Semi-Supervised Hierarchical Graph Classification

Jia Li, Yongfeng Huang, Heng Chang, and Yu Rong

Abstract—Node classification and graph classification are two graph learning problems that predict the class label of a node and the class label of a graph respectively. A node of a graph usually represents a real-world entity, e.g., a user in a social network, or a document in a document citation network. In this work, we consider a more challenging but practically useful setting, in which a node itself is a graph instance. This leads to a hierarchical graph perspective which arises in many domains such as social network, biological network and document collection. We study the node classification problem in the hierarchical graph where a “node” is a graph instance. As labels are usually limited, we design a novel semi-supervised solution named SEAL-CI. SEAL-CI adopts an iterative framework that takes turns to update two modules, one working at the graph instance level and the other at the hierarchical graph level. To enforce a consistency among different levels of hierarchical graph, we propose the Hierarchical Graph Mutual Information (HDMI) and further present a way to compute HDMI with theoretical guarantee. We demonstrate the effectiveness of this hierarchical graph modeling and the proposed SEAL-CI method on text and social network data.

Index Terms—Hierarchical graph, graph representation, graph mutual information, semi-supervised learning

1 INTRODUCTION

Graph has been widely used to model real-world entities and the relationship among them. Two graph learning problems have received a lot of attention recently, i.e., node classification and graph classification. Node classification is to predict the class label of nodes in a graph, for which many studies [1], [2], [3] in the literature make use of the connections between nodes to boost the classification performance. For example, [4] enhances the recommendation precision in LinkDeep by taking advantage of the interaction network, and [5] improves the performance of document classification by exploiting the citation network. Graph classification, on the other hand, is to predict the class label of graphs, for which various graph kernels [6], [7], [8], [9] and deep learning approaches [10], [11], [12] have been designed. In this work, we consider a more challenging but practically useful setting, in which a node itself is a graph instance. This leads to a hierarchical graph in which a set of graph instances are interconnected via edges. This is a very expressive data representation, as it considers the relationship between graph instances, rather than treating them independently. The hierarchical graph model applies to many real-world data, for example, a social network can be modeled as a hierarchical graph, in which a user group is represented by a graph instance and treated as a node in the hierarchical graph, and then a number of user groups are interconnected via interactions or common members. Another example, a document collection can be modeled as a hierarchical graph, in which a document is regarded as a graph-of-words [13], and then a set of documents are interconnected via the citation relationship. In this paper, we study hierarchical graph classification, which predicts the class label of graph instances in a hierarchical graph.

To represent a hierarchical graph, there is a natural and important question: “given multiple levels of inputs and representations of a hierarchical graph, how can we enforce a consistency among different levels of the graph?” In this work, we propose to use mutual information (MI) to enforce this consistency. We are motivated by recent developments of graph MI maximization methods [14], [15], and generalize the MI computation to hierarchical graphs, which is named Hierarchical Graph Mutual Information (HDMI). Our theoretical derivations show that HDMI can be decomposed into a linear combination of node-level MI [14] and graph-level MI [15]. In this regard, we can use non-hierarchical graph MI computational methods to compute HDMI. More specifically, for graph instances, we compute MI between nodes and instance representations via graph-level MI computational methods (e.g., INFOGRAPH [15]); for connections between graph instances, we compute MI between instances and hierarchical graph representations via node-level MI computational methods (e.g., GMI [14]).

Another challenge is that the amount of available class labels is usually very small in real-world data, which limits the classification performance. To address this challenge, we take a semi-supervised learning approach to solving the graph classification problem. We design an iterative...
algorithm framework which takes turns to update two modules: Instance Classifier (IC) and Hierarchical Classifier (HC). We start with the limited labeled training set and build IC, which produces the embedding vectors of graph instances. HC takes the embedding vectors as input and produces predictions. We cautiously select a subset of predicted labels with high confidence to enlarge the training set. The enlarged training set is then fed into IC in the next iteration to update its parameters in the hope of generating more accurate embedding vectors. HC further takes the new embedding vectors for model update and class prediction. This is our proposed solution, called SEmi-supervisedグラフクラッジファイツィオンVia Cautious Iteration (SEAL-CI), to the graph classification problem.

Our contributions are summarized as follows.

- We study semi-supervised hierarchical graph classification, which is scarcely studied in the literature. Our proposed solution SEAL-CI achieves superior classification performance to the state-of-the-other graph kernel and deep learning methods, even when given very few labeled training instances.
- We generalize the MI estimation to the hierarchical graph domain and propose a new concept named Hierarchical Graph Mutual Information (HGMI). We show HGMI can be decomposed by a sum of mutual information in non-hierarchical graph domain. Upon this, HGMI maximization can be achieved with the help of node/graph-level MI maximization.
- We present the HIERARCHICAL GRAPH BENCHMARK 1 with both text data and social network data, with the goal of facilitating reproducible hierarchical graph research. From the social networking platform Tencent QQ, we collect 37,836 QQ groups with 18,422,331 unique anonymized users, in which the hierarchical graph is constructed with common memberships (hierarchy-level) and friendships (instance-level). From the arXiv papers, we collect 4,666 papers, in which the hierarchical graph is constructed with citations (hierarchy-level) and semantics (instance-level).

The remainder of this paper is organized as follows. Section 2 discusses the related works and Section 3 gives the problem definition. Section 4 describes the design of SEAL-CI. We report the experimental results in Section 5. Finally, Section 6 concludes the paper.

2 RELATED WORK

Hierarchical Graph Representation. In the literature, most graph representation methods focus on node-level representations [16], [17], [18], [19] or graph-level representations [20], [21], [22]. One connection between these two kinds of representations is that graph-level representations could be obtained by integrating node-level representations with graph pooling operations, e.g., DIFFPOOL [23], SAGPool [24], Attention-Pool [25], MinCut-Pool [26]. Until recently Li et al. [25] firstly formulate the hierarchical graph representation problem and evaluate the effectiveness of hierarchical graph representation on social network classification tasks. Wang et al. [27] propose a Graph of Graphs Neural Network (GoGNN) and evaluate the proposed GoGNN on entity interaction prediction tasks. Xu et al. [28] consider city-level graph representations and station-level graph representations to facilitate the air quality forecasting tasks. MIRACLE [29] utilizes the multi-view contrast learning method to derive hierarchical graph representations and validated its effectiveness on Drug-Drug Interactions (DDI). However, currently there are still several problems within the hierarchical graph representation research. First, most previous works are based on heuristics and lack the theoretical derivations of consistent representations among different levels of the hierarchical graph. Second, most works mildly evaluate the expressiveness of hierarchical graph representations on one task, which makes the whole story less convincing.

Graph Mutual Information Maximization. There has been a surging interest in using MI to derive unsupervised graph representations recently. For node-level representations, DGI [30] maximizes MI between a graph summary representation and node-level representations and shows that maximizing this kind of MI is equivalent to maximizing the one between the node features and node-level representations. Later, GMI [14] directly approaches MI computation by comparing the node input and node-level representations, without the overheads of an extra graph summary representation. For graph-level representations, INFOGRAPH [15] and MVGRL [31] extend the idea of DGI to the field of graph-level representation and contrasts graph-level and subgraph-level representations. Our work advances this research area by proposing hierarchical graph MI and develops an tractable computation method with guarantees.

Semi-Supervised Graph Learning In many graph applications, the size of labeled instances is typically limited. In addition to labeled instances, semi-supervised graph learning seeks to leverage unlabeled instances within the network structure to obtain gains in performance. In the literature, semi-supervised graph learning techniques can be generally classified into two groups. The first group is consistency regularization. Representative works include graph Laplacian regularization [32], [33], graph contrastive regularization [15], [34], [35]. This group aims to learn a representation function that gives consistent predictions for similar data points. The second group is pseudo-label. As the name indicates, this group uses a trained model on the labeled set to produce additional training examples by labeling instances of the unlabeled set [36]. Representative works include ICA [5], Cautious Iteration [37], [38]. In ICA, for each node a local classifier takes the estimated labels of its neighborhood and its own features as input, and outputs a new pseudo label. The iteration continues until adjacent estimations stabilize. In this work, our proposed method SEAL-CI connects the idea of both consistency regularization and pseudo-label. We show our algorithm has a convergence property w.r.t. the empirical risk.

3 PROBLEM DEFINITION

We denote a set of objects as \( O = \{ o_1, o_2, \ldots, o_N \} \) which represent real-world entities. We use \( d \) attributes to describe properties of objects, e.g., age, gender.
We use a graph instance to model the relationship between objects in $O$, which is denoted as $g = (V, A, X)$, $V \subseteq O$ is the node set and $|V| = n$, $A$ is an $n \times n$ adjacency matrix representing the connectivity in $g$, and $X \in \mathbb{R}^{n \times d}$ is a matrix recording the attribute values of all nodes in $g$.

A set of graph instances $G = \{g_1, g_2, \ldots\}$ can be interconnected, and the connectivity between the graph instances is represented by an adjacency matrix $A$. The graph instances and their connections are modeled as a hierarchical graph $G = (G, A)$.

A graph instance $g \in G$ is a labeled graph if it has a class label, represented by a vector $y \in \{0, 1\}^c$, where $c$ is the number of classes. A graph instance is unlabeled if its class label is unknown. Then $G$ can be divided into two subsets: labeled graphs $G_l$ and unlabeled graphs $G_u$, where $G = G_l \cup G_u$, $|G_l| = L$ and $|G_u| = U$, $Y = \{y_i\}_{i=1}^L$ denotes the labeled ground truth. In this paper, we study the problem of graph classification, which determines the class label of the unlabeled graph instances in $G_u$ from the available class labels in $G_l$ and the hierarchical graph topological structure. As the amount of available class labels is usually very limited in real-world data, we take a semi-supervised learning approach to solving this problem.

Our problem formulation is applicable to many real applications as illustrated below.

Application 1. Consider a social network where we want to infer the topics of user groups, as depicted in Fig. 1. $A, B, C, D$ denote four user groups. Group $A$ has the class label of “game”, $B$ has the label of “non-game”, while the class labels of $C$ and $D$ are unknown. These four groups are connected via some kind of relationships, e.g., interactions or common members. The internal structure of each user group shows the connections between individual users. From this hierarchical graph, we want to determine the class labels of $C$ and $D$.

Application 2. Consider an information retrieval scenario where we want to category documents. We can model a document as a graph-of-words, i.e., graph instances in a hierarchical graph. We then connect these documents by their citation relationships, i.e., connections between graph instances in a hierarchical graph. With this setting, we can achieve better classification performance by taking advantage of both out-of-document relations and inside-of-document semantics (See details in Section 5.2).

Application 3. Protein-Protein Interaction (PPI) sheds light on biological processes, of which one important task is to predict properties of proteins. We can model a protein as a graph of amino-acid interactions [39], i.e., graph instances in a hierarchical graph. We then connect these proteins by their interactions, forming a hierarchical graph. With this formulation, we can achieve better prediction performance by considering both out-of-protein interactions and inside-of-protein interactions.

4 METHODOLOGY

4.1 Problem Formulation

In our problem setting, we have two kinds of information: graph instances and connections between the graph instances, which provide us with two perspectives to tackle the graph classification problem. Accordingly, we build two classifiers: a classifier IC constructed for graph instances and a classifier HC constructed for the hierarchical graph.

For both classifiers, one goal is to minimize the supervised loss, which measures the distance between the predicted class probabilities and the true labels. Another goal is to maximize a hierarchical graph mutual information, which measures the distance among the input hierarchical graph, IC representations and HC representations. The purpose of this hierarchical graph mutual information is to enforce a consistency among different levels of hierarchical graph.

Formally, we formulate the graph classification problem as an optimization problem:

$$\arg\min \xi(G_i) - \xi(G),$$

where $\xi(G_i)$ is the empirical risk for the labeled graph instances, and $\xi(G)$ is the hierarchical graph mutual information for all graph instances.

Specifically, $\xi(G_i)$ includes two parts:

$$\xi(G_i) = \frac{1}{L} \sum_{g_i \in G_i} \left( CE(y_i, e_i) + CE(y_i, \gamma_i) \right),$$

where $e_i$ is a vector of predicted class probabilities by IC, and $\gamma_i$ is a vector of predicted class probabilities by HC. $CE(\cdot, \cdot)$ is the cross-entropy loss function.

The hierarchical graph mutual information (HGMI) $\xi(\cdot)$ is defined as:

$$\xi(G) = HGMI(G) = I(G; E; \Gamma),$$

where $I(\cdot; \cdot)$ denotes the mutual information between a set of three variables, $E = \{e_i\}_{i=1}^{L+U}$ denotes the representations derived by IC and $\Gamma = \{\gamma_i\}_{i=1}^{L+U}$ denotes the representations derived by HC. In the following subsections, we first analyse HGMI and describe the way to compute HGMI; we then give our design of classifiers IC and HC, and our detailed training algorithms.

4.2 Analysis of HGMI

We first show HGMI coincides with graph mutual information between $G$ and $\Gamma$ when the hierarchical graph forms a Markov chain.
Theorem 1. Consider the hierarchical graph forms a Markov chain, i.e., \( G \rightarrow E \rightarrow \Gamma \), then
\[
I(G; E; \Gamma) = I(G; \Gamma),
\]
here \( I(G; \Gamma) \) is the mutual information between the input graph instances and hierarchical graph representations.

The proof is trivial, since the three variables form a Markov chain \( G \rightarrow E \rightarrow \Gamma \), then \( I(G; \Gamma | E) = 0 \), we have
\[
I(G; E; \Gamma) = I(G; \Gamma) - I(G; \Gamma | E) = I(G; \Gamma).
\]

It was worth mentioning that the Markov property is quite reasonable here. For example on text data, upon given document representation, it is reasonable to assume graph-of-words inputs (within documents) are irrelevant when predicting document citations (outside of documents).

We have the following hierarchical graph mutual information decomposition theorem to compute HGMI.

Theorem 2. Consider the hierarchical graph forms a Markov chain, the hierarchical graph mutual information can be decomposed by a sum of non-hierarchical graph mutual information, namely,
\[
I(G; E; \Gamma) = \alpha (I(G; E) + I(E; \Gamma)),
\]
here \( \alpha \in [0,1] \), \( I(G; E) \) is the mutual information between graph instance input and graph representation on instance level, \( I(E; \Gamma) \) is the mutual information between instance representation and hierarchical graph representation on hierarchical level.

To prove Theorem 2, we first introduce two lemmas.

Lemma 1. For any random variable \( Z_1, Z_2, Z_3 \), we have
\[
I(Z_1; Z_2, Z_3) \geq \frac{1}{2} (I(Z_1; Z_2) + I(Z_1; Z_3)),
\]
here \( I(Z_1; Z_2, Z_3) \) is the mutual information between variable \( Z_1 \) and the joint distribution of \( Z_2 \) and \( Z_3 \).

To prove Lemma 1, we make use of the chain rule for mutual information.
\[
I(Z_1; Z_2, Z_3) = I(Z_1; Z_3) + I(Z_1; Z_2 | Z_3) \geq I(Z_1; Z_3).
\]

The last inequality holds as mutual information is non-negative. Accordingly, we have
\[
I(Z_1; Z_2, Z_3) \geq I(Z_1; Z_2).
\]

Based on Eqs. (8) and (9), we complete the proof of Lemma 1.

Lemma 2. For a Markov chain \( G \rightarrow E \rightarrow \Gamma \), we have
\[
I(E; G, \Gamma) \leq I(E; G) + I(E; \Gamma).
\]

Proof. According to [40], if the conditional distribution \( P(Z_3 | Z_1, Z_2) \) is multiplicative, i.e., \( \exists \) two functions \( s_1 \) and \( s_2 \), s.t., \( P(Z_3 | Z_1, Z_2) = s_1(Z_1) s_2(Z_2) \), then \( I(Z_1; Z_2) \geq I(Z_1; Z_3) \). Since \( G \rightarrow E \rightarrow \Gamma \) is a Markov chain, we have \( P(\Gamma | G, E) = P(\Gamma | E) \), which means \( P(\Gamma | G, E) \) is multiplicative. Thus, \( I(G; E; \Gamma) \) holds in our case.

Thus, we complete the proof of Lemma 2. We then prove Theorem 2:
\[
I(G; E; \Gamma) = I(G; E) - I(G; E; \Gamma)
\]
\[
= I(G; E) - (I(E; G, \Gamma) - I(E; \Gamma))
\]
\[
= I(G; E) + I(E; \Gamma) - I(G; E, \Gamma)
\]
\[
= \alpha (I(G; E) + I(E; \Gamma)).
\]

The last equality holds as we have \( \frac{1}{2} (I(G; E) + I(E; \Gamma)) \leq I(E; G, \Gamma) \leq I(G; E) + I(E; \Gamma) \), based on Lemma 1 and Lemma 2. Thus, Theorem 2 is proved.

We use Fig. 2 to illustrate the idea to compute HGMI. One naive method to compute HGMI is according to the definition, which inevitably involves with the computation of joint distribution \( P(G, E, \Gamma) \) and is not tractable in general [41]. With Theorem 2, we decouple the computation of HGMI into a sum of the computation of graph mutual information without hierarchies, which is tractable as used in many previous works [14], [30]. Next we present the design of IC and HC classifier, and then give the detailed way to maximize HGMI.

### 4.3 Design of Classifiers

Classifier IC takes a graph instance as input. As different graph instances have different numbers of nodes, IC is expected to handle graph instances of arbitrary size. Classifier HC takes the hierarchical graph as input, in which individual graph instances are the “nodes”. We propose to embed a graph instance \( g_i \in G \) into a fixed-length vector \( e_i \) via IC first. Then HC can take as input the embedding vectors of graph instances and the adjacency matrix \( A \). In particular, IC takes as input the adjacency matrix \( A \) and attribute matrix \( X \) of an arbitrary-sized graph instance \( g_i \), and outputs an instance representation \( e_i \), i.e., \( e_i = IC(A_i, X) \). HC takes the instance representations \( E = \{ e_i \}_{i=1}^{L+U} \) and \( A \), and outputs the predicted hierarchical graph representations \( \Gamma = \{ \gamma_i \}_{i=1}^{L+U} \). In the following, we illustrate the...
design of IC which performs instance graph representation, and then the design of HC which performs hierarchical graph representation.

4.3.1 Graph Instance Representation

The task of graph instance representation is to produce a fixed-length embedding vector of a graph instance, for which, however, we identify two challenges:

- **Size invariance**: How to design the network structure to flexibly take an arbitrary-sized graph instance and produce a fixed-length embedding vector?
- **Permutation invariance**: How to derive the representation regardless of the permutation of nodes?

While the concept of size invariance is straightforward, we specifically provide the definition of permutation invariance on graphs here:

**Definition 1 (Permutation Invariance on graphs).** Let \( P \in \{0, 1\}^{n \times n} \) be any permutation matrix of order \( n \), then the permutation to a graph instance \( g \) is defined as a mapping of the node indices, i.e.:

\[
P_g = (PAP^T, PX).
\]

Then a function \( f \) is invariant to permutation on \( g \) if

\[
f(A, X) = f(PAP^T, PX).
\]

Recently graph pooling has emerged as a research topic that can be used to tackle the above challenges. Some common practices include DIFFPOOL [23], Attention-based Pool [24], [25], [42], MinCut-Pool [26]. In particular, [42] shows Attention-based Pool is permutation-invariant by connecting with Weisfeiler-Lehman test. To this end, in IC, we first utilize a multi-layer Graph Neural Network (GNN) [18] to smooth each node’s features over the graph’s topology. Then we adopt a graph pooling method and transform a variable number of smoothed nodes into a fixed-length graph instance representation \( e \).

Formally, we are given the adjacency matrix \( A \in \mathbb{R}^{n \times n} \) and the attribute matrix \( X \in \mathbb{R}^{n \times d} \) of a graph instance \( g \) as inputs. We apply a multi-layer GNN network:

\[
H = \text{GNN}(A, X),
\]

where \( H \) is the derived hierarchical graph representation, \( e \) is the output of the pooling function used in [25]:

\[
S = \text{softmax}(W_s \tanh(W_h H^T)), \quad e = SH,
\]

where \( W_s \) and \( W_h \) are two weight matrices. \( H \) is the same after nodes permutation from Eq. (15) and our example pooling function Eq. (16) is only relevant to \( H \), which reveals the permutation invariance of IC. Thus we finish the proof.

To summarize, we use GNN and graph pooling that are permutation invariant to construct the instance-level classifier IC. It produces graph instance representations \( E = \{e_i\}_{i=1}^n \) while successfully dealing with the aforementioned two challenges, which is the input for classifier HC in the next.

4.3.2 Hierarchical Graph Representation

Given \( E \) and the adjacency matrix \( A \in \mathbb{R}^{(L+U) \times (L+U)} \), our next task is to infer the parameters of classifier HC and derive the predicted probabilities \( \Gamma = \{y_i\}_{i=1}^{L+U} \). This problem falls into the setting of non-hierarchical graph setting where \( E \) can be treated as the set of node features. In this context, we consider again a multi-layer GNN. Thus the model becomes:

\[
\Gamma = HC(E, A) = \text{softmax}(\text{GNN}(E, A)),
\]

where \( \Gamma \in \mathbb{R}^{(L+U) \times c} \) is the derived hierarchical graph representation. With \( \Gamma \) and \( E \), next we introduce the way to maximize HGMI.

4.4 Maximization of HGMI

With Theorem 2, the computation of HGMI is reduced into MI between graph input and graph-level representation \( I(G; E) \) and MI between graph input and node-level representation \( I(E; \Gamma) \), in which we have many choices such as INFOGRAPH [15] and GMI [14].

To compute \( I(G; E) \), we use the method of INFOGRAPH [15], where MI is computed between the graph input \( G \) and graph-level representation \( E \). More specifically, it is shown in [15] that maximizing the global \( I(G; E) \) can be estimated by MI between node representations and graph-level representations,

\[
I(G; E) = \sum_{i=1}^{L+U} \frac{1}{L+U} \sum_{j=1}^{n} \frac{1}{n} I(h_{ij}; e_i),
\]

where \( h_{ij} \) is a node representation sampled from \( H_i \). The core thus becomes how we compute \( I(h_{ij}; e_i) \). In this work, we resort to Jensen-Shannon MI estimator (JSD),

\[
I(h_{ij}; e_i) = -sp(-D(h_{ij}, e_i) - E_{h_{ij}}sp(D(h_{ij}, e_i)),
\]

Proposition 1. Let \( P \in \{0, 1\}^{n \times n} \) be any permutation matrix, then \( IC(A, X) = IC(PAP^T, PX) \) as long as the chosen graph pooling method and GNN are permutation invariant.
where $D_1 : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a discriminator constructed by a neural network with parameters $D_1$, i.e., $D_1(h_{ij}, e_i) = \sigma(h_{ij} W_{D_1} e_i)$ and $W_{D_1}$ is a learnable scoring matrix. $h_{ij}$ is a negative example. $sp(x) = \log(1 + \exp(x))$ is the soft-plus function. For a detailed graph-level negative sampling process, please see INFOGRAPH [15].

To compute $I(E; \Gamma)$, we adopt the method of GMI [14], where MI is computed between the graph feature input and node-level representation. More specifically, it is shown in [14] that, for a node-level representation $\gamma_i$, maximizing the global $I(E; \gamma_i)$ can be decomposed as a weighted sum of local MIs,

$$I(E; \gamma_i) = \sum_j w_{ij} I(e_j; \gamma_i),$$

(20)

here we adopt the mean version of GMI, meaning $w_{ij} = \frac{1}{L+U}$. To compute $I(e_j; \gamma_i)$, we use JSD again,

$$I(e_j; \gamma_i) = -sp(-D_{H}(e_j, \gamma_i) - E_{ij} sp(D_{H}(e_j, \gamma_i))),$$

(21)

where $D_{H} : \mathbb{R}^m \times \mathbb{R}^c \rightarrow \mathbb{R}$ is a discriminator constructed by a neural network with parameters $D_{H}$. $e_j$ is a negative example. For a detailed node-level negative sampling process, please see GMI [14].

### 4.5 The Proposed SEAL-CI Model

In this subsection, we present our method to minimize the objective function Eq. (1). A naive way would be directly minimizing Eq. (1) in an end-to-end fashion with the limited labels. This algorithm is called SEMi-supervised Graph Classification (SEAL). However, in real-world scenarios, the number of labeled graph instances $L$ can be quite small compared to the number of unlabeled instances $U$. In this context, neural network based classifiers may suffer from the problem of overfitting.

Following previous works [5], [37], we use the idea of iterative algorithm to alternate optimizing the two modules of IC and HC by trusting a subset of predictions. To be more specific, we combine the instance representation in Section 4.3.1 and hierarchical representation in Section 4.3.2 into one iterative algorithm. We build IC to produce instance representation $E'$ for all graph instances in iteration $t$, and then feed $E'$ into HC to get the predicted probabilities $\Gamma'$. We then make use of pseudo-labeling strategy to update the parameters of IC and generate $E''$, which is then used as the input of HC in iteration $t+1$. Fig. 3 depicts the overall framework of this iterative process.

#### Algorithm 1. SEAL-CI

**Input:** $\mathcal{G} = \{G, A\}$.
**Output:** $\Gamma^{t+1}, E^{t+1}$

1. **Initial:** $G_{tmp} = \emptyset, G^0_t = G_t, t = 0$;
2. **while** $t \lambda \leq U$ **do**
3. $W^{t+1} \leftarrow \arg \min \xi(G^0_t) \mid \Psi^0_t - \xi(G^0_t) \mid \Psi^0$;
4. $E^{t+1} \leftarrow IC(A, X) \mid \Psi^t$;
5. $\Gamma^{t+1} \leftarrow HC(E^{t+1}, A) \mid \Psi^{t+1}$;
6. $G_{tmp} \leftarrow T(t \lambda, (E^t_{G_t}, G_{tmp}^t))$;
7. $G^{t+1}_t \leftarrow G_t \cup G_{tmp}$;
8. $G_{tmp} = \emptyset$;
9. **Return** $\Gamma^{t+1}, E^{t+1}$;

#### 4.5.1 Pseudo-Labeling

To update the instance representations, a naive approach is feeding the whole set of $(E^t, \Gamma^t)$ as pseudo labels for the parameter update in IC, which is the idea of the original ICA [5]. However, not all $(E^t, \Gamma^t)$ are correct in their predictions. The false predictions may lead the update of embedding neural network to the wrong direction. Within the set of unlabeled graphs, different $(E^t, \Gamma^t)$ could contribute differently to the update of embedding neural network. To this end, we make use of the idea of cautious iteration [37] and curriculum learning [43], and cautiously exploit a subset of $(E^t, \Gamma^t)$ to update the parameters of IC in each iteration. Specifically, in iteration $t$, we choose the $t \lambda$ most confident predicted labels in both IC and HC while ignoring the less confident predicted labels. This operation continues until all the unlabeled samples have been utilized or until a predefined iteration number. This algorithm is called SEMi-supervised Graph Classification via Cautious Iteration (SEAL-CI).
and is presented in Algorithm 1. Note here $W$ is the set of all the parameters of IC and HC. In line 3, SEAL is well trained to provide a foundation for SEAL-CI to find informative and accurate pseudo labels thereafter. In line 6, the training set for IC has been enlarged by $t\lambda$ instances and it is done by “committing” these instances’ labels from their maximum probability. In other words, the newly enrolled training instances are found by:

$$T(t\lambda, (E, \Gamma)) = \text{top}(\max((E, \Gamma)), t\lambda),$$  

(22)

here function $\text{top}(\cdot, \lambda)$ is used to select the top $t\lambda$ instances and function $\max(\cdot)$ is used to select the maximum value in a set.

### 4.5.2 Analysis of SEAL-CI

In this part, we discuss about the convergence properties of proposed SEAL-CI algorithm. We first show the empirical risk is a monotonically decreasing function with respect to the iteration step.

**Theorem 3.** Let $\zeta(W)$ be the empirical risk, $W^t(t = 1, 2, \cdots)$ be the parameter set of algorithm 1 at iteration step $t$, $\zeta(W^t)$ be the empirical risk at the corresponding iteration step. Then $\zeta(W^t)$ is a monotonically decreasing function, namely,

$$\zeta(W^{t+1}) \leq \zeta(W^t),$$  

(23)

here $\zeta(W^t)$ is short for $\zeta(G_t|W^t)$ and the size of $G_t$ will be dynamically increased as we add pseudo-labeled instances.

The cautious selection function $T$ in the pseudo labeling process provides a Bayesian prior for incorporating unlabeled instances, which can be formalized as follows:

$$\zeta(W^{t+1}) = \frac{1}{L+1} (\text{CE}(Y, E_t) + \text{CE}(Y, \Gamma_t) + \text{CE}(y'_t, e_t) + \text{CE}(y_t, \gamma_t)),$$  

(24)

here $E_t = \{e_i\}_{i=1}^{p}$ and $\Gamma_t = \{\gamma_i\}_{i=1}^{g}$, $\text{CE}(Y, E_t) = \sum_{e_i \in G_t} \text{CE}(y_i, e_i)$, $\text{CE}(\cdot, \cdot)$ is the cross-entropy function. $y'_t$ denotes the pseudo label that we incorporate with $T(e_i, \gamma_t)$ and $\lambda = 1, i.e., we consider increasing the size of training set by one at each iteration. Let $T$ be short for $T(e_i, \gamma_t)$, we have

$$\text{CE}(y_t, \gamma_t) = (L+U)\text{Cov}[\text{CE}(y_t, \gamma_t), T] + E[\text{CE}(Y, \Gamma_t)],$$  

(25)

$$\text{CE}(y_t, e_t) = (L+U)\text{Cov}[\text{CE}(y_t, e_t), T] + E[\text{CE}(Y, E_t)].$$  

(26)

Based on Eqs. (25) and (26), we can rewrite $\zeta(W^{t+1})$:

$$\zeta(W^{t+1}) = \beta \{\text{Cov}[\text{CE}(y'_t, e_t), T] + \text{Cov}[\text{CE}(y'_t, \gamma_t), T]\} + \frac{1}{L} (\text{CE}(Y, E_t) + \text{CE}(Y, \Gamma_t))$$

$$= \beta \{\text{Cov}[\text{CE}(y'_t, e_t), T] + \text{Cov}[\text{CE}(y'_t, \gamma_t), T]\}$$

$$+ \zeta(W^t) \leq \zeta(W^t),$$  

(27)

here $\beta = \frac{L+U}{L}$. The last inequality holds as the labeling strategy $T$ is similar to a discrete sign function. It is easy to see the positive correlation between $\gamma_t$ and $T$. Besides, $y'_t$ and the pseudo labels $y'_t$ are positively correlated. As cross entropy term $\text{CE}(y'_t, \gamma_t)$ is negatively correlated to $y'_t$, the correlation between $T$ and $\text{CE}(y'_t, \gamma_t)$ is negative. Similarly, the correlation between $T$ and $\text{CE}(y'_t, e_t)$ is negative. We thus finish the proof of Theorem 3.

As the empirical risk is a monotonically decreasing function and bounded (i.e., non-negative), we conclude that the proposed SEAL-CI algorithm has a convergence property.

### 5 Experiments

We evaluate SEAL-CI method on synthetic, text and social network data sets.

#### 5.1 Synthetic Data

We evaluate the performance of SEAL-CI on synthetic data. We first give a description of the synthetic generator, then visualize the learned embeddings. Finally we compare our methods with baselines in terms of classification accuracy.

##### 5.1.1 Synthetic Data Generation

The benchmark data set Cora [44] contains 2708 papers which are connected by the citation relationship. We borrow the topological structure of Cora to provide the skeleton (i.e., edges) of our synthetic hierarchical graph. Then we generate a set of graph instances, which serve as the nodes of this hierarchical graph. Since there are 7 classes in Cora, we adopt 7 different graph generation algorithms, that is, Watts-Strogatz [45], Tree graph, Erdős-Rényi [46], Barbell [47], Bipartite graph, Barabási-albert graph [48] and Path graph, to generate 7 different types of graph instances, and connect them in the hierarchical graph.

Specifically, to generate a graph instance $g$, we randomly sample a number from $[100, 200]$ as its size $n$. Then we generate its structure and assign the class label according to the graph generation algorithm. In this step, the parameter $p$ in Watts-Strogatz, Erdős-Rényi, Bipartite graph and Barabási-albert graph is randomly sampled from $[0.1, 0.5]$, the branching factor for Tree graph is randomly sampled from $[1, 3]$. At last, to make this problem more challenging, we randomly remove $1\%$ to $20\%$ edges in the generated graph $g$. The statistics of the generated graph instances are listed in Table 1.

##### 5.1.2 Baselines and Metrics

We use the following approaches as our baselines:

- GK-SVM [8], which calculates the graphlet count kernel (GK) matrix, then GK-SVM feeds the kernel matrix into SVM [49].
- WL-SVM [9], which is similar as above but using the Weisfeiler-Lehman subtree kernel (WL).

| Type            | Number | Nodes | Edges | Density |
|-----------------|--------|-------|-------|---------|
| Watts-Strogatz   | 351    | 173   | 347   | 2.3%    |
| Tree            | 217    | 127   | 120   | 1.5%    |
| Erdős-Rényi     | 418    | 174   | 3045  | 20%     |
| Barbell         | 818    | 169   | 2379  | 16.3%   |
| Bipartite       | 426    | 144   | 1102  | 10.6%   |
| Barabási albert  | 298    | 173   | 509   | 3.4%    |
| Path            | 180    | 175   | 170   | 1.1%    |


5.1.3 Results

Table 2 shows the experimental results for semi-supervised graph classification. Among all approaches, SEAL-CI achieves the best performance. In the following, we analyze the performance of all methods categorized into 3 groups.

Group *1: All the embedding-based methods perform better than these two kernel methods, which proves that embedding vectors are effective representations for graph instances and are suitable input for graph neural networks.

Group *2: graph2vec-GCN achieves 85.2% accuracy, which is comparable to that of SAGE, but lower than that of SEAL-CI. One possible explanation is that graph2vec is an unsupervised embedding method, which fails to generate discriminative embeddings for classification. Another possibility is that the 300 training instances do not include very informative ones. These limitations of graph2vec motivate us to use supervised graph representation modules such as SAGE and the label enlargement framework in SEAL-CI.

Group *3: SEAL-CI outperforms MIRACLE and SEAL, which proves the effectiveness of our hierarchical graph perspective and the label enlargement algorithm for semi-supervised graph classification.

5.2 Text Data

We evaluate SEAL-CI on the arXiv paper dataset. First, We introduce the compositions of arXiv paper dataset and present how to construct hierarchical graph from text data. Then, we show the evaluation results and give some insights on how to construct a hierarchical graph for text data.

5.2.1 Data Description

arXiv is an open-access repository of electronic preprints. We collect 4666 Computer Science (CS) arXiv papers indexed by Microsoft Academic Graph (MAG) [50]. These papers belong to five subject areas including AI, CL, IT, LG and CV. The arXiv papers data forms a citation network which indicates citation relationships. Each paper consists of two parts: title and abstract. The statistics of the arXiv paper dataset is listed in Table 3.

The connections among graph instances (i.e., papers) are provided by citation relations. Each paper is a semantic graph instance constructed from its title and abstract. There are three type of nodes in a semantic graph instance: abstract node, title node and argument nodes. Specifically, for each sentence in abstract, we parse it into a tuple with Semantic Role Labeling (SRL) toolkit developed by AllenNLP.² For each tuple, we regard its elements as the argument nodes of the semantic graph. The node features of the semantic graph are extracted by pretrained Bert model [51]. The connections within the semantic graph are created as the following: 1) same tuple, if two nodes are from the same tuple, we connect them, and 2) vocabulary overlapping, we connect the node pair if the size of overlapped words is larger than half of the minimum size of any text node. In the Fig. 4 we show an example of hierarchical graph of arXiv papers data.

5.2.2 Baselines and Metrics

In addition to the set of baselines in Section 5.1.2, we compare with the following Bert-feature baselines:

- Bert-MLP, which classifies arXiv papers by feeding average features of title and abstract into a multilayer perceptron (MLP).

² http://demo.allennlp.org/semantic-role-labeling
Bert-IC, which feeds semantic graph instances into IC and classifies arXiv papers in an independent graph classification fashion.

Bert-HC, which feeds average features of title and abstract into HC and classifies arXiv papers in a transductive node classification fashion.

400 graph instances are used as labeled training instances for all methods. We use 2000 instances for testing for all methods. We set the dimension of Bert node feature to 768. We use average pool in IC. We set $\lambda = 1$ and updated times $t = 100$ for SEAL-CI. We run each method 3 times and report its average accuracy.

5.2.3 Results

Table 4 shows the experimental results for semi-supervised graph classification on arXiv paper data. SEAL-CI obtains the state-of-the-art performances among all baselines. To analyze the results of these baselines, all methods are divided into four group as the following.

Group *1: Both GK-SVM and WL-SVM only utilize the structural information of semantic graph instances. And their classification accuracies are lower than 40%, which indicates that structural information of semantic graph instances is not enough to obtain satisfied performance on text data. Graph2vec-GCN tries to exploit both structure of semantic graph instances and the hierarchical graph but still has a weak performance with an accuracy of 42.7%.

Group *2: In this group, we show the performance of several baselines based on Bert. Bert-MLP only uses Bert features extracted from title and abstract, and obtains an accuracy of 63.7%. Bert-IC outperforms Bert-MLP by about 6.4%, by combing both structural and Bert features of text data. Bert-HC, by utilizing the connections between Bert features of title and abstract, achieves the best performance among baselines within Group *2.

Group *3: Both SEAL and MIRACLE aim to minimize mutual information between graph instance representations and hierarchical representations and differ in the measurement of consistency. SEAL and MIRACLE outperform baselines in Group *1 with a margin of 30% and Group *2 with a margin of more than 3%.

Group *4: SEAL-CI outperforms MIRACLE and SEAL, which demonstrates the effectiveness of the cautious iteration algorithm 1.

5.2.4 Influence of the Number of Labeled Training Instances

We examine how the number of labeled training instances affects the performance of our methods. We train Bert-IC, Bert-HC, SEAL and SEAL-CI with a label size of $\{100, 150, 200, 250, 300, 350, 400\}$. We set $\lambda = 1$ in SEAL-CI. We run all methods 5 times, and plot their average accuracy in Fig. 5. As we can see from Fig. 5, SEAL-CI performs the best since it can utilize more training samples. SEAL outperform Bert-HC and Bert-IC in all cases, which makes sense because SEAL makes good use of the hierarchical graph setting and consider the connections between the graph instances for classification. Meanwhile, as SEAL-CI is based on SEAL, the good performance of SEAL provides a foundation for SEAL-CI to find informative and accurate pseudo labels, which could further boost the performance.

5.2.5 Discussion

How to construct hierarchical graphs from text data is an open question. In the above experiment, we use semantic

| TABLE 4 | Comparison of Different Methods on arXiv Paper Data for Semi-Supervised Graph Classification |
|---------|------------------------------------------|
| Algorithm | Accuracy |
| GK-SVM | 38.7% |
| WL-SVM | 35.1% |
| graph2vec-GCN | 42.7% |
| GAT | 69.3% |
| GIN | 69.1% |
| Bert-MLP | 63.7% |
| Bert-IC | 70.1% |
| Bert-HC | 75.2% |
| MIRACLE | 78.4% |
| SEAL | 78.8% |
| SEAL-CI | 79.4% |

Fig. 4. An illustration of the hierarchical graph of arXiv papers data.

Fig. 5. Accuracy with different number of labeled training instances on text data for semi-supervised graph classification.
paraphrasing techniques to construct graph instances for papers, and citation relationships to connect these papers. We think hierarchical graph structures could be a good paradigm to boost the performance of Natural Language Processing classification tasks. Moreover, it is worth mentioning that there are several other ways to construct semantic graphs such as Name Entity Recognition (NER).

### 5.3 Social Network Data

In this section, we evaluate SEAL-CI on Tencent QQ group data. We describe the characteristics of this data set and then present the experimental results. Finally, we have some open discussions on how to construct a hierarchical graph for social network data.

#### 5.3.1 Data Description

Tencent QQ is a social networking platform in China with nearly 800 million monthly active users. There are around 100 million active online QQ groups. In this experiment, we select 37,836 QQ groups with 18,422,331 unique anonymized users. For each user, we extract seven personal features:

- number of days ever since the registration day;
- most frequently active area code in the past 90 days;
- number of friends;
- number of active days in the past 30 days;
- number of logging in the past 30 days;
- number of messages sent in the past 30 days;
- number of messages sent within QQ groups in the past 30 days.

We have 298,837,578 friend relationships among these users. 1,773 groups are labeled as “game” and the remaining groups are labeled as “non-game”.

We construct the hierarchical graph from this Tencent QQ group data as follows. A user is treated as an object, and a QQ group as a graph instance. The users in one group are connected by their friendship. The attribute matrix $X$ is filled with the attribute values of the users. The statistics of the graph instances are listed in Table 5. We build the hierarchical graph from the graph instances via common members across groups, that is, if two groups have more than one common member, we connect them.

#### 5.3.2 Baselines and Metrics

We use the same set of baselines as in Section 5.1.2. 1000 graph instances are used as labeled training instances for all methods. We use 10,000 instances for testing for all methods. We run each method 3 times and report its average performance. For IC, we use SAGE and set the parameters as the same in Section 5.1. Since the class distribution is extremely imbalanced in this data set, we report the Macro-F1 instead of accuracy.

#### 5.3.3 Results

Table 6 shows the experimental results. SEAL-C/AI outperform GK, WL and graph2vec by at least 36% in Macro-F1. SEAL beats MIRACLE by about 4%. SEAL-CI outperforms SEAL and MIRACLE with a margin of more than 1.2%, which demonstrates the effectiveness of the cautious iteration Algorithm 1 on social network data. Next we consider the relation between $\lambda$ and the false prediction rate (i.e., the percentage of misclassified instances) in the pseudo-labeling. Fig. 6 shows the false prediction rate within the most confident predictions of IC. As we can see, the false prediction rate increases as $\lambda$ increases and it reaches 2.4% when $\lambda = 2000$. In the framework of SEAL-CI, as the iteration goes on, we shall bring in more noise to the parameter update, while all the training samples in SEAL are informative and correct. This observation tells us when using SEAL-CI, we need to be “cautious” enough, i.e., $\lambda$ should be small enough, to ensure the false prediction rate is small.

#### 5.3.4 Visualization

We provide visualization of a “game” group and its neighborhood in Fig. 7. The left part is the ego network of the center “game” group. In the one-hop neighborhood of this “game” group, there are 10 “game” groups and 19 “non-game” groups. “Game” groups are densely interconnected with a density of 34.5%, whereas “non-game” groups are sparsely connected with a density of 8.8%. The much higher density among “game” groups validates that common membership is an effective way to relate them in a hierarchical graph for classification. The right part depicts the internal structure of the ego “game” group with 22 users. A bigger
node indicates a larger importance obtained by SAGE, and a darker green color implies a larger node degree. These 22 members are loosely connected and there are no triangles. This makes sense because in reality online “game” groups are not acquaintance networks. Regarding the learned node importance, node 1 has the highest importance as it is the second active member and has a large degree in this group. Node 16 is also important since it has the highest degree in this group. The “border” member 5 has a big attention weight since it has the largest number of days ever since the registration day and is quite active in this group.

5.3.5 Discussion

How to construct a hierarchical graph from social network is an open question. In the above experiment, we connect two QQ groups if they have more than one common member (i.e., > 1). When we change the threshold, it directly affects the edge density in the hierarchical graph, and may influence the classification performance. For example, if we connect two QQ groups when they have one common member or more (i.e., ≥ 1), the edge density is 2.8% compared with 0.27% in the first setting. A proper setting of this threshold is data dependent, and can be determined through a validation set.

6 Conclusion

In this paper, we study the problem of semi-supervised hierarchical graph classification. To enforce a consistency among different levels of the hierarchical graph, we propose Hierarchical Graph Mutual Information (HGMI) and show HGMI can be computed via non-hierarchical graph mutual information computation methods. We build two classifiers IC and HC at the graph instance level and the hierarchical graph level respectively to fully exploit the available information. Our semi-supervised solution SEAL-CI adopts an iterative framework to update IC and HC alternately with an enlarged training set. We present two hierarchical graph benchmarks and demonstrate that SEAL-CI outperforms other competitors by a significant margin.

References

[1] H. Chang et al., “Spectral graph attention network with fast eigen-approximation,” in Proc. 30th ACM Int. Conf. Informat. Knowl. Manage., 2021, pp. 2905–2909.
[2] Y. Rong, W. Huang, T. Xu, and J. Huang, “Dropedge: Towards deep graph convolutional networks on node classification,” in Proc. Int. Conf. Learn. Representations, 2020.
[3] J. Tang, J. Li, Z. Gao, and J. Li, “Rethinking graph neural networks for anomaly detection,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 21076–21089.
[4] R. Ramanath et al., “Towards deep and representation learning for talent search at linkedin,” in Proc. 30th ACM Int. Conf. Informat. Knowl. Manage., 2018, pp. 2253–2261.
[5] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliass-Rad, “Collective classification in network data,” AI Mag., vol. 29, no. 3, pp. 93–106, 2008.
[6] K. M. Borgwardt and H.-P. Kriegel, “Shortest-path kernels on graphs,” in Proc. 5th IEEE Int. Conf. Data Mining, 2005, pp. 74–81.
[7] T. Gärtner, P. Flach, and S. Wrobel, “On graph kernels: Hardness results and efficient alternatives,” in Learning Theory and Kernel Machines, Berlin, Germany: Springer, 2003, pp. 129–143.
[8] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. M. Borgwardt, “Efficient graphlet kernels for large graph comparison,” in Proc. 12th Int. Conf. Artif. Intell. Statist., 2009, pp. 488–495.
[9] N. Shervashidze, P. Schweitzer, E. J. V. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-Lehman graph kernels,” J. Mach. Learn. Res., vol. 12, pp. 2859–2861, 2011.
[10] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in Proc. 33rd Int. Conf. Int. Conf. Mach. Learn., 2016, pp. 2014–2023.
[11] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, “graph2vec: Learning distributed representations of graphs,” 2017, arXiv:1707.05005.
[12] J. Li, T. Yu, D.-C. Juan, A. Gopalan, H. Cheng, and A. Tomkins, “Graph autoencoders with deconvolutional networks,” 2020, arXiv:2012.11888.
[13] F. Rousseau, E. Kligas, and M. Vazirgiannis, “Text categorization as a graph classification problem,” in Proc. Int. Joint Conf. Natural Lang. Process., 2015, pp. 1702–1712.
[14] Z. Feng et al., “Graph representation learning via graphical mutual information maximization,” in Proc. Web Conf., 2020, pp. 299–307.
[15] F.-Y. Sun, J. Hofmann, V. Verma, and J. Tang, “InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in Proc. Int. Conf. Learn. Representations, 2020.
[16] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.
[17] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 855–864.
[18] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017.
[19] M. Deferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Int. Conf. Neural Inform. Process. Syst., 2016, pp. 3844–3852.
[20] P. Yanardag and S. Vishwanathan, “Deep graph kernels,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2015, pp. 1365–1374.
[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in Proc. Int. Conf. Learn. Representations, 2019.
[22] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in Proc. Int. Conf. Learn. Representations, 2018.
[23] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in Proc. 32nd Int. Conf. Neural Inform. Process. Syst., 2018, pp. 4800–4810.
[24] J. Lee, I. Lee, and J. Kang, “Self-attention graph pooling,” in Proc. 33rd Int. Conf. Int. Conf. Mach. Learn., 2019, pp. 3734–3743.
[25] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang, “Semi-supervised graph classification: A hierarchical graph perspective,” in Proc. World Wide Web Conf., 2019, pp. 972–982.
Yu Rong received the PhD degree from the Chinese University of Hong Kong, in 2016. He is a senior researcher with Machine Learning Center, Tencent AI Lab. He joined Tencent AI Lab in June 2017. His main research interests include social network analysis, graph neural networks, and large-scale graph systems. In Tencent AI Lab, he is working on building the large-scale graph learning framework and applying the deep graph learning model to various applications, such as ADME prediction and malicious detection. He has published several papers on data mining, machine learning top conferences, including KDD, WWW, NeurIPS, ICLR, CVPR, ICCV.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.