Heterogeneous Graph Generation: A Hierarchical Approach using Node Feature Pooling

Hritaban Ghosh*, Changyu Chen†, Arunesh Sinha‡, Shamik Sural*
*Indian Institute of Technology Kharagpur, India
†Singapore Management University, Singapore
‡Rutgers University, USA
Email: ghoshhritaban21@gmail.com, cychen.2020@phdcs.smu.edu.sg, arunesh.sinha@rutgers.edu, shamik@cse.iitkgp.ac.in

***Abstract*—Heterogeneous graphs can be used to model systems in various domains like social networks, recommendation systems and biological networks. Unlike their homogeneous counterpart, heterogeneous graphs consist of multiple types of nodes and edges, each representing different entities and relationships. Generating realistic heterogeneous graphs that capture the complex interactions among diverse entities is a difficult task, primarily because the generator has to capture both the node type distribution and the feature distribution for each node type. In this paper, we address the challenges in heterogeneous graph generation by employing a two phase hierarchical approach called HG2NP (Heterogeneous Graph Generation using Node Feature Pooling). The first phase creates a skeleton graph with node types using an existing diffusion based model. In the second phase, we employ an encoder and a sampler structure as generator to assign node type specific features to the nodes. A discriminator is used to guide the training of the generator while feature vectors are sampled from a node feature pool. We conduct extensive experiments with the well-known IMDB and DBLP datasets to show the effectiveness of our method. The need for various architectural components is established through ablation studies.**

***Index Terms*—Heterogeneous graph generation, Node pooling, Type degree distribution, Permutation equivariance and invariance**

I. Introduction

Heterogeneous graphs appear ubiquitously in several domains including social networks, recommendation systems, and biological networks. Such graphs, as opposed to homogeneous ones, have multiple node types and sometimes multiple edge types. They allow for the modeling of diverse entities and their interconnections, providing a richer and more comprehensive view of the underlying system like citation networks [1], social networks [2] and biological networks [3]–[5]. In a social network, for example, an individual can be part of a group while groups can be part of a community. Even in this simplified description of a social network, several node types can be identified, e.g., *individual*, *group* and *community*. In this work, we focus on generating such heterogeneous graphs that in turn can improve the accuracy and reliability of predictive analytics, anomaly detection and risk assessment systems built from synthetic data. It can further lead to improved performance in various tasks like recommendation [6], classification [7] and link prediction [2].

The level of detail and meaningful relationships captured by heterogeneous graphs cannot be achieved by homogeneous graphs without several assumptions and constraints. For example, for modeling node types in social networks using a homogeneous graph, one would have to set aside components in the node feature vector to indicate the discrete node type. This presents a difficulty when the node features are real valued as we mix real valued and discrete valued features in a single vector. To add to the complexity, node features may depend on the type. For example, if a feature vector for *individual* type node is encoded in i-bits, a feature vector for *group* type in j-bits and a feature vector for *community* type in k-bits, then the node feature vector in the homogeneous graph must be at least $\max(i, j, k)$-bits. Zeroes would fill up the unused bits for shorter feature vectors. Such an approach leads to wastage of resources and, therefore, it is imperative that more advanced data structures such as heterogeneous graphs are used.

In recent years, there has been a tremendous effort in the field of homogeneous graph generation [8]–[14]. In the area of graph neural networks [15], heterogeneous graphs such as IMDB and DBLP have been investigated. However, to the best of our knowledge, there is only one prior work in heterogeneous graph generation [16], which has certain shortcomings (see the *Related Work and Background* section). In this paper, we introduce a novel hierarchical approach called Heterogeneous Graph Generation using Node Feature Pooling (HG2NP) which is a significant development over homogeneous graph generation techniques.

Problem Statement: We denote a heterogeneous graph by $G = (V, E)$, where V is the set of nodes/vertices and E is the set of edges. Given a set of M observed heterogeneous graphs $\{G_i\}_{i=1}^{M}$, the heterogeneous graph generation problem is to learn the probability distribution $\mathbb{P}$ of these graphs from which new graphs can be sampled $G \sim \mathbb{P}$.

Our *first contribution* is to divide the problem into two simpler tasks, accomplished in two phases. In the first phase, a skeleton graph with node types is generated, while in the second phase, we assign features to the nodes in the skeleton graph. As a *second contribution*, we make use of message passing (MP) and a node feature vector pool to design an adversarial generator-discriminator architecture for the second phase. The MP module provides neighborhood information that is crucial for generating features of nodes. The node features are sampled from a pool of feature vectors constructed

from all the data, as we find that generation of real valued large dimensional vectors with no restriction produces poor quality outcomes. Our *third contribution* is the concept of labeled permutation equivariance and invariance, which are relevant for graphs with node types, along with a proof showing that HG2NP possesses these properties. Finally, we perform comprehensive experiments with IMDB and DBLP graph data sets to show the superiority of our method compared to the baselines.

To the best of our knowledge, HG2NP is the first ever work on multi-input heterogeneous graph generation.

II. Related Work and Background

Graph generation holds immense practical value, with applications ranging from the design of molecular structures in drug development [12], [17]–[19] to the optimization of model architectures through generative models for computation graphs [20]. In [21], the authors survey deep graph generation techniques and their practical applications. The authors of [8] highlight the formal definitions and taxonomies of several deep generative models for homogeneous graph generation including but not limited to those in [9], [10], [11], [22], [23], [24], [25], [12], [26], [27], [28], [29], [30] and [31]. The work also provides an overview of the evaluation metrics in this specific domain. In [32], the authors introduce the Variational Graph Auto-Encoder (VGAE) - a framework for unsupervised learning on graph-structured data based on the Variational Auto-Encoder (VAE) as originally introduced in [13]. In recent years, alongside the aforementioned approaches, diffusion-based methods have gained prominence, with several studies conducted in this area [14], [33], [34]. The DiGress model [14] utilizes a discrete diffusion process that progressively edits graphs with noise through the process of adding or removing edges and changing the categories. A graph transformer network is trained to revert this process. This model is used as a skeleton graph generator in the first phase of our approach.

As opposed to the vast majority of graph generation papers listed above and in surveys, [35] proposed a generative approach called NetGAN that was trained only on one input graph. This different paradigm aims to generate graphs that match properties of the single graph used for training, thus, there is no learning of distribution over graphs. The only work that we know of for heterogeneous graph generation [16] is also based on the single graph training paradigm. They present a heterogeneous path generator designed to produce path instances. The paths are a sequence of node types and edge types, effectively encapsulating semantic information between their endpoints. Additionally, they propose a heterogeneous graph assembler capable of stitching the generated path instances into one heterogeneous graph in a stratified manner. Expanding on their own work, [36] extends the methodology by incorporating a graph motif generator. It aims to enhance the characterization of higher-order structural distributions within heterogeneous graphs. The significantly different framework and metrics for this work makes it inapplicable to our problem of learning the distribution of graphs and then generating from the learned distribution. Additionally, as the code from this paper is not reproducible for new datasets, we were unable to even adapt the work for comparison.

Message Passing Background: Message passing in a graph allows to effectively incorporate connectivity information between nodes. Consider a graph $G = (V, E)$ with initial node features x_u for node $u \in V$ and edge features $y_{e_{uv}} \in E$ (edge feature might be empty). Let $N(u)$ denote the neighbor nodes of u. Message passing is defined by an operation $h_u = U(x_u, \bigoplus_{v \in N(u)} M(x_u, x_v, y_{e_{uv}}))$ that updates the current node features x_u to h_u and is done repeatedly for all nodes. Here, U is an update function, $\bigoplus$ is an aggregation function and M is a message function (builds message from v to u). There are many variations of these functions in the literature. We use a specific popular one called graph attention networks [37] which builds these functions based on attention mechanism.

DiGress Background: DiGress [14] is a discrete denoising diffusion model for generating homogeneous graphs with categorical node and edge attributes. There are three types of discrete noises in the diffusion model, namely, categorical edits in which the type of a node is changed, edge additions in which a new edge is added, and edge removals in which an existing edge is removed. A graph transformer network is trained to revert this process. For any node, the transition probabilities are defined by the matrices $[Q^t_X]_{ij} = q(x_t = j|x^{t-1} = i)$ and similarly for edges, they are defined as $[Q^t_E]_{ij} = q(e_t = j|e^{t-1} = i)$. Furthermore, adding noise to a current graph to form $G^t = (X^t, E^t)$, implies sampling each node type and edge types from a categorical distribution given by $q(G^t|G^{t-1}) = (X^{t-1}Q^t_X, E^{t-1}Q^t_E)$. Extending this over t timesteps, the following is obtained: $q(G^t|G) = (X^{t-1}\bar{Q}^t_X, E^{t-1}\bar{Q}^t_E)$ where $\bar{Q}^t_X = Q^1_X ... Q^t_X$ and $\bar{Q}^t_E = Q^1_E ... Q^t_E$. Lastly, the denoising neural network ϕ_θ takes a noisy graph $G^t = (X^t, E^t)$ as input and predicts a clean graph G. Once the network is trained, it can be used to sample new graphs. We use a fine-tuned DiGress model on our datasets to obtain skeleton graphs to be processed further in the second phase of HG2NP.

III. Methodology

In this section, we describe our approach HG2NP in detail. First, we note that as articulated in our problem statement in the Introduction section, we aim to generate node types but not explicit edge types. This does not mean that we cannot express edge types, since it is possible to implicitly define an edge type by the node types of its endpoints in many scenarios. In fact, it suffices for the edge relationships found in the standard DBLP and IMDB data available from PyTorch Geometric [38]. Nonetheless, in case of richer edge labels, which cannot be defined by the type of endpoints, one can always convert such a graph into one with node types only by introducing a node n_e for each labeled edge e (edge $e = (v, u)$ is replaced by unlabeled edges (v, n_e) and (n_e, u)), and assigning the edge

label of e to this new node n_e, albeit at the cost of a larger graph.

HG2NP is a two phase scheme, in which the first phase leverages an existing state-of-the-art homogeneous graph generation framework to output a *skeleton graph* with nodes and edges, and node type. Then, in the second phase, we assign feature vectors to the nodes. The second phase is set up as a generative adversarial network. An overview is shown in Figure 1. Details of each of the phases are provided in the following sub-sections.

A. Phase 1: Skeleton Graph Generation

We leverage DiGress [14], which is designed for generating homogeneous graphs. It is a diffusion based generative model which is utilized to learn the distribution patterns of node type underlying the heterogeneous graphs in our datasets. We extract real skeleton graphs from the dataset for training of the generator by keeping only the node, node type and edge information. By breaking down the complex task of heterogeneous graph generation into two distinct phases, we intend to employ effective models for the more manageable tasks in each phase.

B. Phase 2: Heterogeneous Feature Vector Assignment

Our heterogeneous node feature vector assignment step is based on the adversarial generator-discriminator architecture, conditional on the skeleton graph generated in Phase 1.

Multi-type Multi-generator: Given a skeleton graph S with node type vectors x_u from Phase 1, in the second phase, we first utilize a message passing process for updating each node's type vector to obtain a graph $U(S)$ with updated node type vectors h_u for each node u. We utilize graph attention based message passing developed in an existing work [37]. Based on the intuition of message passing, the updated node type vector captures the neighborhood node type information. A copy of the original node type vector is also kept to be used later as described next. Let K be the total number of node types and k be the index to represent a node type. There is a separate conditional generator $g_k(x_u, h_u; \theta_k)$ for each node type k. From the original node type vector, we choose the corresponding g_k and then, conditioned on the updated node type vector for this node, we sample a feature vector from a node type specific pool of node feature vectors. The conditional generators are implemented using a simple fully connected neural network with weights θ_k having Leaky ReLU activation and a softmax at the end followed by sampling of features. Note that the softmax is over the node pool of feature vectors of each type, which is why the network is different for each node type. In this way, all the nodes of the graph are assigned feature vectors. This yields a generated heterogeneous graph. The full generation (including Phase 1) is shown by an example in Figure 2. The intuition behind using a pool of node feature vectors is based on our preliminary empirical study which revealed that learning to generate node feature vectors happens very slowly and always results in poor quality node features. We verified this with an ablation study.

Formally, we let $\theta = \{\theta_k\}_{k=1}^{K}$ be the weights of all the generators and the final generated graph be $g_\theta(U(S))$, where g_θ is a function that takes $U(S)$ and runs $g_k(x_u, h_u; \theta_k)$ for each node u of type k to get the generated heterogeneous graph. Note that we will need to differentiate g_θ, which in turn requires differentiating through $g_k(\cdot, \cdot; \theta_k)$ and U. The MP update U is differentiable by construction, but $g_k(\cdot, \cdot; \theta_k)$ is not, since it involves a sampling step from the softmax output. To handle this, we use the well known Gumbel softmax trick from the literature [39], which approximates non-differentiable argmax by the differentiable softmax in the backward gradient pass to enable passing back gradients through the sampling operation. Thus, we consider $g_\theta(U(S))$ to be differentiable with respect to θ.

Discriminator: The generated heterogeneous graphs and the real heterogeneous graphs from the dataset are used to train a discriminator, which also employs an MP module to obtain an embedding for the heterogeneous graph. In particular, we use a graph attention based MP module that is specifically designed to work with heterogeneous graphs [15]. The MP module itself reduces the dimension of node vectors of different node types to a single latent dimension. Then, the node vectors of each type are averaged independently and the averaged vectors are concatenated in a given order of types to form an embedding of the full heterogeneous graph. Let this averaging and concatenating process be denoted as the function F. The latent vector is passed through a simple fully connected neural network with Leaky ReLU activation and a softmax at the end that outputs the probability of the graph being real. Formally, given a graph G, the heterogeneous MP module produces an updated graph $U_h(G)$ which is processed to obtain a single vector $F(U_h(G))$ that forms the input to discriminator D_w producing $D_w(F(U_h(G))) \in [0, 1]$ as the prediction for fake (0) or real (1). Note that U_h and F are differentiable by construction.

Loss Functions: For the discriminator, since the task is a standard classification step, the loss used is binary cross entropy loss. Also, the generator loss to minimize with respect to θ is given by $\log(1-D_w(F(U_h(g_\theta(U(S))))))$. Recall that we have used the Gumbel softmax trick to make g_θ differentiable.

C. Type Degree Distribution Metric

In this paper, we define a new evaluation metric specific for heterogeneous graphs. In homogeneous graph generation, a popular metric used is degree distribution MMD. This metric is computed between two sets of graphs, a real set and a generated set, by first determining the degree distribution (a histogram of degrees normalized to 1) for every graph in two sets of graphs. Then, the Maximum Mean Discrepancy (MMD) is computed using these degree distributions (see [40] for details). We modify this to work with node types. Essentially, for every node, we define a type i degree as the number of nodes of type i that this node is connected to. Next, we obtain a type degree distribution for each of the K types, followed by a type degree distribution MMD (given graph samples) for each type. Finally, we average the obtained K

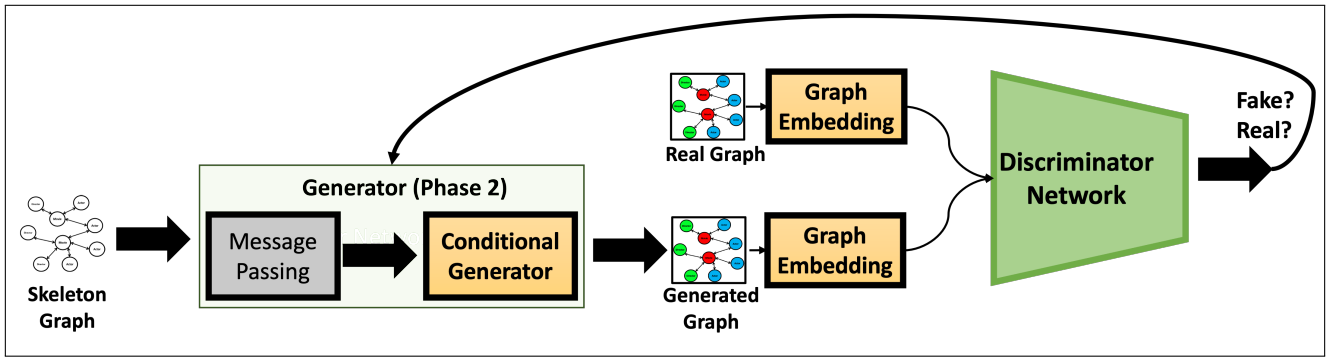


Fig. 1: Overall generation process for HG2NP. The skeleton graph is the output of Phase 1. Phase 2 uses a GAN architecture for training (see details in text and Figure 2).

type degree distribution MMDs (i.e., add up, and divide by K) to get the type degree distribution. The metric is described in more detail in Section IV-B.

D. Permutation Equivariance and Invariance

In graph processing, neural network architectures which produce outputs that are equivariant (in case of output per node) or invariant to isomorphic input graphs are desired, as they remove the need for data augmentation and generally make training simpler. Isomorphism is defined by edge preserving permutations of nodes [41]. Hence, the neural networks that satisfy these properties are called permutation equivariant and permutation invariant. In our work, there are three parts of the architecture: Phase 1 of generator, Phase 2 of generator and the discriminator in Phase 2. Phase 1 generator is DiGress, which has been shown to be permutation equivariant and permutation invariant [14].

For the networks in Phase 2, which handles heterogeneous graphs with node types, we use the notion of isomorphism for labeled graphs [41]. Here, we treat the type of node as the label for the node. Briefly, two labeled graphs are isomorphic iff there exists a bijection f between nodes such that it is (1) edge preserving: (u, v) is an edge iff $(f(u), f(v))$ is an edge and (2) label preserving: u and $f(u)$ have the same label. Hence, we use the term *labeled permutation equivariance or invariance* to denote output equivariance or invariance to isomorphic labeled input graphs. Note that any network that is permutation equivariance or invariance is also labeled permutation equivariance or invariance, as labeled permutation is stricter than permutation because labeled permutation requires preserving labels in addition to preserving edges.

Lemma 1. *Phase 2 generator is labeled permutation equivariant.*

Proof. Consider two graphs, one of which is labeled permuted using function f. The graph attention message passing is permutation equivariant, hence labeled permutation equivariance. This means that the node features x_u after Phase 2, part 1 is the same as $x_{f(u)}$ for permuted node $f(u)$. Also, the label for u and $f(u)$ are the same, say l, since we consider labeled permutation. Thus, the input to the conditional generator of type l is same $x_u = x_{f(u)}$. Then, the probability distribution over node features for node u and node $f(u)$ in permuted graph will also be the same. Thus, the Phase 2 generator is labeled permutation equivariant. □

TABLE I: DBLP graph description

Node type	Representation	Size of feature vector
author	Bag-Of-Words of paper keywords	334
paper	Bag-Of-Words of paper titles	4231
term	GloVe vector	50

Lemma 2. *Phase 2 discriminator is labeled permutation invariant.*

Proof. Consider two graphs, one of which is labeled permuted using function f. Heterogeneous graph attention (HGT) [15] message passing has operations that are specific to the node type (of the nodes involved). Here, we do not consider edge types since our problem has only node types. Thus, we treat all edge types in [15] as just one type (i.e., edge type specific weight matrices are all the same matrix). The labeled permutation also maps the label as is. Hence, all the computed attention weights and the messages are the same for a target node u and $f(u)$ in the permuted graph. The aggregate operation in HGT is just a sum which is permutation invariant. Thus, the updated node vector after one round of HGT is same for all u and $f(u)$ in permuted graph. Hence, HGT is permutation equivariant when graphs have node types only. After HGT, our operation of averaging node vectors over types (labels) and concatenating them (in a fixed order of types) is invariant to isomorphic labeled graphs. Thus, the input to the classifier discriminator neural network is same for two labeled permuted graphs and hence the output is also the same. □

IV. Experimental Results

In this section, we present the results of a detailed set of experiments conducted on multiple datasets using different performance metrics.

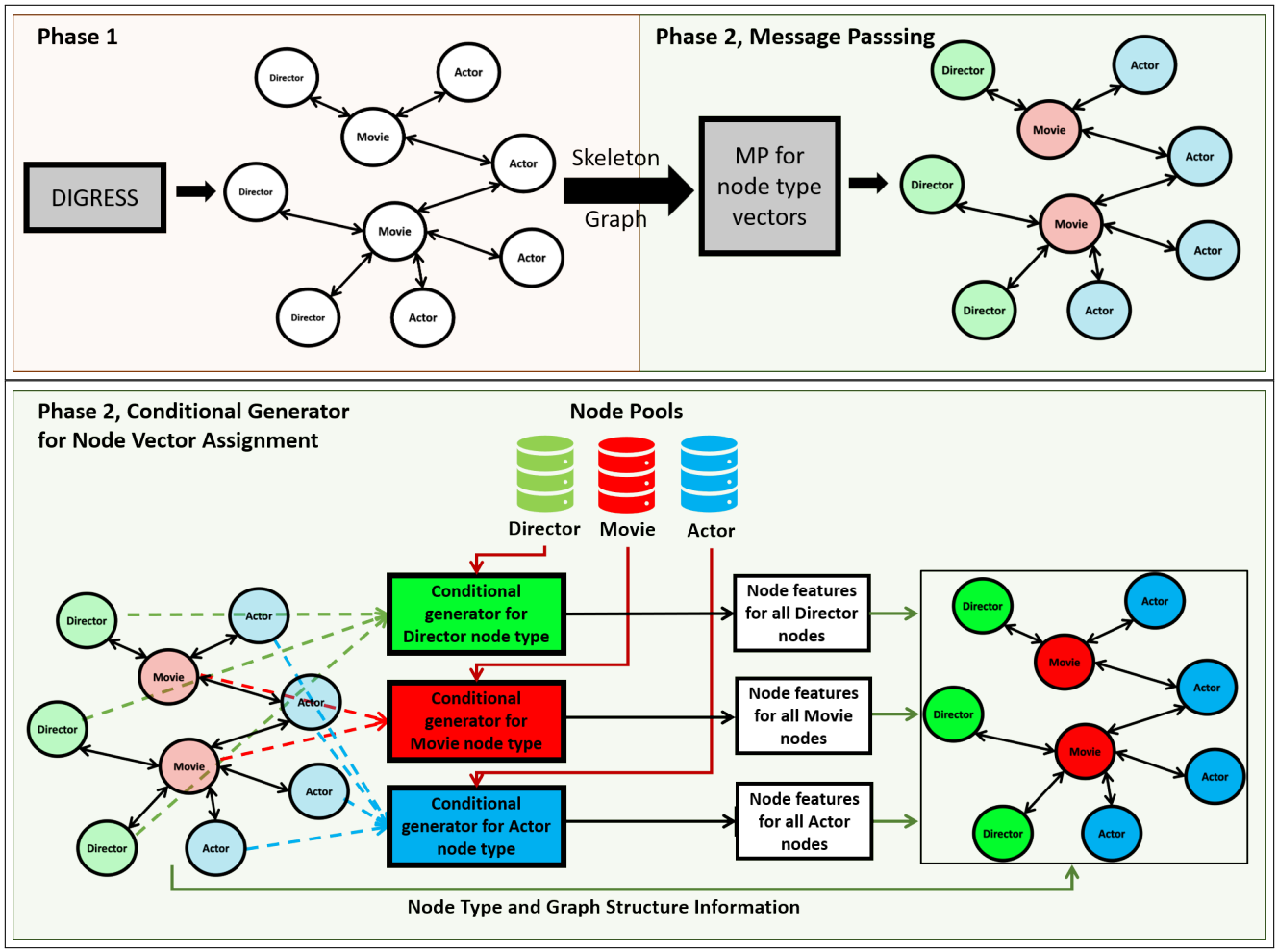


Fig. 2: The generation process using IMDB data as example. In Phase 1, we use DiGress to output a graph with node types Director, Actor and Movie. Subsequently, the first part in Phase 2 is to perform message passing to embed neighbor node type information in each node type vector and the second part is to perform node vector assignment using conditional generators. In this example, the node vectors for say, Director node are picked from the Director feature vector pool based on the probability distribution generated by the output of the conditional generator of Director node type, and then assigned to the Director nodes of the graph. This is done for all node types, thus completing the heterogeneous graph generation.

TABLE II: DBLP graph sets

Split criteria	Number of graphs with nodes ≤ 200
No Split	0
author	0
conference	1
type	4
author and conference	25
author and type	8
conference and type	29
author, conference and type	77

A. Dataset Description

The datasets used in this work were collectively sourced and suitably adapted from [42] and [38]. The former is one of the first attempts that made use of the datasets and has the raw data available with them, while the latter is the PyTorch Geometric framework. Specifically, we look at the Digital Bibliography & Library Project (DBLP) dataset and the Internet Movie Database (IMDB) dataset. The original datasets can be accessed at their corresponding links[1] [2].

1) Digital Bibliography & Library Project (DBLP): The whole DBLP dataset is a heterogeneous graph consisting of authors (4,057 nodes), papers (14,328 nodes), terms (7,723 nodes), and conferences (20 nodes). The authors are divided into four research areas (database, data mining, artificial intelligence, information retrieval). Each author is described by a bag-of-words representation [43] of their paper keywords. Each paper is described using a bag-of-words representation of their paper titles, while each term is described using GloVe vectors [44]. The feature vector sizes are summarized in

[1]DBLP
[2]IMDB

TABLE III: IMDB graph description

Node type	Representation	Feature vector size
movie	Bag-Of-Words of plot keywords	3066
actor	Mean of associated movies' features	3066
director	Mean of associated movies' features	3066

Table I. The raw datasets, though quite convenient to use, contain only a single large heterogeneous graph, and the features available for segmenting these graphs are limited. To address this limitation, we obtained raw base data that was used to contruct the dataset from [42] and a snapshot release of DBLP February 2024 to enrich the dataset with additional categorical features. This allowed us to divide the large, singular heterogeneous graph into multiple smaller ones, providing us with a more diverse and comprehensive dataset for our experiments.

We have modified the DBLP dataset to include another categorical feature which is the type of the publication. The original graph is very large and therefore, we have used some categorical features and their combinations to segregate the large graph into smaller components each corresponding to a unique category of the feature. We used the author research area (indicated as author), conference, and type of paper publication and their combinations to obtain a set of smaller graphs. Furthermore, due to resource constraints, we have limited ourselves to use only those graphs that have number of nodes less than or equal to 200. Post this processing stage, the number of graphs in each set is highlighted in Table II. The split using the values of author research area (indicated as author) and values of conference, yielded 25 graphs with less than 200 nodes. Additionally, the split using the values of author research area (indicated as author), values of conference, and values of publication type, yielded 77 graphs with less than 200 nodes. These are the two datasets of DBLP based graphs we use in our experiments.

Examples of the graphs from the split criteria - (author, conference and type) are visualized in the top row of Figure 3. In all DBLP graphs, authors are in red, papers in blue and terms are in green.

2) Internet Movie Database (IMDB): The original IMDB dataset is a heterogeneous graph consisting of three types of entities - movies (4,278 nodes), actors (5,257 nodes), and directors (2,081 nodes). The movies are divided into three classes (action, comedy, drama) according to their genre. Movie features correspond to elements of a bag-of-words representation of its plot keywords. Features are assigned to directors and actors as the means of their associated movie features. The feature vector sizes are summarized in Table III.

We have modified the dataset to include three categorical features, namely, year, language and country obtained from the raw data provided in [42]. It allows for more categorical features which can be utilized to split the graphs. We used the movie genre classes (indicated as movie), and the three newly extracted categorical features to split the larger graph into a set of smaller graphs. After the above processing stage, the number of graphs in each set is highlighted in Table IV. The number of graphs with less than 200 nodes split by year is 64, and by country, language, movie is 179 and these are the two datasets of IMDB based graphs we subsequently use.

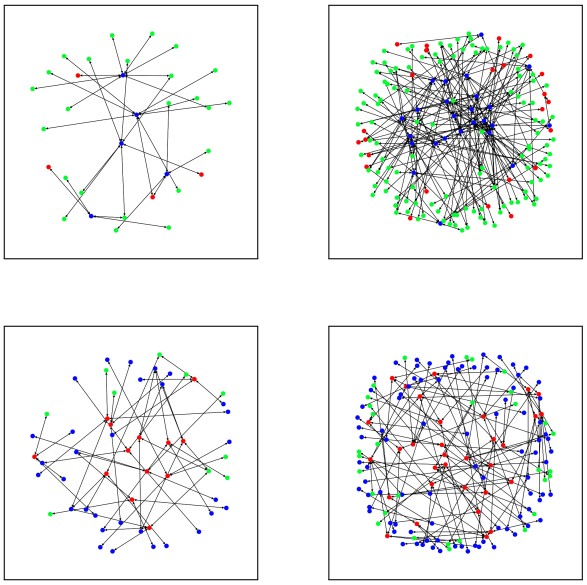


Fig. 3: (Top row) Examples of DBLP graphs from the split criteria - author, conference and type. (Bottom row) Examples of IMDB graphs from the split criteria - country, language and movie

TABLE IV: IMDB graph sets

Split criteria	Number of graphs with nodes ≤ 200
No Split	0
movie	0
year	64
language	43
country	56
movie and year	159
movie and language	75
movie and country	108
year and language	246
year and country	441
language and country	111
movie, year and language	382
movie, year and country	732
movie, language and country	179
year, language and country	523
movie, year, language and country	802

Examples of the graphs from the split criteria - (movie, language and country) are visualized in the bottom side of Figure 3. In all IMDB graphs, movies are in red, actors in blue and directors are in green.

B. Evaluation Metrics

The new metric introduced in our work in Section III-C is first explained in detail below.

Type Degree Distribution MMD: Let there be K number of node types in the graph $G = (V, E)$. Then, for every node i, compute the following K-length vector wherein every position

j in the vector is the number of neighboring nodes of i that are of type j. Thus, they are the number of nodes v, such that $(i, v) \in E$ and v is of type j. Once this vector is computed for all nodes, we have an $n \times K$ matrix for the whole graph. Drawing parallels from degree distribution MMD, note that in that metric, there was only one dimension instead of K, i.e., it had an $n \times 1$ dimensional matrix and we have extended it to K types. Now, for each of the K types, compute a degree distribution MMD separately. An average of these K degree distributions is the Type Degree Distribution MMD.

Other Metrics: In addition to the proposed type degree distribution metric, tailored specifically for the heterogeneous graph generation task (see Section III-B), we employ three widely used metrics for generative model evaluation, namely [45]: *Wasserstein Distance* (W-Dist.), *1-Nearest Neighbor Accuracy* (1-NN Accuracy), and *Fréchet Inception Distance* (FID). Both Wasserstein distance and FID are distance measures between two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, and are approximated in practice using finite samples. The 1-NN accuracy is calculated as the leave-one-out (LOO) accuracy of a 1-NN classifier, trained on samples drawn from $\mathbb{P}_r$ and $\mathbb{P}_g$. When the two distributions are similar, the classifier is expected to achieve approximately 50% LOO accuracy. Here, $\mathbb{P}_r$ denotes the real data distribution, and $\mathbb{P}_g$ denotes the generated data distribution. The feature vectors used for Wasserstein distance, 1-NN accuracy and FID are extracted from our trained discriminator. For further details, we refer readers to Xu et al. [45].

We also use a number of standard metrics used in the literature. All our metrics are computed on samples of real and generated graphs; we generate 10 such sample sets and report the mean and standard deviation in our results. Note that all these metrics in the literature are Maximum Mean Discrepancy distance computed for the quantities listed below. The MMD is estimated using these quantities computed for each of the sampled generated and real graphs with 500 sampled graphs. These quantities are (1) Degree Distribution (called Degree Dist. in tables), (2) Clustering Coefficient (called Clust. Coeff. in tables), and (3) Spectral Density (called Spect. Dens. in tables). In general, all of these metrics capture the structural similarity of the generated graphs to that of the real graphs in the dataset. Additionally, the first four metrics explicitly capture node type similarity by design. Lower values indicate better performance for all metrics, except for the 1-NN accuracy, which is optimal when equal to 50%.

C. Training Details and Hyperparameter Choices

All our experiments were performed on a GPU with 16 GB RAM, except the first phase of our approach using DiGress, which is particularly memory intensive, and required GPUs with 32 GB of RAM. The CUDA version of our GPUs was 12.2. All the experiments were performed with the following split for every dataset: 72% (90% of 80%) train, 8% (10% of 80%) validation and 20% test.

The two phases of our experiments are trained independently and therefore the hyperparameter decisions for each phase are also made independently. For the first phase, several variations of the learning rate and optimizer of the DiGress model were tried and the best settings were chosen after a grid search for each of the datasets. This best setting was used to produce the skeleton graphs for the second phase. In a similar vein, for the second phase, variations of the learning rate and optimizer were tried along with the overall number of parameters in the model. Post grid search, the best setting was chosen based on the evaluation metrics to produce the final graphs. We list the hyperparameter choices made in the first and second phase for all the datasets.

TABLE V: Hyperparameter choices of first phase for IMDB

Split criteria	Learning Rate	Optimizer
year	0.0002	nadam
year and country	0.0002	nadam
movie, language and country	0.0002	adamw
movie, year, language and country	0.0002	adamw

TABLE VI: Hyperparameter choices of first phase for DBLP

Split criteria	Learning Rate	Optimizer
author and conference	0.0002	nadam
author, conference and type	0.0002	nadam

Phase 1: Skeleton Graph Generator: Our first phase is DiGress, and most of the model's parameters are retained to be the default ones as provided by the code of [14]. However, the learning rate and optimizer were grid searched upon and the best model based on negative log-likelihood loss was chosen. The choices for learning rate were $[0.00001, 0.00010.0002, 0.001]$ and the choices for the optimizer were $[nadam, adamw]$ [46] [47]. Finally, the choices for generating the samples are summarized in Tables V and VI for IMDB and DBLP, respectively.

Phase 2: Heterogeneous Feature Vector Assignment For the second phase, the hyperparameter choices were extended to include the overall model's parameters along with the learning rate and optimizer. For brevity, we refer to the overall model's parameters as follows: (1) d - default, (2) d2 - double the parameters from default, and (3) dh2 - double only the number of heads in attention based frameworks in default. The choices for learning rate were $[0.02, 0.0002]$ and the choices for the optimizer were $[adam, radam, nadam, adamw]$ [48] [46] [49] [47]. Finally, the choices used to generate the heterogeneous graphs are summarized in Tables VII and VIII for IMDB and DBLP, respectively.

D. Baselines

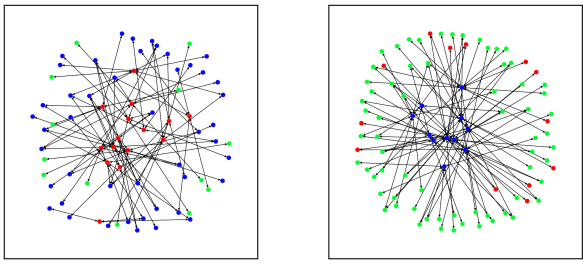
As mentioned in Section II, there is only one prior work [16] on heterogeneous graph generation. However, that approach is not directly applicable to training over multiple graphs. Hence, we make two baselines by appropriately adapting the prior work VGAE [32] to our scenario. VGAE is a Variational Autoencoder based homogeneous graph generative model. It makes use of homogeneous MP to compute intermediate latent vectors which are then used to construct a new adjacency matrix. We trained VGAE to produce skeleton graphs for our

TABLE VII: Second Phase Hyperparameter choice for IMDB

Split criteria	Learning Rate	Optimizer	Parameters
year	0.0002	nadam	d
year and country	0.0200	adam	d
movie, lang. & country	0.0002	adam	d
movie, year, lang. & country	0.0200	adam	dh2

TABLE VIII: Second Phase Hyperparameter choice for DBLP

Split criteria	Learning Rate	Optimizer	Parameters
author and conference	0.0002	nadam	d
author, conference and type	0.0200	radam	d2



(a) IMDB from split criteria - (movie, language and country)

(b) DBLP from split criteria - (author, conference and type)

Fig. 4: Generated Graph Samples

datasets. Then, in order for VGAE to work with heterogeneous graphs, we did a random feature vector assignment from the node pool of the appropriate node type as a post-processing step to construct a heterogeneous graph. It is important to note that node types in VGAE are copied from the input and are not manipulated in its generation process (only edges are manipulated), providing it with significant advantage as a baseline. We also construct a hybrid of VGAE and our HG2NP, which we call VGAE-H. In VGAE-H, we use VGAE in our first phase, instead of DiGress, and then use our model's second phase as a feature vector assignment module.

E. Detailed Results and Comparative Study

Detailed experimental results are presented in Table IX for IMDB graphs and Table X for DBLP graphs. The results show that our approach HG2NP consistently outperforms the baselines. We note that some results could not be produced ($-$ in the table) for the size of graphs that we have in our experiments, given a 24-hour cutoff on the CPU and GPU. In particular, performance of HG2NP is much better on the type degree distribution, showing a better modeling of the distribution of node types and connecting edges in the graphs generated by our approach. The Wasserstein distance between the feature vectors (obtained from our trained discriminator) of real and generated graphs is nearly 93% lower on average than other approaches, and the FID scores are approximately 8.05 times lower on average, indicating that our generated graphs are in synonymity with the real graphs. In the context of standard metrics like Degree Distribution, Clustering Coefficient, and Spectral Density, our approach consistently fares well, thereby reinforcing the idea that our generated graphs are invariable with the real graphs in the dataset.

We show samples of the graphs generated from our models in Figure 4. Some likeness can be seen between the example graphs from the dataset in Figure 3 and the generated graphs. For instance, red nodes (movies in IMDB) are centric in IMDB graphs in both the figures and in DBLP graphs, blue nodes (papers in DBLP) are centric in both the figures.

F. Ablation Study

In ablation studies, we looked at two components of our second phase, namely, the MP module in Phase 2, Part 1 and the node pools in Phase 2, Part 2 (see Figure 2). In the first study, we remove the MP module in Phase 2, Part 1 and re-run the experiments. The sampling modules now directly interact with the skeleton graphs and the node type vectors. We call this *NoMP* in the result tables. In the second study, we remove the node feature pools and replace them with dense neural networks that directly outputs the node features. We call this *NoPool* in the result tables. The ablation results for IMDB and DBLP shown in Tables XI and XII, respectively, clearly indicate that our architectural choices for HG2NP are important in getting the desired performance. In particular, the message passing is critical.

G. Single Graph Training

Even though most graph generative frameworks are trained on multiple graph instances to learn the distributions of graph in the generation process as seen in [8], some, for instance, [35], [16] and [36], use a different approach. In an attempt to generate realistic graphs, they train only on one single graph and try to match properties of the generated graphs to this graph. Although our model uses multiple graph instances for training, we can also train it on a single graph (as if we have only that graph in the training set). We have extended our experiments to NetGAN [35] which uses a single graph, and also run VGAE [32] on a single graph. In all datasets, the largest graph in terms of number of nodes below 200 were used for the purpose of training. The results of these experiments are summarized in Tables XIII and XIV for IMDB and DBLP, respectively. We see that even in this scenario, HG2NP outperforms existing approaches.

V. Limitations and Discussions

An important limitation, and potential future work, is to design our approach to explicitly work with edge types. However, we do note that popular heterogeneous graph datasets in Pytorch Geometric do not have such explicit edge types which are not already implied by the node types. This direction is an interesting future work as it also requires expanding on the labeled permutation equivariance and invariance we have defined, though we note that our transformation of edge labeled graphs to only node labeled graphs at the start of Section III may be the path towards this theoretical extension.

Another potential improvement is to have iterative refinement of feature vector assignments. Vectors once assigned are

TABLE IX: IMDB results for two splits (bold shows best results).

	Year			Country, Language and Movie		
Metrics	VGAE	VGAE-H	HG2NP	VGAE	VGAE-H	HG2NP
Degree Dist.	0.35 ± 0.004	0.34 ± 0.008	$\mathbf{0.18 \pm 0.001}$	0.30 ± 0.002	0.30 ± 0.003	$\mathbf{0.15 \pm 0.001}$
Clust. Coeff.	0.92 ± 0.024	0.88 ± 0.015	$\mathbf{5.2e\text{-}5 \pm 1.4e\text{-}5}$	— ± —	0.75 ± 0.012	$\mathbf{7.8e\text{-}6 \pm 4.6e\text{-}6}$
Spect. Dens.	0.38 ± 0.013	0.36 ± 0.011	$\mathbf{0.06 \pm 0.003}$	0.23 ± 0.003	0.24 ± 0.004	$\mathbf{0.04 \pm 0.002}$
Type Degree	0.28 ± 0.007	0.26 ± 0.007	$\mathbf{0.05 \pm 3.5e\text{-}4}$	0.18 ± 0.002	0.17 ± 0.001	$\mathbf{0.03 \pm 2.3e\text{-}4}$
W-Dist.	$43.85e\text{-}6 \pm 5.47e\text{-}6$	$49.64e\text{-}6 \pm 7.95e\text{-}6$	$\mathbf{2.93e\text{-}6 \pm 2.01e\text{-}6}$	$39.07e\text{-}6 \pm 3.20e\text{-}6$	$35.27e\text{-}6 \pm 3.41e\text{-}6$	$\mathbf{8.63e\text{-}6 \pm 2.35e\text{-}6}$
1-NN Accuracy	0.85 ± 0.112	0.79 ± 0.019	$\mathbf{0.38 \pm 0.196}$	0.47 ± 0.011	0.48 ± 0.009	$\mathbf{0.48 \pm 0.010}$
FID	$2.53e\text{-}3 \pm 5.42e\text{-}5$	$2.61e\text{-}3 \pm 9.19e\text{-}5$	$\mathbf{0.99e\text{-}3 \pm 1.56e\text{-}5}$	$1.18e\text{-}3 \pm 1.20e\text{-}5$	$1.19e\text{-}3 \pm 2.76e\text{-}5$	$\mathbf{1.07e\text{-}3 \pm 3.63e\text{-}5}$

TABLE X: DBLP results for two splits (bold shows best results).

	Author and Conference			Author, Conference, and Pub. Type		
Metrics	VGAE	VGAE-H	HG2NP	VGAE	VGAE-H	HG2NP
Degree Dist.	0.41 ± 0.002	0.43 ± 0.007	$\mathbf{0.16 \pm 0.004}$	0.36 ± 0.003	0.35 ± 0.006	$\mathbf{0.01 \pm 0.001}$
Clust. Coeff.	— ± —	1.08 ± 0.007	$\mathbf{3.6e\text{-}4 \pm 4.7e\text{-}5}$	— ± —	0.92 ± 0.018	$\mathbf{6.0e\text{-}5 \pm 1.3e\text{-}5}$
Spect. Dens.	0.48 ± 0.023	0.52 ± 0.020	$\mathbf{0.07 \pm 0.006}$	0.38 ± 0.015	0.36 ± 0.006	$\mathbf{1.2e\text{-}3 \pm 3.2e\text{-}4}$
Type Degree	0.30 ± 0.004	0.29 ± 0.016	$\mathbf{0.04 \pm 0.001}$	0.22 ± 0.002	0.19 ± 0.003	$\mathbf{1.5e\text{-}3 \pm 9.3e\text{-}5}$
W-Dist.	$28.07e\text{-}6 \pm 3.50e\text{-}6$	$42.38e\text{-}6 \pm 3.80e\text{-}6$	$\mathbf{0.00 \pm 0.000}$	$58.91e\text{-}6 \pm 4.65e\text{-}6$	$63.32e\text{-}6 \pm 6.31e\text{-}6$	$\mathbf{0.00 \pm 0.000}$
1-NN Accuracy	0.92 ± 0.112	0.98 ± 0.011	$\mathbf{0.61 \pm 0.024}$	1.00 ± 0.002	1.00 ± 0.000	$\mathbf{0.47 \pm 0.053}$
FID	$3.01e\text{-}3 \pm 1.54e\text{-}5$	$3.23e\text{-}3 \pm 2.44e\text{-}5$	$\mathbf{0.87e\text{-}3 \pm 6.41e\text{-}5}$	$3.45e\text{-}3 \pm 3.72e\text{-}5$	$3.53e\text{-}3 \pm 2.07e\text{-}5$	$\mathbf{0.14e\text{-}3 \pm 6.11e\text{-}5}$

TABLE XI: Ablation : IMDB results for two splits

	Year			Country, Language and Movie		
Metrics	NoMP	NoPool	HG2NP	NoMP	NoPool	HG2NP
W-Dist.	$4.13e\text{-}6 \pm 2.04e\text{-}6$	$4.21e\text{-}6 \pm 1.83e\text{-}6$	$\mathbf{2.93e\text{-}6 \pm 2.01e\text{-}6}$	$8.63e\text{-}6 \pm 2.73e\text{-}6$	$\mathbf{7.77e\text{-}6 \pm 1.44e\text{-}6}$	$8.63e\text{-}6 \pm 2.35e\text{-}6$
1-NN Accuracy	0.31 ± 0.101	0.38 ± 0.039	$\mathbf{0.38 \pm 0.196}$	$\mathbf{0.48 \pm 0.014}$	0.45 ± 0.124	0.47 ± 0.011
FID	$0.99e\text{-}3 \pm 3.37e\text{-}5$	$0.99e\text{-}3 \pm 2.30e\text{-}5$	$\mathbf{0.99e\text{-}3 \pm 1.56e\text{-}5}$	$1.07e\text{-}3 \pm 1.47e\text{-}5$	$1.07e\text{-}3 \pm 1.16e\text{-}5$	$\mathbf{1.07e\text{-}3 \pm 3.63e\text{-}5}$

TABLE XII: Ablation : DBLP results for two splits

	Author and Conference			Author, Conference, and Pub. Type		
Metrics	NoMP	NoPool	HG2NP	NoMP	NoPool	HG2NP
W-Dist.	0.00 ± 0.000	0.00 ± 0.000	$\mathbf{0.00 \pm 0.000}$	0.00 ± 0.000	0.00 ± 0.000	$\mathbf{0.00 \pm 0.000}$
1-NN Accuracy	0.62 ± 0.032	$\mathbf{0.60 \pm 0.026}$	0.61 ± 0.024	0.47 ± 0.028	$\mathbf{0.51 \pm 0.029}$	0.47 ± 0.053
FID	$0.95e\text{-}3 \pm 6.44e\text{-}5$	$0.95e\text{-}3 \pm 7.09e\text{-}5$	$\mathbf{0.87e\text{-}3 \pm 6.41e\text{-}5}$	$0.14e\text{-}3 \pm 4.67e\text{-}5$	$0.19e\text{-}3 \pm 3.96e\text{-}5$	$\mathbf{0.14e\text{-}3 \pm 6.11e\text{-}5}$

TABLE XIII: Single Graphs : IMDB results for four splits

	Year		
Metrics	VGAE	NetGAN	HG2NP
Degree Dist.	0.37 ± 0.004	0.37 ± 0.004	$\mathbf{0.23 \pm 0.002}$
Clust. Coeff.	0.95 ± 0.016	0.73 ± 0.008	$\mathbf{6.8e\text{-}5 \pm 7.2e\text{-}6}$
Spect. Dens.	0.61 ± 0.178	0.64 ± 0.168	$\mathbf{0.22 \pm 0.131}$
Type Degree	0.30 ± 0.006	$0.24 \pm 4.6e\text{-}4$	$\mathbf{0.07 \pm 4.9e\text{-}4}$
W-Dist.	$5.55e\text{-}5 \pm 5.01e\text{-}6$	$4.52e\text{-}5 \pm 5.06e\text{-}6$	$\mathbf{1.90e\text{-}5 \pm 3.83e\text{-}6}$
1-NN Accuracy	0.71 ± 0.026	0.78 ± 0.063	$\mathbf{0.55 \pm 0.151}$
FID	$2.61e\text{-}3 \pm 7.44e\text{-}5$	$2.32e\text{-}3 \pm 1.07e\text{-}5$	$\mathbf{1.21e\text{-}3 \pm 5.85e\text{-}5}$

TABLE XIV: Single Graphs : DBLP results for two splits

	Author and Conference		
Metrics	VGAE	NetGAN	HG2NP
Degree Dist.	0.41 ± 0.003	0.73 ± 0.000	$\mathbf{0.17 \pm 0.001}$
Clust. Coeff.	— ± —	2.00 ± 0.000	$\mathbf{0.01 \pm 3.3e\text{-}4}$
Spect. Dens.	0.68 ± 0.080	0.75 ± 0.000	$\mathbf{0.29 \pm 0.205}$
Type Degree	0.30 ± 0.004	$0.36 \pm 4.1e\text{-}4$	$\mathbf{0.040 \pm 3.1e\text{-}4}$]
W-Dist.	$2.19e\text{-}5 \pm 3.38e\text{-}6$	$3.05e\text{-}5 \pm 3.61e\text{-}6$	$\mathbf{1.28e\text{-}5 \pm 3.58e\text{-}6}$
1-NN Accuracy	0.98 ± 0.011	0.99 ± 0.005	$\mathbf{0.63 \pm 0.033}$
FID	$3.04e\text{-}3 \pm 2.44e\text{-}5$	$3.11e\text{-}3 \pm 1.20e\text{-}5$	$\mathbf{1.76e\text{-}3 \pm 15.95e\text{-}5}$

not revisited in HG2NP. Assignment is one shot in the sense that all of them are done in parallel in one go. Alternative options include performing iterative refinements, i.e., we still assign all of them at once, but iteratively improve the assignment by updating them all again.

A further option is to sequentially assign feature vectors, that is assign feature vectors to some nodes first, and then based on previous assignments, assign feature vectors to their neighbors and so on.

Overall, our work HG2NP provides a hierarchical approach that successfully generates heterogeneous graphs with promising results for the domains that we experimented with. It is expected work equally well for other domains as well. A number of potential extensions of our work provide ground for further research on this topic.

REFERENCES

[1] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles, "Co-ranking authors and documents in a heterogeneous network," in *Seventh IEEE International Conference on Data Mining*. IEEE, 2007, pp. 739–744.

[2] Y. Dong, J. Tang, S. Wu, J. Tian, N. V. Chawla, J. Rao, and H. Cao, "Link prediction and recommendation across heterogeneous social networks," in *Twelfth IEEE International Conference on Data Mining*. IEEE, 2012, pp. 181–190.

[3] X. Chen, M.-X. Liu, and G.-Y. Yan, "Drug–target interaction prediction by random walk on the heterogeneous network," *Molecular BioSystems*, vol. 8, no. 7, pp. 1970–1978, 2012.

[4] Y. Li and J. C. Patra, "Genome-wide inferring gene–phenotype relationship by walking on the heterogeneous network," *Bioinformatics*, vol. 26, no. 9, pp. 1219–1224, 2010.

[5] W. Wang, S. Yang, X. Zhang, and J. Li, "Drug repositioning by integrating target information through a heterogeneous network model," *Bioinformatics*, vol. 30, no. 20, pp. 2923–2930, 2014.

[6] C. Chen, W. Ma, M. Zhang, Z. Wang, X. He, C. Wang, Y. Liu, and S. Ma, "Graph heterogeneous multi-relational recommendation," in *AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 3958–3966.

[7] L. D. Santos, B. Piwowarski, L. Denoyer, and P. Gallinari, "Representation learning for classification in heterogeneous graphs with application to social networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 5, pp. 1–33, 2018.

[8] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5370–5390, 2022.

[9] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International Conference on Machine Learning*, 2018, pp. 5708–5717.

[10] S.-Y. Su, H. Hajimirsadeghi, and G. Mori, "Graph generation with variational recurrent neural network," *arXiv preprint arXiv:1910.01743*, 2019.

[11] L. D'Arcy, P. Corcoran, and A. Preece, "Deep q-learning for directed acyclic graph generation," *arXiv preprint arXiv:1906.02280*, 2019.

[12] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[13] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *27th International Conference on Artificial Neural Networks*, 2018, pp. 412–422.

[14] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "Digress: Discrete denoising diffusion for graph generation," in *11h International Conference on Learning Representations*, 2023.

[15] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *The Web Conference 2020*, 2020, pp. 2704–2710.

[16] C. Ling, C. Yang, and L. Zhao, "Deep generation of heterogeneous networks," in *IEEE International Conference on Data Mining*, 2021, pp. 379–388.

[17] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, 2018.

[18] Y. Li, L. Zhang, and Z. Liu, "Multi-objective de novo drug design with conditional graph generative model," *Journal of cheminformatics*, vol. 10, pp. 1–24, 2018.

[19] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[20] S. Wasserman and P. Pattison, "Logit models and logistic regressions for social networks: I. an introduction to markov graphs and p," *Psychometrika*, vol. 61, no. 3, pp. 401–425, 1996.

[21] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu, "A survey on deep graph generation: Methods and applications," in *Learning on Graphs Conference*, 2022, pp. 47–1.

[22] M. Khodayar and J. Wang, "Deep generative graph learning for power grid synthesis," in *International Conference on Smart Energy Systems and Technologies*, Sep. 2021, pp. 1–6.

[23] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-vae: A variational autoencoder for directed acyclic graphs," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] R. Assouel, M. Ahmed, M. H. Segler, A. Saffari, and Y. Bengio, "Defactor: Differentiable edge factorization-based probabilistic graph generation," *arXiv preprint arXiv:1811.09766*, 2018.

[25] J. Lim, S.-Y. Hwang, S. Moon, S. Kim, and W. Y. Kim, "Scaffold-based molecular design with a graph generative model," *Chemical Science*, vol. 11, no. 4, pp. 1153–1164, 2020.

[26] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, "Constrained graph variational autoencoders for molecule design," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[27] S. Kearnes, L. Li, and P. Riley, "Decoding molecular graph embeddings with reinforcement learning," *arXiv preprint arXiv:1904.08915*, 2019.

[28] X. Bresson and T. Laurent, "A two-step graph convolutional decoder for molecule generation," *arXiv preprint arXiv:1906.03412*, 2019.

[29] P. Rivas, M. Guarino, and A. Shah, "Dipol-gan: Generating molecular graphs adversarially with relational differentiable pooling," *LatinX in AI at Neural Information Processing Systems Conference*, 2019.

[30] D. Flam-Shepherd, T. Wu, and A. Aspuru-Guzik, "Graph deconvolutional generation," *arXiv preprint arXiv:2002.07087*, 2020.

[31] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, "Permutation invariant graph generation via score-based generative modeling," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 4474–4484.

[32] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[33] Y. Qin, C. Vignac, and P. Frossard, "Sparse training of discrete diffusion models for graph generation," *arXiv preprint arXiv:2311.02142*, 2023.

[34] J. Jo, D. Kim, and S. J. Hwang, "Graph generation with diffusion mixture," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27*, 2024.

[35] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," in *International Conference on Machine Learning*, 2018, pp. 610–619.

[36] C. Ling, C. Yang, and L. Zhao, "Motif-guided heterogeneous graph deep generation," *Knowledge and Information Systems*, vol. 65, no. 7, pp. 3099–3124, 2023.

[37] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *International Conference on Learning Representations*, 2021.

[38] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[39] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.

[40] L. O'Bray, M. Horn, B. Rieck, and K. Borgwardt, "Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions," in *International Conference on Learning Representations*, 2021.

[41] S.-M. Hsieh, C.-C. Hsu, and L.-F. Hsu, "Efficient method to perform isomorphism testing of labeled graphs," in *International Conference on Computational Science and its Applications*, 2006, pp. 422–431.

[42] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *The Web Conference 2020*, 2020, pp. 2331–2341.

[43] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, 2010.

[44] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[45] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, "An empirical study on evaluation metrics of generative adversarial networks," *arXiv preprint arXiv:1806.07755*, 2018.

[46] T. Dozat, "Incorporating Nesterov Momentum into Adam," in *4th International Conference on Learning Representations*, 2016, pp. 1–4.

[47] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, 2015.

[49] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *8th International Conference on Learning Representations, ICLR*, 2020.