Fast Attributed Multiplex Heterogeneous Network Embedding

Zhijun Liu
Yantai University
Yantai, China
liuzhijun9503@126.com

Chao Huang
JD Finance America Corporation
Mountain View, USA
chaohuang75@gmail.com

Yanwei Yu*
Ocean University of China
Qingdao, China
yuyanwei@ouc.edu.cn

Baode Fan
Yantai University
Yantai, China
fanbaodeyt@163.com

Junyu Dong
Ocean University of China
Qingdao, China
dongjunyu@ouc.edu.cn

ABSTRACT

In recent years, heterogeneous network representation learning has attracted considerable attentions with the consideration of multiple node types. However, most of them ignore the rich set of network attributes (attributed network) and different types of relations (multiplex network), which can hardly recognize the multi-modal contextual signals across different relations. While a handful of network embedding techniques are developed for attributed multiplex heterogeneous networks, they are significantly limited to the scalability issue on large-scale network data, due to their heavy computation and memory cost. In this work, we propose a **F**ast **A**ttributed **M**ultiplex heterogeneous network **E**mbedding framework (FAME) for large-scale network data, by mapping the units from different modalities (*i.e.*, network topological structures, various node features and relations) into the same latent space in an efficient way. Our FAME is an integrative architecture with the scalable spectral transformation and sparse random projection, to automatically preserve both attribute semantics and multi-type relations in the learned embeddings. Extensive experiments on four real-world datasets with various network analytical tasks, demonstrate that FAME achieves both effectiveness and significant efficiency over state-of-the-art baselines. The source code is available at: https://github.com/ZhijunLiu95/FAME.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Network embedding; graph representation learning; multiplex heterogeneous networks; attributed networks; large-scale networks; sparse random projection

*Yanwei Yu is the corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411944

ACM Reference Format:
Zhijun Liu, Chao Huang, Yanwei Yu, Baode Fan, and Junyu Dong. 2020. Fast Attributed Multiplex Heterogeneous Network Embedding. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411944

1 INTRODUCTION

Representation learning on network data (*e.g.*, social networks, citation networks and E-commerce networks) has become a pragmatic research field and been applied to many online services, such as advertising, E-commerce and social media platforms. At its core is to learn low-dimensional vector representations of nodes while preserving network topological structure and intrinsic characteristics, which could facilitate various downstream network analytical tasks, *e.g.*, link prediction [6], node classification [27], spatial-temporal data modeling [23, 29] and user behavior analysis [15, 30].

Early methods towards this goal, have made significant efforts on representation learning for homogeneous networks with singular type of nodes [6, 12, 22]. To capture the network heterogeneity properties, many subsequent research works propose to model heterogeneous graph structures based on predefined meta-paths, such as metapath2vec [5]. To capture the rich neighborhood contextual signals, various graph neural network based models have been proposed to aggregate feature information from neighboring nodes, such as graph convolutional networks (GCN) [12], graph attention networks (GAT) [25], inductive graph learning (GraphSAGE) [7] and mutual information maximization schemes (DMGI) [17].

However, in real-world scenarios, we need to deal with more complex contextual network structures–multiplex heterogeneous network in which each connection between multiple types of nodes is exhibited with relationship diversity in nature. Consider the online retailing system as an example, there may exist multiple relations (*e.g.*, click, add-to-cart, add-to-preference and purchase) between the same pair of user and item [28]. The ignorance of such multi-modal relations, makes existing network embedding methods insufficient to distill effective attribute-based structural signals from the collective behaviors of users. While a handful of studies attempt to learn node embeddings on multiplex heterogeneous network [2, 17, 32], a significant deficiency is that they can hardly serve the large-scale network data (*e.g.*, millions of nodes and edges)–which is ubiquitous in practical scenarios. For example, when learning the representation of the Alibaba data with more than 40 million nodes and 500 million edges, GATNE takes nearly 4 hours to converge

with 150 distributed workers [2]. Such methods of excessively high time complexity and relying on large-scale distributed computing platform are not enough to be adopted in practice, especially in frequently updated systems. Hence, an efficient model with stronger joint representation learning capability of heterogeneous multi-type relations and contextual attributes for large-scale network data, is urgently needed.

Nevertheless, there are several key challenges that remain to be solved, in order to realize both efficient and effective representation learning for large-scale ***attributed multiplex heterogeneous networks*** (AMHENs).

- *Heterogeneity.* Heterogeneous multi-typed nodes and relations raise the challenge of automatically capturing various interactive meta-paths among nodes and high-order network structures in a unified embedding framework. Most existing heterogeneous network embedding methods require handcrafted meta-paths for different network datasets.
- *Multiplicity.* Because of heterogeneous multi-typed relations and contextual attributes, performing representation learning for AMHENs with the goal of jointly preserving meta-path interactions, global structural contexts and attributed information, remains a significant challenge.
- *Scalability.* In light of these limitations of existing multiplex heterogeneous network embedding methods, it is important and challenging to develop leaning algorithms that can scale well to large-scale networks with millions of nodes, edges and high-dimensional attributes without relying on large-scale distributed computing platform.

To tackle the aforementioned challenges, we propose a novel **F**ast **A**ttributed **M**ultiplex heterogeneous network **E**mbedding framework, named **FAME**. Specifically, we first decompose the multiplex heterogeneous network into homogeneous and bipartite subnetwork. Then, a spectral transformation module is developed to automatically aggregate the semantic-level decoupled sub-networks with the exploration of their multi-relational topological signals. To significantly reduce the computation and memory costs, we propose to endow the graph convolutional network with the efficient feature aggregation capability via the sparse random projection mechanism. To encode the rich set of network attributes, we further incorporate the external attributed features into the spectral graph transformation architecture, while preserving both semantic and structural information in the attributed multiplex heterogeneous network. The resulting model outperforms several strong baselines on both link prediction and node classification tasks. We also found that with the new designed random projection-based graph convolutional operation, our FAME embedding framework performs significantly faster than the state-of-the-art representation learning methods for AMHENs by up to several orders of magnitudes.

We highlight the key contributions of this work as follows:

- We explore how information interchange on various modalities in the network, to jointly reflect unique node characteristics and graph topological information. The proposed FAME framework can automatically capture relation-aware structural signals between nodes without the prior knowledge of meta-path definitions.
- FAME incorporates both the spectral graph transformation and node attributed features into the sparse random projection architecture, to augment the graph convolutional network with the capability of large-scale network representation learning in an efficient way.
- We perform extensive experiments on four real-world datasets to demonstrate the superiority of our proposed model when competing with other baselines. In addition to its effectiveness advantage, FAME results in up to 6 orders of magnitudes faster than state-of-the-art AMHEN embedding techniques.

2 RELATED WORK

Network Embedding. There are mainly two types of network representation learning methods: network embedding and graph neural networks. Network embedding is to embed network into a low dimensional space while preserving the network structure and property [9, 31], such as random walk based methods [6], deep neural network models [21], matrix factorization based approaches [1, 18]. For graph neural networks, GCN [12] is proposed to incorporate neighbors' features into the center node feature using convolutional operations. GAT [24] leverages attention mechanism to capture the importance of neighbors to the center node more effectively. SGC [27] is a simplified version of GCN, which only uses the product of high-order adjacency matrices and attribute matrix, without nonlinear transformation and training parameters. However, all these algorithms are proposed for the homogeneous networks.

Heterogeneous Network Embedding. Heterogeneous network embedding mainly focuses on preserving the meta-path based structural information. metapath2vec [5] formalizes meta-path based random walk to construct the heterogeneous neighborhood of nodes and then leverages a heterogeneous skip-gram model to learn node embeddings. HERec [19] also uses a meta-path based random walk strategy to generate meaningful node sequences and subsequently integrated into matrix factorization (MF) model to learn network embeddings. HetGNN [33] and HeGAN [8] incorporate bi-directional LSTM, attention mechanism, and generative adversarial networks (GAN) for heterogeneous network embedding. These methods rely on domain knowledge to choose the valuable meta-paths, whereas there also exist several methods [26, 32] which do not require meta-path selection. However, HANE [26] transforms various types of nodes with different attributes into a uniform space, which cannot distinguish the diversity of edges between nodes. GTN [32] needs explicit products of candidate adjacency matrices and requires additional memory space and computing resources.

Multiplex Heterogeneous Network Embedding. Multiplex networks are much richer than simple heterogeneous networks and often used to model complex interaction systems. Most of the existing approaches usually only capture a single view of a heterogeneous network, whereas there are usually multiple types of relations between nodes, yielding networks with multiple views. PMNE [16] proposes three aggregation models to learn one overall embedding from the multiplex network. However, it cannot capture long-distance meta-path information between nodes and ignore the rich content information of the nodes. GATNE [2] learns base embedding, edge embedding and attribute embedding to generate the

overall node embeddings. The base embedding and attribute embedding are shared among edges of different types, while the edge embedding is computed by aggregation of neighborhood information with the self-attention mechanism. HAN [25] extends GAT to heterogeneous networks through meta-path based neighbor discovery strategy and hierarchical attention mechanism. DMGI [17] learns a node encoder that maximizes the mutual information between local patches of each relation of the heterogeneous graph and the global representation of the relation. However, these methods still need to specify the meta-path type manually. Recently, GTN [32] computes the convex combinations of adjacency matrices of sub-networks with different weights to obtain candidate adjacency matrices to generate the useful meta-paths. GMP [10] builds category graphs to model user purchasing behaviors, and uses multi-scale pyramid neural network to predict users' purchasing intentions.

Random Projection. Random projection is based on the Johnson-Lindenstrauss lemma [11], which uses a random matrix with unit Euclidean column norms to find a lower-dimensional subspace that approximately preserves the Euclidean distances between all pairs of data points in the original space. There are several variants of random projection techniques for manifold and network learning [14, 20]. Recently, RandNE [34] and FastRP [3] are proposed to capture high-order structure information of homogeneous network by Gaussian random projection and sparse random projection, respectively. However, these methods ignore the heterogeneity and attributes of nodes and relations, and thus cannot capture the rich semantics on AMHENs.

3 PROBLEM DEFINITION

In this section, we introduce key notations used in this paper and then formally define the studied problem.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a network, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges between the nodes, each representing a relationship between two nodes. Each edge $e_{ij} \in \mathcal{E}$ is an ordered pair $e_{ij} = (v_i, v_j)$ and is associated with a weight $w_{ij} > 0$, which indicates the strength of the relation. If G is undirected, we have $e_{ij} \equiv e_{ji}$ and $w_{ij} \equiv w_{ji}$; if G is directed, we have $e_{ij} \not\equiv e_{ji}$ and $w_{ij} \not\equiv w_{ji}$.

Definition 1 (Heterogeneous Network). *A heterogeneous network is a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ represent the set of all node types and the set of all edge types, respectively. Each node $v \in \mathcal{V}$ belongs to a particular node type, and each edge $e \in \mathcal{E}$ is categorized into a specific edge type. If $|\mathcal{O}| + |\mathcal{R}| > 2$, the network is called **heterogeneous**; otherwise **homogeneous**.*

Definition 2 (Attributed Network). *An attributed network is a network $\mathcal{G}$ endowed with an attribute feature matrix, i.e., $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$. $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the matrix that consists of node attribute features for all nodes, where each row is the associated node feature vector of node v_i. Here, n and m denotes the number of nodes and attributes, respectively.*

Definition 3 (Attributed Multiplex Heterogeneous Network, or AMHEN). *An attributed multiplex heterogeneous network is a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{|\mathcal{R}|}\}$, where $\mathcal{G}_r = \{\mathcal{V}, \mathcal{E}_r, \mathbf{X}\}$ is the graph that contains all edges with edge type $r \in \mathcal{R}$, $\mathcal{E} = \bigcup_{r \in \mathcal{R}} \mathcal{E}_r$, and $\mathcal{V} = \bigcup_{o \in \mathcal{O}} \mathcal{V}_o$. If $|\mathcal{O}| + |\mathcal{R}| > 2$ and there exist different types of edges between same node pairs, the network is called **multiplex heterogeneous**.*

More specifically, in a multiplex heterogeneous network, there may be multiple types of edges between node v_i and v_j. Hence, an edge is denoted as e_{ij}^r in AMHENs, where r corresponds to a certain edge type. Given the above definitions, we next formally define our studied problem for representation learning on networks.

Problem 1 (Attributed Multiplex Heterogeneous Network Embedding). *Given an attributed multiplex heterogeneous network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, the problem of Attributed Multiplex Heterogeneous Network Embedding is to learn a d-dimensional vector representation for each node $v_i \in \mathcal{V}$, i.e., learn a mapping function $f : \mathcal{V} \rightarrow \mathbb{R}^d$, where $d \ll |\mathcal{V}|$.*

Key notations are summarized in Table 1.

Table 1: Main notations and their definitions.

Notation	Definition
$\mathcal{G}$	the input network
$\mathcal{V}, \mathcal{E}$	the node/edge set of $\mathcal{G}$
$\mathcal{O}, \mathcal{R}$	the node/edge type set of $\mathcal{G}$
$\mathbf{X}$	the node attribute matrix of $\mathcal{G}$
$\mathbf{Z}$	the node embedding
$\mathbf{A}$	the adjacency matrix
$\mathbb{A}$	the adjacency matrix with meta-paths
$\mathbf{R}$	the random projection matrix
d	the dimension of embeddings
n, m	the number of nodes/attributes
s	the sparsity of random projection matrix
K	the order of spectral graph transformation
α_i	the weight of spectral graph transformation
β_i	the weight of adjacency matrix

4 METHODOLOGY

In this section, we present the details of our FAME (as shown in Figure 1), consisting of two key components: (i) *spectral graph transformation* and (ii) *fast random projection embedding*. *Spectral graph transformation* aims to capture the short and long meta-paths among heterogeneous nodes across multi-relations and sub-network high-order structures. *Fast random projection embedding* incorporates the spectral graph transformation and node attribute features together into a spare random projection architecture to learn the low-dimensional representation of nodes efficiently.

4.1 Spectral Graph Transformation

Since AMHENs involve different types of nodes and relationships among these nodes (each relationship having a different role and impact on node representation), we first decouple an AMHEN into multiple homogeneous and bipartite sub-networks (the latter involving two types of nodes) to differentiate each relationship between nodes in the network. After that, we perform spectral graph

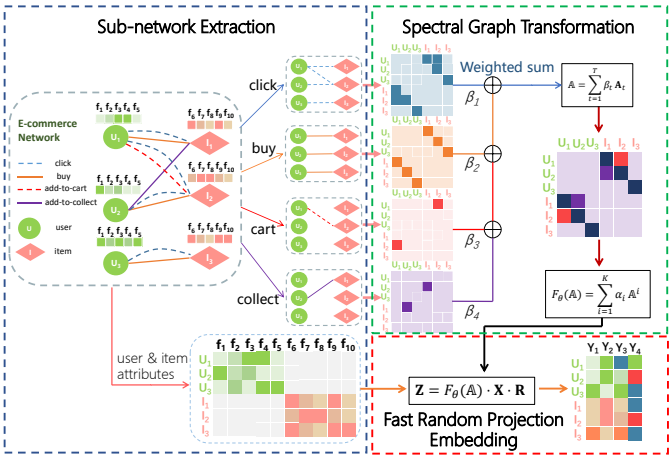


Figure 1: The overview of the proposed FAME.

transformation on these sub-networks to better capture the proximities between nodes.

Let $\{\mathcal{G}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ be the collection of obtained homogeneous networks and bipartite networks and $\{\mathbf{A}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ denote the adjacency matrices corresponding to $\{\mathcal{G}_r\}$. For example, as shown in Figure 1, an E-commerce network contains two node types (*i.e.*, user and item) and four edge types (*i.e.*, click, buy, add-to-cart and add-to-preference). In such a case, we divide the network into four bipartite networks. Notice that we extend the adjacency matrix $\mathbf{A}_r$ of sub-network $\mathcal{G}_r$ to include all types of nodes in the network to fit the dimensions of all adjacency matrices.

Inspired by [13], we can use spectral graph transformation of the adjacency matrices to capture the high-order proximities between nodes in each sub-network. One can exploit the fact that the power $\mathbf{A}^i$ of the adjacency matrix of an unweighted graph contains the number of paths of length i for each node pair. On the basis that nodes connected by many paths should be considered to have a closer relationship to each other than nodes connected by a few paths. We compute a weighted sum of powers of $\mathbf{A}$ as a spectral graph transformation function:

$$F(\mathbf{A}_r) = \sum_{i=1}^{K} \alpha_i \mathbf{A}_r^i, \tag{1}$$

where α_i is the weight for the i-th order proximity, r denotes edge type, and K is the highest order. The result is a matrix polynomial of order K. The coefficients α_i should be decreased to reflect the assumption that links are more likely to arise between nodes that are connected by short paths than nodes connected by long paths. Thus, such a function takes both path length and path count into account in capturing the high-order structures.

Nevertheless, directly applying spectral graph transformation on the separate sub-networks cannot capture the complex meta-paths between nodes in AMHENs. To address this issue, we propose a novel spectral graph transformation incorporating automatic generation of meta-paths for AMHENs.

We first perform a weighted sum of the separate sub-networks to obtain a new combined adjacency matrix:

$$\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r, \tag{2}$$

where the weight β_r of the adjacency matrix $\mathbf{A}_r$ indicates the importance of the corresponding sub-network in the network. Now, we can capture all meta-path interactions across multi-relations among nodes in the network by performing a spectral graph transformation on the new adjacency matrix $\mathbb{A}$:

$$\begin{aligned} F(\mathbb{A}) &= \sum_{i=1}^{K} \alpha_i \mathbb{A}^i \\ &= \sum_{i=1}^{K} \alpha_i \Big(\sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r\Big)^i. \end{aligned} \tag{3}$$

The spectral graph transformation function $F(\mathbb{A})$ allows learning short and long meta-paths across multi-relations including original sub-network high-order structures at the same time.

An example of AMHEN is illustrated in Figure 2, we only consider two relations (*i.e.*, buy and add-to-cart) between users and items in the toy example. $\mathbf{A}_1$ and $\mathbf{A}_2$ denote the adjacency matrices for the two sub-networks corresponding to purchase and add-to-cart relationships, respectively. As shown in Figure 2, if each adjacency matrix is regarded as an unweighted graph, and two relationships are equally important (*i.e.*, $\mathbb{A} = \mathbf{A}_1 + \mathbf{A}_2$), then spectral graph transformation captures the number of meta-paths with different lengths across multi-relations. For example, $(\mathbf{A}_1 + \mathbf{A}_2)^2$ obtains the number of 2-length meth-paths across heterogeneous multi-relations for all node pairs. If the importance of each relationship is taken into account, *e.g.*, $\mathbb{A} = \mathbf{A}_1 + 0.5 * \mathbf{A}_2$ (the importance of add-to-cart is set lower than that of purchase), our spectral graph transformation can capture the summarization of meta-paths with different lengths across multi-relations with importance. For example, the element 2.5 in $(\mathbf{A}_1 + 0.5 * \mathbf{A}_2)^2$ indicates the summarization of 2-length meta-paths from u_1 to u_1 across multi-relations with importance weights.

Furthermore, we find that when calculating higher-order adjacency matrices, the values of some nodes with larger degrees are much higher than those with lower degrees. This obvious data skewness will significantly affect the performance of the learned embedding. Therefore, we first normalize the obtained adjacency matrix $\mathbb{A}$ to reduce the data skewness before performing spectral graph transformation:

$$\mathbb{A} = \mathbf{D}^{-1}\mathbb{A} \tag{4}$$

where $\mathbf{D}$ is the degree matrix of $\mathbb{A}$:

$$\mathbf{D}_{ij} = \begin{cases} \sum_k \mathbb{A}_{ik} & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The main difference from the meta-path generation in GTN [32] is that GTN computes the convex combinations of sub-network adjacency matrices with different weights to obtain candidate adjacency matrices. Then candidate adjacency matrices are used to capture different lengths of meta-paths by stacking equal-length graph transformer layers. This requires the calculation and storage

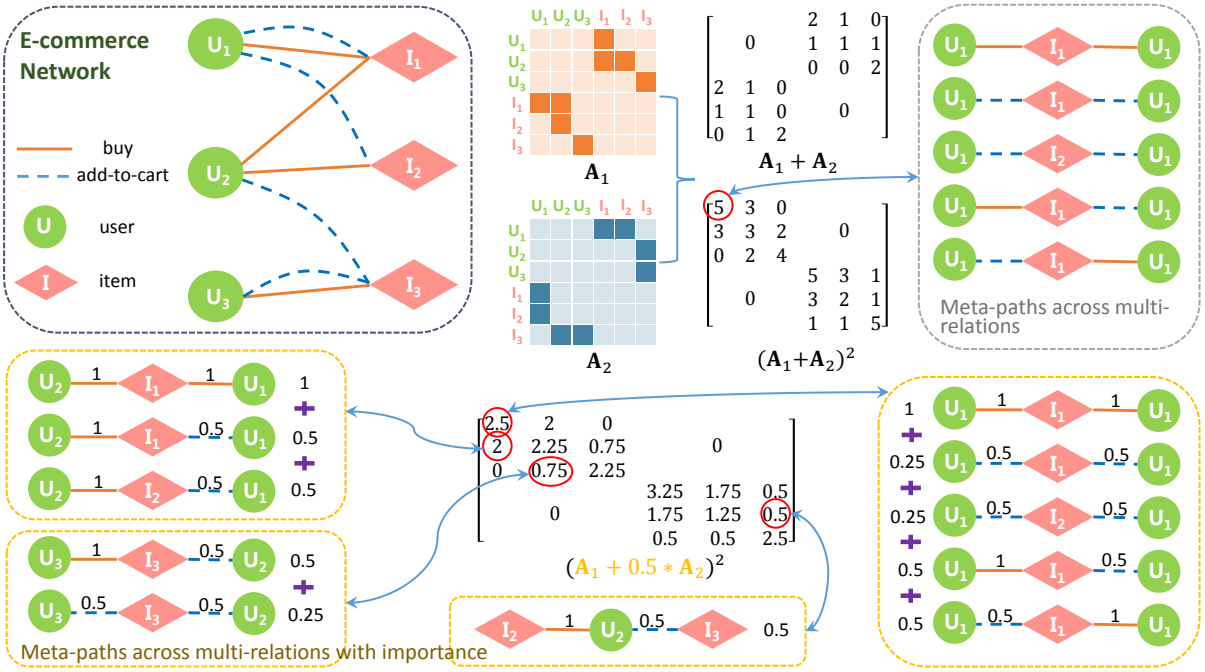


Figure 2: Illustration of spectral graph transformation for an example of AMHEN

of a large amount of candidate adjacency matrices as the meta-path length increases, which significantly increases the time complexity and space complexity of the proposed model. Our proposed spectral graph transformation can directly obtain all meta-path interactions across multi-relations among multi-type nodes in an AMHEN by performing transformation on a single weighted summed adjacency matrix $\mathbb{A}$, without computing and storing redundancy candidate adjacency matrices.

4.2 Fast Random Projection Embedding

GCNs have recently been widely used in heterogeneous network embedding and have achieved great success [12, 25, 26, 32]. Nevertheless, these methods all need to train a large number of parameters as well as a long training time, which is inefficient when dealing with large-scale networks.

To improve the efficiency, we first introduce the spectral graph convolution theorem and analyze its limitations; then we propose a fast random projection embedding method for large-scale AMHENs. Spectral graph convolution theorem defines the convolution in the Fourier domain based on the normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{D} = diag(\sum_j \mathbf{A}_{ij})$ is the degree matrix [12]. Let $\mathbf{L} = \Phi\Lambda\Phi^T$, where Φ is the matrix of eigenvectors of $\mathbf{L}$ and Λ is the diagonal matrix of its eigenvalues. The convolution on the network is defined as follows:

$$\begin{aligned} g_\theta \star \mathbf{X} &= g_\theta(\mathbf{L})\mathbf{X} \\ &= g_\theta(\Phi\Lambda\Phi^T)\mathbf{X} \\ &= \Phi g_\theta(\Lambda)\Phi^T\mathbf{X}, \end{aligned} \tag{6}$$

where $\mathbf{X} \in \mathbb{R}^{n\times m}$ is the node feature matrix, g_θ is a filter parameterized by θ in the Fourier domain, and $\Phi^T\mathbf{X}$ is the graph Fourier transform of signal $\mathbf{X}$.

To convolve the local neighbors of the target node, [4] defines $g_\theta(\Lambda)$ as a polynomial filter up to K order as follows:

$$g_\theta(\Lambda) = \sum_{k=1}^{K} \theta_k \Lambda^k, \tag{7}$$

where $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. To get a low-dimensional representation for a node, one method is to generalize g_θ to $m \times d$ filter for feature maps by replacing parameter $\theta \in \mathbb{R}^K$ with parameter matrix $\Theta \in \mathbb{R}^{K\times m\times d}$:

$$\begin{aligned} g_\theta(\Lambda) &= \sum_{k=1}^{K} \Lambda^k \Theta_k, \\ \mathbf{Z} &= \sigma(g_\theta \star \mathbf{X}) \\ &= \sigma(\Phi(\sum_{k=1}^{K} \Lambda^k)\Phi^T\mathbf{X}\Theta_k), \end{aligned} \tag{8}$$

where $\mathbf{Z}$ denotes the matrix of the node embeddings and $\sigma(\cdot)$ is the softmax activation function. However, the number of parameters that need to be optimized in such method is $K \times m \times d$. In addition, the complexity of this filtering operation is linear in the number of edges. When the size of node feature increases, the number of parameters increases. Therefore, it is difficult to apply to large-scale heterogeneous networks with many types of nodes with huge feature space.

In order to solve the above efficiency problems, we propose fast random projection embedding, which incorporates the spectral graph transformation and node features into random projection architecture to efficiently learn effective node representation. Random projection is a simple but powerful dimension reduction method that preserves pairwise distances between data points. There are many variants of random projection techniques are applied in the manifold learning area. The sparse random projection we use is one of the variations, which achieves a significant speedup with little loss on accuracy. Formally, let projection matrix $\mathbf{R} \in \mathbb{R}^{m \times d}$ with *i.i.d* entries drawn from:

$$r_{ji} = \sqrt{s} \begin{cases} 1 & \text{with prob. } \frac{1}{2s} \\ 0 & \text{with prob. } 1 - \frac{1}{s} \\ -1 & \text{with prob. } \frac{1}{2s} \end{cases} \tag{9}$$

where $s = \sqrt{m}$ or $s = \frac{m}{\log m}$ [14]. With $s = \sqrt{m}$, one can achieve a $\sqrt{m}$-fold speedup because only $\frac{1}{\sqrt{m}}$ of the data need to be processed. Since the multiplications with $\sqrt{s}$ can be delayed, no floating point arithmetic is needed.

In the fast random projection embedding, we replace the filter g_θ with our spectral graph transformation function $F(\mathbb{A})$ to capture the multi-relational structures in the AMHEN, and then incorporate it with node attribute features and finally get the low-dimensional embeddings of nodes by random projection:

$$\begin{aligned} \mathbf{Z} &= F(\mathbb{A}) \cdot \mathbf{X} \cdot \mathbf{R} \\ &= \sum_{i=1}^{K} \alpha_i \mathbb{A}^i \cdot \mathbf{X} \cdot \mathbf{R}, \end{aligned} \tag{10}$$

where $\mathbf{R} \in \mathbb{R}^{m \times d}$ is the projection matrix that can be obtained using Eq. (9). Notice that our fast random projection embedding method also supports multiplex heterogeneous networks without node attribute features. Under such situations, a projection matrix $\mathbf{R} \in \mathbb{R}^{n \times d}$ is directly applied to the spectral graph transformation function.

Although we reduce the number of parameters by random projection, the spectral graph transformation still requires an explicit matrix product on the adjacency matrices. This seriously affects the efficiency of our method, since the complexity of matrix product on the adjacency matrices is extremely high. To further reduce the computation cost, we leverage the associative property of matrix product to reduce the time complexity as in [34]:

$$\begin{aligned} \mathbf{Z}^i &= \left(\mathbb{A}^i \cdot (\mathbf{X} \cdot \mathbf{R})\right) = \Big(\underbrace{\mathbb{A} \ldots \left(\mathbb{A}}_{i} \cdot (\mathbf{X} \cdot \mathbf{R})\right)\Big) \\ \mathbf{Z} &= \sum_{i=1}^{K} \alpha_i \mathbf{Z}^i. \end{aligned} \tag{11}$$

This operation significantly reduces the time complexity from $O(n^3 \cdot K + n^2 \cdot m + n \cdot m \cdot d)$ to $O(K \cdot e \cdot n + n \cdot m \cdot d)$, where e is the number of edges in $\mathbb{A}$.

Algorithm 1 shows the pseudo-code of our proposed framework. It is worth noting that we replace the explicit adjacency matrix product in the spectral graph transformation by Eq. (11) in random projection, as shown in lines 4-8, which significantly improves the efficiency of the algorithm.

Algorithm 1 The Learning Process of FAME

Input: Input AMHEN $\mathcal{G}$, node feature matrix $\mathbf{X}$, embedding dimension d, matrix polynomial order K, weights $\alpha_1, \alpha_2, \ldots, \alpha_K, \beta_1, \beta_2, \ldots, \beta_{|\mathcal{R}|}$
Output: Embedding results $\mathbf{Z}$

1: Decouple the attributed multiplex heterogeneous network into homogeneous networks and bipartite networks to obtain the adjacency matrices $\{\mathbf{A}_r | r = 1, 2, \ldots, |\mathcal{R}|\}$
2: Calculate $\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r$
3: Normalization $\mathbb{A} = \mathbf{D}^{-1}\mathbb{A}$
4: Generate random projection matrix $\mathbf{R}$ by Eq.(9)
5: $\mathbf{Z}^1 \leftarrow \mathbb{A} \cdot \mathbf{X} \cdot \mathbf{R}$
6: **for** $i = 2$ to K **do**
7: Calculate $\mathbf{Z}^i \leftarrow \mathbb{A} \cdot \mathbf{Z}^{i-1}$
8: **end for**
9: $\mathbf{Z} = \alpha_1 \mathbf{Z}^1 + \cdots + \alpha_K \mathbf{Z}^K$

Table 2: Statistics of Datasets (n-type: node type, e-type: edge type, feat.: features, and Mult.: Multiplex edge type)

Dataset	#nodes	#edges	#n-type	#e-type	#feat.	Mult.
Alibaba-S	21,318	41,676	2	4	19	✓
Alibaba	1,310,391	61,354,443	2	4	19	✓
Amazon	10,166	148,865	1	2	1,156	✓
AMiner	58,068	118,939	3	3	4	×
IMDB	12,772	18,644	3	2	1,256	×

5 EXPERIMENT

5.1 Datasets

We conduct extensive experiments on four public real-world datasets. Alibaba dataset[1] has two node types (user and item) and includes four types of edges between users and items. We use the category of item as the class label in node classification. Amazon dataset[2] includes product metadata of Electronics category and co-viewing, co-purchasing links between products. The product attributes include the price, sales-rank, brand, category, etc. AMiner dataset[3] contains three types of nodes: author, paper and conference. The domain of papers is considered as the class label. IMDB dataset[4] contains three types of nodes, *i.e.*, movie, actor and director, and labels are genres of movies. Node features are given as bag-of-words representations of plots. There is no multiplex edge between each pair of node types in the Aminer and IMDB datasets, so they are not multiplex networks. Since some of the baselines cannot scale to the whole graph on Alibaba dataset, we evaluate the model performance on a sampled dataset from Alibaba, denoted by Alibaba-S. The statistics of these four datasets are summarized in Table 2.

[1] https://tianchi.aliyun.com/competition/entrance/231719/information/
[2] http://jmcauley.ucsd.edu/data/amazon/
[3] https://github.com/librahu/
[4] https://github.com/seongjunyun/Graph_Transformer_Networks

5.2 Baselines

We compare our FAME against the following baselines:

- **node2vec** [6] - node2vec is a network embedding method which samples short biased random walks with the balance between DFS and BFS.
- **RandNE** [34] - RandNE is a Gaussian random projection approach to map the network into a low-dimensional embedding space while preserving the high-order proximities between nodes.
- **FastRP** [3] - FastRP is an extension of RandNE by using sparse random projection and normalizing node similarity matrix entries.
- **SGC** [27] - SGC is a simplified version of GCN, which only uses the product of high-order adjacency matrices and attribute matrix, without nonlinear transformation and training parameters.
- **PMNE** [16] - We denote their network aggregation, result aggregation, and co-analysis model as PMNE-n, PMNE-r, and PMNE-c, respectively.
- **GATNE** [2] - GATNE includes GATNE-T and GATNE-I. We use GATNE-I as our baseline method, which considers both the network structure and the node attributes, and then learns an inductive transformation function to obtain embeddings for all nodes.
- **HAN** [25] - HAN applies graph neural network and graph attention mechanism on multiplex network to learn node embedding.
- **GTN** [32] - GTN transforms a heterogeneous graph into multiple new meta-path graphs and then learns node embeddings via graph convolution network on the meta-path graphs.
- **DMGI** [17] - DMGI integrates node embeddings from different types of relations through the consensus of a regularization framework and an universal discriminator.

As RandNE, FastRP, node2vec and SGC can only deal with homogeneous network, we feed separate graphs with different edge types into them to obtain different node embeddings for each separated graph, then perform mean pooling to generate final node embedding.

In addition to the above baselines, we also design a variant **FAME**$_m$ of our model to verify the effectiveness of our proposed spectral graph transformation, which do not consider meta-path interactions across multi-relations and directly perform spectral graph transformation on the decoupled networks.

We also summarize the network types handled by the competitor methods in Table 3.

5.3 Experimental Setting

The purpose of running time comparison task is to evaluate the efficiency of methods, hence we use the entire network as the training set to obtain the embeddings of all nodes. For link prediction task, we hide a set of edges from the original graph and train on the remaining graph. Following [2], we create a validation/test set that contains 5%/10% randomly selected positive edges respectively with the equivalent number of randomly selected negative edges for each edge type. For the node classification task, we first learn the representation of each node and then perform accuracy evaluation. More specifically, we take 80% of the node embeddings as the training set, 10% as the validation set, and 10% as the test set, and then train a logistic regression classifier in the training set, evaluate on the nodes in the test set.

We set d to 200 for all the methods for a fair comparison. We use the source code provided by their authors for baselines. For all random walk based methods, we set walk length to 100 per node, window size to 10, the number of negative samples to 5. We set $p = 2$ and $q = 0.5$ for node2vec and set α_r and β_r to 1 for every edge type r on GATNE. For the PMNE model, we use the hyperparameters given by the original paper. For all deep learning methods (*e.g.*, GATNE, HAN, GTN, DMGI), we tune learning rate in $\{0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005\}$. We set the regularization parameter to 0.001, the number of attention head is set as 8, and the

Table 3: The network types handled by different methods (Heter.: Heterogeneity, Multi.: Multiplex edge type, Attr.: Attribute, Unsup.: Unsupervised, Auto.: Automatic meta-path).

	Heter.		Multi.	Attr.	Unsup.	Auto.
	Node	Edge				
node2vec	×	×	×	×	✓	×
RandNE	×	×	×	×	✓	×
FastRP	×	×	×	×	✓	×
SGC	×	×	×	✓	✓	×
PMNE	×	✓	✓	×	✓	×
DMGI	✓	✓	✓	✓	✓	×
HAN	✓	✓	×	✓	×	×
GTN	✓	✓	✓	✓	×	✓
GATNE	✓	✓	✓	✓	✓	×
FAME	✓	✓	✓	✓	✓	✓

Table 4: Runtime comparison of all methods (Second)

Method	Alibaba-S	Amazon	AMiner	IMDB
node2vec	1031.25	428.68	3841.8	1272.75
RandNE	2.26	1.06	7.31	1.77
FastRP	2.78	1.54	5.95	1.93
SGC	0.192	0.446	0.74	0.224
PMNE-n	4026.25	788.04	12492.55	2481.55
PMNE-r	4389.25	793.03	12378.45	4917.8
PMNE-c	2016.3	376.74	7511.55	2048.15
GATNE	27152.02	51546.15	9h/epoch	57842.78
DMGI	1510.81	4401.83	50026.69	18556.37
HAN	4226.95	/	87105.55	70510
GTN	21166.83	/	OOM	4287.20
FAME$_m$	0.96	1.43	3.29	1.29
FAME	0.51	0.59	1.45	0.28
Speedup*	**53239**×	**87366**×	**22345**×/epoch	**206581**×

* Speedup over GATNE. OOM: Out Of Memory (80 GB). DMGI runs out of memory on the entire AMiner, and the result is only run on a single type of paper node.

Table 5: Link prediction performance comparison of different methods on four datasets

	Alibaba-S			Amazon			AMiner			IMDB		
	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1
node2vec	0.614	0.580	0.593	0.946	0.944	0.880	0.594	0.663	0.602	0.479	0.568	0.474
RandNE	0.877	0.888	0.826	0.950	0.941	0.903	0.607	0.630	0.608	0.901	0.933	0.839
FastRP	0.927	0.900	0.926	0.954	0.945	0.893	0.620	0.634	0.606	0.869	0.893	0.811
SGC	0.686	0.708	0.623	0.791	0.802	0.760	0.589	0.585	0.567	0.826	0.889	0.769
PMNE-n	0.966	0.973	0.891	0.956	0.945	0.893	0.651	0.669	0.677	0.674	0.683	0.646
PMNE-r	0.859	0.915	0.824	0.884	0.890	0.796	0.615	0.653	0.662	0.646	0.646	0.613
PMNE-c	0.597	0.591	0.664	0.934	0.934	0.868	0.613	0.635	0.657	0.651	0.634	0.630
DMGI	0.857	0.781	0.784	0.905	0.878	0.847	OOM	OOM	OOM	0.926	0.935	0.873
GATNE	0.981	0.986	0.952	**0.963**	0.948	**0.914**	OOT	OOT	OOT	0.872	0.878	0.791
FAME$_m$	0.980	0.983	0.947	0.939	0.937	0.887	0.610	0.639	0.677	0.935	0.951	0.894
FAME	**0.993**	**0.996**	**0.979**	0.959	**0.950**	0.900	**0.687**	**0.747**	**0.726**	**0.944**	**0.959**	**0.897**

OOT: Out Of Time (36 hours). OOM: Out Of Memory (80 GB); DMGI runs out of memory on the entire AMiner.

Table 6: Link prediction results on entire Alibaba dataset

	Alibaba		
	ROC-AUC	PR-AUC	F1
RandNE	0.916	0.934	0.835
FastRP	0.929	0.934	0.921
SGC	0.776	0.767	0.747
FAME	**0.949**	**0.951**	**0.932**

dropout ratio of attention is 0.6. For GTN, we use the sparse version of their released source code and set GT layers to 3 for all datasets. For DMGI, we set the self-connection weight $w = 3$ and tune α, β, γ in $\{0.0001, 0.001, 0.01, 0.1\}$. In the experiment, we set matrix polynomial order K to 3 and sparsity s to $\sqrt{m}$ for our FAME. For the weights $\alpha_1, \ldots, \alpha_K, \beta_1, \ldots, \beta_{|\mathcal{R}|}$, we perform optuna[5], a Bayesian hyperparameter optimization method, for 100 rounds on validation set; the search ranges for weights are set to $[10^{-6}, 10^6]$. We evaluate the efficiency of all methods on a machine with Intel Xeon E5-2660 (2.2GHz) CPU and 80GB memory.

5.4 Model Efficiency Study

We first evaluate the superior efficiency of our method by reporting the running time of FAME and all baselines on all datasets in Table 4. As we can see, FAME is significantly faster than all heterogeneous network embedding baselines by orders of magnitude on all datasets. More specifically, FAME achieves at least 53,239× speedup over state-of-the-art AMHEN embedding method GATNE. GTN that learns meta-paths automatically is also slow on the multiplex networks (*e.g.*, Alibaba-S). From the results, we can observe that FAME is 41,503 times and 15,311 times faster than the sparse version of GTN on Alibaba-S and IMDB, respectively. Due to the high computational and space cost of GTN method, its sparse version still has the out of memory issue on AMiner data. Even for homogeneous network embedding node2vec, FAME is more than three orders of magnitude faster. Only random projection-based methods (*i.e.*, RandNE and FastRP) and SGC achieve comparable running time, but RandNE and FastRP are also slightly slower than FAME. This is because RandNE and FastRP need to perform multiple embedding for the decoupled sub-networks. However, in the experiments below, we will show that the quality of embeddings produced by FAME is significantly better than those of RandNE, FastRP and SGC.

[5]https://github.com/pfnet/optuna

5.5 Link Prediction

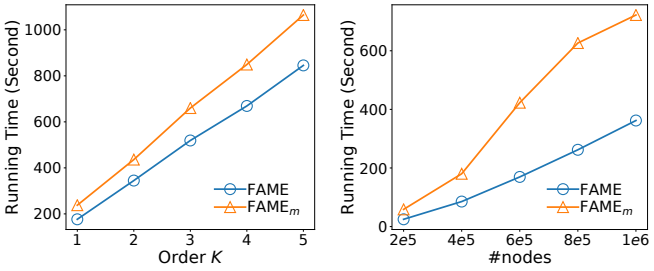
We evaluate the model performance by comparing FAME with nine baselines on link prediction task. Because both HAN and GTN require node labels for training, thus they cannot be applied to the link prediction for nodes of different types. DMGI does not conduct link prediction experiment and explain how to do link prediction between different types of nodes in their paper. Thus, we overlook the node type in the experiment. The results are shown in Table 5. GATNE cannot run on AMiner dataset due to the out of time issue, and DMGI runs out of memory on AMiner dataset. We can see that FAME significantly outperforms all baselines on four networks. More specifically, our FAME achieves comparable results to state-of-the-art GATNE on Amazon dataset. This is because Amazon dataset only has two types of links, co-viewing and co-purchasing, between one type of node (product). Such a network structure is relatively simple, and even manually specified meta path can capture most features in the network. At this time, the network features obtained by the manually specified meta-paths are basically equivalent to the network features obtained by all the meta-paths that we automatically obtain. However, FAME obtains better performance than state-of-the-art GATNE on other two more complex networks as FAME captures effective multi-relational topological structures automatically.

Notably, only RandNE, FastRP, SGC and our FAME can run on the entire Alibaba dataset, with 80G memory and 36 hours limit. As shown in Table 6, FAME achieves state-of-the-art performance on Alibaba dataset compared with best results from previous state-of-the-art methods. Additionally, FAME performs better than the variation FAME$_m$ on all datasets, suggesting that our proposed spectral

Table 7: Node classification performance comparison

	Unsupervised	AMiner		Alibaba-S		IMDB	
		Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
RandNE	✓	0.641 (0.0074)	0.672 (0.0064)	0.319 (0.0170)	0.358 (0.0093)	0.373 (0.0143)	0.392 (0.0185)
FastRP	✓	0.650 (0.0086)	0.690 (0.0074)	0.301 (0.0180)	0.392 (0.0119)	0.363 (0.0236)	0.381(0.0140)
SGC	✓	0.516 (0.0047)	0.587 (0.0157)	0.286 (0.0231)	0.361 (0.0175)	0.489 (0.0106)	0.563 (0.0133)
HAN	×	0.690 (0.0149)	0.726 (0.0086)	0.275 (0.0327)	0.392 (0.0081)	0.552 (0.0112)	0.568 (0.0078)
GTN	×	OOM	OOM	0.255 (0.0420)	0.392 (0.0071)	**0.615** (0.0108)	**0.616** (0.0093)
GATNE	✓	OOT	OOT	0.291 (0.0086)	0.390 (0.0014)	0.169 (0.0132)	0.333 (0.0005)
DMGI	✓	0.473 (0.0155)	0.626 (0.0093)	0.220 (0.0214)	0.392 (0.0026)	0.548 (0.0190)	0.544 (0.0189)
FAME$_m$	✓	0.575 (0.0064)	0.621 (0.0076)	0.306 (0.0119)	0.333 (0.0124)	0.505 (0.0176)	0.513 (0.0092)
FAME	✓	**0.722** (0.0114)	**0.727** (0.0091)	**0.323** (0.0154)	**0.393** (0.0060)	0.593 (0.0135)	0.594 (0.0143)

OOM: Out Of Memory (80 GB), OOT: Out Of Time (36 hours). The standard deviations are reported in the parentheses.



(a) Runtime *w.r.t.* order K (b) Runtime *w.r.t.* #nodes

Figure 3: Runtime of FAME *w.r.t.* order K and #nodes.

transformation effectively captures the heterogeneous meta-paths and works better in multiplex heterogeneous networks.

5.6 Node Classification

We further evaluate the effectiveness of representations on the supervised task of node classification with state-of-the-art methods. The results are shown in Table 7. Based on these results, we have the following observations: (1) FAME is significantly better than state-of-the-art unsupervised embedding methods (*i.e.*, GATNE and DMGI), while even achieves comparable results with sate-of-the-art supervised methods (*i.e.*, HAN and GTN). (2) GTN cannot run on AMiner due to memory constraint as it needs to calculate the product of the adjacency matrices explicitly. Furthermore, GATNE cannot obtain the embeddings for AMiner network within 36 hours, and it cannot well learn the features that distinguish node categories. (3) Meta-path information is more important in node classification for multi-type node networks (*e.g.*, FAME and FAME$_m$ on AMiner vs. FAME and FAME$_m$ on Alibaba-S).

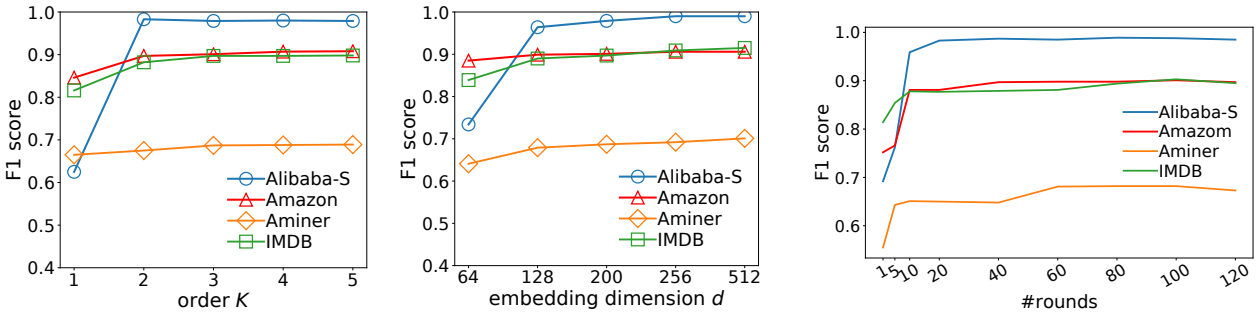
5.7 Scalability Analysis

We now investigate the scalability of the proposed FAME on large-scale AMHEN. Figure 3 shows the speed of our proposed methods *w.r.t.* order K and the number of nodes sampled from the Alibaba dataset. As we can see, FAME is quite scalable in terms of order K and the number of nodes in Alibaba network. FAME not only performs better than FAME$_m$ in effectiveness, but also improves the embedding efficiency by performing spectral transformation for all sub-networks uniformly. More specifically, FAME only takes less than 10 minutes to obtain the embeddings for all nodes in the entire Alibaba on a single server. In summary, apart from the promising performance, FAME is also scalable enough to be adopted in practical scenarios without resorting to large-scale distributed computing platform.

5.8 Parameter Sensitivity

We now investigate the sensitivity of our proposed method with respect to the important parameters, including order k, embedding dimension d, and the number of rounds for tuning weights. To clearly show the influence of these parameters, we report F1 score on link prediction task with different parameter settings on four datasets. Figure 4 shows the experimental results.

As shown in Figure 4(a), at first, the performance of FAME increases as order K increases, and then remains stable when $K \geq 3$. This is mainly because 1-length and 2-length meta-path interactions can significantly improve the effect of link prediction, while longer meta-paths do not lead to significant performance improvements. Then, we examine the effectiveness of FAME when the embedding dimension d varying from 64 to 512, and report the average F1 score on all datasets. From the results in Figure 4(b), we can see that the performance of FAME gradually rises and then remains relatively stable as the dimension increases, because our method uses random projection to reduce the dimension. The increase of the embedding dimension d will definitely reduce the distance error in the embedding space. When it increases to a certain degree, the error reaches a certain boundary and tends to be stable. Figure 4(c) illustrates the performance of our FAME with respect to the number of rounds in tuning weights. We can conclude that our FAME can efficiently determine the weight parameters, and it has achieved stable performance within a few dozen rounds. In addition, the high efficiency of our method verified in the above experiments also makes the parameter tuning process more efficient, which greatly reflects the advantages of our method.



(a) F1 score *w.r.t.* order K (b) F1 score *w.r.t.* embedding dimension d (c) F1 score *w.r.t.* #rounds

Figure 4: Parameter sensitivity of proposed method *w.r.t.* order K, dimension d, and #rounds.

6 CONCLUSION

We present an efficient yet effective unsupervised method FAME for embedding large-scale AMHENs. FAME first performs an spectral graph transformation to capture different proximities between nodes including complete meta-paths across multi-relations and high-order topological structures. Then FAME tactfully incorporates the proposed spectral graph transformation and node attribute features into the sparse random projection architecture for fast and effective node representation learning. Experiments show that FAME significantly drives the embedding time down by up to 6 orders of magnitude over state-of-the-art baselines. Additionally, FAME achieves better performance compared to state-of-the-art methods on both link prediction and node classification tasks.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China under grant Nos. 61773331 and 61403328, the Fundamental Research Funds for the Central Universities under grant No. 201964022, and the Graduate Innovation Foundation of Yantai University under grant No. YDZD2021.

REFERENCES

[1] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.
[2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *KDD*. 1358–1368.
[3] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and Accurate Network Embeddings via Very Sparse Random Projection. In *CIKM*. 399–408.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*. 3837–3845.
[5] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*. 135–144.
[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.
[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[8] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *KDD*. 120–129.
[9] Chao Huang, Baoxu Shi, Xuchao Zhang, Xian Wu, et al. 2019. Similarity-aware network embedding with self-paced learning. In *CIKM*. 2113–2116.
[10] Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V Chawla. 2019. Online purchase prediction via multi-scale modeling of behavior dynamics. In *KDD*. 2613–2622.
[11] William B Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics* 26, 189-206 (1984), 1.
[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[13] Jérôme Kunegis and Andreas Lommatzsch. 2009. Learning spectral graph transformations for link prediction. In *ICML*. 561–568.
[14] Ping Li, Trevor J Hastie, and Kenneth W Church. 2006. Very sparse random projections. In *KDD*. ACM, 287–296.
[15] Ruirui Li, Xian Wu, Xian Wu, and Wei Wang. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *WWW*. 2472–2478.
[16] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *ICDMW*. IEEE, 134–141.
[17] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*. 5371–5378.
[18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, et al. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *WWW*. 1509–1520.
[19] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *TKDE* 31, 2 (2018), 357–370.
[20] Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton Van Den Hengel. 2012. Is margin preserved after random projection?. In *ICML*. 643–650.
[21] Yiwei Sun, Suhang Wang, Tsung-Yu Hsieh, Xianfeng Tang, and Vasant G. Honavar. 2019. MEGAN: A Generative Adversarial Network for Multi-View Network Embedding. (2019), 3527–3533.
[22] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[23] Xianfeng Tang, Boqing Gong, Yanwei Yu, Huaxiu Yao, Yandong Li, Haiyong Xie, and Xiaoyu Wang. 2019. Joint modeling of dense and incomplete trajectories for citywide traffic volume inference. In *WWW*. 1806–1817.
[24] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[25] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.
[26] Yueyang Wang, Ziheng Duan, Binbing Liao, Fei Wu, and Yueting Zhuang. 2019. Heterogeneous Attributed Network Embedding with Graph Convolutional Networks. In *AAAI*. 10061–10062.
[27] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, et al. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.
[28] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex Behavioral Relation Learning for Recommendation via Memory Augmented Transformer Network. In *SIGIR*. 2397–2406.
[29] Yanwei Yu, Xianfeng Tang, Huaxiu Yao, Xiuwen Yi, and Zhenhui Li. 2019. Citywide Traffic Volume Inference with Surveillance Camera Records. *IEEE Transactions on Big Data* (2019).
[30] Yanwei Yu, Hongjian Wang, and Zhenhui Li. 2018. Inferring mobility relationship via graph embedding. *IMWUT* 2, 3 (2018), 1–21.
[31] Yanwei Yu, Huaxiu Yao, Hongjian Wang, Xianfeng Tang, and Zhenhui Li. 2018. Representation learning for large-scale dynamic networks. In *DASFAA*. Springer, 526–541.
[32] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.
[33] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *KDD*. 793–803.
[34] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-Scale Network Embedding with Iterative Random Projection. In *ICDM*. 787–796.