Joint Embedding of Meta-Path and Meta-Graph for Heterogeneous Information Networks

Lichao Sun[†], Lifang He[‡*], Zhipeng Huang[§], Bokai Cao[¶], Congying Xia[†], Xiaokai Wei[¶] and Philip S. Yu[†]
[†]*University of Illinois at Chicago, Chicago, IL* [‡]*Cornell University, New York City, NY*
[§]*The University of Hong Kong, China* [¶]*Facebook, Menlo Park, CA*
Email: {lsun29, cxia8, psyu}@uic.edu, {lifanghescut,caobokai.why,weixiaokai}@gmail.com, zphuang@cs.hku.hk

***Abstract*—Meta-graph is currently the most powerful tool for similarity search on heterogeneous information networks, where a meta-graph is a composition of meta-paths that captures the complex structural information. However, current relevance computing based on meta-graph only considers the complex structural information, but ignores its embedded meta-paths information. To address this problem, we propose MEta-GrAph-based network embedding models, called MEGA and MEGA++, respectively. The MEGA model uses normalized relevance or similarity measures that are derived from a meta-graph and its embedded meta-paths between nodes simultaneously, and then leverages tensor decomposition method to perform node embedding. The MEGA++ further facilitates the use of coupled tensor-matrix decomposition method to obtain a joint embedding for nodes, which simultaneously considers the hidden relations of all meta information of a meta-graph. Extensive experiments on two real datasets demonstrate that MEGA and MEGA++ are more effective than state-of-the-art approaches.**

***Keywords*-node embedding, heterogeneous information networks, tensor learning, meta graph**

I. Introduction

Many information retrieval and mining tasks such as node classification [7], clustering [30], link prediction [28], and information diffusion [25] become time-consuming in large-scale networks. This motivates researchers to develop network embedding techniques which aim to learn a distributed representation vector for each node in a network. An effective network embedding should preserve the similarity between nodes in order to reconstruct the original network.

The word2vec [21] idea has inspired many studies for network representation learning, most of which are in the context of homogeneous information networks, such as DeepWalk [22], LINE [31], and node2vec [9]. A homogeneous information network is a simple structural network, where all nodes and links are considered to belong to a single class.

However, in practice, there are usually multiple types of nodes (*e.g.*, authors and papers in DBLP) and links (*e.g.*, cite and publish) that compose a heterogeneous information network (HIN). To measure the similarity between nodes in HINs, many customized similarity or relevance measures

*Corresponding author.

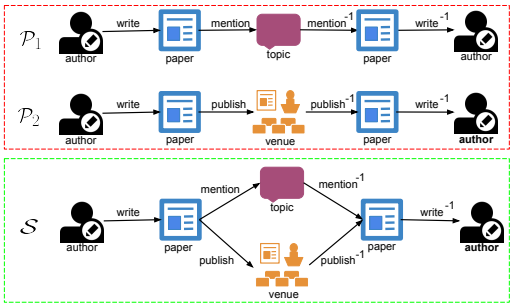

Figure 1: An example of meta-paths $\mathcal{P}_1$, $\mathcal{P}_2$, and meta-graph $\mathcal{S}$ of the bibliographic data.

based on meta-paths have been proposed in recent years [16], [29]. For example, a meta-path $author \rightarrow paper \rightarrow venue \rightarrow paper \rightarrow author$ (denoted as $APVPA$) indicates two authors having their publications in the same venue. Comparing to meta-path-based relevance measures utilizing only simple structural information, meta-graph [15] is recently proposed to capture complex structural information in HINs. In short, meta-graph is a special directed acyclic graph (DAG) which contains at least two embedded meta-paths, such as a DAG containing $APVPA$ and $APTPA$ as shown in Figure 1, where T is the topic of a paper.

Meta-graph is an effective tool to calculate the relevance score between nodes in HINs, where a higher score indicates that there are more meta-graph instances between two nodes, *i.e.*, a closer relationship. How to explore meta-graphs for representation learning in HINs is still an open question. An intuitive idea for meta-graph-based representation learning is to learn the node embedding by leveraging multiple meta-graphs between nodes in HINs. However, existing meta-graph-based relevance measures only utilize the strong relations as defined by the meta-graphs themselves, and they usually ignore the weak relations as indicated by their embedded meta-paths. To address this problem, we propose to learn the node embedding by leveraging both meta-graph and its embedded meta-paths for similarity search. An effective representation learning based on a single meta-graph should contain both strong and weak relations embedded

in this meta-graph. In addition, we explore a novel meta-graph-based similarity measure to compute relevance scores that can better capture the strong relations between nodes in HINs.

In summary, there are three-fold contributions of this paper: 1) We are the first to propose the meta-graph-based node embedding method in HINs. Specifically, we develop two kinds of node embedding methods based on meta-graph, named MEGA and MEGA++ respectively. 2) We introduce GraphSim which is an effective meta-graph-based similarity measure with best performance comparing to previous meta-graph-based similarity measures, such as StructCount and SCSE. 3) Our approaches show the best performance comparing to other competing methods on two real-world datasets.

II. Preliminary and Problem Formulation

In this section, we first introduce some related concepts and notations from multilinear algebra. Then, we review some concepts and approaches involved in HIN analysis including meta-graph and relevance measure. Last part, we formulate the problem of node embedding in HINs.

A. Multilinear Algebra

The basic mathematical object of multilinear algebra is the tensor, a higher order generalization of vectors (first order tensors) and matrices (second order tensors) to multiple indices. The order of a tensor is the number of dimensions, also known as modes or ways. An N-th order tensor is represented as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where I_n is the cardinality of its n-th mode, $n \in \{1, 2, \cdots, N\}$. An element of a vector $\mathbf{x}$, a matrix $\mathbf{X}$, or a tensor $\mathcal{X}$ is denoted by x_i, $x_{i,j}$, $x_{i,j,k}$, etc., depending on the number of modes. All vectors are column vectors unless otherwise specified. For an arbitrary matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, its i-th row and j-th column vector are denoted by $\mathbf{x}^i$ and $\mathbf{x}_j$, respectively.

Definitions of outer product, partial symmetric tensor, mode-n matricization, and CP factorization are given below, which will be applied to present our approach.

Definition 1: (Outer Product) The outer product of N vectors $\mathbf{x}^{(n)} \in \mathbb{R}^{I_n}$ for $n \in [1:N]$ is an N-th order tensor and defined element-wise by $\left(\mathbf{x}^{(1)} \circ \cdots \circ \mathbf{x}^{(N)}\right)_{i_1, \cdots, i_N} = x^{(1)}_{i_1} \cdots x^{(N)}_{i_N}$ for all values of the indices.

Definition 2: (Partial Symmetric Tensor) An N-th order tensor is a rank-one partial symmetric tensor if it is partial symmetric on modes $i_1, ..., i_j \in 1, ..., N$, and can be written as the tensor product of N vectors, i.e.,

$$\mathcal{X} = x^{(1)} \circ \cdots \circ \mathbf{x}^{(N)} \tag{1}$$

where $x^{(i_1)} = \cdots = x^{(i_j)}$.

Definition 3: (Mode-n Matricization) The mode-n matricization or unfolding of an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathbf{X}_{(n)}$ and is of size $I_n \times J_n$, where $J_n = \Pi_{m=1, m \neq n}^{N} I_m$.

Definition 4: (CP Factorization) For a general tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, its CANDECOMP/PARAFAC (CP) factorization is

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{x}^{(1)}_r \circ \cdots \circ \mathbf{x}^{(N)}_r = [\![\mathbf{X}^{(1)}, ..., \mathbf{X}^{(N)}]\!] \tag{2}$$

where for $n \in [1:N]$, $\mathbf{X}^{(n)} = [\mathbf{x}^{(n)}_1, ..., \mathbf{x}^{(n)}_R]$ are factor matrices of size $I_n \times R$, R is the number of factors, and $[\![\cdot]\!]$ is used for shorthand.

To obtain the CP factorization $[\![\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}]\!]$, the objective is to minimize the following estimation error:

$$\mathcal{L} = \min_{\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}} \|\mathcal{X} - [\![\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}]\!]\|^2_F \tag{3}$$

However, $\mathcal{L}$ is not jointly convex w.r.t. $\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}$. A widely used optimization technique is the Alternating Least Squares (ALS) algorithm, which alternatively minimize $\mathcal{L}$ for each variable while fixing the other, that is,

$$\mathbf{X}^{(n)} \leftarrow \arg\min_{\mathbf{X}^{(n)}} \|\mathbf{X}_{(n)} - \mathbf{X}^{(n)} (\odot^{N}_{i \neq n} \mathbf{X}^{(i)})^{\mathrm{T}}\|^2_F \tag{4}$$

where $\odot^{N}_{i \neq n} \mathbf{X}^{(i)} = \mathbf{X}^{(N)} \odot \cdots \mathbf{X}^{(n-1)} \odot \mathbf{X}^{(n+1)} \cdots \odot \mathbf{X}^{(1)}$.

B. Meta Graph

Definition 5: (Meta-Graph [15]) A meta-graph S is a directed acyclic graph (DAG) defined on a HIN schema $T_G = (\mathcal{O}, \mathcal{R})$. A meta-graph S contains a single source node n_s with 0 in degree and a single target node n_t with 0 out degree. Mathematically, a meta-graph $S = (A, B, n_s, n_t)$, where A is a set of nodes, B is a set of edges, n_s is the of source node, and n_t is the target node,.

Since a meta-graph only has one source node and one target node, not all sub-graphs of HINs can be meta-graph.

Definition 6: (Meta-graph-based Relevance Measure) Given a HIN $G = (V, E)$ and a meta-graph $\mathcal{G}$, the similarity of any two nodes $v_s, v_t \in V$ with respect to $\mathcal{G}$ is defined as:

$$s = \sum_{g_{v_s \to v_t} \in \mathcal{G}} s(v_s, v_t \mid g_{v_s \to v_t}) \tag{5}$$

where $g_{v_s \to v_t}$ is a meta-graph instance of $\mathcal{G}$, and $s(v_s, v_t \mid g_{v_s \to v_t})$ is the relevance score between v_s and v_t, which will be determined by the number of meta-graph instances connecting them.

Prior works provide different meta-graph-based relevance measures, such as StructCount, SCSE and BSCSE [15].

C. Problem Formulation

We study the problem of meta-graph-based node embedding in the HIN. Given a HIN $G = (V, E)$, we have two goals in this study. First, we want to explore a customized meta-graph-based relevance measure which can more efficiently capture the complex structural information. Second, we aim at finding an effective node embedding that

can better preserve the closeness between nodes in a HIN based on a meta-graph and its embeded meta-paths analysis. Specifically, we integrate all the similarity information of a meta-graph and its embedded meta-paths into a symmetric matrix and a partial symmetric tensor, and perform multi-linear analysis of the coupled partial symmetric tensor and symmetric matrix to find the node embedding.

III. METHODS

In this section, we will introduce a brand new similarity measure, and the embedding techniques of MEGA++. First, we will introduce a meta-graph-based similarity measure named GraphSim. Then, we proposed a coupled tensor-matrix decomposition to obtain a joint embedding for nodes in HINs.

A. GraphSim: A Normalized version of StructCount

First, we want to propose a new meta-graph-based similarity measure called GraphSim. In previous work, Huang et al. [15] proposed three meta-graph-based similarity measures: StructCount, SCSE, and BSCSE which is a mixed measure based on previous two measures. GraphSim can be viewed as a normalized version of StructCount.

StructCount [15] is a straightforward meta-graph-based similarity measure in HIN, which counts the number of meta-graph instances in the graph G with an n_s as source and an n_t as target object.

Definition 7: (GraphSim) A meta-graph-based similarity measure. Given a symmetric meta-graph $\mathcal{G}$, GraphSim between two nodes $v_t, v_s \in V$ is defined as:

$$s(v_t, v_s) = \frac{2 \times |\{g_{v_t \to v_s} | g_{v_t \to v_s \in \mathcal{G}}\}|}{|\{g_{v_t \to v_t} | g_{v_t \to v_t \in \mathcal{G}}\}| + |\{g_{v_s \to v_s} | g_{v_s \to v_s \in \mathcal{G}}\}|} \tag{6}$$

where $g_{v_t \to v_s}$is a meta-graph instance between v_t and v_s, $g_{v_t \to v_t}$ is that between v_t and v_t, and $g_{v_s \to v_s}$ is that between v_s and v_s.

Comparing to StructCount, GraphSim is normalized version of StructCount. $s_{GraphSim}(v_t, v_s)$ is determined by two parts: First, the number of meta-graph instance between $v_t, v_s \in V$ by following $\mathcal{G}$; Second, the balance of their visibility, where the visibility is defined as the number of meta-graph instances between themselves. Normalized relevance score can present better relation between different nodes. For example, an author A_1 published all his four papers with A_2. A_3 published five papers with A_2, and A_3 published other five papers with other authors. Without normalized process, the relation between A_2 and A_3 is closer than A_2 and A_1. However, for common sense, we should agree A_2 and A_1 have closer relation, which indicates GraphSim is a better measure.

Comparing to three measures in [15], $s(v_t, v_s)$ of GraphSim is between 0 and 1 like SCSE. However, SCSE measures the random walk probability that v_s expands a meta-graph instance to v_t. In our application, we find GraphSim shows better performance than all those three meta-graph measures [15].

B. MEGA++: Node Embedding by CTMD

In this section, we show how to jointly consider similarity information of a meta-graph and its embedded meta-paths to learn a node embedding. The basic idea is the integration of similarity matrices and coupled embedding by joint factorization. Specifically, we first compute a meta-graph similarity matrix using the proposed GraphSim, denoted as $\mathbf{Y} \in \mathbb{R}^{M \times M}$, and for each meta-path $\mathcal{P}$, compute an embedded meta-path similarity matrix using the PathSim, denoted as $\mathbf{M} \in \mathbb{R}^{M \times M}$. Next, we concatenate the embedded meta-path similarity matrices of different embedded meta-paths to form a third-order tensor comprising three modes: nodes, nodes, and paths, denoted as $\mathcal{X} = [\mathbf{M}_1, \cdots, \mathbf{M}_N] \in \mathbb{R}^{M \times M \times N}$. Then, we introduce a novel coupled tensor-matrix decomposition (CTMD) method to find common latent features between $\mathcal{X}$ and $\mathbf{Y}$. Last, we use the latent features to measure the similarity between different nodes in the HIN.

In the following, we detail the CTMD method, which can be seen as a special case of the coupled tensor-matrix decomposition [1] with input partial symmetric tensor $\mathcal{X}$ and symmetric matrix $\mathbf{Y}$. Notice that since similarity matrix is symmetric, the resulting $\mathcal{X}$ is a partial symmetric tensor, and $\mathbf{Y}$ is a symmetric matrix.

Tensors (including matrix) provide a natural and efficient representation for a meta-graph data, but there is no guarantee that such representation will be good for subsequent learning, since learning will only be successful if the regularities that underlie the data can be discerned by the model. Tensor factorization is a powerful tool to analyze tensors. In previous work, it was found that CP factorization (which is a higher order generalization of SVD) is particularly effective to acknowledge the connections and find valuable features among tensor data [32]. Motivated by these observations, we exploit the benefits of CP and SVD factorizations to find an effective embedding in the sense of meta-path-based similarity tensor $\mathcal{X}$ and meta-graph similarity matrix $\mathbf{Y}$.

Based on above analysis, we design our CTMD objective function as below:

$$\min_{\mathbf{P}, \mathbf{T}} \ \|\mathcal{X} - [\![\mathbf{P}, \mathbf{P}, \mathbf{T}]\!]\|_F^2 + \alpha \|\mathbf{Y} - \mathbf{P}\mathbf{P}^{\mathrm{T}}\|_F^2 \tag{7}$$

where $\mathbf{P} \in \mathbb{R}^{M \times R}$ and $\mathbf{T} \in \mathbb{R}^{N \times R}$ are latent matrices. Specifically, $\mathbf{P}$ is jointly learned from both meta-graph and meta-path similarity information.

The objective function in Eq. (7) is non-convex with respect to $\mathbf{P}$ and $\mathbf{T}$ together, thus there is no closed-form solution. We introduce an effective iteration method to solve this problem. The main idea is to decouple the parameters using an Alternating Direction Method of Multipliers (ADMM) approach [4], by alternatively optimizing the objective with respect to one variable, while fixing others.

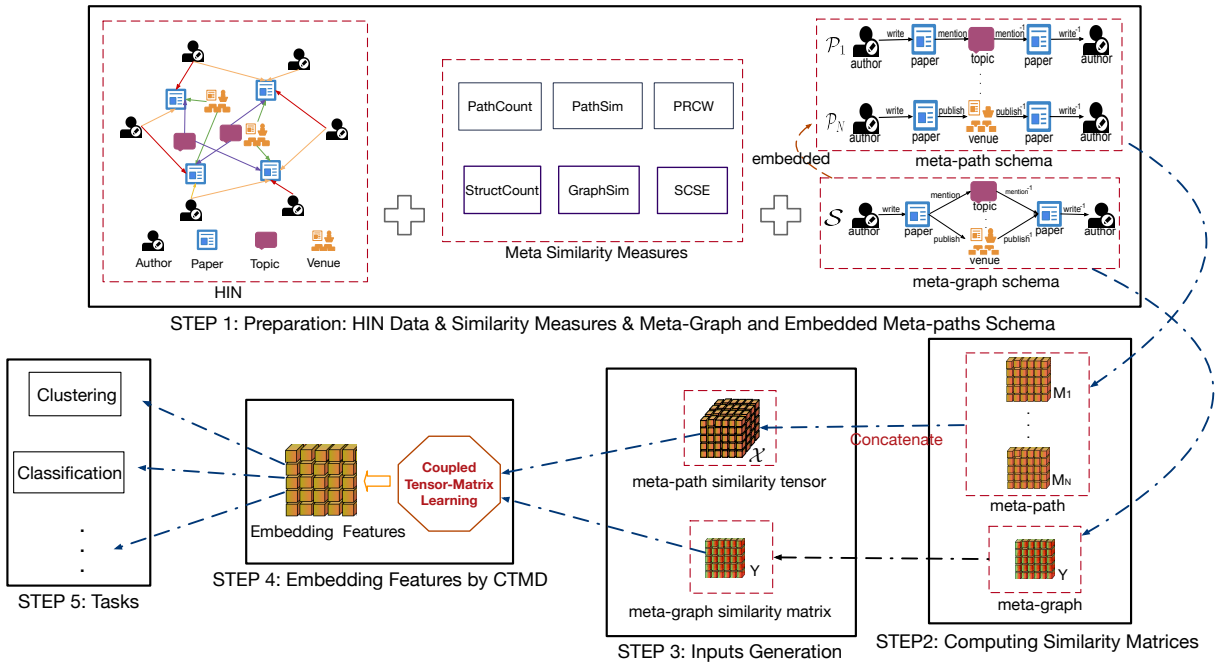

Figure 2: Framework of Meta-graph-based Node Embedding with Coupled Tensor-Matrix Decomposition

Update P: First, we optimize $\mathbf{P}$ while fixing $\mathbf{T}$. Notice that the objective function in Eq. (7) involves a fourth-order term with respect to $\mathbf{P}$ which is difficult to optimize directly. To obviate this problem, we use a variable substitution technique and minimize the following objective function

$$\begin{aligned} \min_{\mathbf{P},\mathbf{Q}} \ & \|\mathcal{X} - [\![\mathbf{P},\mathbf{Q},\mathbf{T}]\!]\|_F^2 + \alpha\|\mathbf{Y} - \mathbf{P}\mathbf{Q}^{\mathrm{T}}\|_F^2 \\ s.t. \ & \mathbf{P} = \mathbf{Q} \end{aligned} \tag{8}$$

where $\mathbf{Q} \in \mathbb{R}^{M\times R}$ is an auxiliary variable.

The augmented Lagrangian function of Eq. (8) is

$$\begin{aligned} \mathcal{L}(\mathbf{P},\mathbf{Q}) =& \|\mathcal{X} - [\![\mathbf{P},\mathbf{Q},\mathbf{T}]\!]\|_F^2 + \alpha\|\mathbf{Y} - \mathbf{P}\mathbf{Q}^{\mathrm{T}}\|_F^2 \\ &+ tr(\mathbf{U}^{\mathrm{T}}(\mathbf{P}-\mathbf{Q})) + \frac{\lambda}{2}\|\mathbf{P}-\mathbf{Q}\|_F^2 \end{aligned} \tag{9}$$

where $\mathbf{U} \in \mathbb{R}^{M\times R}$ is the Lagrange multiplier, and λ is the penalty parameter which can be adjusted efficiently according to [18].

To compute $\mathbf{P}$, Eq. (9) can be transformed as

$$\min_{\mathbf{P}} \|\mathcal{X}_{(1)} - \mathbf{P}\mathbf{F}^{\mathrm{T}}\|_F^2 + \alpha\|\mathbf{Y} - \mathbf{P}\mathbf{Q}^{\mathrm{T}}\|_F^2 + \frac{\lambda}{2}\|\mathbf{P} - \mathbf{Q} + \frac{1}{\lambda}\mathbf{U}\|_F^2 \tag{10}$$

where $\mathbf{X}_{(1)} \in \mathbb{R}^{M\times(MN)}$ is the mode-1 matricization of $\mathcal{X}$, and $\mathbf{F} = \mathbf{T} \odot \mathbf{Q} \in \mathbb{R}^{(MN)\times R}$.

By setting the derivative of Eq. (10) with respect to $\mathbf{P}$ to zero, we obtain the closed-form solution

$$\mathbf{P} = (2\mathbf{X}_{(1)}\mathbf{F} + 2\alpha\mathbf{Y}\mathbf{Q} + \lambda\mathbf{Q} - \mathbf{U})(2\mathbf{F}^{\mathrm{T}}\mathbf{F} + 2\alpha\mathbf{Q}^{\mathrm{T}}\mathbf{Q} + \lambda\mathbf{I})^{-1} \tag{11}$$

To efficiently compute $\mathbf{F}^{\mathrm{T}}\mathbf{F}$, we consider the following property of the Khatri-Rao product of two matrices

$$\mathbf{F}^{\mathrm{T}}\mathbf{F} = (\mathbf{T}\odot\mathbf{Q})^{\mathrm{T}}(\mathbf{T}\odot\mathbf{Q}) = \mathbf{T}^{\mathrm{T}}\mathbf{T} * \mathbf{Q}^{\mathrm{T}}\mathbf{Q} \tag{12}$$

Then the auxiliary matrix $\mathbf{Q}$ can be optimized successively in a similar way, and the solution is

$$\mathbf{Q} = (2\mathbf{X}_{(2)}\mathbf{G} + 2\alpha\mathbf{Y}^{\mathrm{T}}\mathbf{P} + \lambda\mathbf{P} + \mathbf{U})(2\mathbf{G}^{\mathrm{T}}\mathbf{G} + 2\alpha\mathbf{P}^{\mathrm{T}}\mathbf{P} + \lambda\mathbf{I})^{-1} \tag{13}$$

where $\mathbf{X}_{(2)}$ is the mode-2 matricization of $\mathcal{X}$, and $\mathbf{G} = \mathbf{T}\odot\mathbf{P}$.

Moreover, we optimize the Lagrange multiplier $\mathbf{U}$ using the gradient descent method by

$$\mathbf{U} \leftarrow \mathbf{U} + \lambda(\mathbf{P} - \mathbf{Q}) \tag{14}$$

Update T: Next, we optimize $\mathbf{T}$ while fixing $\mathbf{P}$ and $\mathbf{S}$. We need to optimize the following objective function

$$\min_{\mathbf{T}} \ \|\mathbf{X}_{(3)} - \mathbf{T}\mathbf{H}^{\mathrm{T}}\|_F^2 \tag{15}$$

where $\mathbf{X}_{(3)}$ is the mode-3 matricization of $\mathcal{X}$, and $\mathbf{H} = \mathbf{Q}\odot\mathbf{P}$.

By setting the derivative of Eq. (15) with respect to $\mathbf{T}$ to zero, we obtain the closed-form solution as

$$\mathbf{T} = (\mathbf{X}_{(3)}\mathbf{H})(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1} \tag{16}$$

The overall algorithm is summarized in Algorithm 1.

Algorithm 1 Coupled Tensor-Matrix Decomposition (CTMD)

Input: Meta-path similarity tensor $\mathcal{X}$, and meta-graph similarity matrix $\mathbf{Y}$
Output: Embedding matrix $\mathbf{P}$

: Set $\lambda_{max} = 10^6$, $\rho = 1.15$
: Initialize $\mathbf{P}, \mathbf{Q}, \mathbf{T} \sim \mathcal{N}(0,1), \mathbf{U} = \mathbf{0}, \lambda = 10^{-6}$
loop convergence
    : Update P by Eq. (11)
    : Update Q by Eq. (13)
    : Update T by Eq. (16)
    : Update μ by $\mu \leftarrow \min(\rho\mu, \mu_{max})$
end loop

C. Time Complexity

Each iteration in Algorithm 1 consists of simple matrix operations. Therefore, rough estimates of its computational complexity can be easily derived based on ADMM [17].

The estimate for the update of $\mathbf{P}$ according to Eq. (11) is as follows: $O(M^2NR)$ for the computation of the term $2\mathbf{X}_{(1)}\mathbf{F} + 2\alpha\mathbf{Y}\mathbf{Q} + \lambda\mathbf{Q} - \mathbf{U}$; $O((M+N)R^2)$ for the computation of the term $2\mathbf{F}^{\mathrm{T}}\mathbf{F} + 2\alpha\mathbf{Q}^{\mathrm{T}}\mathbf{Q} + \lambda\mathbf{I}$ due to Eq. (12) and $O(R^3)$ for its Cholesky decomposition; $O(MK^2)$ for the computation of the system solution that gives the updated value of $\mathbf{P}$. An analogous estimate can be derived for the update of $\mathbf{Q}$.

Overall, the updates of model parameters $\mathbf{P}$ and $\mathbf{Q}$, require $O(R^3+(M+N)R^2+M^2NR)$ arithmetic operations in total.

Algorithm 2 MEGA++

Input: An HIN G, a particular meta-graph g, a embedded meta-path p_i of a meta-graph g, and an empty array $\mathbf{P}_A$
Output: Embedding matrix $\mathbf{P}$

: Y = GraphSim(G, g)
loop $p_i \in g$
    : $\mathbf{P}_i$ = PathSim(G, p_i)
    : $\mathbf{P}_A$ = [$\mathbf{P}_A$; $\mathbf{P}_i$] store $\mathbf{P}_i$ into $\mathbf{P}_A$
end loop
: $\mathcal{X}$ = concatenate($\mathbf{P}_A$) $\mathcal{X}$ is a tensor of $\mathcal{S}$'s embedded meta-paths
: P = CTMD($\mathcal{X}$, Y)

IV. Experiments

In this section, we conduct extensive experiments in order to test the effectiveness of the proposed methods: GraphSim, MEGA and MEGA++. We first introduce two real-life datasets and a set of methods to be compared. Then, we evaluate the effectiveness of proposed methods on four data mining tasks: clustering, classification, parameter analysis and time analysis.

Table I: Statistics of Datasets

	$\lvert V\rvert$	$\lvert E\rvert$	Avg. degree	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{R}\rvert$
DBLP	15,649	51,377	6.57	4	4
MOVIE	25,643	40,173	3.13	5	4

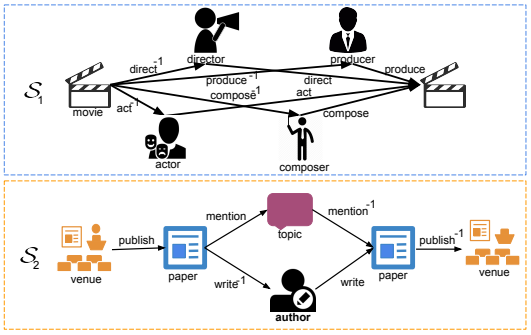

Figure 3: Guided meta-graphs: $\mathcal{S}_1$ is the guided meta-graph for M2M task, and $\mathcal{S}_2$ is for V2V task.

We use two real datasets (*e.g.* DBLP-4-Area and YAGO Movie) in the evaluation. Table I shows some statistics about them. DBLP-4-Area [29] is the subset of original DBLP, which contains 5,237 papers (P), 5,915 authors (A), 18 venues (V), 4,479 topics (T). The authors and venues are from 4 areas: *database*, *data mining*, *machine learning* and *information retrieval*. YAGO Movie is a subset of YAGO [15], which contains 7,332 movies (M), 10,789 actors (A), 1,741 directors (D), 3,392 producers (P) and 1,483 composers (C). The movies are divided into five genres: *action*, *horror*, *adventure*, *sci-fi* and *crime*. The guided meta-graphs are designed for three tasks as shown in the Figures 1 and 3.

The proposed methods are compared with meta-graph-based relevance measures (*e.g.* StructCount, SCSE, and BSCSE [15]), and network embedding approaches (*e.g.* DeepWalk [22], and LINE [31]) in clustering and classification tasks. The experimental results are shown in the following sections.

A. Clustering Results

We first conduct a clustering task to evaluate the performance of the compared methods on DBLP and YAGO Movie datasets. For DBLP, we use the areas of authors as ground-truth label for clustering authors (A2A), and use the areas of venues as labels for clustering venues (V2V). For YAGO Movie, we use the genres of movies as labels (M2M). To be specific, we use k-means on the derived meta-graph-based relevance matrices for the clustering task. To evaluate the results, we use NMI and purity as evaluation metrics.

Clustering results of the three tasks are shown in Table II. Comparing to previous meta-graph-based relevance measures, the proposed GraphSim always shows the best performance of all. We observe at least 19.94% improvement

Table II: Clustering performance

		Pre. Meta-Graph Measures			Pre. Network Embedding		OUR WROKS		
Task	Method	StuctCount	SCSE	BSCSE($\alpha = 1$)	LINE*	DeepWalk*	GraphSim	MEGA	MEGA++
DBLP	NMI	0.2634	0.6309	0.6309	0.7954	0.8258	0.8479	0.8521	**0.8718**
(V2V)	Purity	0.5000	0.7333	0.7333	0.8042	0.8584	0.8744	0.8817	**0.8956**
DBLP	NMI	0.0338	0.0156	0.0156	0.3920	0.4896	0.2150	0.5263	**0.5315**
(A2A)	Purity	0.2997	0.2822	0.2823	0.7135	0.7941	0.4903	0.7956	**0.7989**
Moive	NMI	0.0011	0.0008	0.0008	0.0008	0.0007	0.0021	0.0045	**0.0045**
(M2M)	Purity	0.2991	0.2988	0.2988	0.2981	0.2981	0.3002	0.3017	**0.3032**
Overall	NMI	0.0994	0.2158	0.2158	0.3961	0.4387	0.3550	0.4610	**0.4693**
	Purity	0.3663	0.4381	0.4381	0.6052	0.6502	0.5550	0.6597	**0.6659**

Table III: Classification performance

		Pre. Meta-Graph Measures			Pre. Network Embedding		OUR WROKS		
Task	Method	StuctCount	SCSE	BSCSE($\alpha = 0$)	LINE*	DeepWalk*	GraphSim	MEGA	MEGA++
DBLP	Macro-F1	0.734	0.616	0.734	0.816	0.839	0.818	0.863	**0.867**
(A2A)	Micro-F1	0.730	0.634	0.730	0.817	0.840	0.819	0.863	**0.867**
Movie	Macro-F1	0.126	0.111	0.126	0.186	0.189	0.125	0.298	**0.310**
(M2M)	Micro-F1	0.281	0.276	0.281	0.241	0.245	0.307	0.342	**0.352**
Overall	Macro-F1	0.430	0.364	0.430	0.501	0.514	0.472	0.581	**0.589**
	Micro-F1	0.506	0.455	0.506	0.540	0.543	0.563	0.603	**0.610**

in NMI of GraphSim method when compared with the previous meta-graph-based relevance measure on clustering the venues and authors in DBLP, respectively. The clustering results can be sensitive to initialization of centroid seeds, so we set 100 times of random initializations. All methods show worse performance on YAGO Movie than DBLP, but the proposed methods, especially MEGA++, show the best performance comparing to prior works..

B. Classification Results

We then conduct a classification task. Comparing to the clustering task, in DBLP we do not evaluate the results of classifying the venues, as the total number of venues is only 18. We first apply previous methods and our works to generate the similarity matrices or embedding space of the original network. Then, we randomly partition the samples, and set 80% samples as training set and the rest as testing set. Last, we apply k nearest neighbor (k-NN) classifier with $k = 5$ to evaluate the methods with training and testing dataset [15], [29]. To prevent the special case of random partition, we repeat and use different random partition 10 times in total. For multi-label classification task, we use the average Macro-F1 score and Micro-F1 score as the evaluation metrics.

GraphSim outperforms the existing relevance measures (*e.g* StructCount, SCSE and BSCSE) because it represents a better relations between objects in the HINs by normalizing the presence of meta-graph structures. MEGA++ outperforms all the baselines because it captures both lower-order (*i.e.* meta-path) and higher-order (*i.e.* meta-graph) structural information by facilitating the use of coupled tensor-matrix decomposition method to obtain a joint embedding for nodes in HINs.

Table IV: Time analysis: three tasks of MEGA++: R is the dimension of the embedding, and second is the time scale

R	1	5	10	15
DBLP (A2A)	0.338	4.693	9.731	9.689
DBLP (V2V)	0.129	0.169	0.445	0.667
MOVIE (M2M)	1.625	6.013	12.69	19.57

C. Parameter Analysis

In this section, we first analyze the parameter sensitivity of our methods as shown in Figure 4. We use two evaluation metrics, Normalized Mutual Information(NMI), and Purity (both the larger, the better), to evaluate the performances of our methods for clustering task. In Figure 4 (a)-(b), the penalty parameters λ of MEGA is used for minimizing the Frobenius Norm of embedding space $\mathbf{P}$ and $\mathbf{Q}$ in Eq. (8), and the best performance is achieved when λ is set as 3.2768e-04. From 4 (b)-(c), setting the embedding dimensions as 5 shows the best performance for both MEGA and MEGA++. MEGA++ outperforms MEGA with the same number of embedding dimensions. The Figure 4 (e)-(f) show the two penalty parameters λ and α of MEGA++. The penalty parameter λ of MEGA++ is the same as that in MEGA. The penalty parameter α of MEGA++ is used for minimizing the Frobenius Norm of meta-graph similarity matrix $\mathbf{Y}$ and its embedding space in Eq. (7). We find that $\lambda = 0.6711$ and $\alpha = 1.6$ produce the best performance of clustering task.

D. Time Analysis

In this section, we evaluate the execution time of MEGA++. In Table IV, it shows the execution time is linear with respect to the embedding dimensions. Based

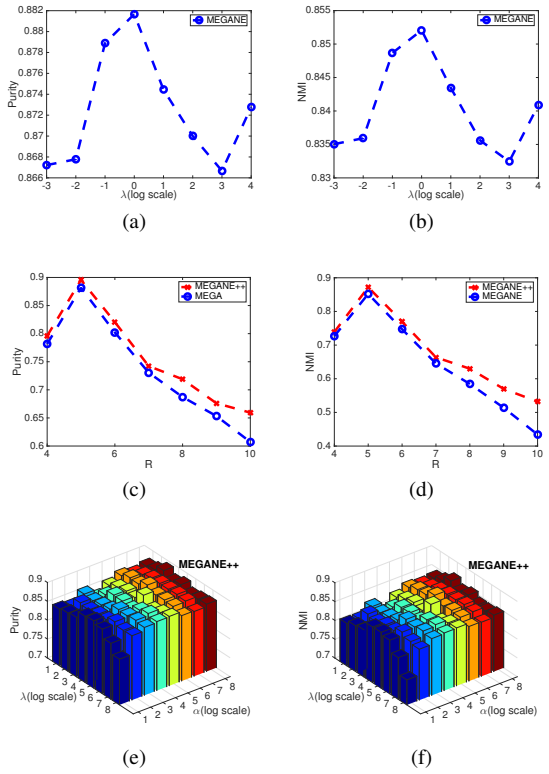

Figure 4: Parameters analysis: two metrics including Purity (a)(c)(e) and NMI (b)(d)(f). (a)-(b) analyzes λ of MEGA. (c)-(d) analyzes the embedding dimensions R of MEGA and MEGA++. (e)-(f) analyzes two parameters λ and α of MEGA++.

on the time complexity of MEGA++, when we have a fixed size of dataset, the embedding dimensions R, and the number of views N are linear with respect to the execution time. Sometimes, MEGA++ can be early stopped when it is already converge, so as to the same time consuming of DBLP (A2A) with $R = 10$ and $R = 15$. The same results are shown in the real testing on three tasks, which show the efficiency of MEGA++.

V. Related Work

A. Network Embedding

Network embedding want to learn a low-dimensional representations from a network. Previous traditional works [3] usually construct the affinity graph using the feature vectors of the vertexes and then compute the eigenvectors of the affinity graph. Some other groups use matrix factorization to represent graph as adjacency matrix [2].

Recently, DeepWalk [22] and LINE [31] are proposed for learning the network embedding. Besides these two most popular node embedding methods, many other network embedding are proposed recent years [6], [7], [10], [33]. [6], [33] learn the node embedding by deep learning encoder methods. However, none previous node embedding methods consider the meta-graph and its embedded meta-paths information.

B. Tensor Learning and Embedding

Just like deep learning, tensor learning becomes very hot and popular topic in recent years due to the stronger computing capability and lower computation cost [5], [11], [12], [14], [19], [20], [24]. Coupled tensor matrix embedding tries to fuse multiple information sources where matrices and tensors sharing some common modes are jointly embedding [8]. A gradient-based optimization approach for joint tensor-matrix analysis is proposed by Acar et al. [1].

C. Multi-view Learning

Multi-view learning is a hot idea to think one object with different views [13], [23], [26], [27]. In this paper, we think the HIN with different views such as meta-paths and meta-graph, and fuse the different information for node embedding. However, none of these frameworks can be directly applicable to learn jointly embedding with a partial symmetric tensor and a symmetric matrix, and also do not leverage meta-path and meta-structure information for similarity search in HINs.

VI. Conclusion and Future Work

In this paper, we proposed a new meta-graph-based relevance measure, *i.e.* GraphSim, and two node embeddings, *i.e.* MEGA and MEGA++, by leveraging a meta-graph and its embedded meta-paths similarity information. In the experiment, MEGA++ shows better performance than other compared methods in different tasks. In the future, we can expend our proposed node embedding for a single meta-graph to multiple meta-graphs node embedding in a HIN. Meanwhile, we can utilize heterogeneous and homogeneous information together for node embedding.

VII. Acknowledge

This work is supported in part by NSFC through grants No. 61503253 and 61672313, NSF through grants No. IIS-1526499, IIS-1763325, and CNS-1626432, and NSF of Guangdong Province through grant No. 2017A030313339.

References

[1] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv:1105.3422*, 2011.

[2] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *WWW*. ACM, 2013.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 2011.

[5] Bokai Cao, Lifang He, Xiaokai Wei, Mengqi Xing, Philip S Yu, Heide Klumpp, and Alex D Leow. t-bne: Tensor-based brain network embedding. In *SDM*. SIAM, 2017.

[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *KDD*. ACM, 2015.

[7] Ting Chen and Yizhou Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *WSDM*. ACM, 2017.

[8] Beyza Ermiş, Evrim Acar, and A Taylan Cemgil. Link prediction in heterogeneous data via generalized coupled tensor factorization. *DMKD*, 2015.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 2016.

[10] Huan Gui, Jialu Liu, Fangbo Tao, Meng Jiang, Brandon Norick, and Jiawei Han. Large-scale embedding learning in heterogeneous event data. In *ICDM*. IEEE, 2016.

[11] Tengjiao Guo, Le Han, Lifang He, and Xiaowei Yang. A ga-based feature selection and parameter optimization for linear support higher-order tensor machine. *Neurocomputing*, 2014.

[12] Lifang He, Xiangnan Kong, Philip S Yu, Xiaowei Yang, Ann B Ragin, and Zhifeng Hao. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*. SIAM, 2014.

[13] Lifang He, Chun-Ta Lu, Hao Ding, Shen Wang, Linlin Shen, S Yu Philip, and Ann B Ragin. Multi-way multi-level kernel modeling for neuroimaging classification. In *CVPR*, 2017.

[14] Lifang He, Chun-Ta Lu, Guixiang Ma, Shen Wang, Linlin Shen, S Yu Philip, and Ann B Ragin. Kernelized support tensor machines. In *ICML*, 2017.

[15] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. Meta structure: Computing relevance in large heterogeneous information networks. In *KDD*. ACM, 2016.

[16] Ni Lao and William W Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 2010.

[17] Athanasios P Liavas and Nicholas D Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *TSP*, 2015.

[18] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *NIPS*, 2011.

[19] Xiaolan Liu, Tengjiao Guo, Lifang He, and Xiaowei Yang. A low-rank approximation-based transductive support tensor machine for semisupervised classification. *TIP*, 2015.

[20] Chun-Ta Lu, Lifang He, Weixiang Shao, Bokai Cao, and Philip S Yu. Multilinear factorization machines for multi-task multi-view learning. In *WSDM*. ACM, 2017.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv*, 2013.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*. ACM, 2014.

[23] Weixiang Shao, Lifang He, Chun-Ta Lu, Xiaokai Wei, and Philip S Yu. Online unsupervised multi-view feature selection. *ICDM*, 2016.

[24] Weixiang Shao, Lifang He, and S Yu Philip. Clustering on multi-source incomplete data via tensor modeling and factorization. In *PAKDD*. Springer, 2015.

[25] Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. Multi-round influence maximization. In *KDD*. ACM, 2018.

[26] Lichao Sun, Yuqi Wang, Bokai Cao, S Yu Philip, Witawas Srisa-An, and Alex D Leow. Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning. In *ECML-PKDD*. Springer, 2017.

[27] Lichao Sun, Xiaokai Wei, Jiawei Zhang, Lifang He, S Yu Philip, and Witawas Srisa-an. Contaminant removal for android malware detection systems. In *BigData*. IEEE, 2017.

[28] Yizhou Sun, Jiawei Han, Charu C Aggarwal, and Nitesh V Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *WSDM*. ACM, 2012.

[29] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 2011.

[30] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S Yu, and Xiao Yu. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *TKDD*, 2013.

[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*. ACM, 2015.

[32] Charles F Van Loan. Structured matrix problems from tensors. In *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications*. Springer, 2016.

[33] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*. ACM, 2016.