maximum mean discrepancy MMD score between validation set and generated graphs.

For evaluation of generation, we follow the approach in Liao et al. (2019) and measure the MMD over 4 graph statistics: (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.[2]

The quality of generated graphs measured by MMD of the 4 graph statistics on the test set is reported in Table 1 and some graphs sampled from the model distribution are presented in Figure 5. More generated graphs sampled with hierarchical graph structure are presented in appendix A.

[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.

[2]MMD depends on Gaussian kernels with the first Wassertein distance. We follow Liao et al. (2019) in using total variation (TV) distance, which is consistent with Wasserstein distance, to speeds up the evaluation. New evaluation metrics for comparing graph sets are (O'Bray et al., 2021; Thompson et al., 2022) but we choose to comply with the experimental setup in GRAN.

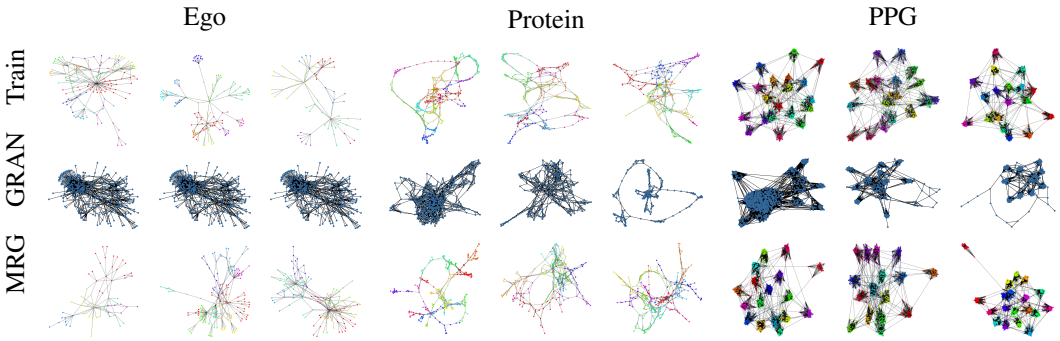


Figure 3: Sample graphs generated by different models.

Table 2: Comparison of models with different number of levels and shared parameters.

	Ego			
	Deg.	Clus.	Orbit	Spec.
MRG-B	$4.1e^{-3}$	0.062	$1.8e^{-2}$	$1.42e^{-2}$
MRG-B 2-level	$4.73e^{-3}$	$5.43e^{-2}$	$1.41e^{-2}$	$1.9e^{-2}$
MRG-B shared	$1.87e^{-2}$	0.368	$3.20e^{-2}$	$3.16e^{-2}$

5.1 Ablation studies

In this section we perform two ablation studies to evaluate more compact form of the MRG. First, we asses the performance MRG with fewer levels by splicing out the middle level of Ego dataset so that their HGs have 2 levels after root ($L = 2$). We can see from the results in table 2 that the generation quality of the models drops slightly once we decrease the number of levels which, therefore it shows that having more number of hierarchical levels improves the expressiveness of the model.

Moreover, we train the MRG with shared parameters across levels so that all levels use similar GNN and output models. The comparison in Table 2 show that using individual models for each level offers better results which can be explained by the that graph at different levels have have different characteristics such as graph sparsity.

6 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 with $\mathcal{O}(\log n)$ complexity for a graph with n nodes, insofar as the graph is amendable 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 that what we used in this work that introduces the new method will be an informative and worthy undertaking.

References

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

David M. Blei, Andrew Y. Ng, and Michael I. Jordan.

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.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

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.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

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

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

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

Yue He, Yancheng Dong, Peng Cui, Yuhang Jiao, Xiaowei Wang, Ji Liu, and Philip S. Yu. Purify and generate: Learning faithful item-to-item graph from noisy user-item interaction behaviors. New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325.

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.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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.

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.

Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *arXiv preprint arXiv:1809.02630*, 2018.

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.

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.

Furuzan Atay Onat and Ivan Stojmenovic. Generating random graphs for wireless actuator networks. In *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–12, 2007. doi: 10.1109/WOWMOM.2007.4351712.

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.

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.

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.

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

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.

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