Hierarchical Representation Learning for Attributed Networks

Shu Zhao, Ziwei Du, Jie Chen, Yanping Zhang, Jie Tang, *Senior Member, IEEE,* and Philip S. Yu , *Fellow, IEEE*

Abstract—Network representation learning, also called network embedding, aiming to learn low dimensional vectors for nodes while preserving essential properties of the network, benefits plenty of practical applications. However, how to do representation learning on the network quickly and effectively is a meaningful and challenging task, especially for the attributed networks. In this paper, we propose HANE, a **H**ierarchical **A**ttributed **N**etwork **E**mbedding framework, which is a fast and effective method by quickly constructing a hierarchical attributed network of different granularities to learn nodes representations. Specifically, for an attributed network, HANE first builds a hierarchy of successively smaller attributed network from fine to coarse by the fast granulation strategy fusing topological structure and node attributes. After using any unsupervised network embedding method to learn nodes representations of the coarsest network, HANE refines the nodes representations of the hierarchical attributed network from coarse to fine. HANE improves the speed of network representation learning while maintaining its performance and the representation learning method of the coarsest network is flexible. We conduct extensive evaluations for the proposed framework HANE on six datasets and two benchmark applications. Experimental results demonstrate that HANE achieves significant improvements over previous state-of-the-art network embedding methods in efficiency and effectiveness.

Index Terms—Network representation learning, network embedding, attributed network, hierarchical attributed network, granulation.

1 Introduction

NETWORK is an important data structure to explore and model complex systems in the real world. More and more machine learning applications conduct classification or prediction based on network data. As a fundamental tool to analyze networks, network representation learning has attracted increasing attention in the recent few years. Network representation learning, also called network embedding, aims to map each node into a low-dimensional vector representation by preserving network structure and inherent properties. It has attracted tremendous attention recently due to significant progress in downstream network analysis tasks such as node classification, link prediction. Recently, a large number of network representation learning methods have been proposed. For instance, DeepWalk [1], node2vec [2], LINE [3] are pioneering works that introduce deep learning techniques into network analysis to learn node embeddings. NetMF [4] gives a theoretical analysis of equivalence for the different network embedding algorithms, such as DeepWalk [1], node2vec [2], PTE [5], and LINE [3]. CNRL [6] simultaneously detects community distribution of each node and learns the representation of both nodes and communities. MCNE [7] to learn multiple conditional network representations, so that various preferences for multiple behaviors could be fully captured. STNE [8], CAN [9], ASNE [10], DANE [11], BANE [12], metapath2vec [13] and GATNE [14] are proposed for attributed network embedding that combines the network topological structure and node attributes simultaneously (homogeneous or heterogeneous). DeepGL [15] is a general inductive network representation learning framework for learning deep node and edge embeddings that generalize across-networks. Nevertheless, these methods are computationally expensive to handle the network in a single granularity that needs to calculate the entire network and most methods to solve the problems of attributed networks are time-consuming and difficult to apply to large-scale attributed networks. More recently, a few embedding methods that have proposed fast approaches to learn node embeddings have been developed. One of them is the study of hierarchical network representation. MILE [16], HARP [17] and LouvainNE [18] adopt hierarchical frameworks to preserve the network's local or global structure across granularities. Some of them can make large-scale problems solvable. However, they do not take into account node attributes that are also important for network analysis. Hierarchical attributed network embedding has been least studied. Recently Deng and Zhao et al. [19] propose that GraphZoom first integrates the fusion of attribute information to generate a new graph, and then adopt hierarchical frameworks to improve accuracy and scalability. However, in many real-world applications, a variety of

- *Shu Zhao, Ziwei Du, Jie Chen are with the Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education. School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, P.R.China .*
  E-mail: zhaoshuzs@ahu.edu.cn, duziwei1225@163.com . chenjie200398@163.com
- *Yanping Zhang is with the Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education. School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, P.R.China. .*
  E-mail: zhangyp2@gmail.com .
  **Corresponding author*
- *Jie Tang is with the Department of Computer Science and Technology . Tsinghua University, and Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, China, 100084. .*
  E-mail: jietang@tsinghua.edu.cn
- *Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA .*
  E-mail: psyu@uic.edu .

networks, such as coauthor network, citation network, are not only accompanied with a rich set of attributes, but presented in a hierarchical way. As shown in Fig.1, according to the attribute of the nodes, the citation network about computer science can be divided into hierarchical structure, a branch of the hierarchical structure from coarse to fine are Artificial Intelligence (AI), Natural Language Processing (NLP), Information Extraction (InfoE) and citations. Among them, computer science represents the coarsest node, and citations represent the finest node. Such attributed networks with hierarchical topology structure and attribute information are helpful to learn the relationship between nodes from different aspect and different granularity. Preserving the hierarchical attribute information is still a challenge. HANE (**H**ierarchical **A**ttributed **N**etwork **E**mbedding) can obtain hierarchical attribute information through fusion layer by layer, and builds the relationship between different granularities.

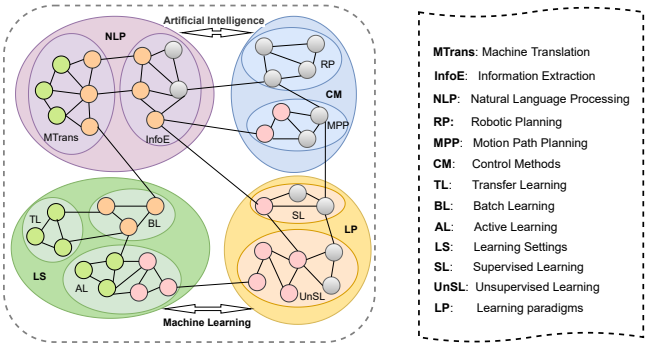


Fig. 1. An illustration of citation network with hierarchical topology structure and attribute information. Among them, nodes represent papers, edges represent citation relationships, and nodes with the same color have similar keywords.

The graph neural network that promotes the deep neural network model to graph structure data has attracted the attention of many researchers in recent years. There are some recent works on hierarchically learning network representation analogous to the pooling step in conventional convolutional neural (CNN) networks, including DIFFPOOL [20], EigenPooling [21], and so on. These methods divide the nodes into subgraphs (supernodes), coarsening them according to the subgraphs, and then reduce the entire graph information to coarsening graphs by generating the features of the supernodes from the corresponding nodes in the subgraphs. However, most of them are used for graph classification or semi-supervised node classification and they need labeled data to train the representation model, which is usually difficult in practice. We focus on the unsupervised network embedding methods rather than supervised/semi-supervised ones [22], [23], [24], [25], [26], [27] in this paper.

The rapid development of large-scale networks, existing unsupervised network embedding approaches are still with the following challenges posed by the characteristics of large-scale attributed networks in real-world applications. First, how to reduce the scale of the network so that the large-scale attributed network can be represented more effectively? Second, how to quickly and effectively preserve the network topology and the hierarchical attribute information during the node representation learning process to ensure the validity of the results? To tackle the above challenges, in this paper, we present HANE, a Hierarchical Attributed Network Embedding framework to learn node embeddings for the attributed network in a hierarchical way. We use the granulation module to compress the network structure and attributes, shrink the network scale, and thus reduce the computational complexity of the network representation learning. HANE mainly includes three modules as follows. (1) Granulation Module (GM). Given an attributed network, a hierarchical attributed network from large-scale to small-scale are generated by the continuous fusion of the granulation of the network structure and node attributes. The network topology and node attributes are preserved between these networks of different scales from fine-grained to coarse-grained, so that we can obtain an approximate solution at the coarsest granularity. (2) Network Embedding Module (NE). At the coarsest granularity, we use one of the existing unsupervised network embedding (attributed or structure-only) methods for network representation learning with high speed because the scale of the attributed network becomes small and the approximate solution expressed of the original network node is obtained. (3) Refinement Module (RM). This module refines the node embeddings from coarse to fine. It mainly inherits the embeddings from the coarse granularity and uses an unsupervised graph convolutional neural network to update them according to the topological structure and node attributes of the network at the current granularity. This process is highly efficient because we do not need to use a network embedding method to relearn node embeddings at each granularity.

In this paper, we study an unsupervised method for fast learning the representation of each node of an attributed network in a hierarchical way. We summarize the contributions of this paper as follows.

- We investigate the large-scale attributed network embedding problem in a hierarchical way that not only incorporates the network topological structure information and node attributes information within one granularity, but infers these two kinds of information across granularities.
- We propose a fast and efficient framework HANE, which consists of a granulation module (GM), a network embedding module (NE), and a refinement module (RM). Also, the NE module of HANE is flexible that structure-only network embedding methods and attributed network embedding methods are both suitable for it at the coarsest granularity.
- Our extensive evaluations show that the effectiveness of HANE, which outperforms state-of-the-art network embedding methods on node classification (e.g., 0.73%∼3.48% average relative lift in $Micro\_F1$ scores) and link prediction (e.g., 0.62%∼3.61% relative lift in AUC scores) tasks. More importantly, HANE is efficient in the representations learning process, showing its high average speedup (e.g., 1.72×∼1016.88×) over state-of-the-art baselines.

2 RELATED WORK

In this section, we review the related state-of-the-art of network embedding methods from four aspects, includ-

ing single-granularity structure-only network embedding methods, single-granularity attributed network embedding methods, hierarchical structure-only network embedding methods, and hierarchical attributed network embedding.

Single-granularity Structure-only Network Embedding. This kind of method projects the nodes of a network into a low-dimension vector space by preserving network local or global structure-property. Representative works include DeepWalk [1], node2vec [2], LINE [3], NetMF [4], GraRep [28], HOPE [29], etc. DeepWalk [1] and node2vec [2] are random walk based methods which generate a corpus on networks by some random walks and then train a skip-gram model on the corpus. LINE [3] learns node representations on large-scale networks while preserving both first-order and second-order proximities. GraRep [28], HOPE [29] and NECS [30] are capable to preserve high-order proximities of networks. NetMF [4] is a unified matrix factorization framework for theoretically understanding and improving DeepWalk and LINE, etc. ProNE [31] is a fast, scalable, and effective model that first initializes network embeddings efficiently by formulating the task as sparse matrix factorization and then enhances the embeddings by propagating them in the spectrally modulated space. DNE [32] is the first work that learns a discrete representation for networks. SPINE [33] jointly captures the local proximity and proximities at any distance while being inductive to deal with unseen nodes efficiently. NodeSketch [34] preserves high-order node proximity via recursive sketching that generates node embeddings in Hamming space. This kind of method has been proved to have good performances on a variety of regular network analysis tasks, such as link prediction or multi-label classification. However, they do not take into account utilizing node attributes information that is important for network analysis.

Single-granularity Attributed Network Embedding. Attributed network embedding aims to seek low-dimensional vector representations for nodes in a network, such that the original network topological structure and node attribute proximity can be preserved in such representations. STNE [8] adopts a content-to-node translation model to preserve contents and structure properties. CAN [9] proposes a variational auto-encoder that embeds each node and attribute with means and variances of Gaussian distributions. AANE [35] enables a joint learning process to be done in a distributed manner for accelerated attributed network embedding. SNE [10] proposes a generic framework for embedding social networks by capturing both the structure proximity and attribute proximity. TADW [36] incorporates text features of nodes into network representation learning under the framework of matrix factorization. CANE [37] learns context-aware embeddings for nodes with mutual attention mechanisms and is expected to model the semantic relationships between nodes more precisely. ANRL [38] uses a neighbor enhancement auto-encoder to model the node attributes and an attribute-ware skip-gram model based on the attribute encoder to capture the network structure. DGENE [39] is a general end-to-end model that leverages the complementary information of network structure and content. ProGAN [40] is a proximity generative adversarial network for network embedding. PGE [41] incorporates not only the network topological structure and node attributes but edge properties into the network embedding procedure. DANE [11] captures the high nonlinearity and preserve various proximities in both topological structure and node attributes. SANE [42] learns the topological structure and sparse node attribute information simultaneously in a united approach. metapath2vec [13], GATNE [14] and HERec [43] are for heterogeneous network embedding. GCN [22] uses graph convolutional networks to learn node embeddings, by merging local graph structures and features of nodes to obtain embeddings from the hidden layers. GraphSAGE [23] uses node property information in neighbor aggregation to efficiently generate node embeddings for previously unseen data. Yet, there are generally time-consuming. In this paper, we increase the speed by reducing the size of the network, many of the above methods can be applied to our method.

Hierarchical Structure-only Network Embedding. This kind of method generally captures the hierarchical network structure. HARP [17] proposes a hierarchical paradigm for network embedding based on iterative learning methods (e.g., DeepWalk and node2Vec). HARP focuses on improving the quality of embeddings by using the learned embeddings from the previous level as the initialized embeddings for the next level. MILE [16] repeatedly coarsens the network into smaller ones by merging nodes with similar local structures. It then applies existing embedding methods on the coarsest and refines the embeddings to the original network through a graph convolution neural network that it learns. LouvainNE [18] creates the partition hierarchy of the original graph by reusing the Louvain algorithm, and generates a level-specific node embedding of each partition in the hierarchy and combines the embeddings of all levels to compute the node embeddings in the original graph. GNE [44] formulates an optimization problem with spherical constraints to describe the hierarchical community structure-preserving network embedding. spaceNE [45] preserves hierarchies formed by communities through subspace, manifolds with flexible dimensionalities and is inherently hierarchical. MINES [46] incorporates multi-dimensional relations and hierarchical structure into a coherent model for node representation learning. HSRL [47] recursively compresses an input network into a series of smaller networks using a community-awareness compressing strategy to capture both the local and global topological information of a network. Ma and Cui et al. [48] propose a network embedding model NetHiex that captures the latent hierarchical taxonomy, which is generally unknown. However, these embedding techniques only account for the network structure. Our model follows the idea to captures the hierarchical network structure, but we integrate the attribute information of the node to obtain a higher quality node representation.

Hierarchical Attributed Network Embedding. This kind of methods fuse the attribute information of nodes while capturing the hierarchical structure. GraphZoom [19] first fused the topology and attributes of the original network to generate a new network. Then, by merging nodes with high spectral similarity, this fused network is repeatedly coarsened into a much smaller network and the embeddings obtained at the coarsest level to increasingly finer networks. However, GraphZoom is based on the spectral method, the scalability is not good enough, and GraphZoom can only capture the attribute information of the original network,

but cannot preserve the hierarchical attribute information. HANE can obtain hierarchical attribute information through fusion layer by layer, and constructs a hierarchical attributed network of different granularities to learn nodes representation.

3 PROBLEM FORMULATION

Let $G = (V, E, \mathbf{X})$ be an attributed network, where V denotes the set of n nodes and E represents the set of m edges. $\mathbf{X} \in \mathbb{R}^{n \times l}$ is a matrix that encodes all node attributes information, and $\mathbf{x}_i$ describes the attributes associated with node v_i. We will give some formal definitions of the problem for a better description. First, the definition of attributed network embedding is given, as in definition 3.1.

Definition 3.1 (Attributed Network Embedding) Given an attributed network $G = (V, E, \mathbf{X})$, we aim to represent each node $v_i \in V$ as a low-dimensional vector $\mathbf{z}_i$ by learning a mapping function $f_G : V \rightarrow \mathbf{Z} \in \mathbb{R}^{n \times d}$, where $d \ll n$ and the mapping function f_G preserves not only network structure but also node attributes proximity.

In this paper, we let $\succ$ to denote finer. $G^i \succ G^{i+1}$ means G^i is at a finer granularity than G^{i+1}. That is, $\mid V^i \mid > \mid V^{i+1} \mid$. Thus, hierarchical attributed network constructed by networks with different granularities can be defined as follows.

Definition 3.2 (Hierarchical Attributed Network) Given an attributed network $G = (V, E, \mathbf{X})$, let $G^0 = G$, a series of networks at different granularities $G^0 \succ G^1 \succ \cdots \succ G^k$ is called a hierarchical attributed network. $G^i \succ G^{i+1}$, $i = 0, 1, 2, \cdots, k-1$, represents that G^{i+1} is at a coarser granularity than G^i, $G^i = (V^i, E^i, \mathbf{X}^i)$, $\mid V^i \mid > \mid V^{i+1} \mid$. k is the granularities of the hierarchical network. $v_j^{i+1} \in V^{i+1}$ denotes the j^{th} node of the network G^{i+1}, which is a supernode formed by a node subset V_j^i composed of several nodes of the network G^i.

In order to granulate G^i to G^{i+1}, we first introduce the definition of equivalence relation and equivalence class. Then we define two equivalence relations used in this paper, i.e., structure-based equivalence relation R_s and attribute-based equivalence relation R_a. Based on these definitions, we give a granulation strategy in section 4 to build a hierarchical attribute network.

Definition 3.3 (Equivalence Relation and Equivalent Class [49]) Assume that A is a set, R is a binary relation on A, for any $a, b, c \in A$,

(1) Reflexivity: aRa.

(2) Symmetry: If aRb, then bRa.

(3) Transitivity: If aRb and bRc, then aRc.

R is called an equivalence relation on A. The equivalence class of a under R, denoted $[a]_R$, is defined as $[a]_R = \{b \in A | aRb\}$. Two elements of the given set are equivalent to each other if and only if they belong to the same equivalence class. Let A/R denote the collection of equivalence classes. Any equivalence relation provides a partition of the underlying set into disjoint equivalence classes.

Inspired by the study of networks that community structure is an important and quite common characteristic in the network [50], we give the structure-based equivalence relation R_s as Definition 3.4.

Definition 3.4 (Structure-based Equivalence Relation R_s) Define a binary relation R_s on node set V, denoted $v_i R_s v_j$, if v_i and v_j are in the same non-overlapping community which is detected based on the network topological structure, where v_i and $v_j \in V$.

Obviously, R_s is an equivalence relation.

According to the proverb that birds of a feather flock together, we give the attribute-based equivalence relation R_a in this paper.

Definition 3.5 (Attribute-based Equivalence Relation R_a) Define a binary relation R_a on the node set V, denoted $v_i R_a v_j$, if v_i and v_j are in a same non-overlapping clustering which is clustered based on node attributes, where v_i and $v_j \in V$.

Similarly, R_a is an equivalence relation.

Lemma 3.1 If R_s and R_a are equivalence relations on node set V, $R_{node} = R_s \bigcap R_a$ is an equivalence relation on node set V.

Based on the definitions above, a hierarchical attributed network embedding framework is proposed, we formulate the hierarchical attributed network embedding problem as follows.

Problem Formulation (Hierarchical Attributed Network Embedding) Given an attributed network $G = (V, E, \mathbf{X})$, granulate G to a hierarchical attributed network $G = G^0 \succ G^1 \succ \cdots \succ G^k$. The goal is to find a function $f_G : V \rightarrow \mathbf{Z} \in \mathbb{R}^{n \times d}$, where $d \ll n$, preserving both the network topological structure and node attributes proximity from the network G^k at the coarsest granularity to the original network G.

4 METHODOLOGY

To address the problem above, we propose a hierarchical attributed network embedding framework HANE, which is a fast and effective method by quickly incorporates both the network topological structure and node attributes in a hierarchical way. We first compress the scale of attributed network by continuous granulation and fusion of attributes and structures. And the small-scale network retains the backbone information of the network structure and attributes. Thereby reducing the complexity of network representation. Fig. 1 shows an overview of the proposed HANE framework, which contains three key modules: GM, NE, and RM. Granulation module (GM) to construct a hierarchical attributed network by considering both the network topological structure and node attributes from fine to coarse. Network embedding module (NE) is to learn the node representations on the coarsest attributed network. The choice of the underlying network representation learning technology at this stage is flexible, we can use any unsupervised method of network representation (attribute or structure-only). Refinement module (RM) utilizes the unsupervised graph convolutional neural networks to refine the node representations on the hierarchical attributed network from coarse to fine. In the following three subsections, we discuss the details of the above three modules.

4.1 Granulation Module (GM)

Given an attributed network $G = (V, E, \mathbf{X})$, the purpose of the granulation module GM is to construct a hierarchical attributed network from fine to coarse $G = G^0 =$

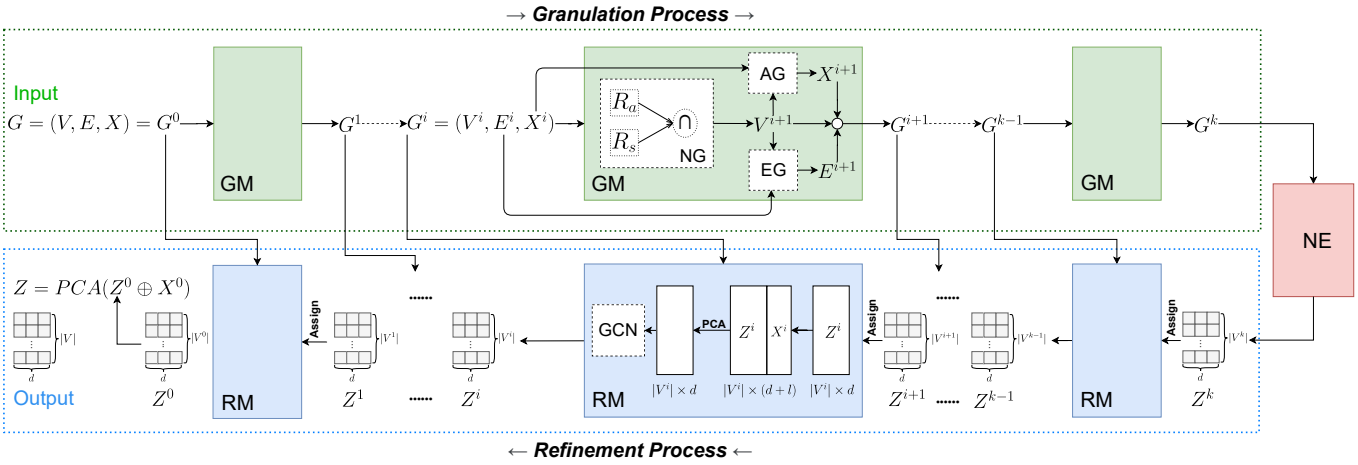


Fig. 2. The overview of our proposed framework HANE. GM is the granulation module which constructs a hierarchical attributed network from fine to coarse $G = G^0 \succ G^1 \succ \cdots \succ G^k$. NE is the network embedding module for the network G^k at the coarsest granularity. RM is the refinement module which refines the node representations on the hierarchical attributed network from coarse to fine $Z^k, Z^{k-1}, \cdots, Z^0, Z$. NG, EG, and AG are the nodes granulation submodule, edges granulation submodule, and attributes granulation submodule in granulation module GM. R_s and R_a are structure-based equivalence relation and attribute-based equivalence relation which are combined to granulate nodes in the NG submodule.

$(V^0, E^0, \mathbf{X}^0) \succ G^1 = (V^1, E^1, \mathbf{X}^1) \succ \cdots \succ G^k = (V^k, E^k, \mathbf{X}^k)$, where G^{i+1} is a new abstraction level of $G^i, i = 0, 1, \cdots, k-1$. In this phase, we propose a granulation strategy that combines attributes and structure to improve the effectiveness of the results. In order to granulate G^i to G^{i+1} preserving attributes and structure, there are three steps: nodes granulation (NG) which granulates V^i to V^{i+1} to generate super-nodes, edges granulation (EG) which forms super-edges E^{i+1} based on V^{i+1} and E^i, attributes granulation (AG) which forms attributes $\mathbf{X}^{i+1}$ of super-nodes based on V^{i+1} and $\mathbf{X}^i$.

Nodes Granulation (NG). Based on the definitions of section 3, we will discuss how to granulate the nodes in G^i to form super-nodes in G^{i+1} by combing the network topological structure and node attributes simultaneously. Then we give the details of nodes granulation by combining R_s and R_a.

First, here the Louvain algorithm [51] is employed, which is one of the most popular and fast community detection methods. Many community detection methods can be also used to detect the non-overlapping communities of G^i. According to Definition 3.4 R_s, node set V^i is partitioned into several equivalence classes, denoted as $V^i/R_s = \{V^i_{s_1}, V^i_{s_2}, \cdots\}$, where V^i_j is the j^{th} equivalence class of V^i. We then use mini-batch k-means algorithm [52] which is one of the widely-used clustering methods to partition the node set V^i into several non-overlapping clusters. According to Definition 3.5 R_a, node set V^i is also partitioned into several equivalence classes, denoted as $V_i/R_a = \{V^i_{a_1}, V^i_{a_2}, \cdots\}$. Finally, according to Leamma 3.1, we fuse the network structure and node attributes to partition V^i of G^i by R_{node}, denoted as $V^i/R_{node} = V^i/(R_s \bigcap R_a) = \{V^i_1, V^i_2, \cdots\}$. Each equivalence class $V^i_j, j = 1, 2, \cdots$, is regarded as a super-node v^{i+1}_j in V^{i+1} of G^{i+1}.

The above nodes granulation process successfully forms $V^{i+1} = \{v^{i+1}_1, v^{i+1}_2, \cdots\}$, from the attributed network $G^i = (V^i, E^i, \mathbf{X}^i)$ by its network structure and node attributes.

Edges Granulation (EG). After acquiring the node set V^{i+1} of network G^{i+1} from the network G^i, we will discuss how to form the edge set E^{i+1} of the network G^{i+1}.

Given an attributed network $G^i = (V^i, E^i, \mathbf{X}^i)$ and $V^{i+1} = \{v^{i+1}_1, v^{i+1}_2, \cdot \cdot \cdot\}$, where v^{i+1}_j is a super-node granulated by an equivalence class V^i_j of V^i. For $\forall v^{i+1}_p$ and $v^{i+1}_q \in V^{i+1}$, $e^{i+1}_{pq} = (v^{i+1}_p, v^{i+1}_q) \in E^{i+1}$,

$$e^{i+1}_{pq} = \begin{cases} True, & if \quad \exists v^i_w \in V^i_p, v^i_s \in V^i_q, (v^i_w, v^i_s) \in E^i, \\ False, & otherwise. \end{cases} \tag{1}$$

This edges granulation process forms edge set E^{i+1} of G^{i+1} according to node set V^{i+1} of G^{i+1} and edge set E^i of G^i. If $e^{i+1}_{pq} = True$, v^{i+1}_p and v^{i+1}_q will be connected.

Attributes Granulation (AG). In order to obtain coarse-grained attribute information, we will discuss how to form the attribute $\mathbf{X}^{i+1}$ of G^{i+1} from $\mathbf{X}^i$ of G^i.

Given the network $G^i = (V^i, E^i, \mathbf{X}^i)$ and $V^{i+1} = \{v^{i+1}_1, v^{i+1}_2, \cdots\}$, each dimension of $\mathbf{X}^i$ represents different attribute information, the attribute $\mathbf{x}^{i+1}_j$ of each super node $v^{i+1}_j \in V^{i+1}$ should contain the attribute information of nodes in G^i it contains. Assume the attribute values are numerical, here we use the mean to represent the attribute information of the super node, the attribute vector $\mathbf{x}^{i+1}_j$ of the node v^{i+1}_j is calculated as Equation (2), $j = 1, 2, \cdots$

$$\mathbf{x}^{i+1}_j = \frac{1}{|V^i_j|} \sum_{p=1}^{|V^i_j|} \mathbf{x}^i_p, \tag{2}$$

where $\mathbf{x}^i_p$ is the attribute vector of the node v^i_p in the j^{th} node subset V^i_j of the node set V^i, i.e., $v^i_p \in V^i_j \subseteq V^i$.

After the process of granulation module, we generate a super-network and iteratively granulation to construct a series of attributed networks $G^0, G^1, \cdots, G^k$ at different granularities which reveal the hierarchical network structures and node attributes of the original network G. It is obvious that $| V^i | > | V^{i+1} |, | E^i | > | E^{i+1} |, i = 0, 1, 2, \cdots, k-1$.

4.2 Network Embedding on the Coarsest Attributed Network

From above, a hierarchical attributed network from fine to coarse $G = G^0 \succ G^1 \succ \cdots \succ G^k$ is constructed by iteratively implementing granulation module (GM).

For the coarsest network $G^k = (V^k, E^k, \mathbf{X}^k)$ which is smaller than the original attributed network G. And it retains the backbone information of the original network structure and attributes. We learn the representation of nodes on G^k to acquire the approximate solution of the original network as Equation (3).

$$\mathbf{Z}^k = PCA(\alpha \cdot f_{G^k}(V^k) \oplus (1-\alpha)\mathbf{X}^k), \tag{3}$$

where $f_{G^k}(\cdot)$ is a network representation learning method to learn the nodes representation of the coarsest attributed network. α is a parameter. If $f_{G^k}(\cdot)$ is a structure-only-based network embedding method, $0 \leq \alpha \leq 1$, here we set $\alpha = 0.5$. If $f_{G^k}(\cdot)$ is an attributed network embedding method, $\alpha = 1$. The attributed network embedding method can combine attribute information well, so operation $\oplus$ and $PCA(\cdot)$ is no longer executed. $\oplus$ means concatenation operator, so that network structure and node attributes are fused for the attributed network embedding at the coarsest granularity. Principal components analysis (PCA) [53] aimed to find a linear projection of the original data, here used to reduce the dimensionality of the learned embedding from $(d+l)$ to d, so that the coarsest network embeddings $\mathbf{Z}^k \in \mathbb{R}^{|V^k| \times d}$. It is worth noting that $\mid V^k \mid$ is much smaller than $\mid V \mid$, so the speed of network embedding for G^k is far faster than for G.

In this stage, we choose network representation learning technology as $f_{G^k}(\cdot)$ is very flexible. Many existing unsupervised network embedding methods can be used, such as structure-only network embedding DeepWalk, node2vec and attributed network embedding STNE, CAN, and so on.

4.3 Refinement Module (RM)

Our goal is to learn the node representation of the original attributed network, preserving both the network topological structure and node attributes proximity from the network G^k at the coarsest granularity to the original network G. So our final stage is to refine the representation of nodes from coarse to fine.

Given a hierarchical attributed network $G = G^0 \succ G^1 \succ \cdots \succ G^k$ and the embedding $\mathbf{Z}^k$ of the coarsest network G^k, we will discuss how to learn the embedding $\mathbf{Z}$ of the original network G by refinement module RM from G^k to G.

For a better description, we first study an easier subtask: learn embedding $\mathbf{Z}^i$ according to G^i and $\mathbf{Z}^{i+1}$. Once we solved this subtask, we can then iteratively apply the technique on each pair of hierarchical attributed networks from the coarsest network G^k to the finest network G^0 and eventually derive the node embeddings on the original network G. Next, we focus on $\mathbf{Z}^i = RM(G^i, \mathbf{Z}^{i+1})$.

First, we inherit coarse-grained embeddings $\mathbf{Z}^{i+1}$ and fuse the network structure and node attributes of G^i to initialize the embedding $\mathbf{Z}^i$ of finer-grained network G^i as Equation (4).

$$\mathbf{Z}^i = PCA(Assign(\mathbf{Z}^{i+1}, G^i) \oplus \mathbf{X}^i), \tag{4}$$

where $Assign(\cdot)$ assigns $\mathbf{Z}^{i+1} \in \mathbb{R}^{|V^{i+1}| \times d}$ to $\mathbf{Z}^i \in \mathbb{R}^{|V^i| \times d}$ as initial embeddings of G^i. Specifically, if $v_p^i \in V_j^i$ and $v_q^i \in V_j^i$, $\mathbf{z}_p^i = \mathbf{z}_q^i = \mathbf{z}_j^{i+1}$, where v_j^{i+1} is granulated as a super-node by a subset V_j^i, and $\mathbf{z}_j^{i+1}$ is the embedding of node v_j^{i+1}. $\oplus$ means concatenation operator, and $PCA(\cdot)$ is used to reduce the dimensionality of the embedding from $(d+l)$ to d.

Second, we optimize the embedding matrix $\mathbf{Z}^i \in \mathbb{R}^{|V^i| \times d}$ inspired by literature [22] as Equation (5).

$$\mathbf{Z}^i = H(\mathbf{Z}^i, \mathbf{M}^i), \tag{5}$$

where $\mathbf{M}^i \in \mathbb{R}^{|V^i| \times |V^i|}$ is the adjacency matrix of the network G^i. $H(\cdot)$ a simple layer-wise linear GCN model that makes calculation faster. And we can stack multiple $H(\cdot)$ to achieve a model of higher capacity.

The j^{th} layer of GCN is updated by Equation (6).

$$H^j(\mathbf{Z}^i, \mathbf{M}^i) = \sigma(\widetilde{\mathbf{D}^i}^{-1/2} \widetilde{\mathbf{M}^i} \widetilde{\mathbf{D}^i}^{-1/2} H^{j-1}(\mathbf{Z}^i, \mathbf{M}^i)\Delta^j), \tag{6}$$

where $\sigma(\cdot)$ is an activation function, $\Delta^j \in \mathbb{R}^{d \times d}$ is a layer-specific trainable weight matrix. $\widetilde{\mathbf{M}^i} = \mathbf{M}^i + \lambda \mathbf{D}^i$, $\mathbf{D}^i \in \mathbb{R}^{|V^i| \times |V^i|}$ is a diagonal matrix. $\mathbf{D}^i(p,p) = \sum_q \mathbf{M}^i(p,q)$. $\lambda \in [0,1]$ is a hyper-parameter for controlling the weight of self-loop. $\widetilde{\mathbf{D}^i}(p,p) = \sum_q \widetilde{\mathbf{M}^i}(p,q)$.

In our model, the learning of the refinement module is essentially learning Δ^j for each $j \in [1,s]$, s is the number of hidden layers. We learn Δ^j only once at the coarsest granularity k and assign it to the refinement module at other granularities. Inspired by MILE [16], here we define a loss function to learn Δ^j.

$$loss = \frac{1}{|V^k|} \| (\mathbf{Z}^k - H^s(\mathbf{Z}^k, \mathbf{M}^k)) \|^2, \tag{7}$$

we adopt AdamOptimizer to learn Δ^j.

Based on the above calculation, we can obtain the node representation of each layer of the hierarchical attributed network. For the original network G, we concatenate $\mathbf{Z}^0$ and its node attributes to compensate for the loss of node attributes information. The embedding $\mathbf{Z}$ is calculated as Equation (8), where $PCA(\cdot)$ is used to reduce the dimensionality of the embedding from $(d+l)$ to d.

$$\mathbf{Z} = PCA(\mathbf{Z}^0 \oplus \mathbf{X}^0). \tag{8}$$

The proposed framework is summarized in Algorithm 1. We first use a fast granulation module in lines 2-7 to reduce the network size gradually and build a hierarchical attribute network. Then, in line 8, we learn the representation of the nodes in the coarsest network by any unsupervised network representation learning method. In the remainder of Algorithm 1, we gradually refine the node representations of the network from coarse to fine to obtain the final node representations of the original network.

5 EXPERIMENTS

We evaluate the effectiveness of the proposed framework HANE on two benchmark applications, node classification and link prediction, which are also commonly used tasks for many existing network embedding methods evaluations. In addition, we also analyze the efficiency and scalability, flexibility of the NE module, and different granulation layers of HANE. Finally, we discuss the significant test of HANE.

Algorithm 1 HANE(G, k)

Input: An attributed network $G = (V, E, \mathbf{X})$, number of granularites k

Output: Embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$ for all nodes in V.

$G^0 \leftarrow G$
for $i = 0, 1, 2, \cdots, k-1$ **do** //Granulation Module
  $V^{i+1} = V^i / R_{node} = \{V_1^i, V_2^i, \cdots\}$ //Nodes Granulation
  $E^{i+1} \leftarrow$ According to Equation (1) //Edges Granulation
  $\mathbf{X}^{i+1} \leftarrow \mathbf{x}_j^{i+1} = \frac{1}{|V_j^i|} \sum_{p=1}^{|V_j^i|} \mathbf{x}_p^i, j = 1, 2 \cdots |V^{i+1}|$ //Attributes Granulation
  $G^{i+1} = (V^{i+1}, E^{i+1}, \mathbf{X}^{i+1})$
end for
$\mathbf{Z}^k = PCA(\alpha \cdot f_{G^k}(V^k) \oplus (1-\alpha)\mathbf{X}^k)$, //NE on the coarsest attributed network
for $i = k-1, \cdots, 1, 0$ **do** //Refinement Module
  $\mathbf{Z}^i = PCA(Assign(\mathbf{Z}^{i+1}, G^i) \oplus \mathbf{X}^i)$ //initialize the embedding $\mathbf{Z}^i$
  $\mathbf{Z}^i = H(\mathbf{Z}^i, \mathbf{M}^i)$
end for
$\mathbf{Z} \leftarrow PCA(\mathbf{Z}^0 \oplus \mathbf{X}^0)$
Return $\mathbf{Z}$

TABLE 1
The statistics of datasets

Datasets	#nodes	#edges	#attributes	#labels
Cora	2,708	5,278	1,433	7
Citeseer	3,312	4,660	3,703	6
DBLP	13,404	39,861	8,447	4
PubMed	19,717	44,338	500	3
Yelp	716,847	6,977,410	300	100
Amazon	1,598,960	132,169,734	200	107

5.1 Datasets

We employ four widely-used datasets for demonstrating the effectiveness, efficiency, scalability and flexibility of HANE.

- **Cora** [54] is a citation network containing 2708 machine learning papers which are classified into seven research fields. There are 5278 edges between all these papers which indicating their citation relations. Each paper is associated with the title and abstract content. The attribute of each node is the bag of words representation of the corresponding paper.
- **Citeseer** [54] is also a citation network dataset. It contains 3312 research papers from six research fields, and there are 4660 edges between them. Each paper is associated with the title and abstract content. The attribute of each node is the bag of words representation of the corresponding paper.
- **DBLP** [55] is a citation network dataset, which consists of 13,404 nodes and 39,861 edges. Each paper is associated with the title. We learn the TF-IDF weighted word frequencies of each paper as nodes attributes.
- **PubMed** [54] is a set of articles (i.e., nodes) related to diabetes from the PubMed database, and edges here represent the citation relationship. The node attributes are TF-IDF weighted word frequencies, and node labels are the types of diabetes addressed in the articles.
- **Yelp** [27] is a social network of friends on Yelp. The node represents an active user. The edge indicates whether there is a friend relationship between the nodes. The node feature contains information of all the reviews by the users. The label of one node represents the types of business that the user has been to.
- **Amazon** [27] is e-commerce network. The node represents one product. The edge represents purchase relationship. The node feature contains information of all the reviews on that product. The label of one node represents the categories of the product.

5.2 Baselines

We categorize the baselines into the following four groups.

Single-granularity Structure-only Network Embedding Methods This group of baselines leverage network structure information only and ignore the node attributes.

- **DeepWalk** [1] and **node2vec** [2] use truncated random walks to generate node sequences, then they are fed into skip-gram model to learn the latent node representations.
- **LINE** [3] exploits the network structure's first-order proximity and second-order proximity.
- **GraRep** [28] considers different powers of the adjacency matrix to preserve higher-order graph proximity for graph embedding.
- **NodeSketch** [34] preserves high-order node proximity via recursive sketching which generates node embeddings in Hamming space.

Single-granularity Attributed Network Embedding Methods This kind of compared methods seek for low-dimensional vector representations for nodes in a network, such that original network topological structure and node attribute proximity can be preserved.

- **STNE** [8] adopts a content-to-node translation model to preserve content and structure properties.
- **CAN** [9] proposes a variational auto-encoder that embeds each node and attribute with means and variances of Gaussian distributions.

Hierarchical Structure-only Network Embedding Methods This kind of compared methods adopt hierarchical frameworks to preserve the network local or global structure in a multi-granularity way without considering node attributes.

- **HARP** [17] proposes a hierarchical paradigm for network embedding based on iterative learning methods (e.g., DeepWalk and node2Vec). HARP focuses on improving the quality of embeddings by using the learned embeddings from the previous level as the initialized embeddings for the next level.
- **MILE** [16] repeatedly coarsens the network into smaller ones using a hybrid matching technique to maintain the backbone structure of the network. It then applies existing embedding methods on the coarsest and refines the embeddings to the original graph through a novel graph convolution neural network that it learns.

Hierarchical Attributed Network Embedding. This kind of compared methods seek for low-dimensional vector

representations for nodes in a network in a hierarchical way, such that original network preserve the network local or global structure while considering node attributes.

- **GraphZoom** [19] GraphZoom encodes graph structure and node attribute in a single graph and exploiting spectral coarsening and refinement methods to remove high frequency noise from the graph.

5.3 Evaluation Metrics

To evaluate the performance of the network representation methods on node classification and link prediction task, we use $Micro\_F1$, $Macro\_F1$, area under curve (AUC) and average precision (AP) as evaluation metrics.

For nodes classification, we use $Micro\_F1$ and $Macro\_F1$ as evaluation metrics, they are defined as follows.

$\boldsymbol{Micro\_F1}$: It does not need to distinguish categories, and directly uses the $Precision$ and $Recall$ of the overall sample to calculate the $F1$ score.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}, \tag{9}$$

where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. TP, FP, FN, TN denote true positive, false positive, false negative and true negative, respectively.

$\boldsymbol{Macro\_F1}$: It needs to be calculated the average $F1$ score of all labels. It is defined as Equation (10).

$$Macro\_F1 = \frac{\sum_{c \in \xi} F1(c)}{|\xi|}, \tag{10}$$

where $F1(c)$ is the $F1$ score for label c.

We employ AUC and AP scores to evaluate the performance of link prediction.

AUC: It is the area under the receiveroperating characteristic (ROC) curve, between 0.1 and 1. As a numerical value, AUC can directly evaluate the quality of the classifier. The ROC curve is a performance indicator of the classifier. The X axis is false postive rate $FPR = \frac{FP}{TN+FP}$, and the Y axis true postive rate $TPR = \frac{TP}{TP+FN}$.

AP: It is the graphic area enclosed by the $precision$-$recall$ (PR) curve and the X axis. The PR curve reflects the trade-off between accuracy of the classifier's recognition of positive examples and the coverage ability of positive examples. The X axis is $Recall$, the Y axis is $Precision$.

5.4 Parameter Settings

Running Environment. The experiments in this paper are conducted on the Inspur NF8460M4 server with 4 Intel E7-4809CPU @ 2.1GHz, 1T of RAM and 12T Hard Disk. The codes of our proposed method are implemented with Linux in python 2.7.

Parameter Configuration. Our all embeddings dimension d is set to 128. We perform PCA dimensionality reduction based on the sklearn.decomposition.PCA class and use the sklearn.svm.LinearSVC method for the node classification task.

Configuration in Granulation Module (GM): We use the Louvain algorithm [51] to detect communities. The Louvain algorithm automatically obtains the number of communities. We initialize the weight of each edge as one and obtain the weight of the super edge by summing. The mini-batch k-means algorithm [52] is used for clustering. The number of clusters is set as the number of node labels. We use the sklearn.cluster.MiniBatchKMeans for clustering based on nodes attributes. The number of granularities k is set to 1, 2 and 3, respectively, in our experiments. Our results are the average of the results of five experiments, and the results are true and valid.

Configuration in Network Embedding module (NE): In our experiments, we use DeepWalk [1] for network embedding at the coarsest granularity. Finally, to prove the flexibility of the NE module, we use GraRep, STNE and CAN for network embedding at the coarsest granularity. Other methods have similar results, which are not shown in this paper because of space limitations. The number of walks for each node is set to 10, and the length of walks is set to 80. The window size is set to 10 for generating node contents. For the operator $\oplus$, we set parameter $\alpha = 0.5$ to fuse the network topological structure and node attributes.

Configuration in Refinement Module (RM): Our model parameters Δ^j are updated and optimized by stochastic gradient descent with AdamOptimizer. For the graph convolution network model, the self-loop weight λ is set to 0.05, the number of hidden layers s is 2 and $tanh(\cdot)$ is used as the activation function. The learning rate is set to 0.001 for Cora, Citeseer, and DBLP and 0.0001 for PubMed. The number of training $epochs$ is 200 for all datasets.

Code Details. The codes of our proposed method on the Inspur NF8460M4 server, together with our public datasets, are available. In our experiments, version 1.10 of Tensorflow is used. Version 2.0 of networkx and version 0.19.0 of scikit-learn are used. We will release the codes and datasets on Github.

5.5 Node Classification

Node classification is a traditional task commonly used to evaluate the quality of the learned node embeddings. Similar to previous studies [1] [2], we employ $Micro\_F1$ and $Macro\_F1$ as metrics to measure the performance of node classification. After the node embeddings are learned, we randomly sample 10%∼90% labeled nodes to train an SVM classifier and the rest nodes are used for testing. Each experiment is independently implemented for five times. We report performances on both average $Micro\_F1$(Mi_F1) and average $Macro\_F1$(Ma_F1).

To show the effectiveness of our proposed HANE method on node classification task, the results of $Micro\_F1$(Mi_F1) and $Macro\_F1$(Ma_F1) in each datasets and each ratio display in Tables 2, 3, 4, 5. The best performances are marked in boldface. For HANE, MILE and GraphZoom the number of granularities k is set to 1, 2, 3 and DeepWalk is selected as the network embedding method for the coarsest network in this section. More discussions on the network embedding method for HANE are given in Section 5.8. From these results shown in Tables 2, 3, 4, 5, we have the following observations and analysis.

First and foremost, HANE consistently generates better results in Micro_F1 and Macro_F1 scores than all the competitors for all settings across four datasets. HANE achieves

TABLE 2
Node classification results on Cora dataset

Algorithm	10%		20%		30%		40%		50%		60%		70%		80%		90%	
	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1
DeepWalk	76.6	75.8	79.9	79.0	80.8	80.1	81.7	81.01	83.4	82.9	83.7	83.1	84.5	84.2	82.3	82.0	81.6	81.7
LINE	67.6	65.7	72.9	71.3	75.5	74.8	76.1	75.1	77.4	76.3	75.6	74.2	77.4	76.3	76.9	76.2	75.6	74.4
Node2vec	76.6	75.2	79.2	78.1	79.9	79.5	79.0	78.4	80.6	80.1	80.3	79.7	81.8	81.5	81.4	80.5	81.5	81.1
GraRep	74.4	72.5	76.6	75.3	77.7	75.9	77.1	75.7	77.2	75.1	78.0	76.3	80.7	79.0	78.8	77.7	77.1	76.9
NodeSketch	76.2	74.7	79.2	78.1	81.2	80.0	82.1	80.8	83.0	81.9	83.8	82.4	85.0	83.9	84.4	83.2	84.7	83.3
STNE	81.1	79.5	83.3	82.3	83.9	82.7	83.9	82.7	85.2	84.1	86.4	85.7	86.7	86.1	87.3	86.9	88.6	**88.9**
CAN	82.2	80.5	84.1	82.9	85.1	83.8	85.8	84.6	86.5	85.3	86.9	85.9	86.7	85.5	86.8	85.8	88.2	87.2
HARP	75.2	73.8	77.2	76.2	79.1	78.0	79.8	78.8	80.3	79.2	80.6	79.7	81.2	80.5	81.7	81.0	82.4	82.1
MILE(k=1)	76.1	75.0	79.5	78.4	80.6	79.6	81.7	80.8	82.1	80.1	82.4	81.7	82.9	82.4	82.6	81.6	83.7	82.8
MILE(k=2)	76.3	74.7	79.8	78.2	81.2	79.8	82.4	81.1	82.3	81.0	83.5	82.4	83.2	81.9	83.0	81.8	84.2	83.1
MILE(k=3)	75.1	73.7	79.0	77.8	80.1	79.0	80.5	79.3	81.6	80.5	82.0	80.9	82.2	81.1	82.7	81.7	82.1	81.5
GraphZoom(k=1)	77.6	76.0	80.8	79.3	83.6	82.4	83.8	82.7	84.9	83.9	84.9	83.8	85.6	84.7	85.6	84.6	85.5	84.6
GraphZoom(k=2)	77.6	76.0	80.4	79.2	82.2	81.1	82.7	81.7	82.9	82.1	83.2	82.5	84.1	83.2	83.0	81.9	84.3	82.9
GraphZoom(k=3)	77.4	75.8	80.5	79.1	82.1	81.0	82.5	81.5	82.7	82.0	83.4	82.5	83.8	82.8	82.9	81.8	83.8	82.3
HANE(k=1)	81.8	80.7	84.5	83.3	85.1	83.9	86.0	84.8	86.1	85.0	86.9	85.6	87.5	86.5	87.6	86.5	88.2	86.7
HANE(k=2)	82.6	81.3	84.6	83.3	**86.0**	**84.7**	85.9	84.5	**87.0**	**85.8**	87.1	85.8	**87.9**	86.5	87.8	86.7	87.8	86.4
HANE(k=3)	**82.7**	**81.3**	**84.7**	**83.4**	85.7	84.2	**86.5**	**85.2**	86.8	85.6	**88.1**	**87.0**	87.8	**86.6**	**88.0**	**86.9**	**88.6**	87.2

TABLE 3
Node classification results on Citeseer dataset

Algorithm	10%		20%		30%		40%		50%		60%		70%		80%		90%	
	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1
DeepWalk	52.9	48.5	53.9	49.3	56.5	51.5	57.6	53.2	57.2	53.0	57.9	53.4	58.3	53.1	59.0	54.0	55.8	51.8
LINE	44.0	40.4	45.9	42.2	47.5	43.2	47.8	43.2	48.8	44.1	48.7	44.3	49.4	45.0	49.5	44.8	48.8	43.8
Node2vec	55.4	50.6	56.6	51.6	57.9	51.6	58.8	53.1	59.1	53.6	60.3	54.5	60.8	55.3	60.3	55.6	58.7	54.1
GraRep	53.6	47.0	54.6	49.0	54.3	47.7	55.3	48.4	55.4	48.5	55.6	48.7	55.7	48.9	55.5	48.8	55.1	49.6
NodeSketch	45.6	41.0	55.1	51.6	60.5	57.0	64.5	61.0	67.3	63.5	69.6	66.1	71.6	67.7	73.0	68.7	75.5	71.7
STNE	69.2	62.0	70.3	63.1	71.7	64.6	72.3	65.8	72.6	67.1	71.8	65.7	70.7	64.6	69.7	63.9	69.6	63.3
CAN	66.6	62.3	69.3	64.9	71.6	66.9	72.2	67.2	73.0	68.3	73.6	68.7	73.6	68.6	74.2	68.8	73.9	68.1
HARP	53.7	49.2	56.2	51.2	57.8	52.9	58.7	53.7	59.4	54.3	60.0	54.7	60.2	54.9	60.4	54.9	60.4	54.0
MILE(k=1)	52.9	48.2	55.5	50.2	56.6	50.8	57.2	51.4	58.2	52.1	58.1	52.2	58.3	52.3	58.8	52.6	58.0	50.9
MILE(k=2)	53.2	48.0	55.6	50.1	56.6	50.1	58.1	51.0	57.7	50.4	57.8	50.5	58.1	50.4	59.3	50.9	58.3	50.7
MILE(k=3)	52.4	47.2	54.3	58.7	55.7	49.6	55.5	49.2	55.7	59.2	55.6	48.9	55.8	49.0	56.5	50.1	56.0	49.0
GraphZoom(k=1)	59.5	55.3	64.3	60.2	66.9	62.4	67.7	63.5	68.6	63.6	69.7	64.7	69.7	64.5	69.8	64.6	69.4	63.9
GraphZoom(k=2)	60.6	55.9	63.4	59.2	65.9	61.2	67.7	63.3	68.4	63.3	68.9	64.0	69.5	64.5	69.4	64.4	70.4	65.6
GraphZoom(k=3)	60.7	56.0	63.9	59.4	66.2	61.1	67.9	63.2	68.7	63.6	68.9	63.8	69.6	64.7	69.0	63.8	69.4	64.1
HANE(k=1)	68.8	64.0	71.0	66.8	72.5	68.7	73.5	69.2	74.6	70.9	**75.1**	71.1	74.8	70.8	**75.7**	71.8	75.3	71.2
HANE(k=2)	**69.7**	**65.5**	71.7	67.4	73.4	69.1	73.8	69.8	**74.8**	70.7	74.9	70.5	75.1	71.2	75.4	70.9	76.4	71.4
HANE(k=3)	69.7	64.9	**71.9**	**67.5**	**73.4**	**69.4**	**74.4**	**70.3**	74.4	**70.8**	74.9	**70.8**	**75.8**	**71.7**	75.6	**71.9**	**76.9**	**72.8**

state-of-the-art performance with 0.73% average relative lift on Cora dataset, 2.18% on Citeseer dataset, 1.37% on DBLP dataset and 3.48% on PubMed dataset in $Micro\_F1$ scores, and with 0.43% on average relative lift Cora dataset, 3.96% on Citeseer dataset, 1.60% on DBLP dataset and 3.73% on PubMed dataset in $Macro\_F1$ scores, compared with best results from previous state-of-the-art algorithms. The performance of HANE is followed by that of the single-granularity attributed network embedding methods STNE and CAN, then followed by that of GraphZoom. Single-granularity structure-only network embedding methods which use only the network structure perform the worst. The attributed network embedding baselines that fuse the network topological structure and node attributes information generally perform better than the structure-only baselines that use network topological structure information without node attributes. This further proves the usefulness of the attributes for network embedding.

Second, it is worth noting that hierarchical representation learning methods outperform most of single-granularity approaches. Among the seven structure-only network embedding method, hierarchical structure-only network embedding methods get a better representation in most cases. Building hierarchical attributed networks and properly modeling can lead to better representation learning and benefit downstream applications. This possible hierarchical attributed networks are obtained in HANE, which strengthens the close interaction between the network structure and attribute information. Although we have the same or slightly worse comparison with the STNE method in the case of a higher test set ratio on Cora dataset with the 90% training ratio in the last column of Table 2, from Tabel 7, we find that the proposed HANE method is much faster than STNE. When STNE is selected as the base network embedding method, as shown in Fig.4, our method HANE(STNE) is superior to STNE in speed and performance. Furthermore, a hierarchical attributed network embedding method GraphZoom is behind single-granularity attributed method. However, GraphZoom is much faster than single-granularity attributed method in speed. GraphZoom fuses the attributes and the topology structure only once and the attribute information of supernodes is very difficult to get, so that the attribute information was not fully utilized in the model.

These observations demonstrate the effectiveness of the proposed HANE.

5.6 Link Prediction

Next, we evaluate the quality of network embedding for link prediction, which is a typical task in network analysis. We perform a link prediction task on four datasets.

In this task, we use the same setting as in [34]. Specifically, we randomly sample 20% of the edges from each network and we randomly sample an equal number of node pairs

TABLE 4
Node classification results on DBLP dataset

Algorithm	10%		20%		30%		40%		50%		60%		70%		80%		90%	
	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1
DeepWalk	80.0	72.8	82.1	76.0	82.7	76.9	83.0	77.4	83.1	77.5	82.9	77.4	83.3	77.9	82.8	77.2	83.3	77.8
LINE	68.7	57.1	70.8	59.9	71.1	60.2	71.4	60.5	72.2	61.4	71.9	61.2	72.1	61.3	72.3	61.4	72.0	61.0
Node2vec	80.1	73.1	81.5	75.3	82.2	75.8	82.6	76.6	82.6	76.7	82.7	76.6	82.8	76.8	82.9	76.9	83.0	77.3
GraRep	81.1	74.1	81.9	75.7	82.1	76.2	82.4	76.2	82.4	76.6	82.5	76.5	82.7	77.1	82.4	76.5	82.8	77.0
NodeSketch	62.5	46.1	68.6	56.9	72.1	62.1	74.6	65.4	76.4	67.9	77.7	69.6	78.9	70.7	80.1	72.0	80.1	72.8
STNE	82.1	75.9	82.3	76.3	83.0	77.1	82.9	77.1	83.1	77.4	83.5	78.1	83.1	77.9	83.2	78.2	83.5	78.4
CAN	82.1	75.4	82.7	76.4	83.2	77.3	83.3	77.5	83.4	77.5	83.4	77.6	83.8	77.9	83.7	77.9	83.6	78.0
HARP	79.6	72.9	80.9	74.4	81.4	75.4	81.6	75.7	81.9	76.0	81.7	75.9	81.7	75.8	81.7	75.8	81.9	76.2
MILE(k=1)	81.8	75.6	82.5	76.6	83.1	77.4	83.0	77.2	83.2	77.5	83.4	77.9	83.7	78.2	83.2	77.8	82.9	77.2
MILE(k=2)	81.3	74.7	82.6	76.7	83.0	77.2	83.1	77.4	83.1	77.5	83.1	77.6	83.4	77.8	83.1	77.5	82.9	77.6
MILE(k=3)	81.1	74.5	82.1	76.0	82.7	76.8	82.7	77.0	82.8	77.0	82.9	77.4	83.1	77.6	83.0	77.7	83.1	77.4
GraphZoom(k=1)	80.9	74.5	82.8	77.2	83.3	78.0	83.4	78.2	83.6	78.4	83.6	78.5	83.6	78.6	83.9	79.0	83.5	78.6
GraphZoom(k=2)	81.3	75.2	82.6	76.9	83.0	77.5	83.2	77.8	83.4	78.2	83.4	78.2	83.7	78.6	83.9	78.6	83.3	77.6
GraphZoom(k=3)	81.3	75.2	82.5	76.7	83.0	77.6	83.2	77.9	83.5	78.4	83.4	78.4	83.7	78.7	83.8	78.6	83.4	77.8
HANE(k=1)	83.0	77.0	83.8	78.0	84.2	78.4	84.3	78.7	84.6	79.2	84.3	78.9	84.4	78.6	84.5	79.0	84.4	78.6
HANE(k=2)	83.4	77.6	**84.1**	**78.7**	**84.3**	**78.9**	84.5	79.2	84.6	79.3	84.8	79.5	84.6	79.6	84.6	**79.6**	**84.8**	**79.6**
HANE(k=3)	**83.4**	**77.8**	84.0	78.6	84.3	78.9	**84.6**	**79.2**	**84.7**	**79.4**	**84.9**	**79.9**	**84.8**	**79.7**	**84.7**	79.5	84.6	79.4

TABLE 5
Node classification results on PubMed dataset

Algorithm	10%		20%		30%		40%		50%		60%		70%		80%		90%	
	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1	Mi_F1	Ma_F1
DeepWalk	79.0	77.4	80.5	79.1	81.2	79.6	80.5	79.1	80.7	79.4	80.6	79.4	80.5	79.3	80.4	78.9	80.0	78.2
LINE	71.3	68.7	72.4	69.8	72.8	70.4	72.9	70.5	73.1	70.8	72.8	70.5	73.1	70.7	73.1	70.7	72.6	70.0
Node2vec	79.8	78.4	80.6	79.1	80.6	79.3	80.3	79.0	80.3	79.0	80.3	79.0	80.5	79.2	80.4	78.9	80.8	79.2
GraRep	79.0	77.5	79.9	78.5	79.9	78.6	79.6	78.3	79.6	78.3	79.3	78.1	79.3	78.0	79.4	77.9	79.1	77.4
NodeSketch	72.5	69.7	77.9	76.1	80.0	78.5	81.2	79.8	82.3	81.0	82.5	81.3	83.1	82.0	83.3	82.1	83.8	82.7
STNE	83.6	82.8	84.3	83.4	84.6	83.9	84.5	83.8	84.6	83.9	84.4	83.7	84.4	83.7	84.2	83.4	83.8	83.1
CAN	83.8	83.3	84.3	83.7	84.5	84.0	84.7	84.2	84.9	84.4	84.9	84.8	84.2	84.4	85.2	84.7	84.9	84.3
HARP	79.3	77.9	80.3	78.9	80.6	79.3	80.8	79.4	80.9	79.6	80.9	79.6	80.7	79.5	81.1	79.8	81.2	79.9
MILE(k=1)	80.5	79.2	81.2	79.9	81.6	80.3	81.5	80.2	81.7	80.4	81.8	80.5	81.5	80.2	81.9	80.6	82.2	81.1
MILE(k=2)	80.7	79.5	81.5	80.3	81.6	80.5	81.8	80.7	81.8	80.8	81.9	80.7	82.2	81.1	81.9	80.8	81.7	80.5
MILE(k=3)	80.2	78.9	80.9	79.7	81.3	80.1	81.4	80.2	81.5	80.3	81.3	80.1	81.6	80.4	81.5	80.3	82.7	81.5
GraphZoom(k=1)	79.7	78.3	80.5	79.3	80.9	79.6	81.0	79.8	81.2	79.9	81.2	79.9	81.3	80.0	81.1	79.9	81.1	79.7
GraphZoom(k=2)	79.7	78.4	80.6	79.3	80.9	79.6	80.8	79.6	80.7	79.4	80.8	79.6	81.0	79.8	81.0	79.7	81.2	79.9
GraphZoom(k=3)	79.6	78.3	80.6	79.2	80.7	79.4	80.8	79.6	80.7	79.4	80.8	79.6	81.0	79.7	80.9	79.6	81.0	79.6
HANE(k=1)	86.2	85.9	87.1	86.9	87.2	87.0	87.4	87.2	87.5	87.3	87.6	87.4	87.6	87.4	87.7	87.4	87.8	87.5
HANE(k=2	86.5	86.3	87.1	86.9	87.4	87.2	**87.6**	87.3	87.5	87.2	87.7	87.5	87.7	87.4	87.8	87.6	87.8	87.6
HANE(k=3)	**86.6**	**86.4**	**87.2**	**87.0**	**87.6**	**87.3**	87.5	**87.3**	**87.8**	**87.6**	**87.8**	**87.6**	**87.9**	**87.6**	**87.9**	**87.7**	**87.8**	**87.6**

TABLE 6
Performance of link prediction

Algorithms	Cora		Citeseer		DBLP		PubMed	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk	90.7	92.1	88.3	88.9	94.8	95.3	94.1	93.7
LINE	83.3	86.4	81.8	83.5	85.3	87.1	73.8	73.9
node2vec	89.3	89.8	85.0	83.0	93.9	93.7	93.7	93.0
GraRep	88.5	90.3	83.4	83.7	93.8	93.3	89.8	87.8
NodeSketch	/	/	/	/	/	/	/	/
STNE	/	/	/	/	/	/	/	/
CAN	82.6	84.6	97.2	96.6	90.8	90.3	74.4	73.7
HARP	90.1	90.7	88.4	88.7	94.0	94.5	93.1	92.6
MILE($k=1$)	89.9	91.3	89.2	90.3	95.3	95.9	94.5	94.4
MILE($k=2$)	90.0	91.5	90.2	91.1	94.7	95.5	93.8	93.9
MILE($k=3$)	88.7	90.3	89.9	90.8	93.3	94.4	92.4	92.8
GraphZoom($k=1$)	91.5	91.8	88.8	89.3	95.8	95.7	94.9	94.1
GraphZoom($k=2$)	91.4	91.2	87.9	88.0	95.7	95.6	94.7	93.8
GraphZoom($k=3$)	90.9	91.0	87.9	87.9	95.6	95.5	94.7	93.9
HANE($k=1$)	93.6	93.8	95.7	95.4	96.4	96.2	94.9	93.7
HANE($k=2$)	**94.8**	**94.6**	**97.8**	**97.4**	**97.1**	**96.9**	**95.7**	**94.7**
HANE($k=3$)	92.9	93.3	95.4	95.2	96.7	96.5	95.6	94.7

without edge connections from the network to generate negative examples, as test data. The rest of the network is used for training. We predict the link between nodes based on the cosine similarity of their embeddings. We employ AUC and AP scores to evaluate the performance of link prediction, which are commonly used in related literature [34] [2] [9]. Higher values of AUC and AP indicate the better performance of the network embedding method. For HANE and MILE, the number of granularities k is set to 1, 2, 3. Without loss of generality, DeepWalk is selected to learn the embeddings of the coarsest network for HANE, HARP and MILE. Each experiment is independently implemented for 10 times and the average performances on the test set are reported in Table 6. Among them, the parameters of NodeSketch have too much influence on the results. We did not find a suitable parameter to make the results of the link prediction normal. The results of STNE is poor here. Maybe STNE is not suitable for link prediction tasks. For Table 6, the best performances are marked in boldface.

Empirical results show, the proposed HANE consistently outperforms all the other baselines on all four datasets. HANE achieves state-of-the-art performance with 3.61%, 0.62%, 1.36% and 0.84% relative lift in AUC scores on Cora, Citeseer, DBLP, and PubMeb datasets, and with 3.05%, 0.83%, 1.25% and 0.64% relative lift in AP scores across these four datasets, compared with best results from previous state-of-the-art algorithms. The proposed HANE consistently outperforms the competitors which further verifies the effectiveness of the proposed method.

From Tabel 6 shows, we can find that the performance of hierarchical network representation learning outperforms single-granularity approaches. It demonstrates that the hierarchical structure helps to capture the potential relationships between nodes in the network.

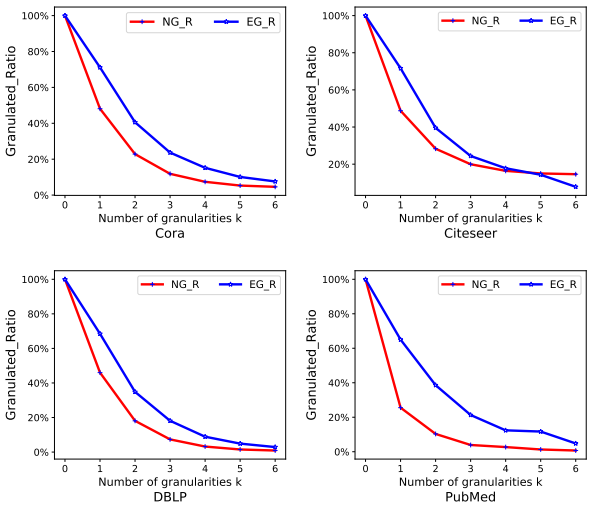


Fig. 3. The $Granulated\_Ratio$ of hierarchical network. X axis is the number of granularities k, we set $k = 0, 1, 2, 3$. Y axis is the $Granulated\_Ratio$ of hierarchical network. The red line represents the $Granulated\_Ratio$ of the nodes(NG_R) and the blue line denotes the $Granulated\_Ratio$ of the edges(EG_R).

TABLE 7
Time comparison for network representation learning (in Seconds)

Algorithm	Cora	Citeseer	DBLP	PubMed	avgSpeedup
DeepWalk	131.94 (4.94×)	125.32 (2.81×)	1247.66 (21.94×)	841.76 (9.56×)	9.81×
LINE	403.27 (15.09×)	395.11 (8.85×)	2,207.13 (38.83×)	3,860.38 (43.86×)	50.92×
node2vec	56.95 (2.13×)	57.54 (1.29×)	672.58 (11.83×)	455.80 (5.18×)	5.12×
GraRep	233.82 (8.75×)	295.53 (6.62×)	9,764.31 (171.73×)	37,904.82 (430.64×)	154.44×
STNE	2,880.50 (107.76×)	1,380.00 (30.91×)	$> 4days$ (/)	231,900.00 (2634.63×)	1016.88×
CAN	345.98 (12.94×)	782.50 (17.53×)	3,495.28 (61.47×)	22,001.38 (249.96×)	85.48×
HARP	36.28 (1.36×)	**32.18** (0.72×)	173.17 (3.05×)	281.42 (3.20×)	2.08×
MILE($k = 1$)	117.58 (4.40×)	115.03 (2.58×)	375.71 (6.61×)	575.86 (6.54×)	5.03×
MILE($k = 2$)	65.85 (2.46×)	68.45 (1.53×)	193.91 (3.41×)	360.83 (4.10×)	5.62×
MILE($k = 3$)	35.25 (1.32×)	41.71 (0.93×)	118.23 (2.08×)	222.62 (2.53×)	1.72×
GraphZoom($k = 1$)	76.82 (2.87×)	110.16 (2.47×)	490.06 (8.62×)	774.85 (8.80×)	5.60×
GraphZoom($k = 2$)	30.89 (1.16×)	49.96 (1.12×)	197.12 (3.47×)	272.40 (3.09×)	2.21×
GraphZoom($k = 3$)	29.01 (1.09×)	47.07 (1.05×)	187.71 (3.30×)	263.56 (2.99×)	2.10×
HANE($k = 1$)	73.01 (2.73×)	98.65 (2.21×)	300.49 (5.28×)	314.60 (3.57×)	3.25×
HANE($k = 2$)	50.54 (1.89×)	63.40 (1.42×)	132.50 (2.33×)	150.05 (1.70×)	1.84×
HANE($k = 3$)	**26.73**	44.64	**56.86**	**88.02**	1

TABLE 8
Time comparison with three base network embedding methods (in Seconds)

Datasets	Cora	Citeseer	DBLP	PubMed
GraRep	233.82(3.42×)	295.53(4.48×)	9,764.31(78.83×)	37,904.82(278.10×)
HANE(GR, k=1)	158.60(2.32×)	160.09(2.43×)	276.05(2.23×)	281.11(2.06×)
HANE(GR, k=2)	113.30(1.66×)	111.83(1.70×)	193.59 (1.56×)	222.20(1.63×)
HANE(GR, k=3)	**68.27**	**65.95**	**123.87**	**136.30**
STNE	2,880.50(29.3×)	1,380.00(11.84×)	$> 4days$ (/)	231,900.00(1629.20×)
HANE(STNE, k=1)	168.39(1.71×)	186.17 (1.60×)	589.46(3.73×)	626.39 (4.40×)
HANE(STNE, k=2)	133.27(1.36×)	151.45 (1.30×)	262.52(1.66×)	200.06(1.41×)
HANE(STNE, k=3)	**98.22**	**116.53**	**158.04**	**142.31**
CAN	345.98 (21.10×)	782.50(24.73×)	3,495.28(61.23×)	22,001.38(252.34×)
HANE(CAN, k=1)	65.86(4.02×)	89.43 (2.83×)	621.17(10.88×)	704.98 (8.09×)
HANE(CAN, k=2)	28.63(1.75×)	45.52 (1.44×)	135.50(2.37×)	202.06 (2.32×)
HANE(CAN, k=3)	**16.40**	**31.64**	**57.08**	**87.19**

5.7 Efficiency and Scalability

We compare the efficiency and demonstrate the scalability of the proposed HANE. Table 7 reports the network representations learning time of HANE at different granularities and nine baselines divided into four groups. The best performances are marked in boldface. The representation learning time of NodeSketch is not given for comparison in this paper because the running environment (Matlab 2017b on MacOS) of the codes from the corresponding author is different from ours. In this section, DeepWalk is selected as the network embedding method for the coarsest network.

We define $Granulated\_Ratio$ to measure the scale change of the networks at different granularity. $Granulated\_Ratio$ means the ratio of numbers of node and edge after granulation to the original numbers of node and edge.

The $Granulated\_Ratio$ of nodes (NG_R): $NG\_R = \frac{n'}{n}$, where n is the number of nodes in the original network, and n' is the number of nodes in the coarse-grained network.

The $Granulated\_Ratio$ of edges (EG_R): $EG\_R = \frac{m'}{m}$, where m is the number of nodes in the original network, and m' is the number of nodes in the coarse-grained network.

Fig. 3 shows $Granulated\_Ratio$ at different granularities on all four datasets. We have the following observations and analysis.

First, from Table 7, HANE shows its notable speed against all the competitors, especially against these single-granularity attributed network embedding competitors. The hierarchical network representation learning methods are much faster than the single-grained network representation learning methods, which further shows that the representation learning can be accelerated in a hierarchical way. HANE significantly outperforms the single-granularity structure-only network embedding baselines with 9.81×∼154.44× speedup, the single-granularity attributed network embedding baselines with 85.48×∼1016.88× speedup and the hierarchical structure-only network embedding baselines with at least 1.71× speedup across these four datasets, averagely. Our method HANE is faster than the hierarchical attributed network embedding GraphZoom with 2.10×∼5.60×, averagely.

Second, HANE achieves state-of-the-art learning time performance on all datasets except for the Citeseer dataset. It seems that HARP and MILE ($k = 3$) are slightly faster than the proposed HANE on this dataset. However, it is worth noting that the proposed HANE fuses the network structure and node attributes while HARP and MILE only consider network topological structure without considering node attributes. Attributed network embedding baselines are generally more time-consuming than structure-only network embedding baselines from Table 7.

Third, with the increase of the number of granularities k, we find that the $Granulated\_Ratio$ dramatically decreases (see Fig. 3) and the speedup increases further (see Table 7) while the quality of the embeddings is preserved reflected by achieving the best Micro_F1 and Macro_F1 (See Table 2 ∼ Table 5). In Fig. 3, we further find that the number of granularities k generally is not large. We granulate them once, the nodes scale are reduced by at least 52% on all datasets. When we granulate them three times, the nodes scale/edges scale of all datasets remain less than 20% /25%. This shows that our granulation module can better capture the community structure in the network. HANE has an efficient granulation module because the granulation process is based on the community detection for network topological structure and clustering for node attributes.

From above, we conclude that HANE is an efficient, scalable, and effective network embedding approach.

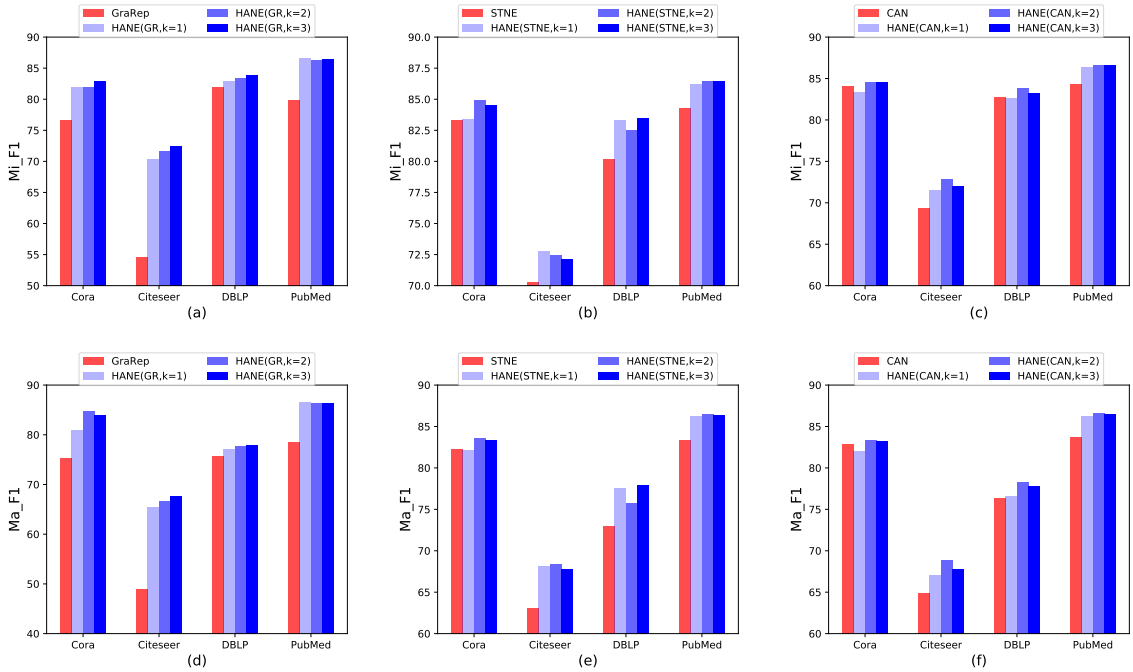


Fig. 4. The performance of node classification with different base network embedding methods for the coarsest network in HANE (i.e., NE module). Red bars indicate the results of methods GraRep, STNE, and CAN. Three different blue bars represent the results of HANE(GR), HANE(STNE), and HANE(CAN) with different granularities k=1, 2, 3, respectively. (a), (b), (c) show the comparison of $Micro\_F1$ scores for GraRep, STNE, CAN, respectively. The $Macro\_F1$ scores of them are shown in (d), (e), (f). The training ratio is set 20%, similar conclusions are on other training ratios.

5.8 Flexibility

In this section, we will show the proposed HANE is flexible that structure-only network embedding methods and attributed network embedding methods are both suitable for the NE module of HANE. Table 8 shows the embedding learning time of HANE on different datasets with three network embedding methods, GraRep, STNE, and CAN, at the coarsest granularity. The best performances are marked in boldface. Fig. 4 summarizes the performances of the proposed HANE with different network embedding methods and the original embedding methods with the 20% training ratio and similar conclusions can be drawn on other training ratios.

As shown in Fig. 4 and Table 8, we have the same observations as the above. First, HANE achieves higher Micro_F1 and Macro_F1 scores than all the competitors across all four datasets within a much shorter time. Second, HANE significantly outperforms GraRep that is a single-granularity structure-only network embedding competitor with 3.42×∼278.10× speedup, STNE and CAN which are attributed network embedding competitors with 29.30×∼1629.20× speedup and 21.10×∼252.34× speedup.

At the coarsest granularity, we can use any of the existing structure-only network embedding methods and attributed network embedding methods for the NE module of HANE that is flexible.

5.9 Different Granulation Layers

In this section, we study how the number of granularities k affect the performance of HANE. Starting from $k = 1$, we increase k until it reaches 6 or the coarsest graph contains less than 100 nodes. From Fig. 5, we find that node classification performance of HANE are not sensitive to the number of granularities k on all four datasets. This is because in the refinement module, we integrate the attribute information of each layer as the supplementary information of the topology, which can better refine the representation of the coarsest network layer by layer. Before the compression reduction rate converges, the larger k is, the shorter the running time is. This further shows that HANE can increase the speed while maintaining performance.

5.10 Large-scale Attributed Network Representation Learning

In Fig.6, we show the $Micro\_F1$ and running time of HANE, MILE and GraphZoom with k varying 1 to 3 on Yelp dataset, and comparison of HANE and MILE with k varying 1 to 4 on Amazon dataset. Yelp and Amazon are large-scale attribute networks. We use DeepWalk for network embedding at the coarsest granularity, the training ratio is set 20%, similar conclusions are on other training ratios. We have executed GraphZoom for more than four days on Amazon dataset, but no results were obtained. When increasing k, HANE achieves a higher speedup while the $Micro\_F1$ decreases slowly while effectively granulate the large-scale network without losing key information, thereby greatly reducing the embedding time, and improve the quality of embedding methods on large-scale attribute networks.

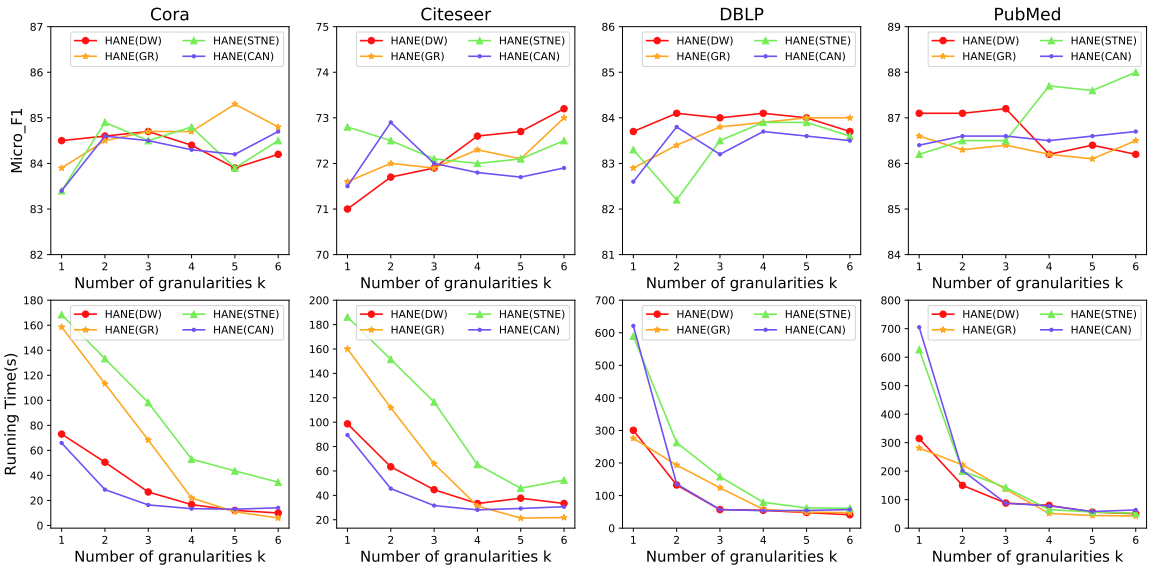


Fig. 5. The performance changes as the number of granulation layers increases. $Micro\_F1$ and running time are reported in the first and second rows, respectively.

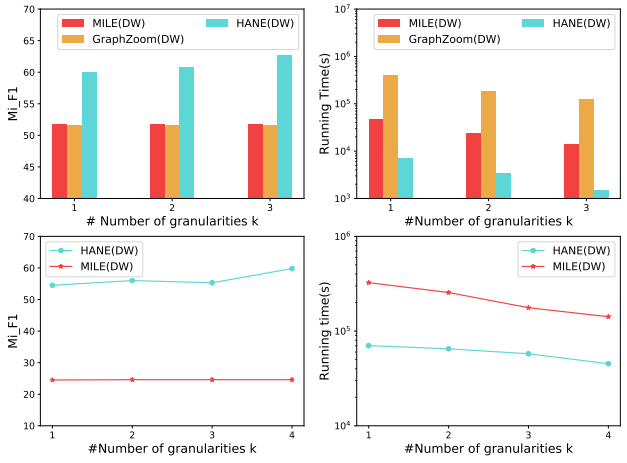


Fig. 6. Comparisons of HANE, MILE and GraphZoom on Yelp dataset and run HANE and MILE on Amazon dataset. $Micro\_F1$ and running time are reported in the first and second columns respectively. The running time is showed in seconds.

TABLE 9
$p\text{-}value$ of independent samples t-test

Datasets	Cora	Citeseer	DBLP	PubMed
DeepWalk	1.31E-06	1.91E-10	2.28E-05	8.42E-11
LINE	3.01E-08	1.13E-14	1.52E-11	2.22E-15
node2vec	1.13E-10	7.67E-11	1.42E-06	3.39E-12
GraRep	5.98E-09	3.07E-10	3.61E-11	4.95E-11
NodeSketch	5.97E-06	6.57E-03	6.6E-04	0.0002
STNE	5.4E-03	2.89E-03	7.23E-09	1.15E-08
CAN	0.0182	1.70E-05	1.47E-07	1.05E-09
HARP	1.43E-09	7.02E-14	2.54E-09	6.24E-14
MILE($k=1$)	2.54E-08	5.34E-13	3.07E-07	1.07E-13
MILE($k=2$)	3.07E-07	1.98E-12	1.96E-07	1.03E-13
MILE($k=3$)	1.08E-08	1.54E-11	1.89E-08	4.51E-11
GraphZoom($k=1$)	4.11E-05	1.14E-06	8.75E-05	1.93E-14
GraphZoom($k=2$)	2.79E-08	1.77E-07	8.41E-06	8.22E-15
GraphZoom($k=3$)	1.75E-08	8.10E-08	7.00E-06	3.11E-15
HANE($k=1$)	0.06	0.03	0.002	0.013
HANE($k=2$)	1.0	1.0	1.0	1.0
HANE($k=3$)	0.15	0.12	0.438	0.021

5.11 Significant Test

In this section, we analyze the statistical significance of method performance improvement through a significant test. First we learn the representation of nodes and randomly sample 10%∼90% labeled nodes to train a SVM classifier and the rest nodes are used for testing. Each experiment is independently implemented for five times. Then, the average performances ($Micro\_F1$) on the testing set of HANE(k=2) performs an independent samples t-test [56] with all baselines on four datasets separately, and calculates $p\text{-}value$ as shown in Tabel 9, significance level $\alpha = 0.05$. In Table 9, the last three row is an independent samples t-test for HANE(k=1, k=2, k=3), they do not differ. It can be seen that the average $p\text{-}value$=6.09$E-04 < \alpha$ of HANE(k=2) over the baselines in four datasets. When $k = 1$ and $k = 3$, there are similar conclusions. This proves that the performance improvement of HANE is statistically significant.

6 CONCLUSION AND FUTURE WORK

In this paper, we formalize the hierarchical attributed network embedding problem and propose an effective framework HANE to solve it with both network topological structure and node attributes in a hierarchical way. HANE not only incorporates these two kinds of information within one granularity but infers them across granularities. Furthermore, HANE is highly efficient for attributed network

embedding. We split the overall network embedding into three modules: granulation module GM, network embedding module NE on the coarsest network, and refinement module RM. GM mainly constructs a hierarchical attributed network from fine to coarse based on both the network topological structure and node attributes so that we can capture the relationships between different granularities and get a series of smaller attributed networks. NE module is flexible with many existing network embedding methods for the attributed network at the coarsest granularity. RM inherits and updates embeddings of the constructed hierarchical attributed network from coarse to fine so that we can quickly get the embeddings of the original network. The proposed framework HANE shows notable speedup and achieves significantly better performances compared to previous state-of-the-art methods on two tasks, including node classification and link prediction, across several datasets.

In future, we will explore the following directions:

1) We seek to investigate the extensibility of our model on dynamic network, such as learning new node representations without repeatedly training the model.

2) Another interesting direction is semi-supervised learning of our model. We will adjust our framework according to specific tasks such as node clustering and graph classification, and consider the label information of the training set to improve the quality of the node representation used for prediction.

ACKNOWLEDGMENT

This work was partially supported the National Natural Science Foundation of China (Grants #61876001, #61602003 and #61673020), National High Technology Research and Development Program (Grant #2017YFB1401903), and the Recruitment Project of Anhui University for Academic and Technology Leader.

REFERENCES

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD'14*. ACM, 2014, pp. 701–710.
[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD'16*. ACM, 2016, pp. 855–864.
[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW'15*, 2015, pp. 1067–1077.
[4] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *WSDM'18*. ACM, 2018, pp. 459–467.
[5] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *KDD'15*. ACM, 2015, pp. 1165–1174.
[6] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *TKDE*, vol. 31, no. 6, pp. 1051–1065, 2018.
[7] H. Wang, T. Xu, Q. Liu, D. Lian, E. Chen, D. Du, H. Wu, and W. Su, "Mcne: An end-to-end framework for learning multiple conditional network representations of social network," in *KDD'19*, 2019, pp. 1064–1072.
[8] J. Liu, Z. He, L. Wei, and Y. Huang, "Content to node: Self-translation network embedding," in *KDD'18*. ACM, 2018, pp. 1794–1802.
[9] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *WSDM'19*. ACM, 2019, pp. 393–401.
[10] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *TKDE*, vol. 30, no. 12, pp. 2257–2270, 2018.
[11] H. Gao and H. Huang, "Deep attributed network embedding," in *IJCAI'18*, 2018, pp. 3364–3370.
[12] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *ICDM'18*. IEEE, 2018, pp. 1476–1481.
[13] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD'17*. ACM, 2017, pp. 135–144.
[14] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *KDD'19*, 2019, pp. 1358–1368.
[15] R. A. Rossi, R. Zhou, and N. Ahmed, "Deep inductive graph representation learning," *TKDE*, pp. 1–15, 2018.
[16] J. Liang, S. Gurukar, and S. Parthasarathy, "Mile: A multi-level framework for scalable graph embedding," in *arXiv preprint arXiv:1802.09612*, 2018.
[17] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *AAAI'18*. AAAI Press, 2018, pp. 2127–2134.
[18] A. K. Bhowmick, K. Meneni, M. Danisch, J.-L. Guillaume, and B. Mitra, "Louvainne: Hierarchical louvain method for high quality and scalable network embedding," in *WSDM'20*, 2020, pp. 43–51.
[19] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng, "Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding," in *ICLR'20*, 2020.
[20] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS'18*.
[21] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *KDD'19*, 2019, pp. 723–731.
[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR'17*, 2017.
[23] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS'17*, 2017, pp. 1024–1034.
[24] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *ICLR'19*, 2019.
[25] W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, and S. Wang, "Hierarchical multi-label text classification: An attention-based recurrent network approach," in *CIKM'19*, 2019, pp. 1051–1060.
[26] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *WSDM'17*. ACM, 2017, pp. 731–739.
[27] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *ICLR'20*, 2020.
[28] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM'15*. ACM, 2015, pp. 891–900.
[29] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *KDD'16*. ACM, 2016, pp. 1105–1114.
[30] Y. Li, Y. Wang, T. Zhang, J. Zhang, and Y. Chang, "Learning network embedding with community structural information," in *IJCAI'19*, 2019, pp. 2937–2943.
[31] J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding, "Prone: fast and scalable network representation learning," in *IJCAI'19*, 2019, pp. 4278–4284.
[32] X. Shen, S. Pan, W. Liu, Y.-S. Ong, and Q.-S. Sun, "Discrete network embedding," in *IJCAI18*, 2018, pp. 3549–3555.
[33] J. Guo, L. Xu, and E. Chen, "Spine: Structural identity preserved inductive network embedding," in *IJCAI'19*, 2019, pp. 2399–2405.
[34] D. Yang, P. Rosso, B. Li, and P. Cudre-Mauroux, "Nodesketch: Highly-efficient graph embeddings via recursive sketching," in *KDD'19*, 2019, pp. 1162–1172.
[35] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *SDM'17*. SIAM, 2017, pp. 633–641.
[36] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *IJCAI'15*, 2015, pp. 2111–2117.
[37] C. Tu, H. Liu, Z. Liu, and M. Sun, "Cane: Context-aware network embedding for relation modeling," in *ACL'17*, 2017, pp. 1722–1731.
[38] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, "Anrl: Attributed network representation learning via deep neural networks." in *IJCAI'18*, 2018, pp. 3155–3161.
[39] J. Liu, N. Li, and Z. He, "Network embedding with dual generation tasks," in *IJCAI'19*. AAAI Press, 2019, pp. 5102–5108.

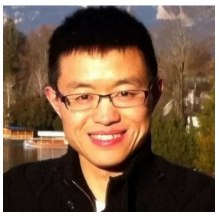
[40] H. Gao, J. Pei, and H. Huang, "Progan: Network embedding via proximity generative adversarial network," in *KDD'19*, 2019, pp. 1308–1316.
[41] Y. Hou, H. Chen, C. Li, J. Cheng, and M.-C. Yang, "A representation learning framework for property graphs," in *KDD'19*. ACM, 2019, pp. 65–73.
[42] H. Wang, E. Chen, Q. Liu, T. Xu, D. Du, W. Su, and X. Zhang, "A united approach to learning sparse attributed network embedding," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 557–566.
[43] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *TKDE*, vol. 31, no. 2, pp. 357–370, 2018.
[44] L. Du, Z. Lu, Y. Wang, G. Song, Y. Wang, and W. Chen, "Galaxy network embedding: A hierarchical community structure preserving approach," in *IJCAI'18*, 2018, pp. 2079–2085.
[45] Q. Long, Y. Wang, L. Du, G. Song, Y. Jin, and W. Lin, "Hierarchical community structure preserving network embedding: A subspace approach," in *CIKM'19*, 2019, pp. 409–418.
[46] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin, "Multi-dimensional network embedding with hierarchical structure," in *WSDM'18*. ACM, 2018, pp. 387–395.
[47] G. Fu, C. Hou, and X. Yao, "Learning topological representation for networks via hierarchical sampling," in *IJCNN'19*. IEEE, 2019, pp. 1–8.
[48] J. Ma, P. Cui, X. Wang, and W. Zhu, "Hierarchical taxonomy aware network embedding," in *KDD'18*. ACM, 2018, pp. 1920–1929.
[49] L. Zhang and B. Zhang, *The Theory of Quotient Space and its Applications. 2nd Version*. Beijing: Tsinghua Press, 2007.
[50] M. Girvan and M. E. Newman, "Community structure in social and biological networks," in *Proceedings of the National Academy of Sciences*, 2002, pp. 7821–7826.
[51] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.
[52] D. Sculley, "Web-scale k-means clustering," in *WWW'10*, 2010, pp. 1177–1178.
[53] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
[54] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.
[55] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *KDD'08*. ACM, 2008, pp. 990–998.
[56] N. A. Weiss and M. J. Hassett, *Introductory statistics*. pearson education London, 2012.

Shu Zhao received her PhD degree in Computer Science from Anhui University in 2007. She is currently a professor in the Department of Computer Science and Technology, Anhui University. Her current research interests include quotient space theory, granular computing, data mining and machine learning.

Ziwei DU is a postgraduate student in the School of Computer Science and Technology, Anhui University. Her research interests include network representation learning, granular computing.

Jie Chen received her PhD degree in Computer Science from Anhui University in 2014. She is currently an associate professor in the Department of Computer Science and Technology, Anhui University. Her current research interests include deep learning, quotient space theory, granular computing, and machine learning.

Yanping Zhang received her PhD degree in Computer Science from Anhui University in 2003. She is currently a professor in the Department of Computer Science and Technology, Anhui University. Her current research interests include deep learning, quotient space theory, granular computing, and machine learning.



Jie Tang is currently a professor and the associate chair of the Department of Computer Science and Technology at Tsinghua University. His research interests include cognitive graph, data mining, social networks, and artificial intelligence. He has been visiting scholar at Cornell University and Southampton University. He has published more than 300 papers. He served as PC co-chair of CIKM'16, WSDM'15, EiC of IEEE Transactions on Big Data, and Acting EiC of Transactions on Knowledge Discovery from Data. He leads the project AMiner.org, an AIenabled research network analysis system. He was honored with the UK Royal Society-Newton Advanced Fellowship Award.

Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is a Distinguished Professor in Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. Before joining UIC, Dr. Yu was with IBM, where he was manager of the Software Tools and Techniques department at the Watson Research Center. His research interest is on big data, including data mining, data stream, database and privacy. He has published more than 1,200 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. Dr. Yu is the recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion and anonymization of big data, the IEEE Computer Societys 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining and anonymization of big data, and the Research Contributions Award from IEEE Intl. Conference on Data Mining (ICDM) in 2003 for his pioneering contributions to the field of data mining. He also received the ICDM 2013 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award (2014). He was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011-2017) and IEEE Transactions on Knowledge and Data Engineering (2001-2004).