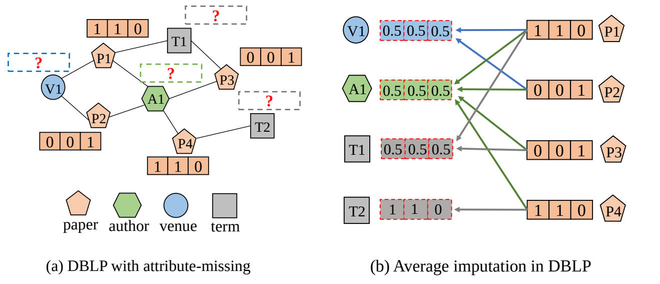
Analyzing Heterogeneous Networks with Missing Attributes by Unsupervised Contrastive Learning

Dongxiao He, Chundong Liang, Cuiying Huo, Zhiyong Feng, Di Jin, Liang Yang, and Weixiong Zhang

***Abstract*—Heterogeneous information networks (HINs) are potent models of complex systems. In practice, many nodes in a HIN have their attributes unspecified, resulting in significant performance degradation for supervised and unsupervised representation learning. We developed an unsupervised heterogeneous graph contrastive learning approach for analyzing HINs with missing attributes (HGCA). HGCA adopts a contrastive learning strategy to unify attribute completion and representation learning in an unsupervised heterogeneous framework. To deal with a large number of missing attributes and the absence of labels in unsupervised scenarios, we proposed an augmented network to capture the semantic relations between nodes and attributes to achieve a fine-grained attribute completion. Extensive experiments on three large real-world HINs demonstrated the superiority of HGCA over several state-of-the-art methods. The results also showed that the complemented attributes by HGCA can improve the performance of existing HIN models.**

***Index Terms*—Heterogeneous information networks, unsupervised learning, contrastive learning, missing data.**


P1
P2
P3
P4
V1
A1
T1
T2
?
1 1 0
0 0 1
0.5 0.5 0.5
paper
author
venue
term
(a) DBLP with attribute-missing
(b) Average imputation in DBLP

(a) DBLP with missing attributes (b) Average imputation in DBLP

Fig. 1. (a) The DBLP dataset with four types of nodes, where only paper nodes have attributes whereas other nodes have no attribute; (b) An average imputation strategy is adopted by the existing methods. Missing attributes are assigned by the vectors of average attributes of their directly connected paper nodes.

I. Introduction

MANY real-world systems, e.g., transportation networks, power grids, and social networks, are best viewed and formulated as networks. The overarching problem of mining and analyzing valuable information in networks has been actively studied for decades [1], [2], [3]. As a more capable representation scheme using multiple types of nodes and edges, heterogeneous information networks (HINs) were recently introduced to model complex systems with various types of entities and relations [4]. Low-dimensional embedding techniques have also been adopted to derive compact representations of HINs and extract network-specific information, such as heterogeneous network structural properties and node semantic relations [4]. Several methods have been developed for heterogeneous network embedding (HNE), including *proximity-preserving* methods, *message-passing* methods, and *relation-learning* methods [5]. Among these methods are the popular ones based on heterogeneous graph neural networks (HGNNs) which have been applied to, e.g., node classification [6], [7] and link prediction [8], [9].

To be effective, HGNN-based methods require node attributes to be well-specified, including their types and values. However, missing data is ubiquitous in data analytic problems. For problems of HIN analysis, node attributes are often missing or incomplete primarily due to the difficulty in data collection or defining attribute types and values. Take a simple heterogeneous network from the database and logic programming website (DBLP) [10] as an example (Fig. 1(a)), the network has four types of nodes (author, paper, term, and venue) and three kinds of links. The attributes of paper nodes come from the keywords in their titles, whereas the other types of nodes have no attributes, which are non-trivial to accurately define.

Missing node attributes impose a huge challenge to developing effective methods for HNE, especially HGNN-based methods. As a work-around scheme to the missing data issue, the existing HGNN methods usually adopt imputation strategies, such as neighbor's average or one-hot vector as done in MAGNN [8]. Consider again the example in Fig. 1(a), the missing attributes are filled in by the average attributes of their directly connected paper nodes (Fig. 1(b)). These imputation methods are non-optimal and may introduce noises or even errors to the data to be analyzed further.

Labeling nodes is costly and time-consuming so that unsupervised learning is desirable, as attempted lately in develop-

This work was supported in part by the Natural Science Foundation of China (61876128, 61772361, 61972442), in part by the Key Research and Development Project of Hebei Province of China (20350802D and 20310802D), in part by the Natural Science Foundation of Hebei Province of China (F2020202040), in part by the Natural Science Foundation of Tianjin of China (20JCYBJC00650), in part by the George Washington University Facilitating Fund (UFF) FY21. *(Corresponding author: Liang Yang.)*

Dongxiao He, Chundong Liang, Cuiying Huo, Zhiyong Feng and Di Jin are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: hedongxiao@tju.edu.cn; liangchundong@tju.edu.cn; huocuiying@tju.edu.cn; zyfeng@tju.edu.cn; jindi@tju.edu.cn).

Liang Yang is with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100195, China (e-mail: yangliang@vip.qq.com).

Weixiong Zhang is with the Department of Health Technology and Informatics, Hong Kong Polytechnic University, Hong Kong, China (e-mail: weixzhang@polyu.edu.hk).

ing some HGNN methods [11], [12]. Nonetheless, the problem of missing data is exacerbated in unsupervised learning because it depends on complete and accurate data to build effective unsupervised models, to be discussed further in the next section with a motivating example. Since imputation may introduce errors, new approaches must be developed when designing unsupervised learning methods for HNE with incomplete data.

We set forth to design an effective unsupervised approach to learn low-dimensional representations of complex HINs with missing node attributes. We developed **HGCA**, an end-to-end unsupervised **h**eterogeneous **g**raph **c**ontrastive learning method for HINs with missing **a**ttributes. We designed a novel contrastive learning strategy to integrate the processes of attribute completion and heterogeneous network embedding into a unified unsupervised framework. We introduced a contrastive objective for maximizing the agreement between two low-dimensional node representations, one for the actual node attributes and the other for learned attributes. In other words, the contrastive learning strategy helped implicitly train the attribute completion module so that complemented attributes were trustworthy and could be used in place of actual attributes. Furthermore, to ensure every complemented attribute to be accurate, we considered a HIN with missing attributes as an augmented heterogeneous network and learned joint embeddings of nodes and attributes to capture their semantic relations to help derive missing attributes. We carried out extensive experiments to compare the new HGCA method with several state-of-the-art approaches. The results showed that HGCA significantly outperformed the existing HIN methods and validated the generality of the new method for complementing missing attributes.

The rest of the paper is organized as follows. In Section II we experimentally investigate the impact of missing attributes on semi-supervised/unsupervised HIN representation learning. We discuss related work in Section III and the notations and definitions used in this paper in Section IV. We present the HGCA method in Section V and experimental results in Section VI. We conclude in Section VII.

II. A Motivating Example

To appreciate the complexity and inherent issues of missing attributes when mining heterogeneous information networks (HINs), particularly in an unsupervised setting, we first considered an example to motivate the development of HGCA. In particular, we analyzed the impact of missing attributes and how inaccurate and erroneous attributes derived from the existing methods could severely affect semi-supervised and unsupervised learning. In the experiment, we compared the average imputation strategy and our new attribute-completion process on DBLP data with missing attributes [10] using a semi-supervised model MAGNN [8] and an unsupervised model we developed for HGCA as discussed in Section V-D.

Figures 2(a) and 2(b) show the classification results of the semi-supervised models with complemented attributes from average imputation (MAGNN-avg) and our attribute completion process (MAGNN-ac). The two models have classification accuracies of 93.02% and 94.40%, respectively.

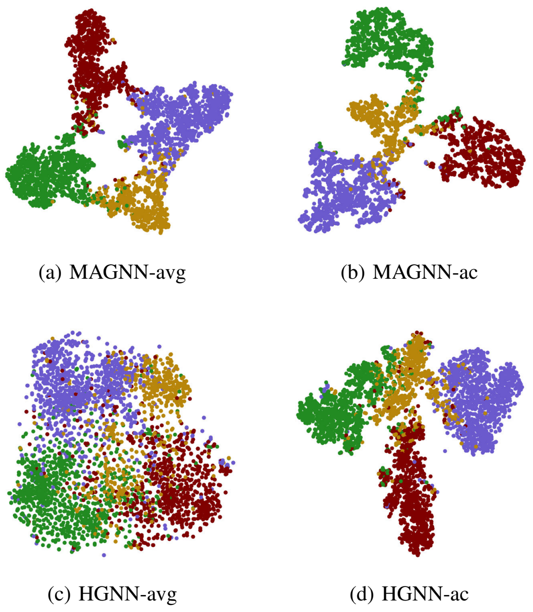

(a) MAGNN-avg (b) MAGNN-ac (c) HGNN-avg (d) HGNN-ac

Fig. 2. Visualization of author node embeddings in DBLP under different settings. Different colors denote different classes. (a) MAGNN-avg. The result of a semi-supervised model MAGNN [8] with complemented attributes from average imputation. The classification accuracy is 93.02%. (b) MAGNN-ac. The result of the semi-supervised model MAGNN with our attribute completion process. The classification accuracy is 94.40%. (c) HGNN-avg. The result of an unsupervised model (HGNN, discussed in Section V-D) with complemented attributes from average imputation. The classification accuracy is 78.45%. (d) HGNN-ac. The result of the unsupervised model HGNN with our attribute completion process. The classification accuracy is 91.06%.

Figure. 2(c) and . 2(d) show the results of the unsupervised models with complemented attributes from average imputation (HGNN-avg) and our attribute completion process (HGNN-ac). Their corresponding classification accuracies are 78.45% and 91.06%.

Attributes complemented in different ways have a negligible impact on semi-supervised learning: their accuracies differ only by 1.38% (Fig. 2). The results of MAGNN-avg and MAGNN-ac have clear separation and denser cluster structures. This may be because attributes of different classes may be easily separated in label-guided training so that inaccurate attributes may still help generate good embedding. Intuitively, semi-supervised learning guided by labels can filter out incorrect information in complemented attributes and avoid significant performance degradation.

However, complemented attributes significantly affect unsupervised learning. The performance of average imputation (HGNN-avg) is 11.61% worse than that of our attribute completion method (HGNN-ac). Unsupervised learning does not use labels (i.e., no labels are used to build the constraint relationship between attributes and classes), but instead needs to mine hidden patterns in the data to guide model training. The noises and errors in complemented attributes from simple imputation methods can make pattern discovery difficult and degrade the performance of unsupervised learning. Therefore, it is imperative to develop effective methods for attribute completion to make unsupervised learning feasible for mining and analyzing HINs.

III. Related Work

A. Heterogeneous Graph Neural Networks

Graph neural networks (GNNs) [1], [2] are effective for analyzing graph-structural data. Most GNNs follow the principle of message passing. They learn node representations by aggregating information from neighboring nodes based on the guidance of topological structure. GNNs have achieved superior performance in many downstream tasks, such as node classification [13], [14], graph classification [15], [16], and some tasks in computer vision (e.g., gait recognition [17]).

Recently, many heterogeneous graph neural networks have been proposed for heterogeneous information network (HIN) analysis, particularly for learning node embeddings based on node attributes by aggregating the information from neighbor nodes while capturing heterogeneous structural properties and node semantic relations. These methods are semi-supervised. HAN [6] uses hierarchical attention to simultaneously learn the importance between nodes and between meta-paths. The node-level attention aims to learn the importance between a node and its meta-path-based neighbors, while the semantic-level attention can learn the importance of different meta-paths. MAGNN [8] further considers intermediate nodes along the meta-paths. It designs several encoder functions for distilling information from meta-path instances and employs a similar hierarchical attention mechanism as HAN for generating node representations. Because domain knowledge is required for defining meta-paths, some methods try not to use meta-paths [18], [19] to remove this limitation. GTN [7] learns a soft selection of edge types and composites relations for generating useful multi-hop connections, which have the same effect as meta-paths. GTN can identify useful connections between unconnected nodes on the original graph while learning effective node representations on the new graphs in an end-to-end fashion. Unlike GTN implicitly learning meta-paths, HetSANN [20] designs aggregation operations directly on the original graph. It uses type-aware attention layers instead of convolutional layers [13] to capture the interactions among different types of nodes. Thus it can incorporate information from high-order neighbors of different types through message passing across layers. HGT [21] holds the same idea as HetSANN. It adopts meta-relation-based mutual attention to learn node embeddings. Besides, HGT argues that most methods are incapable of modeling web-scale heterogeneous graphs and ignore the dynamic nature of HINs. It designs a heterogeneous mini-batch graph sampling algorithm for scalable training and uses a relative temporal encoding technique to capture graph dynamics.

There are also some unsupervised methods. HetGNN [9] considers jointly heterogeneous structural information and heterogeneous content information of each node. It encodes the interactions of heterogeneous content within a node and aggregates node content embeddings based on different neighboring groups. It performs unsupervised learning by a graph context loss. R-GCN [22] considers edge heterogeneity by multiple convolution processes, each of which corresponds to one type of edge. It maximizes the likelihood of observing edges in HINs to optimize parameters in an unsupervised setting. CompGCN [23] extends R-GCN by leveraging a variety of entity-relation composition operations to jointly learn embeddings of nodes and relations. DMGI [11] models the global properties of a HIN. It maximizes the mutual information (MI) [24] between node representations and a global representation of the entire graph and uses a regularization to learn a consensus embedding for each node by taking into account the inter-relationship among meta-path-based relations. NSHE [12] preserves the node pair similarity and network schema structure and learns embeddings in an unsupervised way. Although the above methods have achieved good performance, they all require complete attributes which are often not available in practice.

B. Graph Contrastive Learning

Contrastive learning becomes popular first in visual representation learning [25], [26]. It uses data augmentation to obtain two correlated views of the given data and adopts a contrastive loss to learn representations that maximize the agreement between the two views [27], [28], [29]. As a type of unsupervised method, it performs comparably with semi-supervised methods. Contrastive learning has recently been extended to graphs. One of the primary challenges of graph contrast learning is how to design data augmentation strategies for generating correlated data views. Taking advantage of the properties of a graph, GRACE [30] generates two graph views by removing edges and masking node attributes and then uses a contrastive loss to maximize the agreement of node embeddings in these two views. GCA [31] extends the augmentation strategy of GRACE to adaptive augmentation that incorporates various priors for topological and semantic aspects of the graph. Besides, some existing works try to generate contrastive samples in an automatical way, which is similar with automated graph learning [32], [33], [34]. JOAO [35] proposes a bi-level optimization approach to automate the augmentation selection, which alleviates the reliance on the design of augmentations. AutoGCL [36] uses embedded node features to predict the probability of selecting a certain augment operation, which learns to generate an augmented graph view in a data-driven manner.

To explore the pre-training scheme for node-level tasks, GCC [37] introduces pre-training tasks as subgraph instance discrimination in and across networks and adopts contrastive learning to empower graph neural networks to learn the intrinsic and transferable structural representations. To explore the pre-training scheme for graph-level tasks, GraphCL [38] adopts different graph-level augmentations and uses a graph contrastive loss to make representations invariant to perturbation. IGSD [39] is also a graph-level representation method. It improves the idea of knowledge distillation [40], [41] to iteratively performs teach-student distillation by contrasting augmented views of graph instances.

More recently, HeCo [42] applied contrastive learning to HINs. It employs network schema and meta-paths as two views to capture both local and high-order structures, and performs contrastive learning across the two views. Despite the success of HeCo, it ignores the important role of attributes in creating data views and fails to deal with HINs with missing attributes.

C. Learning with Attribute-Missing Data

The problem of missing attributes is ubiquitous in data analytic problems. Several methods have been proposed to address this problem. GRAPE [43] argues that some existing methods for learning with attribute-missing data rely too much on assumptions of data distribution without paying attention to the relationship between samples. GRAPE tackles the attribute-missing problem using a graph-based approach. It constructs the relationships among samples and attributes and can use graph neural networks to perform attribute completion and label prediction simultaneously with no data distribution assumption. There are also some methods for graph-structured data with missing attributes. GCNMF [44] is one of these methods. Arguing that conventional strategies separate attribute imputation and graph representation learning, which degrade performance, GCNMF integrates the processing of attribute completion and graph representation learning within the same graph neural network (GNN) architecture. The main idea of GCNMF is to represent missing attributes by a Gaussian Mixture Model (GMM), transform GMM into a layer of GNN, and train the overall system by an end-to-end learning process. SAT [45] is a method for graphs with attributes of some nodes entirely missing. SAT makes a shared-latent space assumption so that attribute and topology information can be projected into a common space where missing attributes can be complemented by topology representations. It then uses dual encoders to learn the representations of attributes and topology separately and aligns the paired latent representations via adversarial learning [46], [47], [48]. It is important to note that the above methods do not apply to HINs. The complexity of missing attributes and complex network structures make representation learning a great challenge for HINs.

IV. Definitions and Notations

We now define the key terms and notations used throughout the paper.

Definition 1 (Heterogeneous Information Networks (HINs)): A HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{R}, X)$ consists of a set of nodes $\mathcal{V}$ with a corresponding node-type set $\mathcal{F}$, a set of edges $\mathcal{E}$ with a corresponding edge-type set $\mathcal{R}$, and a set of attribute X, along with a function $\varphi : \mathcal{V} \rightarrow \mathcal{F}$ to map nodes to node types and function $\phi : \mathcal{E} \rightarrow \mathcal{R}$ to map edges to edge types, where $|\mathcal{F}| + |\mathcal{R}| \geq 2$. The edge set can also be represented by an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $n = |\mathcal{V}|$, and $A_{i,j}$ is 1 if there is an edge between nodes v_i and v_j or 0, otherwise.

Definition 2 (Attribute-Missing in HINs): IN general, for a node and its attributes, attribute-missing means that the attribute values of this node are partially or completely unspecified. In HINs, a common scenario of attribute-missing, is that the values of some attribute types of some nodes are entirely missing, while the nodes with other attribute types have no missing attributes. Formally, given a HIN $\mathcal{G}$ with node set $\mathcal{V}$ and node-type set $\mathcal{F}$, attribute-missing in HINs means that there exists a subset $\mathcal{F}' \subset \mathcal{F}$ and $\mathcal{F}' \neq \varnothing$, and a node $v_i \in \mathcal{V}$ having no attribute iff its type belongs to $\mathcal{F}'$.

TABLE I
NOTATIONS AND EXPLANATIONS

Notations	Explanations
$\mathcal{M}$	A meta-path
$\mathcal{V}$	The set of nodes
$\mathcal{V}^{+}$, $\mathcal{V}^{-}$	The set of nodes with existing/missing attributes
$\mathcal{N}(v_i)$	The neighbor set of node v_i
S_i	A group of neighbor nodes of v_i
A, $A^{\mathcal{M}}$	Adjacency matrix and meta-path-based adjacency matrix
X, $\bar{X}$	Original/Complemented node attributes
$X^{(1)}, X^{(2)}$	Two attribute views
Z^{node}, Z^{attr}	Joint embeddings of nodes/attributes
H	The embeddings of nodes in HGNN

Definition 3 (Meta-Path and Meta-Path-based Adjacency): A meta-path $\mathcal{M}$ is defined as a path $F_1 \xrightarrow{R_1} F_2 \xrightarrow{R_2} ... \xrightarrow{R_l} F_{l+1}$ with node types $F_1, F_2, ..., F_{l+1} \in \mathcal{F}$ and edge types $R_1, R_2, ..., R_l \in \mathcal{R}$, which can be simplified as $F_1 F_2 ... F_{l+1}$. Given a HIN and a meta-path $\mathcal{M}$, there are many node sequences in the HIN following the schema defined by $\mathcal{M}$. We name these node sequences meta-path instances of $\mathcal{M}$. A meta-path-based adjacency is an adjacency matrix $A^{\mathcal{M}} \in \mathbb{R}^{n \times n}$, where if node v_i and v_j are connected via a meta-path instance of $\mathcal{M}$, then $A^{\mathcal{M}}_{i,j} = 1$, or 0 otherwise.

The notations that we will use and their explanations are summarized in Table I. Uniformly, uppercase notations denote sets, lowercase notations denote nodes, and subscript indicates node numbers.

V. Method

We start with an overview of our proposed HGCA method and then present its major components, including the joint embedding, attribute completion, heterogeneous graph neural network (HGNN), and contrastive learning strategy.

A. Overview

HGCA is an end-to-end unsupervised contrastive learning method designed for analyzing heterogeneous graphs with missing attributes. It integrates an attribute completion process and HGNN module into a unified unsupervised method using a contrastive learning strategy. The new method is designed not only for completing attributes but also for learning node representations.

HGCA contains four components: joint embedding, attribute completion, HGNN, and contrastive learning strategy (Fig. 3). The joint embedding module transforms heterogeneous information networks (HIN) with missing attributes into an augmented heterogeneous network where the original attributes are treated as a new type of nodes. HGCA then uses predefined meta-paths to learn embeddings of nodes and attributes jointly through a collaborative random walk method(Fig. 4). In the attribute completion module, HGCA computes the values of missing attributes using joint embeddings as guidance. Subsequently, HGCA produces two data views by using different attributes (one on the complemented attributes and the other on the original attributes) and learns node representations by

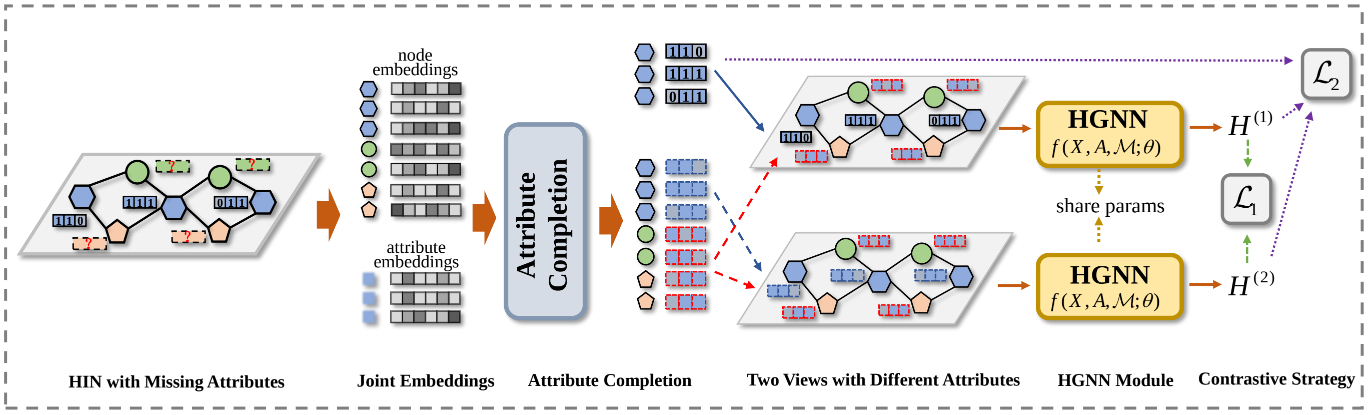


Fig. 3. The flow chart of the HGCA method. The patterns with different shapes and colors represent different types of nodes, among which squares without borders represent a new type of nodes abstracted from attributes. Given a HIN with missing attributes, HGCA first learns joint embeddings by using a collaborative random walk. It then uses the joint embeddings as priors to complete missing attributes and generates two data views with different attributes. A pair of parameter-shared HGNNs is applied to learn the final node representations. Two contrastive objective functions are used to optimize the overall model.

two parameter-sharing HGNN modules. These two HGNN modules are composed of intra-group aggregation, inter-group aggregation within a meta-path, and semantic aggregation between meta-paths. Finally, HGCA adopts a contrastive strategy to optimize the attribute completion module and the HGNN module together.

B. Joint Embedding

A network has many characteristics. In particular, ever node has its characteristic values, represented by node attributes. We introduce joint embeddings of nodes and characteristics to explore available information in HGCA to better estimate missing attributes. Specifically, HGCA first transforms a HIN with missing attributes into an augmented heterogeneous network where attribute nodes are introduced to represent characteristics as a new type of nodes (Fig. 4). It then adds edges linking the original nodes and attribute nodes based on characteristic (or attribute) values. The high-order relations among original and attribute nodes can be captured by meta-path instances among the original nodes and meta-path instances between original and attribute nodes. HGCA then explores this augmented network to mine semantic relations among characteristics and nodes by collaborative random walk [49] and represent nodes and characteristics in joint low-dimension embeddings, which are used as a prior for attribute completion.

The meta-path-based random walk in HGCA is to capture the local neighbor structures in the augmented heterogeneous network. Given a meta-path $\mathcal{M}_p : F_1 \xrightarrow{R_1} F_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} F_{l+1}$ and node v_i with type F_i, the transition probability at step i is defined as

$$p(v_{i+1}|v_i,\mathcal{M}) = \begin{cases} \frac{1}{|\mathcal{N}_{F_{i+1}}(v_i)|} & A_{i,i+1}=1, \varphi(v_{i+1}) = F_{i+1} \\ 0 & otherwise \end{cases} \tag{1}$$

where $\mathcal{N}_{F_{i+1}}(v_i)$ presents neighboring nodes of v_i with type F_{i+1} and $\varphi(v_{i+1}) = F_{i+1}$ denotes the type of v_{i+1}. The random walk ensures the semantic relationships among nodes and characteristics to be properly preserved. HGCA then uses skip-gram [50] to learn joint embeddings of nodes and characteristics by maximizing the predicted probability of the local neighbor structures captured by the random walk. The objective function can be formulated as

$$\arg\max_{\theta} \sum_{v\in\mathcal{V}} \sum_{F\in\mathcal{F}} \sum_{u\in\mathcal{N}_F(v)} \log p(u|v;\theta) \tag{2}$$

where $\mathcal{V}$ is the set of nodes, $\mathcal{F}$ the set of node types, $\mathcal{N}_F(v)$ the set of type F neighbor nodes of node v, which is sampled by the random walk. HGCA uses a softmax function $p(u|v;\theta) = \frac{\exp(z_u\cdot z_v^T)}{\sum_{o\in\mathcal{V}}\exp(z_o\cdot z_v^T)}$ to calculate the probability that v and u are local neighbors, where z_v is the embedding of node v, a vector learned by a skip-gram model with one-hot vector as input. The embeddings of original and attribute nodes can be formulated as Z^{node} and Z^{attr}, respectively. Each row of Z^{node} represents an embedding vector of a node denoted as z_v^{node}. Likewise, each row of Z^{attr} represents an embedding vector of a characteristic of attributes, denoted as z_c^{attr} .

C. Attribute Completion

Deviated from the idea of imputing attribute values using neighbor information, HGCA derives missing attribute values using the joint embeddings described above. Consider a node v_i with missing attributes whose d characteristics are unavailable. To accurately estimate these attributes, HGCA evaluates the similarity between the node embedding $z_{v_i}^{node}$ and the embeddings Z^{attr} of the d characteristics. This evaluation is based on the rationale that the joint embeddings of nodes and attributes contain semantic relations among nodes and these characteristics so that the similarity between a node and a characteristic can be used as the attribute value of the node. Given node embedding $z_{v_i}^{node}$ and the embedding $z_{c_j}^{attr}$ of the jth characteristic, the attribute value can be computed as

$$value = z_{v_i}^{node}\bar{W}(z_{c_j}^{attr})^T \tag{3}$$

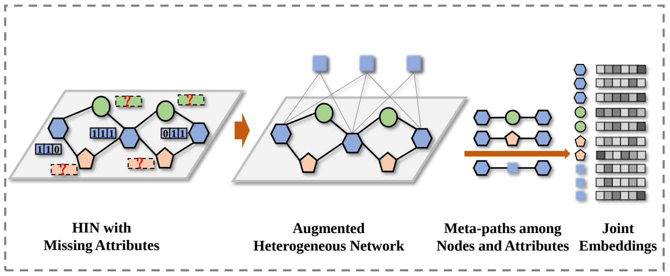

HIN with Missing Attributes
Augmented Heterogeneous Network
Meta-paths among Nodes and Attributes
Joint Embeddings

Fig. 4. The process of learning joint embeddings in HGCA. The patterns with black borders represent original nodes and the blue squares without borders represent a new type of nodes abstracted from attributes (each square represents a characteristic). HGCA first transforms a HIN with missing attributes into an augmented heterogeneous network and then learns joint embeddings by using a collaborative random walk.

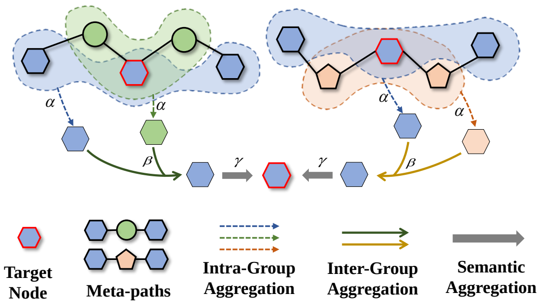


Fig. 5. The architecture of the HGNN module, including intra-group aggregation, inter-group aggregation within a meta-path, and semantic aggregation between meta-paths. Two kinds of meta-paths (four meta-path instances in total) are used here for simplicity.

where $\bar{W}$ is a learnable parameter to make similarity value more accurate and the inner product is used to compute similarities. Given the joint embeddings Z^{node} and Z^{attr} of all nodes and characteristics, the complemented attributes can be expressed as

$$\bar{X} = Z_{node}\bar{W}Z_{attr}^{T} \tag{4}$$

where $\bar{X}$ is the complemented attributes. As a result, each dimension value of missing attributes can be learned accurately under the guidance of joint embeddings.

With missing attributes completed, the complemented attributes and the heterogeneous topology are sent to the HGNN module to derive the final representations. As unsupervised contrastive learning, HGCA adopts two different data views, one from complemented attributes and the other from the original attributes. Specifically, given the complemented attributes $\bar{X}$ of all nodes and the original attributes X of some nodes, the two attribute views can be expressed as:

$$X^{(1)} = \{x_i, \bar{x}_j | \forall v_i \in \mathcal{V}^+, \forall v_j \in \mathcal{V}^-\} \tag{5}$$

$$X^{(2)} = \{\bar{x}_i | \forall v_i \in \mathcal{V}\} \tag{6}$$

where $\mathcal{V}^+$ is the set of nodes with given attributes, $\mathcal{V}^-$ the set of nodes with missing attributes, $x_i/\bar{x}_i$ the ith row of $X/\bar{X}$ representing the original/complemented attribute vector for node v_i, and $X^{(1)}$ and $X^{(2)}$ are the two attribute views with each row being an attribute vector of a node.

Even though the most common case of attribute missing in HINs is node-type-specific, the attribute completion module is also applied to other cases. This is because complemented and original attributes can be combined in $X^{(1)}$. When the original attributes are given in the dataset, we use them in Formula 5, while when the attributes are missing, we use complemented attribute instead.

D. Heterogeneous Graph Neural Network

HGCA learns transformed representations of nodes by a multi-layer perceptron (MLP) and then uses the heterogeneous graph neural network (HGNN) to learn the final node representations. It computes

$$h_i{}' = \sigma\left(MLP\left(x_i\right)\right) \tag{7}$$

where x_i is the attribute vector of node v_i, $h_i^{'}$ the transformed representation of v_i, and $\sigma(\cdot)$ an activation function.

The HGNN module adopts an attention-based [51] hierarchical aggregation process, which consists of intra-group aggregation, inter-group aggregation within a meta-path, and semantic aggregation between meta-paths (Fig. 5). In the first step of intra-group aggregation, the context nodes within a meta-path instance contribute differently to the target node, so it is reasonable to divide them into different groups. For a meta-path $\mathcal{M}_p : F_1 \xrightarrow{R_1} F_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} F_{l+1}$, HGCA groups neighbor nodes of node v_i into

$$S_i^{\mathcal{M}_p - F_l} = \{v_j | dist(i,j) = l, \mathbb{1}(v_i, v_j, \mathcal{M}_p) = 1\} \tag{8}$$

where $dist(i,j)$ is the distance between node v_i and v_j, $\mathbb{1}(v_i, v_j, \mathcal{M}_p)$ an indicator that equals to 1 iff node v_i and v_j are in the same meta-path instance of $\mathcal{M}_p$. For simplicity, we denote $S_i^{\mathcal{M}_p - F_l}$ as $S_i^{F_l}$. The intra-group aggregation can then be formulated as

$$\alpha_{ij}^{F_l} = \frac{\exp\left(\sigma\left(\mathrm{a}_{\mathcal{M}_p}^T \cdot [Q^{F_l}(h_i{}')||K^{F_l}(h_j{}')]\right)\right)}{\sum_{v_k \in S_i^{F_l}} \exp\left(\sigma\left(\mathrm{a}_{\mathcal{M}_p}^T \cdot [Q^{F_l}(h_i{}')||K^{F_l}(h_k{}')]\right)\right)} \tag{9}$$

$$h_i^{F_l} = h_i{}' + \sum_{v_j \in S_i^{F_l}} \alpha_{ij}^{F_l} \cdot V^{F_l}(h_j{}') \tag{10}$$

where $Q^{F_l}, K^{F_l}, V^{F_l}$ are linear functions with parameters specific to S^{F_l}, $\mathrm{a}_{\mathcal{M}_p}$ is a learnable parameter specific to meta-path $\mathcal{M}_p$, $\sigma(\cdot)$ is an activation function, and $||$ denotes a concatenation operation.

The second step is inter-group aggregation within a meta-path. HGCA uses the attention mechanism to distinguish different groups within a meta-path and performs weighted aggregation to derive the embedding under a meta-path as

$$w^{F_l} = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} (q^{\mathcal{M}_p})^T \cdot \tanh(W^{\mathcal{M}_p} \cdot h_i^{F_l} + b^{\mathcal{M}_p}) \tag{11}$$

$$\beta^{F_l} = \frac{\exp(w^{F_l})}{\sum_k \exp(w^{F_k})} \tag{12}$$

$$H^{\mathcal{M}_p} = \sum_k \beta^{F_k} \cdot H^{F_k} \tag{13}$$

where $W^{\mathcal{M}_p}$ and $b^{\mathcal{M}_p}$ are the weight matrix and bias vector specific to meta-path $\mathcal{M}_p$, respectively. $q^{\mathcal{M}_p}$ is also a learnable attention vector specific to $\mathcal{M}_p$.

The last step is semantic aggregation between meta-paths. Similar to inter-group aggregation, HGCA uses a semantic attention mechanism to distinguish the semantic importance of different meta-paths [6]. Given P meta-paths, the final representation of node v_i can be express as

$$H_{final} = \sum\nolimits_{p=1}^{P} \gamma^{\mathcal{M}_p} \cdot H^{\mathcal{M}_p} \tag{14}$$

where $\gamma^{\mathcal{M}_p}$ is the weight of meta-path $\mathcal{M}_p$, which can be computed similarly as in Formula 12.

E. Contrastive Learning

HGCA uses a parameter-sharing network (HGNN module in Section V-D) to learn node embeddings in the two attributes views $X^{(1)}$ and $X^{(2)}$ in Section V-C, denoted as $H^{(1)} = HGNN(X^{(1)}, A, \mathcal{M}; \theta)$ and $H^{(2)} = HGNN(X^{(2)}, A, \mathcal{M}; \theta)$, where θ are shared parameters. Contrastive objectives have been used to train GNN [30], [31], [38], which enforce two embeddings of a node in different views to agree with each other and to be distinguishable from the embeddings of other nodes [52]. Since using a contrastive objective is to maximize the agreement of node embeddings in two views, it can make the complemented attributes and the real attributes play the same role in constructing the final embeddings.

Since meta-paths provide a strong prior that two nodes connected by a valid meta-path instance are closely related, a meta-path-based modified contrastive objective is used in HGCA. In particular, HGCA maximizes the similarity not only between two representations of the same node but also the representations of the two nodes connected by a meta-path instance. That is because two nodes connected by a meta-path instance have a strong semantic relationship and can be seen as a positive sample pair. As a result, node representations are distinguishable and semantic information in HINs is preserved at the same time. Given P meta-paths and corresponding meta-path-based adjacency matrices, HGCA first computes a contrastive coefficient of the node pairs as

$$\Omega = \sum\nolimits_{p=1}^{P} \eta_p \cdot A^{\mathcal{M}_p} \tag{15}$$

where η_p is a hyper-parameter to indicate the importance of meta-path $\mathcal{M}_p$ in the contrastive objective function. For embeddings in each view, the contrastive loss function can be formulated as

$$\psi(H, H') = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} -\log \frac{e^{\langle h_i, h'_i \rangle} + Pos(H, H'; v_i)}{e^{\langle h_i, h'_i \rangle} + Neg(H, H'; v_i)} \tag{16}$$

$$Pos(H, H'; v_i) = \sum_{k \neq i} \omega_{i,k} \cdot e^{\langle h_i, h_k \rangle} + \sum_{j \neq i} \omega_{i,j} \cdot e^{\langle h_i, h'_j \rangle} \tag{17}$$

$$Neg(H, H'; v_i) = \sum_{k \neq i} (1 - \omega_{i,k}) \cdot e^{\langle h_i, h_k \rangle} + \sum_{j \neq i} (1 - \omega_{i,j}) \cdot e^{\langle h_i, h'_j \rangle} \tag{18}$$

$$\langle h_i, h_j \rangle = \cos(h_i, h_j)/\tau \tag{19}$$

where $\cos(\cdot)$ is the cosine function, τ a temperature parameter, and $\omega_{i,j}$ a contrastive coefficient in Ω. The overall loss function of embeddings of two views are then defined as

$$\hat{H}^{(1)}, \hat{H}^{(2)} = f_1(H^{(1)}), f_1(H^{(2)}) \tag{20}$$

$$\mathcal{L}_1 = -\frac{\psi(\hat{H}^{(1)}, \hat{H}^{(2)}) + \psi(\hat{H}^{(2)}, \hat{H}^{(1)})}{2} \tag{21}$$

where $f_1(\cdot)$ is a MLP with one hidden layer and non-linearity activation. Furthermore, HGCA adopts a contrastive loss between the final embeddings and the original attributes so that the node embeddings can contain more valuable information

$$\tilde{H}^{(1)}, \tilde{H}^{(2)} = f_2(H^{(1)}), f_2(H^{(2)}) \tag{22}$$

$$\tilde{X} = f_3(X) \tag{23}$$

$$\mathcal{L}_2 = -\frac{\psi(\tilde{H}^{(1)}, \tilde{X}) + \psi(\tilde{H}^{(2)}, \tilde{X})}{2} \tag{24}$$

where $f_2(\cdot), f_3(\cdot)$ are two MLPs with hidden layers and non-linearity activation. Note that in $\mathcal{L}_2$, Ω is a zero-matrix because HGCA only computes a contrastive loss between the embedding of a node and its attribute vector. The final loss function is defined as

$$\mathcal{L}_{final} = \lambda \mathcal{L}_1 + (1 - \lambda) \mathcal{L}_2 \tag{25}$$

where λ is a hyper-parameter for adjusting the balance between $\mathcal{L}_1$ and $\mathcal{L}_2$.

VI. Experiments

We first discuss the experiment setup, including datasets, baselines, and experimental settings. We then present the comparison results on three tasks (node classification, node clustering, and visualization). We discuss additional experiments to demonstrate the generality of the new HGCA method and consider a parameter analysis.

A. Experimental Setup

To analyze the effectiveness of HGCA, we performed extensive experiments on three real-world HIN datasets (Table II).

1) DBLP[1] [10]: We extracted a subset of DBLP containing information of 4057 authors (A), 14328 papers (P), 8789 terms (T), and 20 venues (V). Authors were divided into four research areas (Database, Data Mining, Artificial

[1]https://dblp.uni-trier.de

Intelligence, and Information Retrieval). For this dataset, paper nodes have attributes derived from their titles and other nodes have no attribute.

2) ACM[2] [6]: We extracted a subset of ACM for 4019 papers (P), 7167 authors (A), and 60 subjects (S). The papers were labeled according to their fields (Database, Wireless Communication, and Data Mining). For the ACM dataset, paper nodes have attributes derived from their abstract, and other nodes have no attribute.
3) Yelp[3] [53]: We extracted a subset from Yelp Open Dataset containing 2614 businesses (B), 1286 users (U), 4 services (S), and 9 rating levels (L). The business nodes were labeled by their categories. For this dataset, business nodes have attributes composed of keywords about their description and other nodes have no attribute.

We comprehensively compared HGCA with eight state-of-the-art embedding methods, which can be grouped into four homogeneous and four heterogeneous methods as well as divided into three semi-supervised and five unsupervised methods.

1) metapath2vec (mp2vec) [54]: An unsupervised heterogeneous embedding method. It uses meta-path-based random walk and skip-gram to generate embeddings. We tested all meta-paths separately and report the best result.
2) GAT [14]: A semi-supervised homogeneous embedding method. It learns node embeddings using the attention mechanism and local neighborhood structures. We tested all meta-paths separately and report here the best results.
3) GAE [55]: An unsupervised homogeneous embedding method. It uses a graph convolution encoder to learn latent representations and uses a decoder to reconstruct network topology. We tested on all meta-paths separately and report here the best results.
4) DGI [56]: An unsupervised homogeneous embedding method. It maximizes the mutual information (MI) between the graph-level summary representation and the local patches to learn embeddings. We tested on all meta-paths separately and report here the best results.
5) GMI [57]: An unsupervised homogeneous embedding method. It learns node embeddings by directly maximizing the MI between the input and output of a graph neural encoder in terms of node attributes and topological structure. We tested on all meta-paths separately and report here the best results.
6) HAN [6]: A semi-supervised heterogeneous embedding method. It uses node-level attention to learn node embeddings in each meta-path-based homogeneous graph and uses semantic attention to aggregate them to form final embeddings.
7) MAGNN [8]: A semi-supervised heterogeneous embedding method. It learns embeddings through three steps (Node Content Transformation, Intra-meta-path Aggregation, Inter-meta-path Aggregation), and designs different intra-meta-path encoders to further improve performance.
8) DMGI [11]: An unsupervised heterogeneous embedding method. It maximizes the MI between the graph-level summary representation and the local patches in each meta-path-based homogeneous graph to learn corresponding embeddings of each graph and uses the consensus regularization to minimize the disagreements among them to form final embeddings.

[2]http://dl.acm.org/

[3]https://www.yelp.com/dataset

TABLE II
STATISTICS OF DATASETS

Dataset	Nodes	Edges	hasAttributes	Meta-Paths
DBLP	Author(A):4057 Paper(P):14328 Term(T):8789 Venue(V):20	A-P:19645 P-T:85810 P-V:14328	Paper	APA APTPA APVPA
ACM	Paper(P):4019 Author(A):7167 Subject(S):60	P-P:9615 P-A:13407 P-S:4019	Paper	PAP PSP
Yelp	Business(B):2614 User(U):1286 Service(S):4 Level(L):9	B-U:30838 B-S:2614 B-L:2614	Business	BUB BSB BLB

For all baseline methods that require attributes, we used the average imputation strategy to complete the missing attributes in the above three datasets. The embedding dimensions of all methods evaluated were set to 64 for a fair comparison. For semi-supervised methods, the labeled nodes were divided into training, validation, and testing sets in the ratio of 10%, 10%, and 80%, respectively. For the proposed HGCA method, we set temperature parameter $\tau = 0.5$ and loss coefficient $\lambda = 0.5$. For the contrastive coefficients in $\mathcal{L}_2$, we set $\eta_{APA}, \eta_{APTPA}, \eta_{APVPA} = 0.1, 0.1, 0.8$ in DBLP, $\eta_{PAP}, \eta_{PSP} = 0.4, 0.6$ in ACM, and $\eta_{BUB}, \eta_{BSB}, \eta_{BLB} = 0.2, 0.6, 0.2$ in Yelp. Unless specified, we used $H^{(1)}$ as final embeddings of HGCA for evaluation.

B. Node Classification

Node classification has been extensively used to evaluate the quality of learned node embeddings. After learning the node embeddings, we used a linear support vector machine (SVM) [58] classifier to classify nodes. To ensure a fair comparison, we only classified the nodes in the test set because labels in the training and validation sets were already used in semi-supervised methods. We trained the SVM with different training ratios from 10% to 80%. We report here the results of average Macro-F1 and Micro-F1 over 10 runs.

The results are shown in Table III where the best results are in bold fonts and the second-best results are underlined. The newly developed unsupervised HGCA method outperformed the baseline methods compared, including three semi-supervised methods (GAT, HAN, MAGNN), in most cases considered. Compared with the best results from the baseline methods (i.e., DMGI on ACM and MAGNN on Yelp & DBLP), HGCA improved the best results by 0.33%-4.10% and reduces errors by 4.11%-32.03%. The improvement is primarily due to HGCA's capacity of attribute completion and unsu-

TABLE III
PERFORMANCE EVALUATION OF NODE CLASSIFICATION ON THREE DATASETS

Datasets	Metrics	Training	Semi-supervised			Unsupervised			
			GAT	HAN	MAGNN	metapath2vec	DGI	DMGI	HGCA
ACM	Macro-F1	10%	0.8951	0.8923	0.8882	0.6947	0.8955	0.9161	**0.9210**
		20%	0.8977	0.9038	0.8941	0.7011	0.9006	0.9222	**0.9254**
		40%	0.8992	0.9099	0.8983	0.7043	0.9019	0.9251	**0.9300**
		60%	0.9007	0.9130	0.9018	0.7073	0.9034	0.9279	**0.9328**
		80%	0.8976	0.9141	0.9011	0.7113	0.9020	0.9257	**0.9321**
	Micro-F1	10%	0.8942	0.8913	0.8881	0.7381	0.8954	0.9149	**0.9201**
		20%	0.8967	0.9026	0.8936	0.7444	0.8992	0.9207	**0.9245**
		40%	0.8983	0.9089	0.8981	0.7480	0.9004	0.9237	**0.9292**
		60%	0.8998	0.9119	0.9011	0.7522	0.9017	0.9261	**0.9318**
		80%	0.8970	0.9130	0.9006	0.7557	0.9000	0.9238	**0.9303**
Yelp	Macro-F1	10%	0.5403	0.7385	0.8686	0.5396	0.5404	0.7242	**0.9096**
		20%	0.5407	0.7724	0.8786	0.5396	0.5407	0.7506	**0.9157**
		40%	0.5407	0.7848	0.8985	0.5400	0.5407	0.7649	**0.9284**
		60%	0.5400	0.7858	0.9058	0.5396	0.5400	0.7709	**0.9303**
		80%	0.5382	0.7893	0.9057	0.5370	0.5382	0.7793	**0.9359**
	Micro-F1	10%	0.7301	0.7598	0.8668	0.7286	0.7303	0.7852	**0.9029**
		20%	0.7306	0.7885	0.8784	0.7289	0.7306	0.7988	**0.9087**
		40%	0.7314	0.7992	0.8986	0.7295	0.7314	0.8068	**0.9219**
		60%	0.7297	0.7997	0.9064	0.7297	0.7297	0.8100	**0.9242**
		80%	0.7282	0.8041	0.9062	0.7278	0.7282	0.8155	**0.9303**
DBLP	Macro-F1	10%	0.8190	0.8923	**0.9252**	0.7482	0.6892	0.9188	0.9079
		20%	0.8220	0.9038	**0.9270**	0.7666	0.7711	0.9224	0.9228
		40%	0.8217	0.9100	0.9269	0.8214	0.8109	0.9250	**0.9302**
		60%	0.8212	0.9130	0.9275	0.8425	0.8217	0.9260	**0.9325**
		80%	0.8202	0.9141	0.9301	0.8420	0.8268	0.9288	**0.9382**
	Micro-F1	10%	0.8323	0.8913	**0.9308**	0.7586	0.7610	0.9251	0.9191
		20%	0.8351	0.9026	**0.9325**	0.7761	0.8067	0.9287	0.9310
		40%	0.8346	0.9089	0.9325	0.8289	0.8313	0.9295	**0.9369**
		60%	0.8342	0.9119	0.9334	0.8502	0.8382	0.9315	**0.9380**
		80%	0.8332	0.9130	0.9357	0.8495	0.8406	0.9331	**0.9434**

TABLE IV
PERFORMANCE EVALUATION OF NODE CLUSTERING ON THREE DATASETS

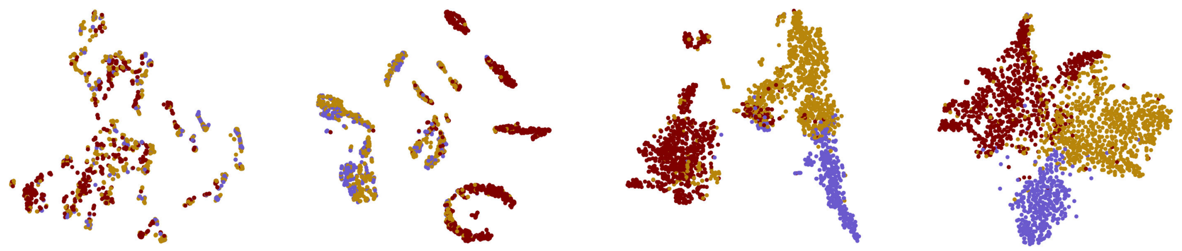
Dataset	Metrics	mp2vec	GAE	DGI	GMI	DMGI	HGCA
ACM	NMI	0.3765	0.4059	0.6153	0.3763	0.6066	**0.6816**
	ARI	0.3026	0.3319	0.6370	0.3022	0.5827	**0.6793**
Yelp	NMI	0.3890	0.3919	0.3942	0.3942	0.3684	**0.3943**
	ARI	0.4249	0.4257	0.4262	0.4260	0.3244	**0.4278**
DBLP	NMI	0.5413	0.5257	0.6615	0.4101	**0.7029**	0.6816
	ARI	0.5756	0.4986	0.7050	0.4056	0.7086	**0.7118**

pervised contrastive representation learning and the effects of integrating the fine-grained attribute completion process and the representation learning process in a unified framework. In addition, metapath2vec, GAT, and DGI performed worse than the other methods (Table III). Metapath2vec only used network topology to learn node representation while ignored attributes. Its poor performance revealed the importance of attribute information. GAT and DGI only used one meta-path-based homogeneous adjacent matrix as input, which only contained one type of semantic relationship. The poor performance of GAT and DGI showed the importance of heterogeneous semantic relations in HIN analysis. Our new HGCA method not only considered a variety of semantic relations between nodes but also accurately completed the missing attributes, which improved performance when combined.

We also compared the performance of $H^{(1)}$ and $H^{(2)}$ of HGCA (see Fig. 3) on the node classification task. For simplicity, we computed the average Macro-F1 and Micro-F1 with different training ratios and reported the difference between the two. Compared to $H^{(1)}$, the performance of $H^{(2)}$ drops to 0.58% and 0.61% for Macro-F1 and Micro-F1, respectively, on ACM dataset, and it decreases to 0.66% and 2.04% on Yelp dataset. This may be because $H^{(2)}$ is generated based on the attribute view $X^{(2)}$, which consists entirely of complemented attributes, whereas $H^{(1)}$ uses attribute view $X^{(1)}$, which contains some existing real attributes. Inevitably, attribute completion cannot fully restore the real attributes so that $H^{(2)}$ is slightly worse than $H^{(1)}$. Despite this fact, the performances of both $H^{(1)}$ and $H^{(2)}$ are better than all baselines. Generally, the performance difference between $H^{(1)}$ and $H^{(2)}$ is very small, showing the success of attribute completion in HGCA. Moreover, $H^{(2)}$ has performance improvements of 0.96% and 0.25% in terms of Macro-F1 and Micro-F1 respectively on the DBLP dataset, which further demonstrates that complemented attributes have great contributions for good embeddings.

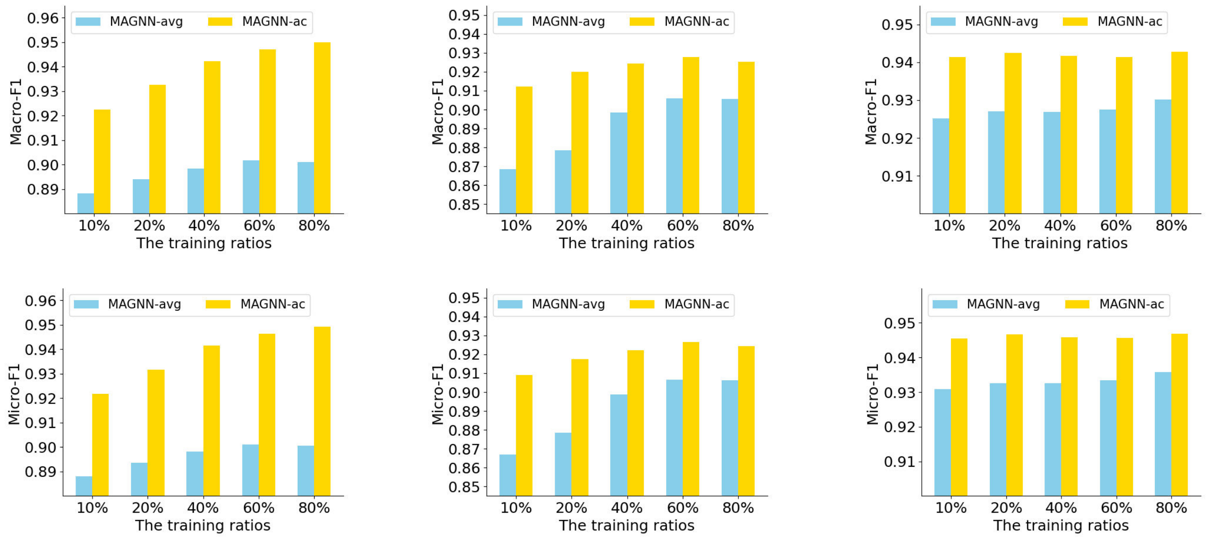
C. Node Clustering

We considered the problem of node clustering in our experiments. We chose five unsupervised methods (metapath2vec,



(a) HAN (b) DMGI (c) MAGNN (d) HGCA

Fig. 6. Visualization of embeddings of business nodes in Yelp. Different colors correspond to different business categories. Among the four methods, HAN and MAGNN are semi-supervised methods, DMGI and our HGCA are unsupervised methods.



(a) ACM (b) Yelp (c) DBLP

Fig. 7. The result(%) of node classification under different attribute inputs. The basic model is MAGNN. In the above histograms, the blue and yellow bars correspond to the result of MAGNN with complemented attributes from average imputation (MAGNN-avg) and that from HGCA (MAGNN-ac), correspondingly. We report Macro-F1 (on the top) and Micro-F1 (on the bottom) on three datasets.

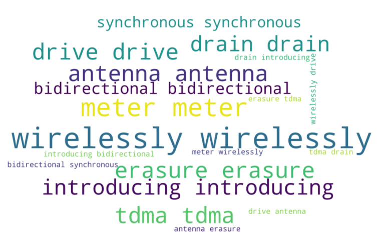
GAE, DGI, GMI, and DMGI) as the baselines for comparison. After learning node embeddings, we ran the K-Means algorithm 10 times. We report here the average normalized mutual information (NMI) and adjusted rand index (ARI) as evaluation metrics. HGCA was the best performer on five of the six cases considered (Table IV). DGI achieved a competitive performance, being the second-best in four of the six cases. This is probably in part because we tested all pre-defined meta-paths separately and reported the best results for DGI. The proposed HGCA used all pre-defined meta-paths simultaneously and automatically learned weights for different meta-paths, which is an advantage over DGI.

D. Visualization

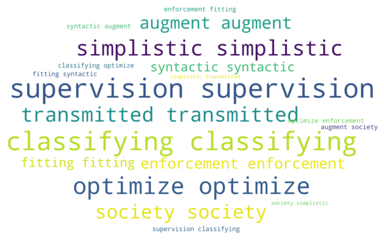
Different from analyzing the impact of missing attributes on semi-supervised and unsupervised representation learning in Section II, we intuitively compared the performance of different methods to visually examine the performance of HGCA. We selected two semi-supervised methods and one unsupervised method for comparison. We utilized t-SNE [59] to project the embeddings of business nodes in Yelp into a 2-dimensional space, where nodes with different colors belong to different classes (Fig. 6). As shown, the results of HGCA have clearer boundaries and denser cluster structures than the other methods compared, demonstrating the superior performance of HGCA. In contrast, HAN and DMGI performed poorly where nodes of different classes were mixed and overlapped. These two methods transformed HINs into multi-relational networks that contained only nodes of target types and ignored other nodes and attributes. HGCA not only used all nodes in HINs for learning embeddings but also accurately completed missing attributes to make unsupervised learning effective.

E. The Generality of HGCA

Accurate attributes are the key to good performance. We designed an experiment to assess whether the attributes complemented by HGCA can be used as a beneficial input to other HIN models. We employed attributes complemented by HGCA as the input to a HGNN model i.e., MAGNN, which was denoted as MAGNN-ac. For comparison, we also

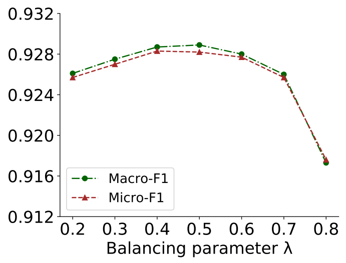


(a) The 298-th paper node (Wireless Communication)

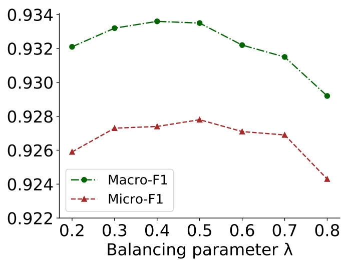


(b) The 410-th paper node (Data Mining)

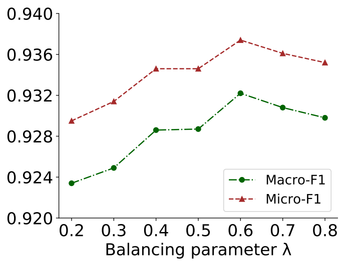
Fig. 8. Visualization of attributes of papers in the form of a word cloud. Take the ACM network as an example, we randomly select two paper nodes with different fields (labels) and use the top 10 words in attributes to generate the word clouds. The title of each sub-figure is the paper's field and paper ID. The bigger the font of the word, the more important its semantics in complemented attributes.



(a) ACM



(b) Yelp



(c) DBLP

Fig. 9. Analysis of the parameter sensitivity of λ. We report the average result of the node classification task under different training ratios on three datasets.

employed attributes complemented by average imputation as the input to MAGNN, which was denoted as MAGNN-avg. We compared the classification results of embedding generated by the two variants of MAGNN. MAGNN-ac performed significantly better than MAGNN-avg in all datasets tested (Fig. 7). This is most likely because the average imputation strategy introduced errors and noises to the complemented attributes, making MAGNN perform poorly. The new HGCA method estimated missing attributes accurately. Importantly, the attributes complemented by HGCA have a good generality and can be used as input to improve the performance of other HIN models, which demonstrates the value of this work for HIN representation learning.

F. Illustration of complemented attributes

We selected two paper nodes in ACM as examples to visualize their attributes in the form of a word cloud (Fig. 8). In this work, we aimed to classify papers according to their three fields of Wireless Communication, Data Mining, and Database. The paper nodes in the ACM dataset have attributes in the form of bag-of-words vectors derived from their abstracts. Therefore, each dimension of a paper node's attribute vector will correspond to a word in the abstract. We restored the semantics of every dimension and generated a word cloud based on the complemented attribute values by our HGCA. As shown in Figure 8, the most important words correctly reflect the paper's field, demonstrating the effectiveness of attribute complementation of our HGCA model.

G. Parameter Analysis

We investigated how model parameters affected HGCA performance. The most important hyperparameters of HGCA are the balancing parameter of loss λ in $\mathcal{L}_{final}$ and the weight of meta-paths η in $\mathcal{L}_1$. Here we report the average result of node classification with different training ratios.

We first analyzed the balancing parameter λ in all three datasets, by performing a grid search with a step size of 0.1 (e.g., varying in $\{0.2, 0.3, ..., 0.8\}$). HGCA achieved the best performance when $\lambda = 0.5$ on ACM and Yelp and $\lambda = 0.6$ on DBLP. Its performance degraded when λ was large or small (Fig. 9). This observation indicates that the contrastive loss between the embeddings of the two views and the contrastive loss between the embeddings and attributes are both important.

We also analyzed the weight of meta-paths η in the ACM dataset where two commonly used meta-paths are "paper-author-paper" (PAP) and "paper-subject-paper" (PSP). We performed a grid search of η from 0.0 to 1.0 with a step size of 0.2. The larger the value of η, the greater the contribution of the corresponding meta-path in constructing positive samples. HGCA achieved good performance when $0.2 \le \eta_{PSP} \le 0.8$ and it was not sensitive to η_{PAP} (Fig. 10). This result indicates that a larger η_{PSP} can help select more effective positive sample pairs to improve model performance. It is reasonable that two papers sharing the same subject are more likely to be in the same field so that PSP is more important than PAP. Therefore, well-designed meta-paths can improve the performance of HGCA, which may very well be true for all HIN models based on meta-path priors.

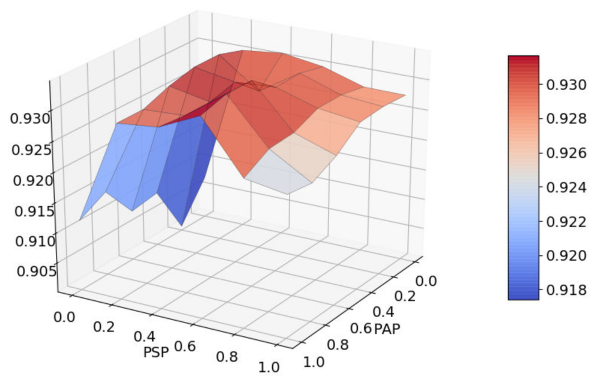


Fig. 10. Analysis of the parameter sensitivity of η on ACM dataset. Shown are the average accuracies of node classification under different training ratios.

VII. Conclusion

We studied and confirmed that the problem of missing attributes can cause severe performance degradation for unsupervised representation learning methods on heterogeneous information networks (HINs). To fill the gap between attribute-missing and unsupervised representation learning on HINs, we developed an unsupervised heterogeneous graph contrastive learning method to integrate missing-attribute completion and node representation learning into an end-to-end architecture. The proposed HGCA method learned joint embeddings via a collaborative random walk to mine the semantic relations among nodes and attributes to achieve an accurate and fine-grained attribute completion. HGCA used a meta-path-based heterogeneous graph neural network (HGNN) to learn node representations. The attribute completion and HGNN modules were integrated by a contrastive learning strategy. This learning strategy made the complemented attributes reliable and beneficial to the HGNN module. It also made node representations discriminative for downstream tasks. Extensive experiments on three large real-world datasets demonstrated the superiority of the new HGCA approach over state-of-the-art methods. Experimental results also showed that the complemented attributes by the new model can generally benefit other HIN models, showing the generality and rigor of the new method for HIN representation learning.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[3] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, 2018.

[4] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, 2017.

[5] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Trans. Knowl. Data Eng.*, 2020.

[6] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.

[7] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *NeurIPS*, 2019, pp. 11 960–11 970.

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

[9] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *SIGKDD*, 2019, pp. 793–803.

[10] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *SIGKDD Explor.*, vol. 14, no. 2, pp. 20–28, 2012.

[11] C. Park, J. Han, and H. Yu, "Deep multiplex graph infomax: Attentive multiplex network embedding using global information," *Knowl. Based Syst.*, vol. 197, p. 105861, 2020.

[12] J. Zhao, X. Wang, C. Shi, Z. Liu, and Y. Ye, "Network schema preserving heterogeneous information network embedding," in *IJCAI*, 2020, pp. 1366–1372.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[14] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[15] L. Bai, L. Cui, Y. Jiao, L. Rossi, and E. Hancock, "Learning backtrackless aligned-spatial graph convolutional networks for graph classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[16] L. Bai, Y. Jiao, L. Cui, L. Rossi, Y. Wang, P. Yu, and E. Hancock, "Learning graph convolutional networks based on quantum vertex information propagation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[17] W. Sheng and X. Li, "Multi-task learning for gait-based identity recognition and emotion recognition using attention enhanced temporal graph convolutional network," *Pattern Recognition*, vol. 114, p. 107868, 2021.

[18] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proc. VLDB Endow.*, vol. 4, no. 11, pp. 992–1003, 2011.

[19] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng, "Meta-path guided embedding for similarity search in large-scale heterogeneous information networks," *arXiv preprint arXiv:1610.09769*, 2016.

[20] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *AAAI*, 2020, pp. 4132–4139.

[21] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020, pp. 2704–2710.

[22] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018, pp. 593–607.

[23] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *ICLR*, 2020.

[24] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, R. D. Hjelm, and A. C. Courville, "Mutual information neural estimation," in *ICML*, vol. 80, 2018, pp. 530–539.

[25] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020.

[26] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *arXiv preprint arXiv:2011.00362*, 2020.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.

[28] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *NeurIPS*, 2020.

[29] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *NeurIPS*, 2020.

[30] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep Graph Contrastive Representation Learning," in *ICML*, 2020.

[31] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu and L. Wang, "Graph contrastive learning with adaptive augmentation," *arXiv preprint arXiv:2010.14945*, 2020.

[32] R. Zhu, Z. Tao, Y. Li, and S. Li, "Automated graph learning via population based self-tuning GCN," in *SIGIR*, 2021, pp. 2096–2100.

[33] Z. Tao, Y. Li, B. Ding, C. Zhang, J. Zhou, and Y. Fu, "Learning to mutate with hypergradient guided population," in *NeurIPS*, 2020.

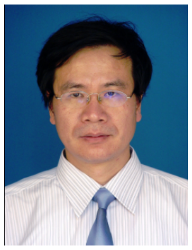
[34] R. Zhu, Z. Tao, Y. Li, and S. Li, "Automated graph learning via population based self-tuning GCN," in *SIGIR*, 2021, pp. 2096–2100.

[35] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," *arXiv preprint arXiv:2106.07594*, 2021.
[36] Y. Yin, Q. Wang, S. Huang, H. Xiong, and X. Zhang, "Autogcl: Automated graph contrastive learning via learnable view generators," *arXiv preprint arXiv:2109.10259*, 2021.
[37] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "GCC: graph contrastive coding for graph neural network pre-training," in *SIGKDD*, 2020, pp. 1150–1160.
[38] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, 2020.
[39] H. Zhang, S. Lin, W. Liu, P. Zhou, J. Tang, X. Liang, and E. P. Xing, "Iterative graph self-distillation," *arXiv preprint arXiv:2010.12609*, 2020.
[40] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Comput. Sci.*, vol. 14, no. 7, pp. 38–39, 2015.
[41] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.
[42] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *KDD*, 2021.
[43] J. You, X. Ma, D. Y. Ding, M. J. Kochenderfer, and J. Leskovec, "Handling missing data with graph representation learning," in *NeurIPS*, 2020.
[44] H. Taguchi, X. Liu, and T. Murata, "Graph convolutional networks for graphs containing missing features," *Future Gener. Comput. Syst.*, vol. 117, pp. 155–168, 2021.
[45] X. Chen, S. Chen, J. Yao, H. Zheng, Y. Zhang, and I. W. Tsang, "Learning on attribute-missing graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.
[46] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017, pp. 214–223.
[47] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *NeurIPS*, 2014, p. 2672–2680.
[48] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proc. ICCV*, 2017, pp. 2794–2802.
[49] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *SIGKDD*, 2019, pp. 732–740.
[50] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.
[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.
[52] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *arXiv preprint arXiv:2103.00111*, 2021.
[53] Y. Lu, C. Shi, L. Hu, and Z. Liu, "Relation structure-aware heterogeneous information network embedding," in *AAAI*, 2019, pp. 4456–4463.
[54] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017, pp. 135–144.
[55] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NeurIPS Workshop on Bayesian Deep Learning*, pp. 1–3, 2016.
[56] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.
[57] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *WWW*, 2020, pp. 259–270.
[58] J. A. K. Suykens, "Support vector machines: A nonlinear modelling and control perspective," *Eur. J. Control*, vol. 7, no. 2-3, pp. 311–327, 2001.
[59] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *J. Mach. Learn. Res.*, vol. 9, no. 2605, pp. 2579–2605, 2008.

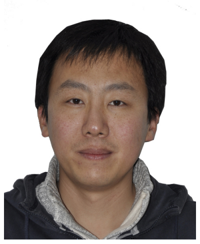
Dongxiao He received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2014. She was a Post-Doctoral Research Fellow in Department of Computer Science, Dresden University of Technology, Germany, from 2014 to 2015. She is an Associate Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. She has published over 40 international journal and conference papers. Her current research interests include data mining and analysis of complex networks.

Chundong Liang received the B.S. degree from the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China, in 2019. He is currently working toward the M.S. degree in the College of Intelligence and Computing, Tianjin University, China. His research interests include social networks, deep learning and data mining.

Cuiying Huo received the B.S degree from Yanshan University, China, in 2019. She is currently a Ph.D. student of the College of Intelligence and Computing, Tianjin University, China. Her research interests are mainly related to data mining, social network analysis and machine learning.



Zhiyong Feng received the Ph.D. degree from Tianjin University. He is currently a full professor in the School of Computer Science and Technology, Tianjin University, China. He is the author of one book, more than 130 articles, and 39 patents. His research interests include knowledge engineering, services computing, and security software engineering. He is a member of the IEEE and the ACM.



Di Jin received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2012. He was a Post-Doctoral Research Fellow at the School of Design, Engineering, and Computing, Bournemouth University, Poole, U.K., from 2013 to 2014. He is currently an Associate Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. He has published more than 50 papers in international journals and conferences in the areas of community detection, social network analysis, and machine learning. He serves as invited reviewers for journals including TKDE, and senior program committees for conferences including AAAI.

Liang Yang received the B.E. and M.E. degrees both in computational mathematics from Nankai University, Tianjin, China, in 2004 and 2007, and the Ph.D. degree in computer science from the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2016. He is an Assistant Professor in the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China. His current research interests include community detection, graph neural networks, low-rank modeling, and data mining.

Weixiong Zhang received the Ph.D. degree from the Department of Computer Science, University of California, Los Angeles, in 1994. He is a professor of computer science and a professor of genetics with Hong Kong Polytechnic University in Hong Kong, China. His research interests include computational molecular and systems biology, artificial intelligence, and combinatorial optimization.