Dynamic Heterogeneous Information Network Embedding with Meta-path based Proximity

Xiao Wang*, Yuanfu Lu*, Chuan Shi†, *Member, IEEE*, Ruijia Wang, Peng Cui, *Member, IEEE*, Shuai Mou

Abstract—Heterogeneous information network (HIN) embedding aims at learning the low-dimensional representation of nodes while preserving structure and semantics in a HIN. Existing methods mainly focus on static networks, while a real HIN usually evolves over time with the addition (deletion) of multiple types of nodes and edges. Because even a tiny change can influence the whole structure and semantics, the conventional HIN embedding methods need to be retrained to get the updated embeddings, which is time-consuming and unrealistic. In this paper, we investigate the problem of dynamic HIN embedding and propose a novel Dynamic HIN Embedding model (DyHNE) with meta-path based proximity. Specifically, we introduce the meta-path based first- and second-order proximities to preserve structure and semantics in HINs. As the HIN evolves over time, we naturally capture changes with the perturbation of meta-path augmented adjacency matrices. Thereafter, we learn the node embeddings by solving generalized eigenvalue problem effectively and employ eigenvalue perturbation to derive the updated embeddings efficiently without retraining. Experiments show that DyHNE outperforms the state-of-the-arts in terms of effectiveness and efficiency.

Index Terms—Dynamic Heterogeneous Information Network, Network Embedding, Social Network Analysis

1 Introduction

Heterogeneous information network (HIN) has shed a light on the analysis of network (graph) data, which consists of multiple types of nodes connected by various types of edges [1]. For example, the DBLP network has four types of nodes: Author (A), Paper (P), Conference (C) and Term (T); and multiple types of relations: writing/written relations between authors and papers, and publish/published relations between papers and conferences, etc. Moreover, a meta-path, describing a composite relation between nodes, is widely used to exploit rich semantics in HINs [2]. In DBLP, the meta-path *APA* means the co-author relation, while *APCPA* represents that two authors publish papers in the same conference. Hence, a HIN contains much complex structure and semantics and studying HIN is of great importance for applications in practice.

Recently, HIN embedding, as a promising way of HIN analysis, has attracted considerable attention [3], [4]. It aims at learning the low-dimensional representation of nodes while preserving the HIN structure and semantic information, so that various downstream applications, such as node classification [5] and link prediction [6], [7], can be benefited from HIN embedding. Several HIN embedding methods have been proposed. For example, the random walk based methods [8], [9], the decomposition based methods [10], [11], [12], the deep neural network based methods [7], [13], [14] and some task-specific methods [15], [16]. However, all of these methods are designed for static HINs, i.e., the structure and semantics do not change over time. In reality, a HIN usually exhibits high dynamics with the evolution of various types of nodes and edges, e.g., the newly added (deleted) nodes or edges. Moreover, the changes of nodes and edges in a dynamic HIN may vary by types. Still taking the DBLP as an example, an advisor collaborates with different students on different papers, resulting in the continuous evolutions of co-author relations and emerging papers. Besides, a large number of new papers are added to the network while the number of conferences remains almost unchanged each year.

Actually, the current HIN embedding methods can hardly handle such complex evolutions effectively in a dynamic HIN. Even with a tiny change in a HIN, these methods have to be retrained repeatedly at each time step, which is very time-consuming and does not meet the realtime processing demand. Although some methods are proposed to deal with dynamic networks [17], [18], [19], they do not consider the heterogeneity of networks and largely ignore various semantic relations in HINs. Directly utilizing these methods for dynamic HINs will inevitably lose some structure and semantics, and reduce the performance of downstream tasks. Thus, an effective and efficient dynamic HIN embedding method is highly desirable in a real HIN analysis scenario.

Basically, there are two fundamental problems which need to be carefully considered for dynamic HIN embedding. *One is how to effectively preserve the structure and semantics in a dynamic HIN.* Since the network structure and semantic relations are the two most important and direct information in HINs, they essentially ensure the effectiveness of the learned embeddings. As the HIN evolves with a newly added node, the local structure centered on this node will be changed, and such changes will be gradually propagated across all the nodes via different meta-paths, leading to changes in the global structure. Moreover, the new node will not only establish direct links with neighborhoods, but also establish complex relations with other nodes through various meta-paths, which will inevitably influence the semantic

- * *indicates equal contribution and* † *indicates corresponding author.*
  Xiao Wang, Yuanfu Lu, Chuan Shi and Ruijia Wang are with the Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, China.
  E-mail: {xiaowang, shichuan, wangruijia}@bupt.edu.cn
- *Yuanfu Lu is also with WeChat Search Application Department, Tencent, China.*
- *Peng Cui is with the Department of Computer Science and Technology in Tsinghua University, Beijing 100084, China.*
  E-mail: cuip@tsinghua.edu.cn
- *Shuai Mou is with Tencent , China.*
  E-mail: harrymou@tencent.com

relations in HINs. Thus, both structure and semantics will change with the evolution of the dynamic HIN. Modeling the changes and encoding the (high-order) structure and semantics in the learned embeddings simultaneously are very critical yet challenging for an effective dynamic HIN embedding method.

The other problem is how to efficiently update the node embeddings without retraining on the whole HIN, when the HIN evolves over time. For each time step, retraining a HIN embedding method is the most straightforward way to get the optimal embeddings. However, apparently, this strategy is very time consuming, especially when the change of network structure is very slight. In the era of big data, retraining manner becomes unrealistic. These problems motivate us to seek an effective and efficient method to preserve the structure and semantics for dynamic HIN embedding.

In this paper, we propose a **Dy**namic **H**eterogeneous information **N**etwork **E**mbedding model (**DyHNE**) with meta-path based proximity to effectively and efficiently learn the node embeddings. Inspired by the perturbation theory [20] widely used for capturing changes of a system, we learn the node embeddings by solving the generalized eigenvalue problem and model the evolution of the HIN with the eigenvalue perturbation. Along this line, we firstly adopt meta-path augmented adjacency matrices to model the typology of the HIN, and build a basic static HIN embedding model (i.e., StHNE) to preserve both of the meta-path based first- and second-order proximities. Thus we can better capture the structure and semantics in dynamic HINs. For capturing the evolution of the HIN, we then utilize the perturbations of multiple meta-path augmented adjacency matrices to model the changes of the structure and semantics of the HIN in a natural manner. Finally, we employ the eigenvalue perturbation theory to incorporate the changes and derive the node embeddings efficiently. In this way, there is no need to retrain StHNE to get the optimal embeddings.

The contributions of our work are summarized as follows:

- For the first time, we study the problem of incrementally learning node embeddings for dynamic HINs , which makes the HIN embedding more practical in the real-world scenario.
- We initiate a static HIN embedding model (StHNE) to preserve structure and semantics in a HIN. Based on StHNE, a dynamic HIN embedding model (DyHNE) with meta-path based proximity is proposed to derive the updated embeddings efficiently, which can be applied to large-scale HINs with the linear time complexity with respect to the number of nodes.
- We conduct comprehensive evaluations to show that our model significantly outperforms several state-of-the-arts in terms of effectiveness and efficiency.

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 describes notations used in the paper and presents some definitions. Then, we propose the dynamic HIN embedding method in Section 4. Experiments and detailed analysis are reported in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

In this section, we first introduce the related methods of general network embedding, and then discuss the recent works on HIN embedding. At last, we briefly present recent works on dynamic network embedding.

2.1 Network Embedding

Network embedding aims to to project a network into a low-dimensional latent space while preserving the original structural information and properties in networks [3], [4], [21]. In the literature, network embedding can be traced back to the dimensionality reduction technique, which typically learns the latent low-dimensional vectors for nodes or edges by decomposing a network [22], [23]. Ahmed et al. [24] propose to represent a graph as a matrix where matrix elements correspond to edges between nodes, and then conduct matrix factorization to learn a low-dimensional representation of a graph. Isomap [22] aims to find the low-dimensional representations for a data set by approximately preserving the geodesic distances between data pairs. These decomposition-based graph embedding methods have achieved good performance in some cases. However, they suffer from the complex computation of a large-scale matrix decomposition, which makes them neither practical nor effective for addressing data mining tasks in large-scale networks.

Along with word2vec [25], which embeds words with low-dimensional vectors, many advances have been made toward this emerging network analysis paradigm [17], [18], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38]. For instance, [26], [30] combine random walk and skip-gram [39] to learn node representations. These methods construct some node sequences by randomly walking on a network, and then leverage skip-gram based models to learn node embeddings. In order to preserve the first-order and second-order proximities between nodes, Tang et al. [27] present a large-scale information network embedding model. GraRep [28] and HOPE [40] are both designed to model the high-order proximity between nodes in networks. [32], [36] perform matrix factorization to find a low-rank space to represent a network. Some deep neural network based models are also proposed for network embedding, such as autoencoder based methods [31], [37]. Besides network topology, some works focus on utilizing the side information, e.g., node content in networks [29], [35]. Recent graph neural networks (GNN) have achieve large attention and some GNN-based models are proposed to solve various data mining tasks (e.g., classification) [41], [42], [43], [44]. Although these methods achieve promising performance, all of them can only handle homogeneous networks and cannot be directly applied to embed HINs which contain multiple types nodes and edges.

2.2 HIN Embedding

Due to the heterogeneity of networks, HIN embedding focuses on preserving structural and semantic information in a network [1], [3], which provides a new perspective for heterogeneous data analysis and makes network embedding more practical in the real world. Analogous to homogeneous network embedding mentioned before, HIN embedding methods can be broadly categorized into four types. The first is meta-path based random walk [8], [9], [45]. Dong et al. [9] propose to randomly walk on a HIN based on meta-paths and then embed different types of nodes into their corresponding latent spaces. HIN2Vec [8] conducts random walk and learns latent vectors of nodes by conducting multiple prediction training tasks jointly. Secondly, some methods decompose a HIN into simple networks and then model them separately [10], [11], [16]. For example, EOE [11] decomposes the complex academic heterogeneous network into a word co-occurrence network and an author cooperative network, and simultaneously performs representation learning on node pairs in sub-networks. Thirdly, there are also some neural network based methods that are designed to embed HINs [7], [13], [14], [46]. Wang et al. [7] model heterogeneous information with an autoencoder and then obtain the final node embeddings by aggregating

multiple feature representations. At last, some HIN embedding methods are proposed for exploring HIN unique properties (e.g., heterogeneous structures) [47], [48] or conducting specific tasks (e.g., recommendation and link prediction) [15], [49]. In PME [15], Chen et al. propose to map different types of nodes into the same relation space and conduct heterogeneous link prediction. All of the above methods only focus on embedding static HIN networks, while ignoring that the network itself is dynamically changing over time.

2.3 Dynamic Network Embedding

Recently, some researchers have begun to pay attention to dynamic network embedding and some attempts have been done [17], [33], [38], [50], [51], [52]. DANE [33] is proposed to learn node embeddings in dynamic attribute networks, which learns node embeddings with an offline method and updates embeddings as network and attribute change over time. Based on the generalized eigenvalue problem, DANE captures the changes of structures with the adjacency matrix and models the changes of attributes with the attribute matrix, which only considers the first-order proximity. In order to preserve high-order proximity between nodes in a dynamic network, Zhu et al. [17] design a GSVD based method DHPE to learn and update node embeddings as network evolves. By transforming the GSVD problem to a generalized eigenvalue problem, DHPE incorporates the changes of dynamic networks with Katz Index based matrix, so as to preserve high-order proximity in homogenous networks. In DynamicTriad [38], Zhou et al. model the evolution of a network as a triadic closure process and learn node embeddings for each network snapshot at different timesteps. DynamicTriad imposes triad (i.e., a group of three vertices) to model the dynamic changes of network structures, and models how a closed triad develops from an open triad. Song et al. [51] extend skip-gram based models and propose a dynamic network embedding framework. Most recently, DHNE [53] is proposed to learn node embeddings in dynamic heterogeneous networks. DHNE constructs comprehensive historical-current networks based on subgraphs of snapshots, on which random walks are performed and a dynamic heterogeneous skip-gram model is used to learn the embeddings. DHNE focuses on preserving dynamic characteristics of nodes with a dynamic heterogeneous skip-gram mode, which cannot incrementally update node embeddings without retraining model.

All in all, the aforementioned methods are either designed for homogeneous networks which contains relatively simple structures, or cannot handle dynamic HINs which have to be retrained on the whole HIN to obtain fresh embeddings as the structure changes.

3 Notations and Definitions

As a HIN evolves over time, nodes and edges may be added (deleted) and changes will vary by types. Formally, we define a dynamic HIN as follows:

Definition 1. **Dynamic Heterogeneous Information Network**. A dynamic Heterogeneous Information Network (HIN) at time step t is defined as $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \phi, \varphi)(1 \leq t \leq T)$, where $\mathcal{V}^t$ and $\mathcal{E}^t$ denote the set of nodes and edges at time step t. In a HIN, each node v and edge e are associated with their type mapping functions $\phi : \mathcal{V}^t \rightarrow \mathcal{T}_{\mathcal{V}}$ and $\varphi : \mathcal{E}^t \rightarrow \mathcal{T}_{\mathcal{E}}$. $\mathcal{T}_{\mathcal{V}}$ and $\mathcal{T}_{\mathcal{E}}$ denote the sets of node and edge types, where $|\mathcal{T}_{\mathcal{V}}| + |\mathcal{T}_{\mathcal{E}}| > 2$.

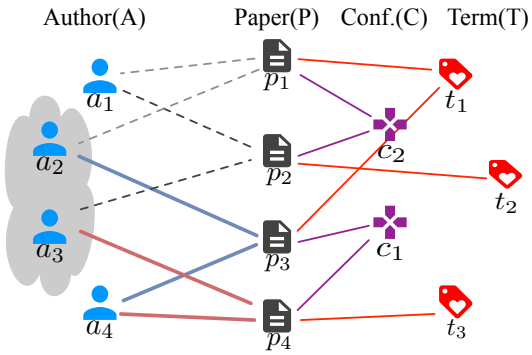


Fig. 1: A toy example of the meta-path based first- and second-order proximities in a HIN. With the meta-path *APA*, nodes a_1 and a_2 should be placed closely in the low-dimensional space as they are connected with a meta-path instance $a_1p_1a_2$, which indicates the meta-path based first-order proximity. Since the neighborhoods of a_1 is the same as a_4's, i.e., $\{a_2, a_3\}$, under the meta-path *APA*, nodes a_1 and a_4 should also be close to each other in the low-dimensional space even though they are not directly connected, which indicates the meta-path based second-order proximity.

A meta-path connects two nodes via semantic paths, which is superior to capture structure and semantics in a HIN [1], [2].

Definition 2. **Meta-path**. A meta-path m is defined as a sequence of node type $t_{v_i} \in \mathcal{T}_{\mathcal{V}}$ or edge type $t_{e_j} \in \mathcal{T}_{\mathcal{E}}$ in the form of $t_{v_1} \xrightarrow{t_{e_1}} t_{v_2} \cdots \xrightarrow{t_{e_l}} t_{v_{l+1}}$ (abbreviated as $t_{v_1}t_{v_2}\cdots t_{v_l+1}$) which describes a composite relation between v_1 and v_{l+1}. A path $(v_1, v_2, \cdots, v_{l+1})$ following the meta-path m is called as path instance of meta-path m.

Example 1. As shown in Figure 1, a meta-path $A \xrightarrow{write} P \xrightarrow{Published} C \xrightarrow{publish} P \xrightarrow{written} A$ (abbreviated as *APCPA*) describes a composite relation between two authors, which indicates that *'two authors write papers published in the same conference'*. Notice that two nodes can be connected via multiple meta-paths, and thus there are multiple path instances between two nodes. In Figure 1, nodes a_1 and a_3 can be connected via *APA* (e.g., path instance $a_1p_2a_3$) and *APCPA* (e.g., path instance $a_1p_1c_2p_2a_3$).

Since meta-paths have shown their superiority in terms of capturing structure and semantics [1], we define the meta-path based first- and second-order proximities and meta-path augmented adjacency matrix in HINs.

Definition 3. **Meta-path based First-Order Proximity**. For a pair of nodes (v_i, v_j), the number of path instances following the meta-path m connecting them represents the first-order proximity between v_i and v_j, which measures the local structure proximity between nodes in HINs.

Definition 4. **Meta-path based Second-Order Proximity**. Under a meta-path m, the neighborhoods $\mathcal{N}(v_i)^m$ of node v_i contain nodes connected to v_i via path instances following m. The proximity between v_i's neighborhoods $\mathcal{N}(v_i)^m$ and v_j's neighborhoods $\mathcal{N}(v_j)^m$ is defined as the second-order proximity based on meta-path m, which measures the similarity of two nodes in term of their neighborhood structure.

Example 2. Fig. 1 shows a toy example of the meta-path based first- and second-order proximities in a HIN. Note that the meta-path based first-order proximity indicates the pairwise similarity between nodes, while the meta-path based second-order proximity means the similarity between a node and its neighborhood set.

Since the adjacency matrix is the most basic and common way to model the network structure, we integrate meta-path and adjacency matrix to define the meta-path augmented adjacency, which can capture the structure and semantics of the HIN in a natural manner.

Definition 5. **Meta-path Augmented Adjacency Matrix**. Given a meta-path m, we define a meta-path augmented adjacency matrix as $\mathbf{W}^m = [w_{ij}^m]$, where w_{ij}^m is the number of path instances following m connecting nodes v_i and v_j. Naturally, $\mathbf{W}^m$ combines the topological structure and semantics of the HIN. And it is symmetric, if m is a symmetric meta-path.

4 The DyHNE Model

In this section, we firstly present the static HIN embedding model (StHNE) as a basic model for preserving the meta-path based first- and second-order proximities, which learns the node embeddings by solving the generalized eigenvalue problem. Then we introduce the eigenvalue perturbation theory to derive the updated embeddings, so that our dynamic model (DyHNE) with meta-path based proximity can learn the node embeddings effectively while capturing the structure and semantics efficiently. We present an overall schematic illustration of StHNE and DyHNE in Fig. 2.

4.1 Basic Idea

The core idea of DyHNE is to build an effective and efficient architecture that can capture the changes of structure and semantics in a dynamic HIN and derive the node embeddings efficiently. To achieve this, we first introduce the meta-path based first-order and second-order proximities to preserve structure and semantics in HINs. As shown in Figure 2, three augmented adjacency matrices based on meta-path *APA*, *APCPA* and *APTPA* are defined and fused with weights, which gives rise to the fused matrix $\mathbf{W}^{(t)}$ at time t. Then, we propose a basic static HIN embedding model (StHNE) which learns node embeddings $\mathbf{U}^{(t)}$ by solving the generalized eigenvalue problem in terms of the fused matrix $\mathbf{W}^{(t)}$. As the HIN evolves from time t to $t+1$, new nodes and edges are added into the network (i.e. nodes a_3, p_4 and t_3; edges $(a_3, p_4), (a_1, p_4), (p_4, c_2), (p_4, t_2)$ and (p_4, t_3)), leading to the changes of meta-path augmented adjacency matrices. Since these matrices are actually the realization of structure and semantics in the HIN, we naturally capture changes of structure and semantics with the perturbation of the fused matrix (i.e. $\Delta\mathbf{W}$). Further, we tailor the embeddings update formulas for dynamic HIN with matrix perturbation theory, so that our dynamic HIN embedding model (DyHNE) can efficiently derive the changed embedding $\Delta\mathbf{U}$ and update network embedding from $\mathbf{U}^{(t)}$ to $\mathbf{U}^{(t+1)}$ with $\mathbf{U}^{(t+1)} = \Delta\mathbf{U} + \mathbf{U}^{(t)}$.

In a nutshell, the proposed StHNE is capable of capturing the structures and semantics in a HIN with meta-path based first-order and second-order proximities, and DyHNE achieves the efficient update of network embeddings with the perturbation of meta-path augmented adjacency matrices.

4.2 Static HIN Embedding

Before achieving effective update node embeddings when the HIN evolves over time, a proper static HIN embedding for capturing structural and semantic information is a must. Hence, we next propose a static HIN embedding model (StHNE), which preserves the meta-path based first- and second-order proximities.

4.2.1 StHNE with Meta-path based First-order Proximity

The meta-path based first-order proximity models the local proximity in HINs, which means that the nodes connected via path instances are similar. Given a node pair (v_i, v_j) connected via path instances following m, we model the meta-path based first-order proximity as follows:

$$p_1^m(v_i, v_j) = w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2, \tag{1}$$

where $\mathbf{u}_i \in \mathbb{R}^d$ is the d-dimension representation vector of node v_i. To preserve the meta-path based first-order proximity in HINs, we minimize the following objective function:

$$\mathcal{L}_1^m = \sum_{v_i, v_j \in \mathcal{V}} w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2. \tag{2}$$

As larger w_{ij}^m indicates that v_i and v_j have more connections via the meta-path m, which makes nodes v_i and v_j closer in the low-dimensional space.

4.2.2 StHNE with Meta-path based Second-order Proximity

The meta-path based second-order proximity is determined through the shared neighborhood structure of nodes. Given the neighbors of node v_p under the meta-path m, denoted as $\mathcal{N}(v_p)^m$, we can model the second-order proximity based on meta-path as follows:

$$p_2^m(v_p, \mathcal{N}(v_p)^m) = ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2. \tag{3}$$

Here, we normalize w_{pq}^m so that $\sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq} = 1$.

With Eq. (3), we keep the node p close to its neighbors under a specific meta-path. As shown in Fig. 1, under the meta-path *APA*, nodes a_1 and a_4 share the same neighborhood set $\{a_2, a_3\}$, Eq. (3) guarantees that node a_1 is close to set $\{a_2, a_3\}$, and node a_4 is also close to $\{a_2, a_3\}$, so nodes a_1 and a_4 will be close even if they are not directly connected. This implicitly preserves the meta-path based second-order proximity of two unconnected nodes, as defined in Definition 3. To preserve the meta-path based second-order proximity in HINs, we minimize the following object function:

$$\mathcal{L}_2^m = \sum_{v_p \in \mathcal{V}} ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2. \tag{4}$$

Intuitively, minimizing Eq. (4) will cause the small distance between node v_p and its neighbors in the low-dimensional space. Thus, nodes that shares the same neighbors with node v_p will also be close to v_p. In this way, the meta-path based second-order proximity defined in Def. 3 can be preserved.

4.2.3 The Unified StHNE Model

Considering multiple semantic relations in a HIN, we define a set of meta-paths $\mathcal{M}$ and assign weights $\{\theta_1, \theta_2, ..., \theta_{|\mathcal{M}|}\}$ to each meta-path, where $\forall \theta_i > 0$ and $\sum_{i=1}^{|\mathcal{M}|} \theta_i = 1$. Thus, our unified model, which combines multiple meta-paths while preserving both of the meta-path based first- and second-order proximities, is as follows:

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \theta_m (\mathcal{L}_1^m + \gamma \mathcal{L}_2^m), \tag{5}$$

where γ is the trade-off factor. Now, the static HIN embedding problem is turned to:

$$\arg \min_{\mathbf{U}^\top \mathbf{D} \mathbf{U} = \mathbf{I}} \sum_{m \in \mathcal{M}} \theta_m (\mathcal{L}_1^m + \gamma \mathcal{L}_2^m), \tag{6}$$

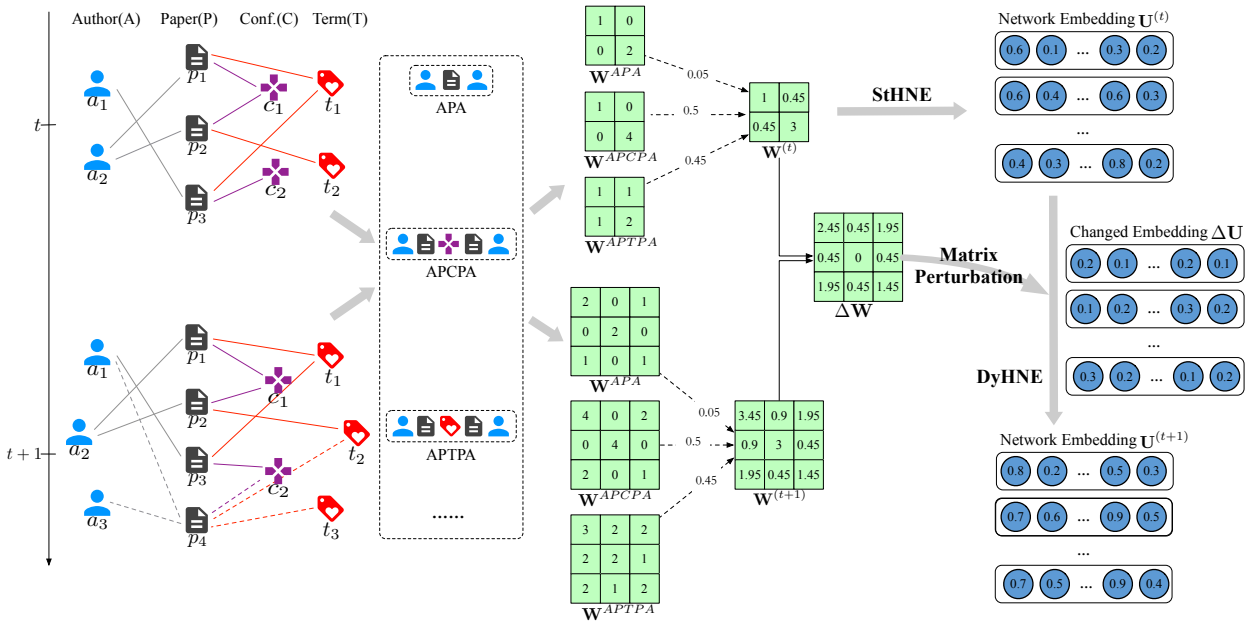


Fig. 2: The overall architecture of the proposed StHNE and DyHNE. At time step t, StHNE extracts three meta-path augmented adjacency matrices (i.e., $\mathbf{W}^{APA}$, $\mathbf{W}^{APCPA}$ and $\mathbf{W}^{APTPA}$) based on meta-path *APA*, *APCPA* and *APTPA*, and fuses them with weights. By solving the generalized eigenvalue problem, StHNE derives the node embeddings that preserve the meta-path based first- and second-order proximities. At the next time stem $t+1$, three new nodes (i.e., author a_3, paper p_4 and term t_3) appear in the HIN, and correspondingly, five edges (i.e., $(a_3, p_4), (a_1, p_4), (p_4, c_2), (p_4, t_2), (p_4, t_3)$) join the network. The changes of structure and semantics in the HIN are captured with the perturbation of fused meta-path augmented adjacency matrix $\Delta\mathbf{W}$. Based on matrix perturbation theory, DyHNE efficiently derives the changed embedding $\Delta\mathbf{U}$ and updates network embeddings from $\mathbf{U}^{(t)}$ to $\mathbf{U}^{(t+1)}$ with $\mathbf{U}^{(t+1)} = \Delta\mathbf{U} + \mathbf{U}^{(t)}$.

where $\mathbf{D}$ is the degree matrix that will be described later. The constraint $\mathbf{U}^\top\mathbf{D}\mathbf{U} = \mathbf{I}$ removes an arbitrary scaling factor in the embedding and avoids the degenerate case where all node embeddings are equal.

4.3 Optimization with Spectral Theory

Inspired by spectral theory [23], [54], we transform the problem of Eq. (6) as the generalized eigenvalue problem, so that we can get a closed-form solution and dynamically update embeddings with the eigenvalue perturbation theory [20]. Hence, we reformulate Eq. (2) as follows:

$$\begin{aligned}\mathcal{L}_1^m &= \sum_{v_i, v_j \in \mathcal{V}} w_{ij}^m ||\mathbf{u}_i - \mathbf{u}_j||_2^2 \\ &= 2tr(\mathbf{U}^\top \mathbf{L}^m \mathbf{U}),\end{aligned} \tag{7}$$

where $tr(\cdot)$ is the trace of the matrix, $\mathbf{U}$ is the embedding matrix, $\mathbf{L}^m = \mathbf{D}^m - \mathbf{W}^m$ is the Laplacian matrix under the meta-path m, and $\mathbf{D}^m$ is a diagonal matrix with $\mathbf{D}_{ii}^m = \sum_j w_{ij}^m$. Similarly, Eq. (4) can be rewritten as follows:

$$\begin{aligned}\mathcal{L}_2^m &= \sum_{v_p \in \mathcal{V}} ||\mathbf{u}_p - \sum_{v_q \in \mathcal{N}(v_p)^m} w_{pq}^m \mathbf{u}_q||_2^2 \\ &= 2tr(\mathbf{U}^\top \mathbf{H}^m \mathbf{U}),\end{aligned} \tag{8}$$

where $\mathbf{H}^m = (\mathbf{I} - \mathbf{W}^m)^\top (\mathbf{I} - \mathbf{W}^m)$ is symmetric. As discussed earlier, we fuse all meta-paths in $\mathcal{M}$, which gives rise to:

$$\mathbf{W} = \sum_{m \in \mathcal{M}} \theta_m \mathbf{W}^m, \qquad \mathbf{D} = \sum_{m \in \mathcal{M}} \theta_m \mathbf{D}^m. \tag{9}$$

Hence, the StHNE can be reformulated as:

$$\mathcal{L} = tr(\mathbf{U}^\top (\mathbf{L} + \gamma \mathbf{H}) \mathbf{U})), \tag{10}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{H} = (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W})$. Now, the problem of static HIN embedding reduces to:

$$\arg \min_{\mathbf{U}^\top \mathbf{D} \mathbf{U} = \mathbf{I}} tr(\mathbf{U}^\top (\mathbf{L} + \gamma \mathbf{H}) \mathbf{U}), \tag{11}$$

where $\mathbf{L} + \gamma\mathbf{H}$ is symmetric.

The problem of Eq. (11) boils down to the generalized eigenvalue problem as follows [55]:

$$(\mathbf{L} + \gamma \mathbf{H}) \mathbf{U} = \mathbf{D} \mathbf{\Lambda} \mathbf{U}, \tag{12}$$

where $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, ..., \lambda_{N_\mathcal{M}})$ is the eigenvector matrix, $N_\mathcal{M}$ is the number of nodes in the meta-path set $\mathcal{M}$.

Having transformed the StHNE as the generalized eigenvalue problem, the embedding matrix $\mathbf{U}$ is given by the top-d eigenvectors with the smallest non-zero eigenvalues. As the HIN evolves from time t to $t+1$, the dynamic HIN embedding model focuses on efficiently updating $\mathbf{U}^{(t)}$ to $\mathbf{U}^{(t+1)}$. That is, update the eigenvectors and eigenvalues.

4.4 Dynamic HIN Embedding

The core idea of a dynamic HIN embedding model is to learn node embeddings efficiently in a dynamic manner, thus we next develop an efficient way to update the eigenvectors and eigenvalues based on matrix perturbation.

4.4.1 *Matrix Perturbation in DyHNE*

Following the previous works [17], [33], we assume that the network evolves on a common node set of cardinality N. A nonexistent node is treated as an isolated node with zero degree and thereby the evolution of a network can be regarded as the change of edges [56]. Besides, the addition (deletion) of edges may vary by types. It is naturally appealing to capture the

evolution of a dynamic HIN with the perturbation of meta-path augmented adjacency matrix $\Delta\mathbf{W} = \sum_{m\in\mathcal{M}} \theta_m \Delta\mathbf{W}^m$. Thus, the changes of $\mathbf{L}$ and $\mathbf{H}$ can be calculated as follows:

$$\Delta\mathbf{L} = \Delta\mathbf{D} - \Delta\mathbf{W}, \tag{13}$$

$$\Delta\mathbf{H} = \Delta\mathbf{W}^\top\Delta\mathbf{W} - (\mathbf{I}-\mathbf{W})^\top\Delta\mathbf{W} - \Delta\mathbf{W}^\top(\mathbf{I}-\mathbf{W}). \tag{14}$$

Since perturbation theory can give approximate solution to a problem by adding a perturbation term [20], we can update eigenvalues and eigenvectors from the eigenvalues and eigenvectors at the previous time with the eigenvalue perturbation. Hence, at new time step, we have the following equation based on Eq. (12):

$$\begin{aligned}&(\mathbf{L}+\Delta\mathbf{L}+\gamma\mathbf{H}+\gamma\Delta\mathbf{H})(\mathbf{U}+\Delta\mathbf{U})\\ &=(\mathbf{D}+\Delta\mathbf{D})(\mathbf{\Lambda}+\Delta\mathbf{\Lambda})(\mathbf{U}+\Delta\mathbf{U}),\end{aligned} \tag{15}$$

where $\Delta\mathbf{U}$ and $\Delta\mathbf{\Lambda}$ are the changes of the eigenvectors and eigenvalues. Here, we omit the (t) superscript for brevity since the perturbation process for any time step t is the same. Let us focus on a specific eigen-pair $(\mathbf{u}_i, \lambda_i)$, Eq. (15) can be rewritten as follows:

$$\begin{aligned}&(\mathbf{L}+\Delta\mathbf{L}+\gamma\mathbf{H}+\gamma\Delta\mathbf{H})(\mathbf{u}_i+\Delta\mathbf{u}_i)\\ &=(\lambda_i+\Delta\lambda_i)(\mathbf{D}+\Delta\mathbf{D})(\mathbf{u}_i+\Delta\mathbf{u_i}).\end{aligned} \tag{16}$$

Hence, the dynamic HIN embedding problem is how to calculate the changes of the i-th eigen-pair $(\Delta\mathbf{u}_i, \Delta\lambda_i)$, because if we have $\Delta\mathbf{U}$ and $\Delta\mathbf{\Lambda}$ between t and $t+1$, we can efficiently update the embedding matrix with $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \Delta\mathbf{U}$.

We first introduce how to calculate $\Delta\lambda_i$. By expanding Eq. (16) and removing the higher order terms that have limited effects on the accuracy of the solution [20], such as $\Delta\mathbf{L}\Delta\mathbf{u}_i$ and $\Delta\lambda_i\Delta\mathbf{D}\Delta\mathbf{u}_i$, then based on the fact $(\mathbf{L}+\gamma\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{D}\mathbf{u}_i$, we have the following:

$$\begin{aligned}&(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i + (\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\ &=\lambda_i\mathbf{D}\Delta\mathbf{u}_i + \lambda_i\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{D}\mathbf{u}_i.\end{aligned} \tag{17}$$

Furthermore, left multiplying both sides by $\mathbf{u}_i^\top$, we have:

$$\begin{aligned}&\mathbf{u}_i^\top(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i + \mathbf{u}_i^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\ &=\lambda_i\mathbf{u}_i^\top\mathbf{D}\Delta\mathbf{u}_i + \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i.\end{aligned} \tag{18}$$

As $\mathbf{L}+\gamma\mathbf{H}$ and $\mathbf{D}$ are symmetric, then based on the fact $(\mathbf{L}+\gamma\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{D}\mathbf{u}_i$ and right multiplying both side by $\Delta\mathbf{u}_i$, we have $\mathbf{u}_i^\top(\mathbf{L}+\gamma\mathbf{H})\Delta\mathbf{u}_i = \lambda_i\mathbf{u}_i^\top\mathbf{D}\Delta\mathbf{u}_i$. Thus, we can rewrite Eq. (18) as follows:

$$\mathbf{u}_i^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i = \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i. \tag{19}$$

Based on Eq. (19), we get the changes of the eigenvalue λ_i:

$$\Delta\lambda_i = \frac{\mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i + \gamma\mathbf{u}_i^\top\Delta\mathbf{H}\mathbf{u}_i - \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i}{\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i}. \tag{20}$$

It is easy to see that $\mathbf{D}$ is a positive-semidefinite matrix, so we have $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i = 1$ and $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_j = 0(i\neq j)$ [20], [57]. Thus,

$$\Delta\lambda_i = \mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i + \gamma\mathbf{u}_i^\top\Delta\mathbf{H}\mathbf{u}_i - \lambda\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i. \tag{21}$$

Having got the change of eigenvalue $\Delta\lambda_i$ between two continuous time steps, our next goal is to calculate the changes of eigenvectors $\Delta\mathbf{u}_i$.

As a HIN usually evolves smoothly [56], the network changes based on meta-paths (i.e., $\Delta\mathbf{W}$) are subtle. We assume the perturbation of the eigenvectors $\Delta\mathbf{u}_i$ is linearly weighted by the top-d eigenvectors with the smallest non-zero eigenvalues [20]:

$$\Delta\mathbf{u}_i = \sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j, \tag{22}$$

where α_{ij} indicates the weight of $\mathbf{u}_j$ on $\Delta\mathbf{u}_i$. Thus, the problem of calculating $\Delta\mathbf{u}_i$ now is transformed into how to determine these weights. Considering Eq.(16), by replacing $\Delta\mathbf{u}_i$ with Eq.(22) and removing the higher order terms that have limited effects on the accuracy of the solution [58], we obtain the following:

$$\begin{aligned}&(\mathbf{L}+\gamma\mathbf{H})\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j + (\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\ &=\lambda_i\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j + \lambda_i\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{D}\mathbf{u}_i.\end{aligned} \tag{23}$$

With the fact that $(\mathbf{L}+\gamma\mathbf{H})\sum_{j=2}^{d+1}\alpha_{ij}\mathbf{u}_j = \mathbf{D}\sum_{j=2}^{d+1}\alpha_{ij}\lambda_j\mathbf{u}_j$, and by multiplying $\mathbf{u}_p^\top(2\leq p\leq d+1, p\neq i)$ on both sides of Eq. (23), we get:

$$\begin{aligned}&\mathbf{u}_p^\top\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\lambda_j\mathbf{u}_j + \mathbf{u}_p^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i\\ &=\lambda_i\mathbf{u}_p^\top\mathbf{D}\sum_{j=2,j\neq i}^{d+1}\alpha_{ij}\mathbf{u}_j + \lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i + \Delta\lambda_i\mathbf{u}_p^\top\mathbf{D}\mathbf{u}_i.\end{aligned} \tag{24}$$

Based on $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_i = 1$ and $\mathbf{u}_i^\top\mathbf{D}\mathbf{u}_j = 0(i\neq j)$, we can simplify the above formula and get:

$$\lambda_p\alpha_{ip} + \mathbf{u}_p^\top(\Delta\mathbf{L}+\gamma\Delta\mathbf{H})\mathbf{u}_i = \lambda_i\alpha_{ip} + \lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i. \tag{25}$$

Finally, we obtain the weight α_{ip} as follows:

$$\alpha_{ip} = \frac{\mathbf{u}_p^\top\Delta\mathbf{L}\mathbf{u}_i + \gamma\mathbf{u}_p^\top\Delta\mathbf{H}\mathbf{u}_i - \lambda_i\mathbf{u}_p^\top\Delta\mathbf{D}\mathbf{u}_i}{\lambda_i - \lambda_p}, i\neq p. \tag{26}$$

To sum up, we now have the changes of eigenvalues and eigenvectors based on Eq. (21), (22) and (26). The new eigenvalues and eigenvectors at $t+1$ can be updated as follows:

$$\mathbf{\Lambda}^{(t+1)} = \mathbf{\Lambda}^{(t)} + \Delta\mathbf{\Lambda}, \qquad \mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \Delta\mathbf{U}. \tag{27}$$

4.4.2 Acceleration

Until now, a straightforward idea to update the embeddings is to calculate Eq. (21), (22) and (26) for Eq. (27). However, the calculation of Eq. (21) is time-consuming due to the definition of $\Delta\mathbf{H}$ (i.e., Eq. (14)). Thus, we propose an acceleration solution tailored for dynamic HIN embedding.

Let us focus on $\Delta\lambda_i$ and α_{ij} in a more detailed way. We replace $\Delta\mathbf{H}$ with Eq. (14) and remove the higher order terms as earlier, Eq. (21) and Eq. (26) can be reformulated as follows:

$$\begin{aligned}\Delta\lambda_i =&\mathbf{u}_i^\top\Delta\mathbf{L}\mathbf{u}_i - \lambda_i\mathbf{u}_i^\top\Delta\mathbf{D}\mathbf{u}_i\\ &+\gamma\{[(\mathbf{W}-\mathbf{I})\mathbf{u}_i]^\top\Delta\mathbf{W}\mathbf{u}_i + (\Delta\mathbf{W}\mathbf{u}_i)^\top(\mathbf{W}-\mathbf{I})\mathbf{u}_i\},\end{aligned} \tag{28}$$

$$\begin{aligned}\alpha_{ij} =&\frac{\mathbf{u}_j^\top\Delta\mathbf{L}\mathbf{u}_i - \lambda_i\mathbf{u}_j^\top\Delta\mathbf{D}\mathbf{u}_i}{\lambda_i-\lambda_j}\\ &+\frac{\gamma\{[(\mathbf{W}-\mathbf{I})\mathbf{u}_j]^\top\Delta\mathbf{W}\mathbf{u}_i + (\Delta\mathbf{W}\mathbf{u}_j)^\top(\mathbf{W}-\mathbf{I})\mathbf{u}_i\}}{\lambda_i-\lambda_j}.\end{aligned} \tag{29}$$

For convenience sake, we rewrite $\Delta\lambda_i$ and α_{ij} as follows:

$$\Delta\lambda_i = \mathbf{C}(i,i) + \gamma[\mathbf{A}(:,i)^\top \mathbf{B}(:,i) + \mathbf{B}(:,i)^\top \mathbf{A}(:,i)], \quad (30)$$

$$\alpha_{ij} = \frac{\mathbf{C}(j,i) + \gamma[\mathbf{A}(:,j)^\top \mathbf{B}(:,i) + \mathbf{B}(:,j)^\top \mathbf{A}(:,i)]}{\lambda_i - \lambda_j}, \quad (31)$$

where $\mathbf{A}(:,i) = (\mathbf{W} - \mathbf{I})\mathbf{u}_i$, $\mathbf{B}(:,i) = \Delta\mathbf{W}\mathbf{u}_i$ and $\mathbf{C}(i,j) = \mathbf{u}_i^\top \Delta\mathbf{L}\mathbf{u}_j - \lambda_i \mathbf{u}_i^\top \Delta\mathbf{D}\mathbf{u}_j$.

Obviously, the calculation of $\mathbf{A}$ is time-consuming. Hence, we define $\mathbf{A}^{(t+1)}(:,i)$ at time step $t+1$ as follows:

$$\mathbf{A}^{(t+1)}(:,i) = (\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})(\mathbf{u}_i + \Delta\mathbf{u}_i). \quad (32)$$

Replacing $\Delta\mathbf{u}_i$ with Eq. (22), we have:

$$\begin{aligned}\mathbf{A}^{(t+1)}(:,i) &= (\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})(\mathbf{u}_i + \sum_{j=2, j\neq i}^{d+1} \alpha_{ij}\mathbf{u}_j) \\ &= \sum_{j=2}^{d+1} \beta_{ij}(\mathbf{W} - \mathbf{I} + \Delta\mathbf{W})\mathbf{u}_j,\end{aligned} \quad (33)$$

where $\beta_{ij} = \alpha_{ij}$ if $i \neq j$, otherwise, $\beta_{ij} = 1$. Furthermore, we can obtain the following:

$$\mathbf{A}^{(t+1)}(:,i) = \sum_{j=2}^{d+1} \beta_{ij}(\mathbf{A}^t(:,j) + \mathbf{B}^t(:,j)). \quad (34)$$

Now, we reduce the time complexity of updating $\mathbf{A}^{(t+1)}$ from $O(ed)$ to $O(d^2)$, which guarantees the efficiency of DyHNE.

4.4.3 Complexity Analysis

Since we only need to run the static model once at the very beginning of the dynamic model, we can dynamically update the representation of the network over T time steps.

For the StHNE, given a meta-path set $\mathcal{M}$ with $N_\mathcal{M}$ nodes, the time complexity of the generalized eigenvalue problem is $O(dN_\mathcal{M}^2)$, where d is the embedding dimension. Although the theoretical time complexity of the StHNEl is high, the real running time is very low since we only need to calculate top-d eigenvalues and eigenvectors of a sparse matrix.

For the DyHNE, let us denote T as the total number of the time steps. f and g are the number of non-zeros entries in the sparse matrices $\Delta\mathbf{D}$ and $\Delta\mathbf{W}$, respectively. In each time step, the time complexities of calculating $\mathbf{B}$ and $\mathbf{C}$ are $O(fd)$ and $O((f+g)d^2)$, respectively. To calculate α_i and top-d eigenvectors, our model takes $O(d^2)$ and $O(d^2 N_\mathcal{M})$, respectively. Finally, updating $\mathbf{A}^{(t+1)}$ takes $O(d^2)$. Overall, the time complexity of the proposed dynamic update method over T time steps is $O(T(f+g+N_\mathcal{M})d^2)$. Since $d \ll N_\mathcal{M}$, f and g are often small, the time complexity of our dynamic model is linear with the number of nodes in the network.

5 Experiments

In this section, we conduct comprehensive and extensive experiments to demonstrate the effectiveness and efficiency of the proposed model. Specifically, we first evaluate the effectiveness of our proposed StHNE and DyHNE. Next is the evaluation of the efficiency of the dynamic update methods. At last, we investigate the parameter sensitivity. Code and dataset are available online[1].

1. https://github.com/rootlu/DyHNE

TABLE 1: Statistics of datasets.

Datasets	Node Types	#Nodes	Meta-path	Time Steps
Yelp	Star (S) User (U) Review (R) Business (B)	9 1,286 33,360 2,614	BSB BRURB	10
DBLP	Term (T) Paper (P) Author (A) Conference (C)	8,833 14,376 14,475 20	APA APCPA APTPA	10
AMiner	Term (T) Paper (P) Author (A) Conference (C)	8,811 18,181 22,942 22	APA APCPA APTPA	10

5.1 Datasets and Settings

5.1.1 Datasets

We evaluate models on three datasets, including two academic networks (i.e., DBLP and AMiner) and a social media network (i.e., Yelp). The statistics of these datasets are summarized in Table 1.

- **Yelp**[2] is a social media dataset provided by Yelp Challenge. We extract information related to restaurants of three subcategories: "American (New) Food", "Fast Food" and "Sushi Bars" [59], and construct a HIN. The meta-paths that we are interested in are *BRURB* (i.e., the user reviewed on two businesses) and *BSB* (i.e., the same star level businesses).
- **DBLP**[3] is an academic network in computer science. In this data set, 4057 authors are labeled with their research areas such as data mining. We consider meta-paths including *APA* (i.e., the co-author relationship), *APCPA* (i.e., authors sharing conferences) and *APTPA* (i.e., authors sharing terms).
- **AMiner**[4] is also an academic network, which evolved from 1990 to 2005 in five research domains. For each author who published in these five domains, his/her label is assigned to the category with the majority of his/her publications. As in DBLP, we are also interested in *APA*, *APCPA* and *APTPA*.

Please note that we can actually take much more meta-paths into consideration for HIN embedding. However, there are infinite meta-paths in a HIN, not all meta-paths have a positive effect on embeddings [45], [60]. We consider two aspects to choose the meta path. On the one hand, we analyze the structure and semantics of different meta-path. For example, in DBLP dataset, meta-path APA contains much more information than PCP, since APA indicates a co-author relationship but PCP only means two paper published in the same conference. Hence, meta-path APA is much important for co-author relationship prediction. On the other hand, we select widely used meta-paths according to the previous works [9], [45], [59]. In Esim [45], authors give different weights to various meta-paths, so as to evaluate the importance of different meta-path to downstream tasks, which inspires us to select meta-path with higher weights. Since the selection of meta-path is still an open question and our work does not focus on this point, we select the most used and meaningful meta-paths based on prior knowledge and previous works [9], [45], [59].

2. https://www.yelp.com/dataset/challenge
3. https://dblp.uni-trier.de
4. https://www.aminer.cn/data

TABLE 2: Performance evaluation of node classification on static HINs. (Tr.Ratio means the training ratio.)

Datasets	Metric	Tr.Ratio	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE-1st	StHNE-2nd	StHNE
Yelp	Macro-F1	40%	0.6021	0.5389	0.5438	0.6387	0.5872	0.6193	0.5377	**0.6421**
		60%	0.5954	0.5865	0.5558	0.6464	0.6081	0.6639	0.5691	**0.6644**
		80%	0.6101	0.6012	0.6068	0.6793	0.6374	0.6909	0.5783	**0.6922**
	Micro-F1	40%	0.6520	0.6054	0.6105	0.6896	0.6427	0.6838	0.6118	**0.6902**
		60%	0.6472	0.6510	0.6233	0.7011	0.6681	0.7103	0.6309	**0.7017**
		80%	0.6673	0.6615	0.6367	0.7186	0.6875	0.7232	0.6367	**0.7326**
DBLP	Macro-F1	40%	0.9295	0.9271	0.9172	0.9354	0.9213	0.9392	0.9283	**0.9473**
		60%	0.9355	0.9298	0.9252	0.9362	0.9311	0.9436	0.9374	**0.9503**
		80%	0.9368	0.9273	0.9301	0.9451	0.9432	0.9511	0.9443	**0.9611**
	Micro-F1	40%	0.9331	0.9310	0.9219	0.9394	0.9228	0.9421	0.9312	**0.9503**
		60%	0.9383	0.9328	0.9291	0.9406	0.9305	0.9487	0.9389	**0.9519**
		80%	0.9392	0.9323	0.9347	0.9502	0.9484	0.9543	0.9496	**0.9643**
AMiner	Macro-F1	40%	0.8838	0.8929	0.8972	0.9449	**0.9487**	0.9389	0.9309	0.9452
		60%	0.8846	0.8909	0.8967	0.9482	0.9490	0.9401	0.9354	**0.9499**
		80%	0.8853	0.8947	0.8962	0.9491	0.9493	0.9412	0.9381	**0.9521**
	Micro-F1	40%	0.8879	0.8925	0.8958	0.9465	**0.9469**	0.9407	0.9412	0.9467
		60%	0.8881	0.8936	0.8960	0.9482	0.9497	0.9423	0.9431	**0.9509**
		80%	0.8882	0.8960	0.8962	0.9500	0.9511	0.9448	0.9423	**0.9529**

TABLE 3: Performance evaluation of relationship prediction on static HINs.

Datasets	Metric	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE-1st	StHNE-2nd	StHNE
Yelp	AUC	0.7404	0.6553	0.7896	0.6651	0.8187	0.8046	0.8233	**0.8364**
	F1	0.6864	0.6269	0.7370	0.6361	0.7355	0.7348	0.7397	**0.7512**
	ACC	0.6819	0.6115	0.7326	0.6386	0.7436	0.7286	0.7526	**0.7661**
DBLP	AUC	0.9235	0.8368	0.7672	0.9074	0.9291	0.9002	0.9246	**0.9385**
	F1	0.8424	0.7680	0.7054	0.8321	0.8645	0.8359	0.8631	**0.8850**
	ACC	0.8531	0.7680	0.6805	0.8416	0.8596	0.8266	0.8577	**0.8751**
AMiner	AUC	0.7366	0.5163	0.5835	0.8691	0.8783	0.8935	**0.9180**	0.8939
	F1	0.5209	0.5012	0.5276	0.6636	0.6697	0.7037	**0.8021**	0.7085
	ACC	0.6686	0.6475	0.6344	0.7425	0.7506	0.7622	**0.8251**	0.7701

5.1.2 Baselines

We compare our proposed models StHNE and DyHNE with comprehensive state-of-the-art network embedding methods, including two homogeneous network embedding methods (i.e., DeepWalk [26] and LINE [27]); two heterogeneous information network embedding methods (i.e., ESim [45] and metapath2vec [9]); and two dynamic homogeneous network embedding methods (i.e., DANE [33], DHPE [17] and DHNE [53]). Additionally, in order to verify the effectiveness of the meta-path based first-order and second-order proximities, we test the performance of StHNE-1st and StHNE-2nd. We use codes of the baseline methods provided by their authors.

- **DeepWalk** [26] [5] performs random walks on networks and then learns low-dimensional node vectors via Skip-gram model.
- **LINE** [27] [6] considers first-order and second-order proximities in networks. We denote the model that only uses first-order or second-order proximity as LINE-1st or LINE-2nd respectively.
- **ESim** [45] [7] takes a given set of meta-paths as input to learn the representation of nodes. For fair comparison, we tune the weights of meta-paths as we do in our models.
- **metapath2vec** [9] [8] leverages meta-path based random walk and Skip-gram to learn node embedding. Since metapath2vec cannot handle multiple meta-paths at the same time, we tune the weights of meta-paths as we do in our models and then fuse embeddings learned from single meta-paths with the optimal weights.
- **DANE** [33] [9] is a framework for dynamic attributed network embedding. We train this model without considering node types and learn the representation of nodes.
- **DHPE** [38] [10] adopts the generalized SVD to preserve the high-order proximity in homogeneous network, which is also designed for incrementally updating the embeddings of nodes.
- **DHNE** [53] [11] constructs comprehensive historical-current networks based on subgraphs of snapshots, on which random walks are performed and a dynamic heterogeneous skip- gram model is used to learn the embeddings.
- **StHNE-1st** is our static model for HIN embedding only utilizing the first-order proximity.
- **StHNE-2nd** is our static model for HIN embedding only utilizing the second-order proximity.

5.1.3 Parameters

For a fair comparison, we set the embedding dimension $d = 100$ for all models. The trade-off factor γ in our method is set as 1. We adopt grid search with a range of $(0, 1)$ to obtain the best-weighted combination of meta-paths in evaluations assuming that we have the ground truth. The size of negative samples is set as

5. https://github.com/phanein/deepwalk
6. https://github.com/tangjianpku/LINE
7. https://github.com/shangjingbo1226/ESim
8. https://ericdongyx.github.io/metapath2vec/m2v.html
9. http://www.public.asu.edu/ jundongl/code/DANE.zip
10. http://pengcui.thumedialab.com
11. https://github.com/Yvonneupup/DHNE

TABLE 4: Performance evaluation of node classification on dynamic HINs. (Tr.Ratio means the training ratio.)

Datasets	Metric	Tr.Ratio	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE	DANE	DHPE	DHNE	DyHNE
Yelp	Macro-F1	40%	0.5840	0.5623	0.5248	0.6463	0.5765	0.6118	0.6102	0.5412	0.6293	**0.6459**
		60%	0.5962	0.5863	0.5392	0.6642	0.6192	0.6644	0.6342	0.5546	0.6342	**0.6641**
		80%	0.6044	0.6001	0.6030	0.6744	0.6285	0.6882	0.6471	0.5616	0.6529	**0.6893**
	Micro-F1	40%	0.6443	0.6214	0.5901	0.6932	0.6457	0.6826	0.6894	0.5823	0.6689	**0.6933**
		60%	0.6558	0.6338	0.5435	0.6941	0.6656	0.7074	0.6921	0.5981	0.6794	**0.6998**
		80%	0.6634	0.6424	0.6297	0.7104	0.6722	0.7281	0.6959	0.6034	0.6931	**0.7298**
DBLP	Macro-F1	40%	0.9269	0.9266	0.9147	0.9372	0.9162	0.9395	0.8862	0.8893	0.9302	**0.9434**
		60%	0.9297	0.9283	0.9141	0.9369	0.9253	0.9461	0.8956	0.8946	0.9351	**0.9476**
		80%	0.9322	0.9291	0.9217	0.9376	0.9302	0.9502	0.9051	0.9087	0.9423	**0.9581**
	Micro-F1	40%	0.9375	0.9310	0.9198	0.9383	0.9254	0.9438	0.8883	0.8847	0.9352	**0.9467**
		60%	0.9346	0.9245	0.9192	0.9404	0.9281	0.9496	0.8879	0.8931	0.9404	**0.9505**
		80%	0.9371	0.9297	0.9261	0.9415	0.9354	0.9543	0.9071	0.9041	0.9489	**0.9617**
AMiner	Macro-F1	40%	0.8197	0.8219	0.8282	0.8797	0.8673	0.8628	0.7642	0.7694	0.8903	**0.9014**
		60%	0.8221	0.8218	0.8323	0.8807	0.8734	0.8651	0.7704	0.7735	0.9011	**0.9131**
		80%	0.8235	0.8238	0.8351	0.8821	0.8754	0.8778	0.7793	0.7851	0.9183	**0.9212**
	Micro-F1	40%	0.8157	0.8189	0.8323	0.8729	0.8652	0.8563	0.7698	0.7633	0.8992	**0.9117**
		60%	0.8175	0.8182	0.8361	0.8734	0.8693	0.8574	0.7723	0.7698	0.9045	**0.9178**
		80%	0.8191	0.8201	0.8298	0.8751	0.8725	0.8728	0.7857	0.7704	0.9132	**0.9203**

TABLE 5: Performance evaluation of relationship prediction on dynamic HINs.

Datasets	Metric	DeepWalk	LINE-1st	LINE-1st	ESim	metapath2vec	StHNE	DANE	DHPE	DHNE	DyHNE
Yelp	AUC	0.7316	0.6549	0.7895	0.6521	0.8164	0.8341	0.7928	0.7629	0.8023	**0.8346**
	F1	0.6771	0.6125	0.7350	0.6168	0.7293	0.7506	0.7221	0.6809	0.7194	**0.7504**
	ACC	0.6751	0.6059	0.7300	0.6185	0.7395	0.7616	0.7211	0.7023	0.7024	**0.7639**
DBLP	AUC	0.9125	0.8261	0.7432	0.9053	0.9196	0.9216	0.5413	0.6411	0.8945	**0.9278**
	F1	0.8421	0.7840	0.7014	0.8215	0.8497	0.8621	0.7141	0.6223	0.8348	**0.8744**
	ACC	0.8221	0.7227	0.6754	0.8306	0.8405	0.8436	0.5511	0.5734	0.8195	**0.8635**
AMiner	AUC	0.8660	0.6271	0.5648	0.8459	0.8694	0.8659	0.8405	0.8412	0.8289	**0.8823**
	F1	0.7658	0.5651	0.6071	0.7172	0.7761	0.7567	0.7167	0.7158	0.7386	**0.7792**
	ACC	0.7856	0.5328	0.5828	0.7594	0.7793	0.7733	0.7527	0.7545	0.7498	**0.7889**

5. We set the number of walks per node as 10, the walk length as 50 and the window size as 5. To apply the homogeneous network embedding models for HINs, we ignore the types of nodes and edges. We will make our code publicly available after the review.

5.2 Effectiveness of StHNE

To evaluate the effectiveness of StHNE, here we learn the node embeddings with the static embedding methods on the whole HIN without considering the evolution of the network. In other words, given a dynamic network with 10 time steps $\{\mathcal{G}^1, \cdots, \mathcal{G}^{10}\}$, we conduct all static network embedding methods, including StHNE, on the union network, i.e., $\mathcal{G}^1 \cup \mathcal{G}^1 \cup \cdots \mathcal{G}^{10}$.

5.2.1 Node Classification

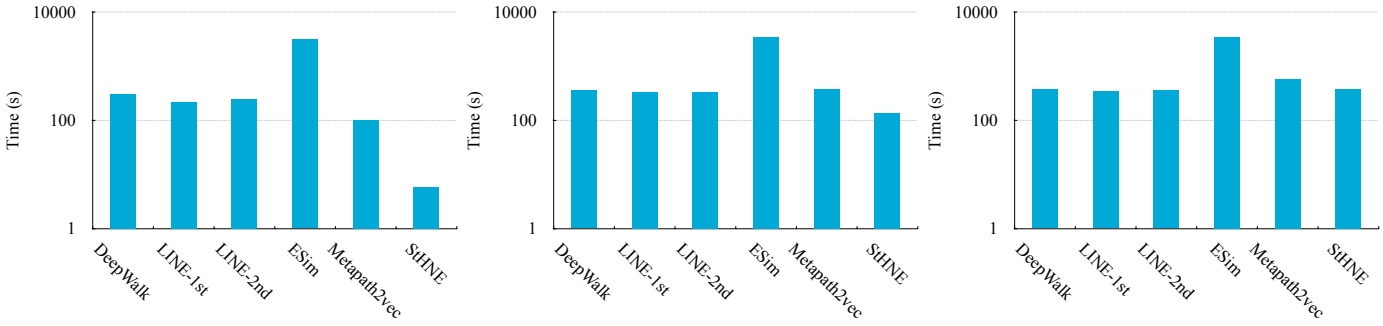
Node classification is a common task to evaluate the performance of representation learning on networks. In this task, after learning the node embeddings on the fully evolved network, we train a logistic regression classifier with node embeddings as input features. The ratio of training set is set as 40%, 60%, and 80%. We set the weights of *BSB* and *BRURB* in Yelp to 0.4 and 0.6. In DBLP, we assign weights $\{0.05, 0.5, 0.45\}$ to $\{$*APA*, *APCPA*, *APTPA*$\}$. In AMiner, we assign weights $\{0.25, 0.5, 0.25\}$ to $\{$*APA*, *APCPA*, *APTPA*$\}$. We report the results in terms of Macro-F1 and Micro-F1 in Table 2.

As we can observe, the StHNE outperforms all baselines on three datasets. It improves classification performance by about 8.7% in terms of Macro-F1 averagely with 80% training ratio, which is due to the weighted integration of meta-paths and the preservation of network structure. Both our model StHNE, ESim and metapath2vec fuse multiple meta-paths with weights, but the performances of ESim and metapath2vec are slight worse on three datasets. This may be caused by the separation of meta-paths fusion and model optimization, which lose some information between multiple relationships for HIN embedding. We also notice that StHNE-1st and StHNE-2nd both outperform LINE-1st and LINE-2nd in most cases, which shows the superiority of the meta-path based first- and second-order proximities in HINs. From a vertical comparison, our StHNE continues to perform best against different sizes of training data, which implies the stability and robustness of our model.

5.2.2 Relationship Prediction

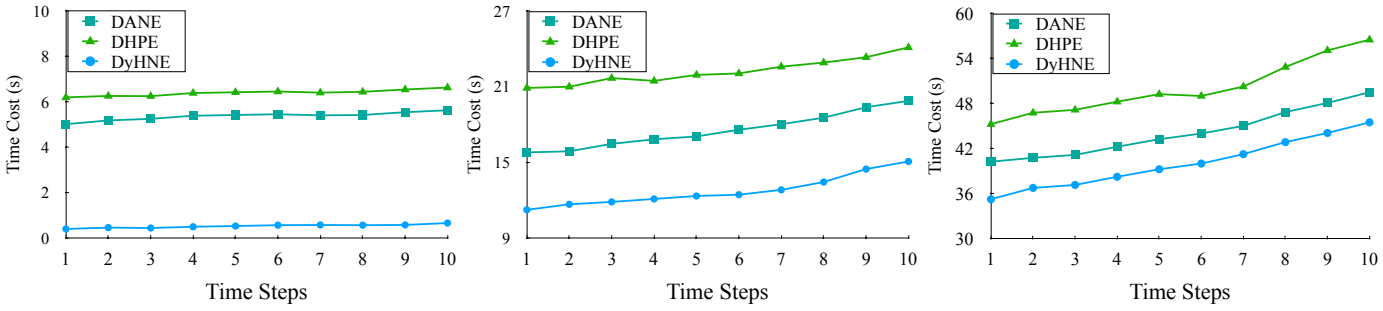
For DBLP and AMiner, we are interested in the co-author relationships (*APA*). Hence, we generate training networks by randomly hiding 20% *AP* in DBLP and 40% *AP* in AMiner as AMiner is much larger. For Yelp, we want to find two businesses that one person has reviewed (*BRURB*), which can be used to recommend businesses for users. Thus, we randomly hide 20% *BR* to generate the training network. We set the weights of *BSB* and *BRURB* in Yelp to 0.4 and 0.6. In DBLP, we assign weights $\{0.9, 0.05, 0.05\}$ to $\{$*APA*, *APCPA*, *APTPA*$\}$. In AMiner, we assign weights $\{0.4, 0.3, 0.3\}$ to $\{$*APA*, *APCPA*, *APTPA*$\}$. We evaluate the prediction performance on testing networks with AUC and Accuracy.

Table 3 shows the comparison results of different methods. Overall, we can see that StHNE achieves better relation pre-



(a) Yelp (b) DBLP (c) AMiner

Fig. 3: Efficiency of StHNE.



(a) Yelp (b) DBLP (c) AMiner

Fig. 4: Efficiency of the DyHNE compared to baselines.

diction performance than other methods on two metrics. The improvement indicates the effectiveness of our model to preserve structural information in HINs. Benefiting from the second-order proximity preserved based on meta-path, StHNE-2nd outperforms than StHNE-1st significantly. The reason is that the higher order proximity is more conducive for preserving complex relationships in HINs.

5.3 Effectiveness of DyHNE

In this section, our goal is to verify the effectiveness of DyHNE compared with these baselines designed for dynamic networks (i.e., DANE and DHPE). Since some baselines (e.g., DeepWalk, LINE and StHNE) cannot handle dynamic networks and we have reported the performance of these methods in Section 5.2, here we only apply these methods to initial networks as in [17], [33]. Specifically, given a dynamic network with 10 time steps $\{\mathcal{G}^1, \cdots, \mathcal{G}^{10}\}$, for the static network embedding methods, including StHNE, we only conduct them on $\mathcal{G}^1$ and report the results, while for the dynamic network embedding methods, i.e., DANE, DHPE and DyHNE, we conduct them from $\mathcal{G}^1$ to $\mathcal{G}^{10}$ to update the embedding incrementally, and report the final results to evaluate their performance in a dynamic environment.

5.3.1 Node Classification

For each dataset, we generate the initial and growing HIN from the original network. Each growing HIN contains ten time steps. In Yelp, reviews are time-stamped, we randomly add 0.1% new *UR* and *BR* to the initial network at each time step. For DBLP, we randomly add 0.1% new *PA*, *PC* and *PT* to the initial network at each time step. Since AMiner itself contains the published year of each paper, we divide the edges appearing in 2005 into 10 time steps uniformly.

As settings in Subsection 5.2.1, we vary the size of the training set from 40% to 80% with the step size of 20% and the remaining nodes as testing. We repeat each classification experiment for ten times and report the average performance in terms of both Macro-F1 and Micro-F1 scores, as shown in Table 4. We can see that DyHNE consistently performs better than other baselines on all datasets with all varying sizes of training data, which demonstrates the effectiveness and robustness of our learned node embeddings when served as features for node classification. Especially, our DyHNE significantly outperforms the two dynamic homogeneous network embedding methods, DANE and DHPE. The reason is that our model considers the different types of nodes and relations and can capture the structure and semantic information in HINs. We also notice that our DyHNE achieves better performance than DHNE which is also designed for dynamic HINs. We believe that the improvement is due to the preserved meta-path based first-order and second-order proximities in node embeddings learned by our DyHNE.

Compared with the baselines designed for static HINs (i.e., DeepWalk, LINE, ESim and metapath2vec), our method also achieves the best performance, which proves the effectiveness of the update algorithm without losing important structure and semantic information in HINs.

5.3.2 Relationship Prediction

For each dataset, we generate the initial, growing and testing HIN from the original HIN. For Yelp, we first build the testing

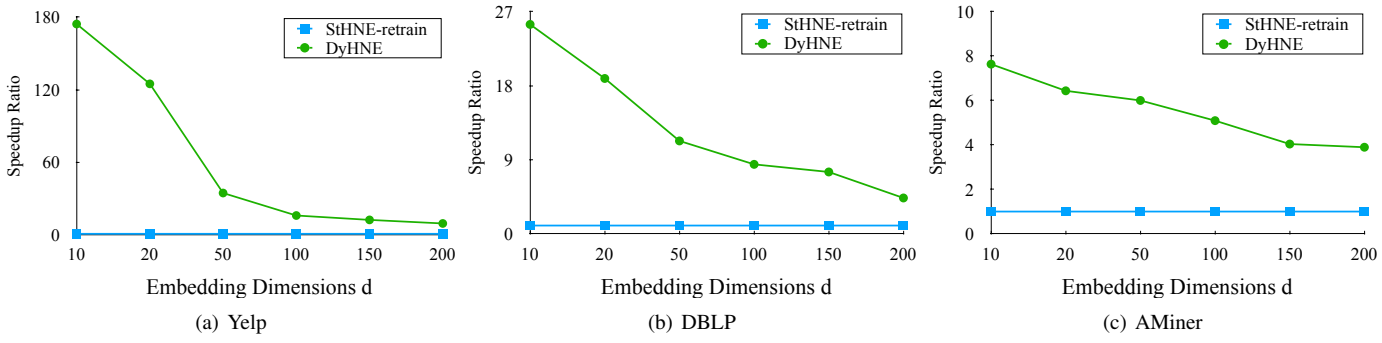


Fig. 5: The speedup ratio of DyHNE.

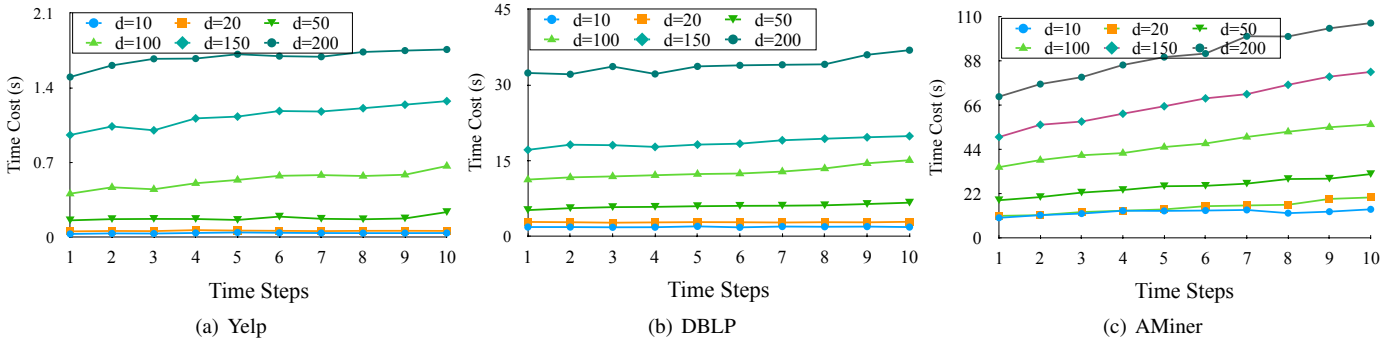


Fig. 6: The running time w.r.t embedding dimensions.

network containing 20% *BR*. The remaining constitutes the initial and growing network, where the growing network is divided into 10 time steps, and 0.1% new *UR* and *BR* are added to the initial network at each time step. For DBLP, we use the similar approach as described above. For AMiner, we take the data involved in 1990-2003 as the initial network, 2004 as the growing network and 2005 as the testing network.

We report the prediction performance in Table 5 and have some findings: (a) Our method consistently improves the relationships prediction accuracy on the three datasets, which is attributed to the structural information preserved by the meta-path based first-order and second-order proximities. (b) DANE and DHPE obtain poor performance due to the neglect of multiple types of nodes and relations in HINs. (c) Compared to DHNE, our DyHNE consistently performances better on three datasets, which is benefit from the effectiveness of update algorithm. Additionally, the meta-path based second-order proximity ensures that our DyHNE captures the high order structures of HIN, which is also preserved with the updated node embeddings.

5.4 Efficiency of StHNE

In this section, we evaluate the efficiency of the proposed static HIN embedding model StHNE. Specifically, we compare StHNE to other static model w.r.t the running time on three datasets, and plot it in a log scale.

As we can see from Fig. 3, our method StHNE is much faster than other methods, though the time complexity is large in theory. Compared with models designed for static networks (i.e., DeepWalk, LINE, ESim and metapath2vec), the running time of StHNE is in the same order with that of them, because StHNE does not require iterative optimization and only needs to calculate top-d eigenvalues and eigenvectors of the sparse matrices.

5.5 Efficiency of DyHNE

In this section, we evaluate the efficiency of our proposed DyHNE, including not only the comparison with the efficiency of the baselines that are designed for dynamic network embedding (i.e., DANE and DHPE), but also the comparison with our proposed static HIN embedding model StHNE.

5.5.1 *Efficiency compared to baselines*

Since DANE and DHPE can also handle the dynamic changes in the network, we compare the running time of DyHNE with DANE and DHPE in Fig. 4. Obviously, DyHNE is much faster than DHPE and DANE as the time complexity of DHPE is $O(T(f+g+N)d^2+d^4)$ and that of DANE is $O(T(f+g+N)d^2)$. Here, N is the total number of nodes in the network. Since DyHNE models the changes of HINs with meta-path augmented adjacency matrices, we only need to update the representation of nodes in the meta-path set instead of all nodes with $O(T(f+g+N_{\mathcal{M}})d^2)$ time complexity.

5.5.2 *Efficiency compared to StHNE*

In order to validate the superiority of DyHNE against retraining StHNE (i.e., StHNE-retrain), we compare the speedup ratio of DyHNE against StHNE with respect to embedding dimensions in Fig. 5. As the dimension is actually the number of eigenvalues to be solved, DyHNE obtains around 174× speedup ratio on Yelp when the embedding dimension is around 10. At the default embedding dimension of 100, DyHNE is also 16× faster than

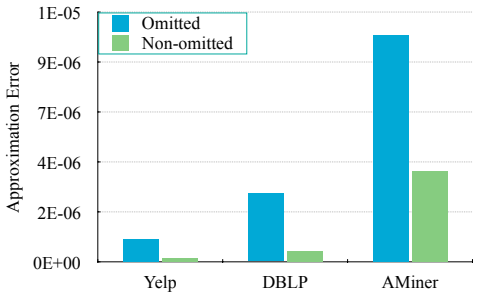


Fig. 7: The approximation error of matrix eigendecomposition in DyHNE, with respect to one timestamp update.

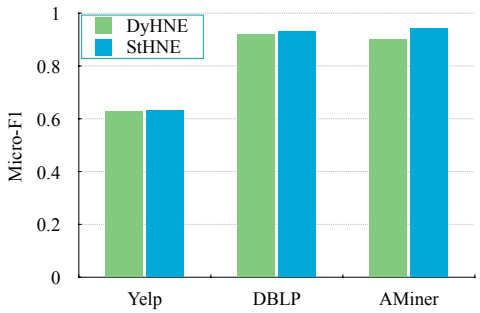


Fig. 8: New node classification. DyHNE means that the classifier takes the learned embeddings of new nodes with DyHNE as input, while StHNE means that the input of the classifier are the learned embeddings of new nodes with StHNE.

StHNE. Overall, although the speedup ratio decreases with the increase of the dimension, DyHNE is still much faster than retraining by StHNE.

To further explore the efficiency of DyHNE, we count the running time w.r.t embedding dimensions. Fig. 6 shows that the time required to update the embedding increases gradually as the dimension increases, which is consistent with our analysis of the time complexity.

5.6 Approximation Error Analysis

As mentioned before, we drop the higher-order terms that have limited effects on the accuracy of matrix decomposition [20]. In order to evaluate the effects of higher-order terms on the accuracy of the solution (the learned node embeddings), we design an approximation error analysis experiment from two aspects, i.e., matrix eigendecomposition and performance comparison. In particular, with respect to matrix eigendecomposition, we have the changed eigenvalues $\Delta\mathbf{\Lambda}$ and eigenvector $\Delta\mathbf{U}$ by our proposed DyHNE at timestamp $t+1$, (i.e., drop the higher-order terms), and then we calculate the relative approximation errors w.r.t. ommited the higher-order terms as $\frac{||(\mathbf{D}+\Delta\mathbf{D})^{-1}(\mathbf{L}+\Delta\mathbf{L}+\lambda\mathbf{H}+\lambda\Delta\mathbf{H})-(\mathbf{U}^{(t)}+\Delta\mathbf{U})(\mathbf{\Lambda}^{(t)}+\Delta\mathbf{\Lambda})(\mathbf{U}^{(t)}+\Delta\mathbf{U})^\top||_F^2}{||(\mathbf{D}+\Delta\mathbf{D})^{-1}(\mathbf{L}+\Delta\mathbf{L}+\lambda\mathbf{H}+\lambda\Delta\mathbf{H})||_F^2}$. Let $\mathbf{U}_g^{(t+1)}$ and $\Lambda_g^{(t+1)}$ denote the eigenvector and eigenvalue of the recomputed factorization for matrix $(\mathbf{L}+\Delta\mathbf{L}+\lambda\mathbf{H}+\lambda\Delta\mathbf{H})\mathbf{U}_g^{(t+1)}=(\mathbf{D}+\Delta\mathbf{D})\Lambda_g^{(t+1)}\mathbf{U}_g^{(t+1)}$, then we can get the relative approximation errors w.r.t. non-ommitted the higher-order terms as $\frac{||(\mathbf{D}+\Delta\mathbf{D})^{-1}(\mathbf{L}+\Delta\mathbf{L}+\lambda\mathbf{H}+\lambda\Delta\mathbf{H})-\mathbf{U}_g^{(t+1)}\mathbf{\Lambda}^{(t+1)_g}\mathbf{U}_g^{(t+1)\top}||_F^2}{||(\mathbf{D}+\Delta\mathbf{D})^{-1}(\mathbf{L}+\Delta\mathbf{L}+\lambda\mathbf{H}+\lambda\Delta\mathbf{H})||_F^2}$. As shown in Figure 7(a), we report the approximation error w.r.t. Non-omitted and Omitted the higher-order terms on three datasets. We can find that the approximation error is quite small with respect to one timestamp update, which indicates the omission of the higher-order terms in our DyHNE brings much little error. Compared to the approximation error w.r.t. non-omitted the higher-order terms, the approximation error w.r.t. omitted is larger, which makes sense that losing high-level terms will inevitably lead to information loss. However, the approximation error is still quite small ($< 1e-5$) which can be ignored.

On the other hand, we analyze the approximation error of task performance with the learned embedding. Specifically, we conduct two models, namely Non-omitted and Omitted models. The non-omitted means that we keep the higher-order terms in the derivation, then our final model evolves into retraining on the whole network as the HIN evolves. The omitted is actually our proposed DyHNE model, which efficiently updates node embeddings when the HIN evolves. On three datasets, we report the results of node classification and relationship prediction tasks in Table 6. As shown in the table, we can find that the overall approximation error is around $0.2\% \sim 3\%$ on three datasets in two tasks (calculated as $(|Non\text{-}omitted - Omitted|)/Non\text{-}omitted$). Compared with not omitting higher-order terms, omitting higher-order terms leads to a slight decrease in model performance on three datasets. The approximation error on AMiner dataset is larger than that on the other two datasets, we believe this is due to the large time span of AMiner dataset. Overall, the approximation errors on all datasets are small enough to be ignored. Hence, omitting the higher order terms have limited effects on the accuracy of the learned node embeddings.

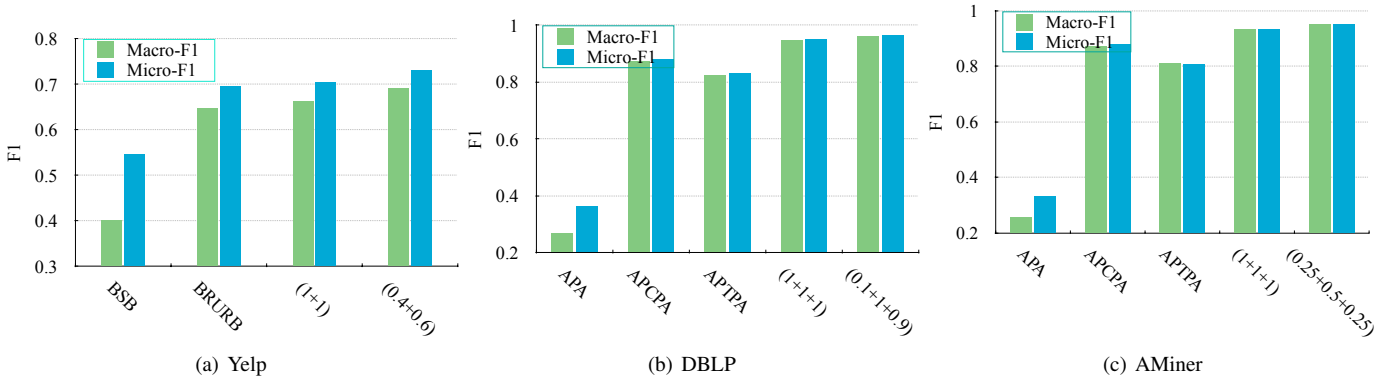
TABLE 6: The approximation error that whether to omit higher-order terms in DyHNE, w.r.t. node classification and relationship prediction.

Task	Model	Yelp	DBLP	AMiner
Classification (Mircro-F1)	Non-omitted	0.6922	0.9611	0.9521
	Omitted	0.6893	0.9581	0.9212
	Error	0.004189	0.003124	0.032454
LinkPrediction (AUC)	Non-omitted	0.8364	0.9385	0.8939
	Omitted	0.8346	0.9278	0.8821
	Error	0.002152	0.011401	0.013201

5.7 New Node Classification

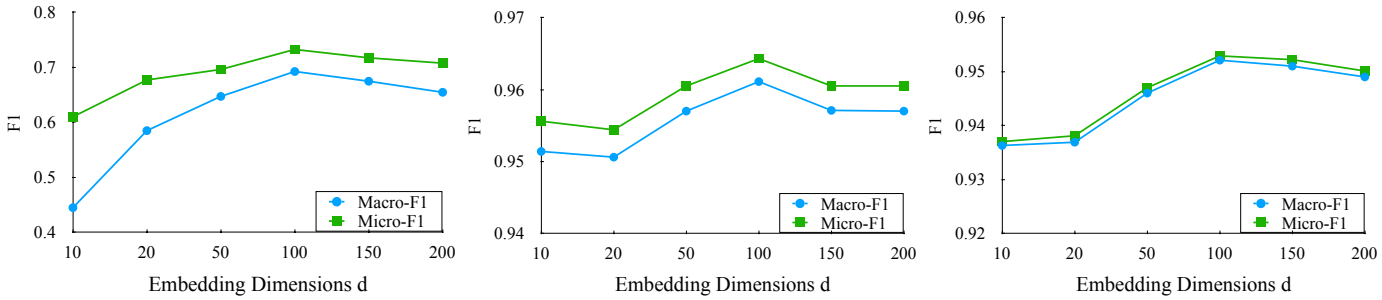
Since we assume that the newly-introduced nodes as isolated nodes following the previous works [17], [33], we regard the evolution of a network as the changes of edges in our proposed DyHNE. In order to verify the effectiveness of the learned embedding for these isolated nodes (i.e., newly-introduced nodes), we conduct node classification task on the newly-introduced node embeddings. After learning node embeddings with our StHNE (i.e., retraining model on the whole HIN) and DyHNE (i.e., training model with the dynamic HIN). We train a logistic classifier with 80% of the learned newly-introduced node embeddings as input features, and test the classifier with the rest data. The result in terms of Macro-F1 and Micro-F1 is reported in Figure 8.

Obviously, we can find that our proposed DyHNE is capable of achieving very competitive performance as StHNE. On Yelp and DBLP datasets, the classifier can accurately classify the new nodes with the updated embeddings by DyHNE, which indicates the effectiveness of the our DyHNE in dealing with these isolated nodes as we assumed. Even on AMiner with large time spans, the



(a) Yelp (b) DBLP (c) AMiner

Fig. 9: The Macro- and Micro-F1 w.r.t. fusion of meta-path. (1+1) means the weight assigned to BSB and BRURB are 1 and 1, and so on.



(a) Yelp (b) DBLP (c) AMiner

Fig. 10: The Macro-F1 and Micro-F1 w.r.t. the embedding dimensions d.

new node classification performance of DyHNE is slightly worse than that of StHNE.

5.8 Parameter Analysis

5.8.1 Meta-paths fusion

Since we fuse multiple meta-paths with weights, we explore the effect of different meta-paths on classification. Specifically, we use a single meta-path to learn the node embeddings, then weight meta-paths uniformly, and finally fuse meta-paths with the optimal weights. As shown in Fig. 9, using a single meta-path can not fully model the structure of HINs, leading to worse performance. The performance of combining multiple meta-paths with weights is improved. Since the structure and semantics of different meta-paths are different, fusing them with non-uniform weights achieves the best performance.

5.8.2 Dimension Selection

We investigate the sensitivity of the number of embedding dimension on node classification. Specifically, we vary the number of embedding dimensions as 10, 20, 50, 100, 150 and 200. The results of node classification are reported in Figure 10. As we can see, the performance of our model improves with the increase of the number of embedding dimensions, and the performance tends to be stable once the dimension of the representation reaches around 100. It is evident that our model are capable to capture rich information of various relations in HINs using the low-dimensional representation.

6 Conclusion

In this paper, we investigate the problem of dynamic HIN embedding and propose a novel representation learning model for dynamic HINs (DyHNE). Based on the designed static HIN embedding model (StHNE), DyHNE captures the structure and semantics by preserving the meta-path based first- and second-order proximities. With the evolution of the dynamic HIN, DyHNE incorporates the change of structure and semantics with meta-path augmented adjacency matrices, and efficiently learns the embedding of nodes based on perturbation theory. Experimental evaluations show that DyHNE not only significantly outperforms the state-of-the-arts, but also is much more efficient.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61702296, 61772082, 61806020, 61972442, U1936220, U1936104), the National Key Research and Development Program of China (2018YFB1402600), the CCF-Tencent Open Fund, and the Fundamental Research Funds for the Central Universities. Yuanfu Lu is also supported by 2019 Tencent Rhino-Bird Elite Training Program.

References

[1] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017.

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

[3] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[4] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[5] Y. Jacob, L. Denoyer, and P. Gallinari, "Learning latent representations of nodes for classifying in heterogeneous social networks," in *Proceedings of WSDM*. ACM, 2014, pp. 373–382.

[6] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, "Co-author relationship prediction in heterogeneous bibliographic networks," in *Proceedings of ASONAM*. IEEE, 2011, pp. 121–128.

[7] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "Shine: signed heterogeneous information network embedding for sentiment link prediction," in *Proceedings of WSDM*. ACM, 2018, pp. 592–600.

[8] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of CIKM*. ACM, 2017, pp. 1797–1806.

[9] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of SIGKDD*. ACM, 2017, pp. 135–144.

[10] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of SIGKDD*. ACM, 2015, pp. 1165–1174.

[11] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks," in *Proceedings of WSDM*. ACM, 2017, pp. 741–749.

[12] Z. Liu, V. W. Zheng, Z. Zhao, Z. Li, H. Yang, M. Wu, and J. Ying, "Interactive paths embedding for semantic proximity search on heterogeneous graphs," in *Proceedings of SIGKDD*. ACM, 2018, pp. 1860–1869.

[13] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of SIGKDD*. ACM, 2015, pp. 119–128.

[14] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *Proceedings of SIGKDD*, 2018, pp. 2190–2199.

[15] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "PME: projected metric embedding on heterogeneous networks for link prediction," in *Proceedings of SIGKDD*, 2018, pp. 1177–1186.

[16] C. Shi, B. Hu, X. Zhao, and P. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[17] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, "High-order proximity preserved embedding for dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[18] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, "Dynamic network embedding: An extended approach for skip-gram based network embedding." in *Proceedings of IJCAI*, 2018, pp. 2086–2092.

[19] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *CoRR*, vol. abs/1805.11273, 2018.

[20] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012.

[21] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *arXiv preprint arXiv:1801.05852*, 2017.

[22] M. Balasubramanian and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[23] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proceedings of NIPS*, 2002, pp. 585–591.

[24] A. Ahmed, N. Shervashidze, S. M. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of WWW*, 2013, pp. 37–48.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of SIGKDD*. ACM, 2014, pp. 701–710.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of WWW*, 2015, pp. 1067–1077.

[28] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of CIKM*. ACM, 2015, pp. 891–900.

[29] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *Proceedings of IJCAI*, 2015, pp. 2111–2117.

[30] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of SIGKDD*. ACM, 2016, pp. 855–864.

[31] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of SIGKDD*. ACM, 2016, pp. 1225–1234.

[32] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding." in *Proceedings of AAAI*, 2017, pp. 203–209.

[33] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proceedings of CIKM*. ACM, 2017, pp. 387–396.

[34] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of SIGKDD*. ACM, 2017, pp. 385–394.

[35] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation." in *Proceedings of IJCAI*, 2016, pp. 3889–3895.

[36] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifyingdeepwalk, line, pte, and node2vec," in *Proceedings of SIGKDD*. ACM, 2018, pp. 459–467.

[37] H. Gao and H. Huang, "Deep attributed network embedding," in *Proceedings of IJCAI*, 2018, pp. 3364–3370.

[38] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proceedings of AAAI*, 2018, pp. 571–578.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013, pp. 3111–3119.

[40] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of SIGKDD*. ACM, 2016, pp. 1105–1114.

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

[42] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of NIPS*, 2017, pp. 1025–1035.

[43] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Proceedings of ICLR*, vol. 1, no. 2, 2017.

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

[45] 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.

[46] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and S. Y. Philip, "Blmne: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder," in *Proceedings of ICDM*. IEEE, 2017, pp. 605–614.

[47] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary?: Revisiting heterogeneous graph embeddings," in *Proceedings of CIKM*. ACM, 2018, pp. 437–446.

[48] Y. Lu, C. Shi, L. Hu, and Z. Liu, "Relation structure-aware heterogeneous information network embedding," in *Proceedings of AAAI*, 2019, pp. 4456–4463.

[49] X. Han, C. Shi, S. Wang, S. Y. Philip, and L. Song, "Aspect-level deep collaborative filtering via heterogeneous information networks." in *Proceedings of IJCAI*, 2018, pp. 3393–3399.

[50] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Representation learning over dynamic graphs," *CoRR*, vol. abs/1803.04051, 2018.

[51] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, "Dynamic network embedding : An extended approach for skip-gram based network embedding," in *Proceedings of IJCAI*, 2018, pp. 2086–2092.

[52] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye, "Temporal network embedding with micro- and macro-dynamics," in *Proceedings of CIKM*, 2019, pp. 469–478.

[53] Y. Yin, L.-X. Ji, J.-P. Zhang, and Y.-L. Pei, "Dhne: Network representation learning method for dynamic heterogeneous networks," *IEEE Access*, vol. 7, pp. 134 782–134 792, 2019.

[54] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proceedings of NIPS*, 2002, pp. 849–856.

[55] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[56] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *CSUR*, vol. 47, no. 1, p. 10, 2014.

[57] B. N. Parlett, *The symmetric eigenvalue problem*. siam, 1998, vol. 20.

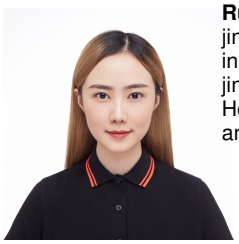
[58] T. Kato, *Perturbation theory for linear operators*. Springer Science & Business Media, 2013, vol. 132.

[59] X. Li, Y. Wu, M. Ester, B. Kao, X. Wang, and Y. Zheng, "Semi-supervised clustering in attributed heterogeneous information networks," in *Proceedings of WWW*, 2017, pp. 1621–1629.
[60] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 3, p. 11, 2013.

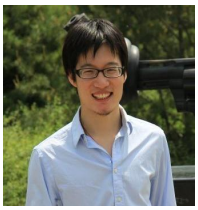
Xiao Wang is an Assistant Professor in the School of Computer Science, Beijing University of Posts and Telecommunications. He received his Ph.D. degree from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2016. He was a post-doctoral researcher in Department of Computer Science and Technology, Tsinghua University, Beijing, China. He got the China Scholarship Council Fellowship in 2014 and visited Washington University, as a joint training student from 2014 to 2015. His current research interests include data mining, social network analysis, and machine learning. Until now, he has published more than 30 papers in conferences such as AAAI, IJCAI, KDD, etc. and journals such as IEEE TKDE, IEEE Trans. on Cybernetics, etc.

Yuanfu Lu received the B.S. degree from Nanjing University of Posts and Telecommunications in 2017. He is currently a master student in Beijing University of Posts and Communications. His current research interests are in graph embedding, data mining and machine learning.

Chuan Shi received the B.S. degree from the Jilin University in 2001, the M.S. degree from the Wuhan University in 2004, and Ph.D. degree from the ICT of Chinese Academic of Sciences in 2007. He joined the Beijing University of Posts and Telecommunications as a lecturer in 2007, and is a professor and deputy director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia at present. His research interests are in data mining, machine learning, and evolutionary computing. He has published more than 40 papers in refereed journals and conferences.



Ruijia Wang received the B.S. degree from Beijing University of Posts and Telecommunications in 2019. She is currently a Ph.D student in Beijing University of Posts and Communications. Her current research interests are in data mining and machine learning.



Peng Cui received his Ph.D. degree in computer science in 2010 from Tsinghua University and he is an Associate Professor at Tsinghua. He has vast research interests in data mining, multimedia processing, and social network analysis. Until now, he has published more than 20 papers in conferences such as SIGIR, AAAI, ICDM, etc. and journals such as IEEE TMM, IEEE TIP, DMKD, etc. Now his research is sponsored by National Science Foundation of China, Samsung, Tencent, etc. He also serves as Guest Editor, Co-Chair, PC member, and Reviewer of several high-level international conferences, workshops, and journals.

ShuaiMou received his B.S degree and M.S degree in software engineering from South China University of Technology in 2014 and 2017, respectively. He was a research assistant at the Singapore Management University from 2015 to 2016. Now he is a researcher at Tencent and his main research interests are in the area of computer vision, unsupervised learning, data mining, graph computing, etc.