GRAHIES: Multi-Scale Graph Representation Learning with Latent Hierarchical Structure

Lei Yu[†‡], Ling Liu[†], Calton Pu[†], Ka Ho Chow[†], Mehmet Emre Gursoy[†], Stacey Truex[†], Hong Min[‡], Arun Iyengar[‡], Gong Su[‡], Qi Zhang[‡] and Donna Dillenberger[‡]
[†] *College of Computing, Georgia Institute of Technolgoy, Atlanta, GA, USA*
[‡] *IBM research, Yorktown Heights, NY, USA*

***Abstract*—A wide variety of deep neural network models for graph-structured data have been proposed to solve tasks like node/graph classification and link prediction. By effectively learning low-dimensional embeddings of graph nodes, they have shown state-of-the-art performance. However, most existing models learn node embeddings by exploring flat information propagation across the edges within the local neighborhood of each node. We argue that incorporating hierarchical node embeddings can capture the inherently hierarchical topological features of many realistic graphs such as social networks, biological network and World Wide Web. In this paper we propose GRAHIES, a general framework for graph neural networks to learn node representations that preserve hierarchical graph information at higher-orders. GRAHIES adaptively learns a multi-level hierarchical structure of the input graph, which consists of successively coarser (smaller) graphs that preserve the global structure of the original graphs at different levels. By combining the graph representations from different levels of the graph hierarchy, the final node representation captures the inherent global hierarchical structure of the original graph. Our experiments show that applying GRAHIES's hierarchical paradigm yields improved accuracy for existing graph neural networks on the node classification tasks.**

***Index Terms*—graph representation learning, node embedding, deep learning, node classification, multi-scale**

I. Introduction

Graph data is a ubiquitous type of data that widely occurs in many real world applications, such as social networks, biological networks, financial networks and World Wide Web. It is often desired to perform a set of machine learning tasks on graph data including node classification [1], link prediction [2], [3] and community detection [4]. Because deep neural networks have demonstrated significant success on image data [5] and text data [6], in recent years there has been a surge of interest in Graph Neural Networks (GNN) that apply deep neural networks on graph tasks and many variants of GNN [1], [7]–[12] have been proposed, which aim to effectively learn low-dimensional representation vectors /embeddings of nodes with using the graph structure and node features. The generated node representations are then used as the input to downstream machine learning tasks on graph data.

Generally, most of GNN variants follow a neighborhood aggregation framework that learn the representation of a node by iteratively aggregating the features of its neighboring nodes. The aggregation in each iteration is parameterized as a GNN layer. The composition of k GNN layers/iterations let the node representation capture the structure and feature information within the node's k-hop neighborhood. Within this framework, the information is propagated flatly via edges and the representation of every node is influenced indiscriminately by any other nodes in the k-hop neighborhood. It may cause GNNs unable to effectively exploit the hierarchical information of graph data and distinguish nodes from different clusters/communities. A small k for GNN models can lead to insufficient information aggregation, missing higher-order structure information, while a large k leads to the aggregation of features from irrelevant nodes/communities. Many real-world networks, however, are inherently hierarchical. For example, social networks usually have many communities that are groups of related nodes that correspond to social spheres; biological networks may have communities corresponding to functional subunits such as protein complexes. The communities can be further recursively grouped into larger communities to present a deep hierarchical structure. Hence, GNN models that disregard important hierarchical structural information may not lead to the best representations for all nodes on these graph data.

In this paper we propose GRAHIES, a multi-scale graph learning framework that allows GNN models to exploit latent hierarchical structure information. GRAHIES recursively coalesces the nodes and edges in the original graph to produce a series of successively smaller and coarser graphs. These coarsened graphs provide a view of the original graph's global structure at different levels/granularities. GNN modules are stacked in a hierarchical fashion in GRAHIES: each GNN module operates on a graph at different coarsening levels(including the original graph) in the hierarchy. The node representations of a coarsened graph are then projected to the original graph. By aggregating the node representations from different graph levels to produce the final representation for every node in the original graph, GRAHIES is able to effectively capture the latent hierarchical structure of the graph. We show that applying our framework to various state-of-the-art neighborhood-aggregation models achieves an average gain in accuracy.

II. Preliminaries

Let $G = (V, E)$ be a graph where V is the set of nodes and E is the set of edges. Let $|V| = N$ and A be $N \times N$ adjacency matrix of the graph with each entry A_{uv} being the weight of the edge between node u and v. We use $X \in \mathbb{R}^{N \times d}$ to represent the node feature matrix and each row X_v is the

d-dimensional feature vector of node $v \in V$. The goal of graph representation learning is to map a node or an entire graph to a point in a low-dimensional continuous vector space. Such representation vectors (embeddings) of nodes and graphs could be used for any downstream task such as node/graph classification, clustering, and link prediction.

A. Graph Neural Networks

Graph neural networks (GNNs) are general deep learning models that operate on graph G and node features X to learn a representation vector of a node or a whole graph. Generally a GNN model follows a neighborhood aggregation framework [13] that iteratively updates the feature vector of every node in the graph by aggregating the feature vectors of its neighbors. A single GNN layer performs one iteration of neighorhood feature aggregation for every node in G. The stacking of k GNN layers makes k iterations of aggregation, which let a nodes' representation to capture the structural information within its k-hop network neighborhood. Considering the k-th GNN layer, the update to v's feature follows the framework represented as:

$$\begin{aligned} a_v^{(k)} &= AGGREGATE^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\}, W_A^{(k)}) \\ h_v^{(k)} &= COMBINE(h_v^{(k-1)}, a_v^{(k)}, W_C^{(k)}) \end{aligned} \tag{1}$$

where h_u^{k-1} is the feature vector of node u from the previous iteration/layer and $h_v^{(k)}$ is the feature vector of node V generated by the k-th iteration/layer. $W_A^{(k)}$ and $W_C^{(k)}$ are trainable model parameters. The initial node feature $h_v^{(0)}$ used for the first iteration is initialized with input node feature X_v.

Many GNN variants [1], [8]–[12], [14]–[16] follow the neighborhood aggregation framework. One of popular variant of GNNs, GraphSAGE [8], has proposed a number of different functions for AGGREGATE including mean, max pooling and LSTM aggregator. For example, the max pooling aggregator is formulated as

$$a_v^{(k)} = max\big(\{\sigma(W_{pool} h_u^{(k-1)} + \mathbf{b}) : \forall u \in \mathcal{N}(v)\}\big) \tag{2}$$

where σ is the sigmoid function. The COMBINE step of GraphSAGE is

$$h_v^{(k)} = \sigma\big(W \cdot CONCAT(h_v^{(k-1)}, a_v^{(k)})\big) \tag{3}$$

which applies a concatenation of $a_v^{(k)}$ and node feature $h_v^{(k-1)}$.

Graph Convolutional Networks (GCN) [1] is another popular variant of GNN. Each GCN layer can be expressed as

$$H^{(k)} = \sigma\big(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(k-1)} W^{(k-1)}\big) \tag{4}$$

where $H^{(k)} \in \mathbb{R}^{N \times d^{(k)}}$ represents the output node feature matrix of layer k. A row in $H^{(k)}$ is $d^{(k)}$ dimensional feature for a node. $\hat{A} = A + I_N$ is the adjacency matrix with added self-loop connections to ensure that the old node feature $h_v^{(k-1)}$ is also taken into consideration to produce updated v's feature $h_v^{(k)}$ during aggregation. $\hat{D}$ is the diagonal node degree matrix where $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. In essence, GCN performs AGGREGATE and COMBINE as a whole for node v by a weighted sum of the feature vectors of all of its adjacency nodes followed by a linear mapping.

For a full GNN module with K layers/iterations of aggregation on a graph G, the node feature $h_v^{(K)}$ of the last layer is used as the learned node representation vector, that is further used for downstream tasks such as node/graph classification.

B. Node Classification

The task of node classification assumes that each node $v \in V$ is associated with a label y_v, and the goal of the graph representation learning for it is to learn a representation vector h_v such that v's label can be predicted as $y_v = f(h_v)$. For this task, GNN is usually followed by a fully connected neural network that takes the node representation from the final layer of GNN as its input and makes label predictions. Semi-supervised multi-class node classification assumes only a small portion of nodes in the graph have label information and the loss function for the training is evaluated over all the labeled nodes.

GNN is also used to learn the representation of an entire graph for graph classification by effectively aggregating all node features in the graph from the final layer of GNN. Assuming a set of graphs and each graph is associated with a a label, the task here is to learn a representation vector for every graph that can be further used to predict the label of an entire graph. In this paper we focus on the task of node classification while our node representation learning approach can be combined with other graph-level pooling schemes [17], [18] to produce graph-level representations.

III. Our approach

The goal of GRAHIES provides a framework to construct deep, multi-layer GNN models that incorporate inherent hierarchical structures of the graph data to generate node representation vectors. In this section, we introduce the details of our multi-layer GNN model GRAHIES.

A. Framework

GRAHIES defines a model architecture that stacks K ($K > 1$) GNN modules and each GNN takes as input a coarsened version of the graph that the preceding GNN module works on. By combining the node representations of graphs at different coarsening levels, GRAHIES lets the model make local node predictions that respect global structure. In particular, GRAHIES iteratively applies a GNN module to generate node representation for current input graph, followed by a graph coarsening module to produce a coarsened graph as the input graph for the next GNN module. Formally, let $A^{(1)} = A$ be the adjacency of the graph $G^{(1)} = G$ and $X^{(1)} = X$ be the node feature matrix. Generally, GRAHIES can be expressed as

$$H^{(k)} = GNN^{(k)}\big(A^{(k)}, X^{(k)}\big) \tag{5}$$

$$(S^{(k)}, X^{(k+1)}) = COARSEN^{(k)}\big(A^{(k)}, X^{(k)}, H^{(k)}\big) \tag{6}$$

$$A^{(k+1)} = {S^{(k)}}^T A^{(k)} S^{(k)} \tag{7}$$

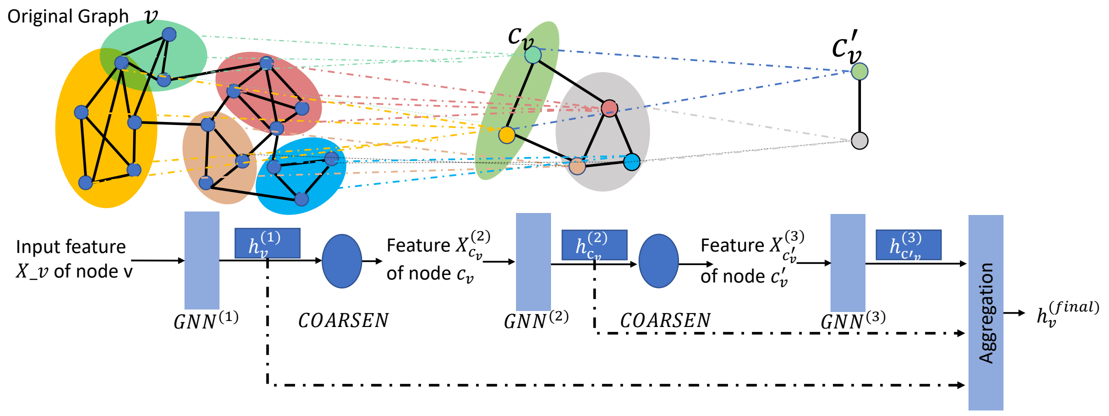

Fig. 1. An illustration of the GRAHIES model. We consider a stacking of three GNNs and the input graph is coarsened to two levels. Given a node v in the input graph, we assume that v only belongs to a single cluster represented by c_v in the coarsened graph for simplicity, and the same for c_v and c'_v. By propagating the graph representation of each coarsening level to the original graph, we can obtain v's representations at different levels $h_v^{(1)}$, $h_{c_v}^{(2)}$, $h_{c'_v}^{(3)}$ from the output of GNNs in the stack for v itself, c_v and c'_v respectively. An aggregation function is applied to the three levels representation to generate the final representation of v.

In this formulation, the k-th GNN module $GNN^{(k)}(A^{(k)}, X^{(k)})$ generates node representations $H^{(k)}$ for current input graph $G^{(k)}$ with adjacency matrix $A^{(k)} \in \mathbb{R}^{N^{(k)} \times N^{(k)}}$ with node feature matrix $X^{(k)}$. $COARSEN$ denotes a either trainable or predetermined non-trainable graph coarsening function that maps the graph $G^{(k)}$ to a coarsened smaller graph $G^{(k+1)}$ represented by $A^{(k+1)} \in \mathbb{R}^{N^{(k+1)} \times N^{(k+1)}}$, where the sizes of graphs $N^{(k)} > N^{(k+1)}$. Each node in $G^{(k+1)}$ represents a cluster of nodes in $G^{(k)}$. The mapping is represented by a soft cluster assignment matrix $S^{(k)} \in \mathbb{R}^{N^{(k)} \times N^{(k+1)}}$ where $S_{ij}^{(k)}$ represents the the probability of node i in $G^{(k)}$ belonging to the cluster node j in $G^{(k+1)}$. Given $S^{(k)}$, the adjacency matrix $A^{(k+1)}$ can be obtained by Equation (7). Note that for the last (K-th) GNN module, the graph is not further coarsened and only 5 is applied to generate $H^{(K)}$.

Figure 1 illustrates the proposed GRAHIES model. Each GNN module learns node representation for a graph at a different coarsening level. GRAHIES projects the representation of higher-level graphs to the original graph by recursively propagating the representation of every node to the nodes in its represented cluster in the previous finer graph. This step resembles the upsampling in convolutional neural networks [19] that upsample a feature map to a desired higher-resolution feature map. In Figure 1, for example, the node representation $h_{c'_v}^{(3)}$ in the coarsest graph is propagated along the hierarchy to all the nodes in its represented cluster containing c_v, and then is further propagated to the nodes in the clusters they represent in the original graph at the finer level. As a result, for a node v in the original graph, we obtain K features from K different levels, denoted by $h_v^{(1)}, h_v^{(2)}, ..., h_v^{(K)}$. Each $h_v^{(k)}$ is the graph representation propagated to v from different levels, representing a cluster in the original graph containing v at different scales. The final representation of a node v in the original graph is generated via an aggregation function such as concatenation, pooling and LSTM to combine these K features.

B. Graph Coarsening

The COARSEN function in GRAHIES creates a graph hierarchy $G^{(1)} = G, G^{(2)}, \dots, G^{(K)}$ by successively clustering the nodes in the preceding graph $G^{(k)}$ to a smaller graph $G^{(k+1)}$. We present two types of approaches for the graph coarsening based on if the GNN modules are coupled with the coarsening process: decoupled coarsening and coupled coarsening. Decoupled coarsening is a separate process to the graph representation learning that produces a graph hierarchy beforehand and GNN modules works on a fixed graph hierarchy, while in the coupled coarsening graph representation learning and the formation of graph hierarchy are joint learning processes.

1) Decoupled Graph Coarsening: The decoupled graph coarsening creates a graph hierarchy independent of graph representation learning. In this case, Equation 6 is simplified as

$$(S^{(k)}, X^{(k+1)}) = COARSEN^{(k)}(A^{(k)}, X^{(k)}) \tag{8}$$

It does not involve the output of GNNs $H^{(k)}$ and only depends on the input graph data. It provides a fixed graph hierarchy for learning node representations.

Graph coarsening can be solved by graph clustering (also known as community detection) where each cluster is represented by a node in the coarsened graph. Graph clustering is a widely studied problem and many clustering techniques has been proposed, including well-knows ones based on modularity [4], normalized cut [20] and matrix factorization [21]. A number of algorithms have been proposed to address graph clustering in attributed networks via probabilistic generative model [22] and matrix factorization [23]. In this paper we present a network embedding approach to learn a graph hierarchy, which later can be easily combined with GNNs for coupled graph coarsening. It is based on some intuitive properties for graph clustering in attributed networks [22] that nodes in the same cluster are likely to share common attributes and two nodes are more likely to be connected if they belong to multiple common clusters.

For a graph hierarchy, we have the graph sizes at each coarsening level $N_1 = N > N_2 > \ldots > N_K$ as hyperparameters in GRAHIES. We define the feature matrix $X^{(k)} \in \mathbb{R}^{N_k \times d}$ for k-th coarsened graph where d_X is the feature dimension of X and $X^{(1)} = X$. A membership propensity matrix is defined as

$$U^{(k)} = X^{(k)} X^{(k+1)^T} \tag{9}$$

in which $U_{ij}^{(k)}$ is the similarity of node i's feature at level k to cluster node j at level $k+1$, representing the propensity (unscaled logits) of node i belonging to the cluster j. Accordingly, the cluster assignment matrix

$$S^{(k)} = softmax(U^{(k)}) \tag{10}$$

where the softmax function is applied in a row-wise manner on $U^{(k)}$.

Considering two nodes u and v in $G^{(k)}$, row $S_u^{(k)}$ and $S_v^{(k)}$ in $S^{(k)}$ represent their probability distributions of belonging to every cluster. Because the probability of two nodes belonging to a common cluster indicates their probability of being connected, the product of $S_u^{(k)} S_v^{(k)}$ represents the connectivity strength between u and v that should be consistent with the adjacency matrix $A^{(k)}$. Thus, the clustering problem can be formulated as

$$\min ||A^{(k)} - S^{(k)} S^{(k)^T}||_F \tag{11}$$

where $|| \cdot ||_F$ denotes the Frobenius norm.

On the other hand, to achieve a good distinction between different clusters/communities, it is expected that each node should be strongly affiliated with only one cluster, which means the cluster assignment probability vector for each node should be close to a one-hot vector. Therefore, an entropy regularization for cluster assignment is introduced to the loss function. For a cluster assignment matrix $S^{(k)}$, each row $S_i^{(k)}$ is a probability distribution of cluster assignment for a node i in $G^{(k)}$, the entropy function $H(S^{(k)}) = \sum_i H(S_i^{(k)})$ where $H(S_i^{(k)}) = -\sum_j S_{ij}^{(k)} \log S_{ij}^{(k)}$.

The graph coarsening is then transformed to learn the feature matrix $X^{(k)}$ ($k > 1$) for each level of coarsened graphs with the goal to minimize

$$\sum_{k=1}^{K-1} \left(||A^{(k)} - S^{(k)} S^{(k)^T}||_F + H(S^{(k)}) \right) \tag{12}$$

Once the feature matrix $X^{(k)}$ ($k > 1$) is learned, we can apply Equation 9 and 10 to obtain a fixed graph hierarchy with soft assignment. We can use the pre-trained graph hierarchy (including node features of coarsened graphs and cluster assignments) in the framework of GRAHIES. However, the decoupled graph coarsening process does not interact with the optimization process of graph representation learning, leading to a suboptimal graph hierarchy, as shown in our experiment.

2) Coupled Graph Coarsening: In the coupled graph coarsening, given an input graph $G^{(k)}$, the generation of a coarser graph $G^{(k+1)}$ with its node features depends on the output node features $H^{(k)}$ of the preceding GNN module on $G^{(k)}$. Therefore, graph coarsening and graph representation learning are joint together. Its benefit lies on the resulting adaptive graph hierarchy that can be jointly optimized for the node representation learning.

The coupled graph coarsening approach is similar to the decoupled one. The difference is that, we define the membership propensity matrix as

$$U^{(k)} = H^{(k)} X^{(k+1)^T} \tag{13}$$

which means that the cluster assignment depends on the output of GNN $H^{(k)}$ and the feature $X^{(k)}$ corresponds to the clustering embeddings with respect to hidden node features produced by GNN modules. In this way, the learning of graph hierarchy (in terms of $X^{(k)}$) and GNN modules for graph representation are joint together, which allows an adaptive graph hierarchy/cluster assignments that can be optimized against node representation. The loss function under joint optimization is the sum of (12) and the loss for a specific task like node classification.

In this paper we employ coupled graph coarsening in our GRAHIES model, because our experiment demonstrates that the coupling of coarsening and representation learning achieves better performance than the decoupled approach.

C. Feature Projection and Aggregation

The final node representation is a combination of GNN output on graphs at different coarsening levels. However, because the graph coarsening is a soft cluster assignment, a node $v \in G$ can have non-zero probabilities in multiple clusters at each level. The probability of any node in the original graph G belonging to a cluster node at level k ($k > 1$), denoted by $B^{(k)}$, can be computed as

$$B^{(k)} = \begin{cases} S^{(k-1)} & k = 2 \\ B^{(k-1)} S^{(k)} & k > 2 \end{cases} \tag{14}$$

where $B^{(k)} \in \mathbb{R}^{N \times N^{(k)}}$. Then, the cluster representation at k-th level ($k > 1$) for v, denoted by $h_v^{(k)}$, can be computed as the sum of cluster node representations weighted by v's probabilities of being in their represented clusters. Accordingly, the node feature matrix $H^{(k)}$ for graph $G^{(k)}$ is projected to the original graph G via $B^{(k)}$

$$H_G^{(k)} = B^{(k)} H^{(k)} \tag{15}$$

A row $H_{i_G}^{(k)}$ of $H_G^{(k)}$ is the projected feature on node i of G. It can regarded as a sum of node features of graph $G^{(k)}$ weighted by $B_i^{(k)}$ that are the probabilities of node i belonging to the corresponding clusters, i.e., $H_{i_G}^{(k)} = B_i^{(k)} H^{(k)}$.

Top-k clusters based projection. Because of the soft assignment, the projected feature derived from Equation 15 integrates the features of all clusters at each level, which may lead to a weak distinction between nodes from different clusters.

Therefore, we only choose features of clusters that node i has top-k largest probabilities being associated with, and project them to node i. Formally, for the probability vector $B_i^{(k)}$, we keep top-k largest entries in it and set others to zero, to obtain a new probability vector denoted by $\hat{B}_i^{(k)}$. With L1-normalization $\hat{B}_i^{(k)}/\sum_j \hat{B}_{ij}^{(k)}$, the projected feature on node i becomes $H_{i_G}^{(k)} = \hat{B}_i^{(k)} H^{(k)}/\sum_j \hat{B}_{ij}^{(k)}$.

Trainable layer importance. The information on graphs at different coarsening levels may be correlated with original graph in varying degrees. To capture that in the model, we introduce trainable layer-wise coefficients on feature projection. For each level $k(k > 1)$, a trainable coefficient parameter $\beta^{(k)} \in \mathbb{R}$ is introduced to (15), i.e.,

$$H_G^{(k)} = \beta^{(k)} B^{(k)} H^{(k)} \tag{16}$$

$\beta^{(k)} = 1$ represents lossless projection which means the original graph G has highest correlation to the k-th level graph in terms of node feature, while $\beta^{(k)} = 0$ means the k-th level graph is not informative and important for learning the node features of G.

After feature projection, for every node v in the original graph G, we obtain K features $h_v^{(k)}$ ($1 \leq k \leq K$) from different graph levels. We can apply different types of aggregation to produce the final node representation. A simple aggregation is a concatenation $h_v^{(1)}||h_v^{(2)}||\ldots||h_v^{(K)}$. The aggregated node features are then used as the input to a class prediction layer/model for node classification.

D. Model Complexity

The model parameters of GRAHIES include the model parameters of each GNN module, the feature matrix $X^{(k)}$ ($k > 1$) for each level of graphs, k number of layer importance coefficients, and parameters of the final aggregation function and prediction layer for classification. Here we consider $X^{(k)}$ ($k > 1$) since it depends on input size but others do not. Given a specific graph coarsening ratio α, the graph at k-th level has size of $N\alpha^{k-1}$ which decreases with levels exponentially and thus the total is $O(N)$.

IV. Experiment

In this section, we evaluate GRAHIES against a number of multi-label node classification tasks on several real-life networks. In particular, we compare our approach with prior state-of-the-art models and analyze the impact of model hyperparameters on the performance.

A. Datasets

Our experiments use a set of standard benchmark datasets for node classification. The dataset statistics are summarized in Table I. In the semi-supervised node classification setting, the unlabeled testing data are accessible during training time. We use three standard citation network benchmark datasets-Cora, Citeseer and Pubmed and closely follow the transductive experimental setup of [1]. In these datasets, nodes correspond to documents and edges to citations. Node features correspond to a bag-of-words representation of a document Each node has a class label. The training data has 20 nodes per class. There are 500 nodes for validation and 1000 nodes for testing.

TABLE I
Summary of the datasets used in our experiment

Dataset	# Nodes	# Edges	# Classes	#Features
Cora	2,708	5,429	7	1,433
Citeseer	3,312	4732	6	3,703
Pubmed	19,717	44,338	3	500

B. Experimental Setup

Unless otherwise noted, we employ coupled graph coarsening based GRAHIES. To examine the capability of GRAHIES framework for improving the performance of existing GNN models, we consider two representative GNN models in Section II-A, GCN [1] and GraphSage [8], and use them as GNN modules in GRAHIES, referred to as GRAHIES-GCN, GRAHIES-Sage respectively. In particular, we use the "mean" variant of GraphSage in which the aggregation takes the element-wise mean value of feature vectors in the neighborhood and the concatenation is used to combine neighborhood features with nodes's own features.

GRAHIES's hyperparameters include the number of graph hierarchy levels and the size of coarsened graphs at different levels. To facilitate hyperparameter tuning, we introduce a coarsening ratio α such that each coarsening level reduces the graph size N to αN. Let L be the number of levels in the graph hierarchy and N be the size of the original graph, the final coarsened graph has size $N\alpha^{L-1}$ which should be no less than one. The optimal settings of these hyperpapameters depends on the dataset since it is correlated with the latent hierarchical structure of graph data. Another hyperparameter is the k value for top-k clusters based projection. The settings for our experiment are given in Table III.

For the transductive tasks on citation networks, we use the same data splitting/preprocessing and follow the exactly same setup as in [1]: we train all models for a maximum of 200 epochs using Adam optimizer [24] with a learning rate of 0.01 ; We employ early stopping with a window size of 10, i.e. we stop training if the validation loss does not decrease for 10 consecutive epochs; We initialize weights using the initialization described in [25] and normalize input feature vectors; We have the hyperparameter settings: 0.5 (dropout rate), 5·10-4 (L2 regularization) and 16 (number of hidden units). For GRAHIES-GCN, we use a single GCN layer as GNN module in each graph level while without introducing trainable layer importance since we found it achieved higher accuracy. The final aggregation function performs concatenation to produce final node representation For node classification, we apply a GCN layer with final node representation as input for prediction. In order to fairly assess the benefits of GRAHIES exploiting hierarchical information, we further compare GRAHIES-GCN with the 2-layer GCN model given in [1] where a GCN layer produces node representation followed by another GCN layer for prediction. The only difference between them is that the final node feature in GRAHIES integrates the features of

TABLE II
HYPERPARAMETER SETTINGS OF GRAHIES-GCN FOR THREE DATASETS

Dataset	#levels	coarsening ratio	k value
Cora	2	0.01	2
Citeseer	3	0.15	1
Pubmed	3	0.08	1

TABLE III
HYPERPARAMETER SETTINGS OF GRAHIES-SAGE FOR THREE DATASETS

Dataset	#levels	coarsening ratio	k value
Cora	2	0.01	2
Citeseer	3	0.08	1
Pubmed	3	0.08	1

coarsened graphs at higher levels. For GRAHIES-Sage, each GNN module consists of two GraphSage aggregator layers. Following the GraphSage model, a fully connected layer is used for label prediction. Similarly, we compare GRAHIES-Sage with a 2-layer GraphSage model.

C. Results Analysis

The results of our comparative evaluation are shown in Table IV. we report the classification accuracy and reuse the metrics reported in [1] for state-of-the-art models including DeepWalk [26], Planetoid [27], and Chebyshev [7]. We group GRAHIES and its corresponding non-hierarchy GNN model together in the table. As we can see, GRAHIES is able to improve upon its base model GCNs by a margin of 1.5%, 1.8% and 2% on three datasets respectively. It suggests that exploiting hierarchy sturcutre in the graph can be beneficial for graph representation learning.

Note the previous works [1], [16] have shown that GCNs with more than 2 layers, even with residual connections, donot perform as well as the 2-layer GCN on citation networks. Our results on GRAHIES show that adding more GCN layers but on different scales(coarsened graphs) improves the performance. The reason could be that, adding more GCN layers on the original graph enlarges the neighborhood for feature aggregation, which allows the node representation to capture higher-order information but at the same time may integrate a lot of irrelevant information and cause the distinct node feature be “washed out” via averaging. However, adding a GCN layer on the cluster-level coarsened graph with feature projection to the original graph avoids this problem. It allows the node representation to capture higher-order graph information while distinguishing the information from different clusters and integrating features from the most relevant clusters via top-k clusters based projection.

Next we examine our design choice for GRAHIES and the impact of model parameters on the performance.

1) Decoupled v.s. Coupled graph coarsening: To understand the benefit of joint learning of graph hierarchy and graph representation, we compare GRAHIES-GCN with its variant with decoupled graph coarsening process. For the decoupled variant, we first run the decoupled graph coarsening on the graph data with sufficient training epochs (1000 epochs in our experiment) to derive the graph hierarchy and then train GNN modules for graph representation learning. We compare GRAHIES-GCN with its decoupled variant and the results are given in Table V As we can see, the decoupled variant has lower accuracy by a margin up tp 2.2%.

TABLE IV
SUMMARY OF RESULTS IN TERMS OF CLASSIFICATION ACCURACIES.

Model	Cora	Citeseer	Pubmed
DeepWalk	67.2%	43.2%	65.3%
Planetoid	75.7%	64.7%	77.2%
Chebyshev	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	76.9%
GRAHIES-GCN	**83.0%**	**72.1%**	**78.9%**
GraphSage	77.5%	65.9%	75.9%
GRAHIES-Sage	**78.9%**	**66.9%**	**N/A**

TABLE V
COUPLED COARSENING V.S. DECOUPLED COARSENING

Model	Cora	Citeseer	Pubmed
GRAHIES-GCN	83.0%	72.1%	78.9%
Decoupled variant	81.0%	71.1%	79.0%

2) Impact of Top-k clusters based projection: To assess the benefit of Top-k clusters based projection, we evaluate the performance of GRAHIES-GCN in the cases of without top-k projection (k=0), and with k values from 1 to 5 on the Cora and Citeseer dataset with other settings as the same as before. The result is show in Table VI. The maximum accuracy occurs at k-2 and 1 for Cora and Citeseer respectively.

3) Impact of coarsening ratio: We choose different coarsening ratio for graph hierarchy given a fixed number of coarsening levels and evaluate the accuracy on Cora dataset. The results are given in Table VII. As we can see that the accuracy is affected by the value of coarsening ratio and highest accuracy got is at 0.01. By evaluating the model accuracy under different parameters settings, we can see that the importance of hierarchy parameters for model accuracy.

V. RELATED WORK

The hierarchy approach to improving the performance on learning graph features have been exploited in a number of recent works [16], [18], [28]. HARP [28] is a hierarchical enhancement to existing random walk based network embedding that learns node embeddings only with topology structure, which is different from GNN that consider both node

TABLE VI
DIFFERENT k FOR TOP-K IMPORTANCE BASED PROJECTION ON CORA

k	0	1	2	3	4	5
cora	82.7	82.6	83.2	83.1	83.1	83.1
citeseer	68.3	72.1	71.4	71.4	71.3	71.3

TABLE VII
DIFFERENT k FOR TOP-K CLUSTERS BASED PROJECTION

coarsening ratio	0.005	0.01	0.02	0.03	0.04	0.05
accuracy	82.4	83.2	80.5	80.4	82.1	81.4

features and network topology. In addition, the approach of HARP for building hierarchy is pure topology based node/edge merging process. In contrast, our graph coarsening approach is designed for attributed graph data. It exploits node features and adaptively learns the hierarchy and also cluster/community level node representations along with GNN training.

DiffPool [18] addresses graph classification that is to is to label a graph, instead of node classification, and aim to learn representation of a whole graph instead of node representation. The hierarchy structure is built through learn-able cluster assignment and used for pooling. Our approach learns the hierarchy structure through node features and used for feature projection. JK-Net [16] considers how to effectively combine the output of different layers within a GNN model on a single graph to improve the node representation. GRAHIES considers the combination at GNN module level. It is straightforward for GRAHIES to use JK-Net to replace GCN/GraphSage as GNN module at each level of graphs in the hierarchy and we will examine its performance gain in the future.

VI. Conclusion

This paper presents a general framework GRAHIES for stacking GNNs to capture inherent hierarchical structure in many realistic networks. By jointly optimizing the hierarchy structure and graph representation, GRAHIES captures the latent hierarchical structure and generates the final node embeddings by effectively projecting the graph representation from higher-level graphs to lower-level. With concatenating the node embedding from different layers, the final node embeddings can be effectively used for node classification. In our future work, we will explore useful herusitic ways to chosse the best hierarchical parameters setting.

Acknowledgment

This work was a part of XDefense umbrella project for robust diversity ensemble against deception. This research is partially sponsored by NSF CISE SaTC (1564097), SAVI/RCN (1402266, 1550379), CNS (1421561), CRISP (1541074), and an REU supplement (1545173). The first author started this research during summer intern 2017 and completed at Georgia Institute of Technology during his PhD (2016-2019). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

References

[1] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," sep 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[2] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 991–1001. [Online]. Available: http://papers.nips.cc/paper/6700-schnet-a-continuous-filter-convolutional-neural-network-for-modeling-quantum-interactions.pdf

[3] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Am. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, May 2007. [Online]. Available: http://dx.doi.org/10.1002/asi.v58:7

[4] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. [Online]. Available: https://www.pnas.org/content/103/23/8577

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3065386

[6] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111. [Online]. Available: http://aclweb.org/anthology/W14-4012

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3844–3852, 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=3157527

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Advances in Neural Information Processing Systems 30*, no. Nips. Curran Associates, Inc., 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs

[9] S. Verma and Z.-L. Zhang, "Graph Capsule Convolutional Neural Networks," may 2018. [Online]. Available: http://arxiv.org/abs/1805.08090

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," 2017. [Online]. Available: http://arxiv.org/abs/1710.10903

[11] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated Graph Sequence Neural Networks," nov 2015. [Online]. Available: http://arxiv.org/abs/1511.05493

[12] H. Gao, Z. Wang, and S. Ji, "Large-Scale Learnable Graph Convolutional Networks," in *ACM SIGKDD - KDD '18*. New York, New York, USA: ACM Press, 2018, pp. 1416–1424. [Online]. Available: http://arxiv.org/abs/1808.03965%0Ahttp://dx.doi.org/10.1145/3219819.3219947

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" *ICLR*, vol. 69, no. 4, pp. 659–677, oct 2019. [Online]. Available: http://arxiv.org/abs/1810.00826

[14] A. Garcia-Duran and M. Niepert, "Learning Graph Representations with Embedding Propagation," 2017. [Online]. Available: http://arxiv.org/abs/1710.03059

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," apr 2017. [Online]. Available: http://arxiv.org/abs/1704.01212

[16] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks," in *ICML*, jun 2018. [Online]. Available: http://arxiv.org/abs/1806.03536

[17] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI*, 2018.

[18] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," in *NeurIPS*, jun 2018.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[20] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug 2000.

[21] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, vol. 22, no. 3, pp. 493–521, may 2011. [Online]. Available: http://link.springer.com/10.1007/s10618-010-0181-y

[22] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *2013 IEEE 13th International Conference on Data Mining*, Dec 2013, pp. 1151–1156. [Online]. Available: https://cs.stanford.edu/ jure/pubs/cesna-icdm13.pdf

[23] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large

attribute networks," in *AAAI*, 2016. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11964

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[25] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9. PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *ACM SIGKDD - KDD '14*. ACM Press, 2014, pp. 701–710. [Online]. Available: http://arxiv.org/abs/1403.6652%0Ahttp://dx.doi.org/10.1145/2623330.2623732

[27] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting Semi-Supervised Learning with Graph Embeddings," mar 2016. [Online]. Available: http://arxiv.org/abs/1603.08861

[28] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "HARP: Hierarchical Representation Learning for Networks," *aaai*, 2017. [Online]. Available: http://arxiv.org/abs/1706.07845