Heterogeneous Information Network Hashing for Fast Nearest Neighbor Search

Zhen Peng[1], Minnan Luo[1(✉)], Jundong Li[2], Chen Chen[2], and Qinghua Zheng[1,3]

[1] School of Electronic and Information Engineering, Xi'an Jiaotong University, China
zhenpeng27@outlook.com, {minnluo,qhzheng}@xjtu.edu.cn
[2] Computer Science and Engineering, Arizona State University, USA
{jundongl,chen_chen}@asu.edu
[3] National Engineering Lab for Big Data Analytics, Xi'an Jiaotong University, China

Abstract. Heterogeneous information networks (HINs) are widely used to model real-world information systems due to their strong capability of capturing complex and diverse relations between multiple entities in real situations. For most of the analytical tasks in HINs (*e.g.*, link prediction and node recommendation), network embedding techniques are prevalently used to project the nodes into real-valued feature vectors, based on which we can calculate the proximity between node pairs with nearest neighbor search (NNS) algorithms. However, the extensive usage of real-valued vector representation in existing network embedding methods imposes overwhelming computational and storage challenges, especially when the scale of the network is large. To tackle this issue, in this paper, we conduct an initial investigation of learning binary hash codes for nodes in HINs to obtain the remarkable acceleration of the NNS algorithms. Specifically, we propose a novel heterogeneous information network hashing algorithm based on collective matrix factorization. Through fully characterizing various types of relations among nodes and designing a principled optimization procedure, we successfully project the nodes in HIN into a unified Hamming space, with which the computational and storage burden of NNS can be significantly alleviated. The experimental results demonstrate that the proposed algorithm can indeed lead to faster NNS and requires lower memory usage than several state-of-the-art network embedding methods while showing comparable performance in typical learning tasks on HINs, including link prediction and cross-type node similarity search.

1 Introduction

A heterogeneous information network (HIN) is a logical network that contains multiple types of entities and various relations between them [22]. Since it can capture rich information in a wide diversity of realistic scenarios via characterizing the heterogeneity of entities and their interactions, HINs become increasingly important and have widespread usage in many high-impact domains, such as the critical infrastructure systems, the knowledge graphs and the social media networks. Over the past few decades, many prevalent network learning tasks have been studied in HINs, most of which heavily depend on the nearest neighbor search (NNS) algorithms, examples include network clustering [16], link prediction [14], personalized recommendation [15], chemical similarity search [29], to name a few. Therefore, it is essential to provide an effective and trustworthy search function to advance these tasks in HINs. More specifically, we need a measure (*e.g.*, Euclidean distance or cosine distance) to quantify the node closeness in HINs such that given a query node in the HIN, we can retrieve the sets of nodes that are most similar to the query node by computing their similarity. Moreover, it should be noted that for the NNS algorithms, their utility is normally determined by two leading factors: (1) time complexity - the retrieval time for the nearest neighbors; and (2) space complexity - the storage costs of the used data structures. Typically, network embedding [9] is often regarded as a popular way to tackle the NNS problem by projecting nodes into a low-dimensional real-valued space while maximally preserving the node proximity in the HIN.

In fact, a vast majority of existing network embedding approaches such as PTE [23], metapath2vec [5] and HIN2Vec [7] generate real-valued node embeddings, which brings formidable computational and storage challenges to the NNS algorithms. The main reason is that a large number of indispensable similarity calculations in the NNS algorithms are directly performed on the learned real-valued node vectors, while the storage and the associated floating-point arithmetic operations could be very expensive. For instance, recommending a movie of interest for all users in a social media network including n users and m movies has a time complexity of $O(nmd)$ and a space complexity of $O((n+m)d)$, where d is the dimensionality of the node embeddings. Considering that the real-world social media networks may contain millions or billions of users and movies, the computational and storage costs of the NNS algorithms will be extremely high and are difficult to scale. Fortunately, hashing has been widely recognized as a promising technique to accelerate the similarity search [26]. Through transforming the real-valued vectors to compact binary codes consisting of a sequence of bits, hashing not only gains the advantage of reducing the memory storage costs but also lowers the time complexity of the nearest neighbor retrieval, as the similarity of two binary hash codes can be easily obtained with fast Hamming distance calculations.

The success of the hashing technique in many applications motivates our initial investigation of learning binary hash codes for nodes in HINs. However, it is a nontrivial problem with the following two unique challenges. The first core problem lies in how to elaborately characterize the diversity of connectivity patterns observed in HINs for the binary hash code learning. Secondly, after imposing the binary constraints on the learning model, designing a principled optimization algorithm is imperative as binary constraints can make the learning procedure NP-hard. To tackle these challenges, in this paper, we develop a novel heterogeneous information network hashing algorithm, called *HIN2Hash*. Benefiting from the collective matrix factorization (CMF) [21] technique which not only supports multi-relational learning but also improves the prediction accuracy, *HIN2Hash* models the diverse relations among nodes in HINs delicately and then learns unified binary hash codes for each node via a well-designed optimization framework. The superiority of *HIN2Hash* over other conventional methods rests with lower time cost and memory usage, and meanwhile, *HIN2Hash* exhibits competitive performance in some typical learning tasks on HINS, including link prediction and cross-type node similarity search. The main contributions of our work are as follow:

- Studying the problem of heterogeneous information network hashing which aims at (1) speeding up the downstream graph mining tasks on HINs that heavily depend on the NNS algorithms; and meanwhile (2) reducing the memory storage usage.
- Proposing a novel CMF based heterogeneous information network hashing algorithm *HIN2Hash* which carefully addresses the diversity of node types and relations among nodes, and encodes nodes into binary hash codes.
- Evaluating the performance of the proposed algorithm on three real-world datasets and the results illustrate that *HIN2Hash* can indeed accelerate similarity calculation and reduce storage with little sacrifice of performance.

2 Related Work

With the rise of representation learning research on networked data, a wide range of techniques have been developed in the past few years, especially for homogeneous network embedding. For instance, DeepWalk [17] and node2vec [8] are two typical algorithms based on the skip-gram model by using uniform and parameterized random walks, respectively. LINE [24] preserves 1st-order and 2nd-order proximities separately, to learn two different types of node embeddings. NetMF [18] is a matrix factorization based framework for network embedding. Different from the above methods, heterogeneous network embedding approaches fully consider the heterogeneity of nodes and edges for representation learning. For example, metapath2vec [5] learns node embeddings based on meta-path-based random walks and the heterogeneous skip-gram model.

Learning to hash (*a.k.a.* Hashing) [26], a widely studied solution to the approximate nearest neighbor search, attracts an enormous number of research efforts due to its comparable performance and superiority in time and storage. Typically, hashing algorithms can be classified into two classes: (1) two-stage approaches which first derive real-valued vectors and then convert them into binary codes, and (2) discrete hashing learning for binary hash codes directly. Recently, a number of network embedding algorithms adopt this idea to learn binary hash codes of nodes and are referred to as network hashing. However, these methods [13, 20] only focus on homogeneous networks, and ignore the various types of nodes and edges in HINs. Different from existing works, our algorithm is proposed for heterogeneous information network hashing and derive binary hash codes straightly through joint optimization of hashing and network embedding.

3 The Proposed Methodology

We first introduce the notations used in this paper. Following the standard notation, we use bold uppercase letters (e.g., $\mathbf{A}$) for matrices and bold lowercase letters (e.g., $\mathbf{b}$) for vectors. Scalars are indicated as normal lowercase letters (e.g., c) and uppercase italic letters (e.g., V) represent sets. Also, we denote the i-th row of the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ as $\mathbf{A}(i,:)$, the j-th column as $\mathbf{A}(:,j)$, and the (i,j)-th entry as $\mathbf{A}(i,j)$. As for the matrix norms, the only used matrix norm is the Frobenius norm, written as $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{A}(i,j)^2}$. We denote $tr(\cdot)$ as the matrix trace, and $sign(\cdot) : \mathbb{R} \rightarrow \{\pm 1\}$ as the round-off function.

3.1 Problem Formulation

With the above-mentioned notations, we now formally define the studied problem of heterogeneous information network hashing. It should be noted that we follow the definition of HINs as in [5, 7].

Definition 1. ***Heterogeneous Information Networks.*** *A heterogeneous information network is defined as a graph $G = (V, E, \Phi, \Psi)$ in which each node $v \in V$ and each edge $e \in E$ is associated with its own mapping function $\Phi(v) : V \rightarrow \mathcal{E}$ and $\Psi(e) : E \rightarrow R$,* i.e., *each node is mapped to one particular node type in $\mathcal{E}$ and each edge belongs to a specific edge type in R. Furthermore, we have $|\mathcal{E}| + |R| > 2$.*

According to the definition, it is obvious that an HIN G contains two different types of links, one is the relations among nodes of the same type (intra-type relations), and the other is the relations among nodes across two different node types (inter-type relations). Formally, suppose $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_g\}$ indicate a set of g node types, then there are g intra-type relations and C_g^2 inter-type relations theoretically. We adopt a set of matrices $A = \{\mathbf{A}_1, \ldots, \mathbf{A}_g\}$ to describe the proximity among nodes within each node type, where $\mathbf{A}_i \in \{0,1\}^{n_i \times n_i}$ $(i = 1, \ldots, g)$; and a set of matrices $X = \{\mathbf{X}_{ij}, (i,j = 1, \ldots, g)(i \neq j)\}$ to represent interactions among nodes of different types, where $\mathbf{X}_{ij} \in \{0,1\}^{n_i \times n_j}$ denotes the inter-type relations between $\mathcal{E}_i$ and $\mathcal{E}_j$. Taking an HIN represented by A and X as an input, the problem of heterogeneous information network hashing is formally defined as follows.

Problem 1. ***Heterogeneous Information Network Hashing.*** *Given an HIN G, the problem of heterogeneous information network hashing aims to learn a function $\mathcal{F} : V \rightarrow \{\pm 1\}^d$ that projects each node $v \in V$ to a low-dimensional Hamming space $\{\pm 1\}^d$ shown in $U = \{\mathbf{U}_1, \ldots, \mathbf{U}_g\}$, where $\mathbf{U}_i \in \{\pm 1\}^{d \times n_i}$ $(i = 1, \ldots, g)$ and $d \ll |V|$. In the learned hashing space $\{\pm 1\}^d$, two additional constraints on the binary codes introduced by [28],* i.e., *bit uncorrelation and bit balance (more details later), should be satisfied as much as possible.*

3.2 Heterogeneous Information Network Hashing

As shown in [18], matrix factorization provides a principled framework to unify the prevalent network embedding methods such as DeepWalk, node2vec, LINE and achieves comparable performance. Hence, in the following context, we resort to the matrix factorization to elaborate our developed framework for heterogeneous information network hashing. Distinct from conventional homogeneous networks, in HINs, multiple types of nodes and edges are presented together, and a particular node type may be involved in more than one types of relations. In order to fully characterize these complex and diverse connectivity patterns among nodes, we expand our view from separately factoring the individual connectivity matrix of each node type to collective matrix factorization as it is able to take full advantage of different types of connections among nodes in HINs for representation learning.

Mathematically, we first define a pairwise relational schema $S = \{(i,j)|\mathcal{E}_i \sim \mathcal{E}_j, i < j\}$ to include all possible C_g^2 inter-type relations. Then the objective function is formulated as:

$$\min_{\mathbf{U}_i \in U_*} \sum_{(i,j)\in S} \alpha_{ij}\|\mathbf{X}_{ij} - \mathbf{U}_i^T\mathbf{U}_j\|_F^2 + \sum_{i=1}^{g} \beta_i \|\mathbf{A}_i - \mathbf{U}_i^T\mathbf{U}_i\|_F^2, \tag{1}$$

where $\mathbf{U}_i$ denotes the binary hash codes of all n_i nodes belonging to $\mathcal{E}_i$. It should be noted that $\mathbf{U}_i$ has to satisfy the bit uncorrelation and balance constraints such that $U_* = \{\mathbf{U}_i \in \{\pm 1\}^{d\times n_i}|\mathbf{U}_i\mathbf{U}_i^T = n_i\mathbf{I}_d, \mathbf{U}_i\mathbf{1}_{n_i} = \mathbf{0}\}$. The constraints in U_* maximize the information encoded in the binary embedding space. To be more specific, the bit uncorrelation constraint ensures that each bit should be as independent as possible. Meanwhile, the bit balance constraint makes each bit of almost equal chance of being 1 or -1. In other words, it maximizes the entropy of each bit. By simultaneously decomposing the relation matrices $\mathbf{X}_{ij}$ and $\mathbf{A}_i$ into a product of two matrices in a low-dimensional Hamming space, we obtain unified binary codes in which different types of relations among nodes can be captured. The trade-off parameters $\alpha_{ij}, \beta_i \geq 0$ measure the importance of each relation matrix in the reconstruction process.

Admittedly, solving Eq. (1) is a challenging task since the existence of binary constraints makes the optimization process of learning binary hash codes NP-hard [10] and may cause Eq. (1) infeasible to solve. For the sake of solving it in a computationally tractable manner, we reformulating Eq. (1) by softening the bit balance and uncorrelation constraints [30, 13]. Considering that the bit balance condition is easier to deal with, we intend to handle these two constraints separately by splitting the constraint set U_* into $U_0 = \{\mathbf{U}_i \in \{\pm 1\}^{d\times n_i}|\mathbf{U}_i\mathbf{1}_{n_i} = \mathbf{0}\}$ and $U_\perp = \{\mathbf{U}_i \in \{\pm 1\}^{d\times n_i}|\mathbf{U}_i\mathbf{U}_i^T = n_i\mathbf{I}_d\}$. As for the uncorrelation constraint, we approximate it by introducing $Z = \{\mathbf{Z}_i \in \mathbb{R}^{d\times n_i}|\mathbf{Z}_i\mathbf{Z}_i^T = n_i\mathbf{I}_d\}$ and a penalty term recording the deviation between a feasible $\mathbf{U}_i$ and the set Z. Hence, the original objective function can be softened as:

$$\min_{\mathbf{U}_i \in U_0, \mathbf{Z}_i \in Z} \sum_{(i,j)\in S} \alpha_{ij} \|\mathbf{X}_{ij} - \mathbf{U}_i^T \mathbf{U}_j\|_F^2 + \sum_{i=1}^{g} \beta_i \|\mathbf{A}_i - \mathbf{U}_i^T \mathbf{U}_i\|_F^2 + \sum_{i=1}^{g} \varphi_i \|\mathbf{U}_i - \mathbf{Z}_i\|_F^2. \tag{2}$$

The above equation allows a certain discrepancy between the binary hash codes (e.g., $\mathbf{U}_i$) and their delegate continuous values (e.g., $\mathbf{Z}_i$), which makes these constraints computationally tractable. In fact, we can impose a very large parameter φ_i to force the matrix $\mathbf{U}_i$ to be equal to $\mathbf{Z}_i$ such that the constraints in Eq. (1) become feasible. By jointly optimizing the binary codes and the delegate real variables, we can obtain nearly balanced and uncorrelated Hamming codes for each node v in the HIN G.

4 The Optimization Algorithm

Due to the existence of binary constraints, we employ the alternating direction method of multipliers (ADMM) [1] to solve Eq. (2). First, we introduce variables $\widetilde{\mathbf{U}}_i$ $(i = 1, \dots, g)$ to convert Eq. (2) to the following equivalent objective function:

$$\begin{aligned} \min_{\mathbf{U}_i \in U_0, \mathbf{Z}_i \in Z} & \sum_{(i,j)\in S} \alpha_{ij} \|\mathbf{X}_{ij} - \mathbf{U}_i^T \mathbf{U}_j\|_F^2 + \sum_{i=1}^{g} \beta_i \|\mathbf{A}_i - \widetilde{\mathbf{U}}_i \mathbf{U}_i\|_F^2 \\ & + \sum_{i=1}^{g} \varphi_i \|\mathbf{U}_i - \mathbf{Z}_i\|_F^2 \qquad s.t.\, \mathbf{U}_i^T = \widetilde{\mathbf{U}}_i. \end{aligned} \tag{3}$$

The augmented Lagrange function of Eq. (3) is:

$$\begin{aligned} & \mathcal{L}_{\{\rho_i\}_{i=1}^g}(\{\mathbf{U}_i, \widetilde{\mathbf{U}}_i, \mathbf{Z}_i, \Lambda_i\}_{i=1}^g) \\ = & \sum_{(i,j)\in S} \alpha_{ij} \|\mathbf{X}_{ij} - \mathbf{U}_i^T \mathbf{U}_j\|_F^2 + \sum_{i=1}^{g} \beta_i \|\mathbf{A}_i - \widetilde{\mathbf{U}}_i \mathbf{U}_i\|_F^2 + \sum_{i=1}^{g} \varphi_i \|\mathbf{U}_i - \mathbf{Z}_i\|_F^2 \\ + & \sum_{i=1}^{g} \langle \Lambda_i, \mathbf{U}_i^T - \widetilde{\mathbf{U}}_i \rangle + \sum_{i=1}^{g} \frac{\rho_i}{2} \|\mathbf{U}_i^T - \widetilde{\mathbf{U}}_i\|_F^2 \qquad s.t.\, \mathbf{U}_i \in U_0, \mathbf{Z}_i \in Z, \end{aligned} \tag{4}$$

where Λ_i is Lagrange multiplier, and ρ_i is the constraint violation penalty parameter. Normally, in ADMM, the basic Gauss-Seidel structure in $(t+1)$-th iteration is as follows:

$$\begin{cases} \mathbf{U}_i^{t+1} = \arg\min \mathcal{L}(\mathbf{U}_i, \{\widetilde{\mathbf{U}}_i^t, \mathbf{Z}_i^t, \Lambda_i^t\}_{i=1}^g) \ \ (i = 1, \dots, g), \\ \widetilde{\mathbf{U}}_i^{t+1} = \arg\min \mathcal{L}(\widetilde{\mathbf{U}}_i, \{\mathbf{U}_i^{t+1}, \mathbf{Z}_i^t, \Lambda_i^t\}_{i=1}^g) \ \ (i = 1, \dots, g), \\ \mathbf{Z}_i^{t+1} = \arg\min \mathcal{L}(\mathbf{Z}_i, \{\mathbf{U}_i^{t+1}, \widetilde{\mathbf{U}}_i^{t+1}, \Lambda_i^t\}_{i=1}^g) \ \ (i = 1, \dots, g), \\ \Lambda_i^{t+1} = \Lambda_i^t + \rho_i((\mathbf{U}_i^{t+1})^T - \widetilde{\mathbf{U}}_i^{t+1}) \ \ (i = 1, \dots, g). \end{cases} \tag{5}$$

Next, we give the details to show how to solve above-mentioned subproblems.

$\mathbf{U}_i^{t+1}$-subproblem. When the other variables are fixed, via the basic algebraic operations, the optimization problem for $\mathbf{U}_i^{t+1}$ is formulated as:

$$\begin{aligned}\mathbf{U}_i^{t+1} = \arg\max_{\mathbf{U}_i\in U_0} \sum_{(i,j)\in S_i} \alpha_{ij} tr(\mathbf{X}_{ij}(\mathbf{U}_j^t)^T\mathbf{U}_i) + \sum_{(k,i)\in S_i} \alpha_{ki} tr(\mathbf{X}_{ki}^T(\mathbf{U}_k^{t+1})^T\mathbf{U}_i) \\ + \beta_i tr(\mathbf{A}_i^T\widetilde{\mathbf{U}}_i^t\mathbf{U}_i) + \frac{\rho_i}{2} tr(\widetilde{\mathbf{U}}_i^t\mathbf{U}_i) - \frac{1}{2} tr(\varLambda_i^t\mathbf{U}_i) + \varphi_i tr((\mathbf{Z}_i^t)^T\mathbf{U}_i),\end{aligned} \tag{6}$$

where S_i is a subset of S which contains all inter-type relations involving $\mathbf{U}_i$. Mathematically, Eq. (6) is also equivalent to:

$$\mathbf{U}_i^{t+1} = \arg\min_{\mathbf{U}_i\in U_0} \|\mathbf{U}_i - \varPi_i\|_F^2, \tag{7}$$

where $\varPi_i = (\sum_{(i,j)\in S_i} \alpha_{ij}\mathbf{X}_{ij}(\mathbf{U}_j^t)^T + \sum_{(k,i)\in S_i} \alpha_{ki}\mathbf{X}_{ki}^T(\mathbf{U}_k^{t+1})^T + \beta_i\mathbf{A}_i^T\widetilde{\mathbf{U}}_i^t + \frac{\rho_i}{2}\widetilde{\mathbf{U}}_i^t - \frac{1}{2}\varLambda_i^t + \varphi_i(\mathbf{Z}_i^t)^T)^T$. It can also be interpreted as projecting $\varPi_i$ onto a balanced Hamming space and the optimal solution can be easily obtained by:

$$\mathbf{U}_i^{t+1} = sign(\varPi_i - \lambda\mathbf{1}_{n_i}^T), \tag{8}$$

where $\lambda = median(\varPi_i)$ is the row median of $\varPi_i$ and can be viewed as a multiplier of the bit balance constraint.

$\widetilde{\mathbf{U}}_i^{t+1}$-subproblem. To achieve $\widetilde{\mathbf{U}}_i^{t+1}$, we need to solve:

$$\widetilde{\mathbf{U}}_i^{t+1} = \arg\min \beta_i\|\mathbf{A}_i - \widetilde{\mathbf{U}}_i\mathbf{U}_i^{t+1}\|_F^2 + \frac{\rho_i}{2}\|(\mathbf{U}_i^{t+1})^T - \widetilde{\mathbf{U}}_i + \frac{\varLambda_i^t}{\rho_i}\|_F^2. \tag{9}$$

By setting the derivative of Eq. (9) w.r.t. $\widetilde{\mathbf{U}}_i$ to zero, we get a closed-form solution of $\widetilde{\mathbf{U}}_i^{t+1}$ as follows:

$$\widetilde{\mathbf{U}}_i^{t+1} = \frac{1}{2n_i\beta_i + \rho_i}(2\beta_i\mathbf{A}_i(\mathbf{U}_i^{t+1})^T + \rho_i(\mathbf{U}_i^{t+1})^T + \varLambda_i^t). \tag{10}$$

$\mathbf{Z}_i^{t+1}$-subproblem. The subproblem regarding $\mathbf{Z}_i^{t+1}$ is:

$$\mathbf{Z}_i^{t+1} = \arg\min_{\mathbf{Z}_i\in Z} \varphi_i\|\mathbf{U}_i^{t+1} - \mathbf{Z}_i\|_F^2. \tag{11}$$

It is easy to derive the update rule of $\mathbf{Z}_i$ which is equivalent to projecting $\mathbf{U}_i^{t+1}$ to the Stiefel manifold [6] and an analytical can be derived. Specifically, on the basis of Von Neumann's trace inequality [11], we have $tr(\mathbf{U}_i^{t+1}\mathbf{Z}_i^T) \leq \sum_{k=1}^d \sigma_k(\mathbf{Z}_i)\sigma_k(\mathbf{U}_i^{t+1})$, where $\sigma_k(\mathbf{Z}_i)$ is the k-th largest singular value of $\mathbf{Z}_i$. Assume that $\mathbf{U}_i^{t+1} = \mathbf{P}_i\varSigma\mathbf{Q}_i^T$ is the thin SVD decomposition of $\mathbf{U}_i^{t+1}$, then $\mathbf{Z}_i^{t+1}$ can be updated according to:

$$\mathbf{Z}_i^{t+1} = \sqrt{n_i}\mathbf{P}_i\mathbf{Q}_i^T. \tag{12}$$

The key steps of the proposed algorithm are summarized in Algorithm 1. By optimizing the network embedding and hashing in a joint fashion, *HIN2Hash* generates compact binary hash codes for nodes in HINs. After analysis, the total time complexity is $\#iterations * O((g-1)n^2d + n^2d + nd^2)$, *i.e.*, $\#iterations * O(n^2d)$, where $n = \max\{n_i\}_{i=1}^{g}$ (detailed analysis omitted for brevity).

Algorithm 1 *HIN2Hash*: Heterogeneous information network hashing for fast nearest neighbor search

```
Input: An HIN $G$ including intra-type relations $A$ and inter-type relations $X$, the
    embedding dimension $d$, parameters $\alpha_i, \beta_i$ and $\varphi_i$ $(i = 1, \dots, g)$.
Output: The binary hash codes $U$ of all nodes.
 1: Initialize $\mathbf{U}_i$, $\widetilde{\mathbf{U}}_i$ and $\Lambda_i$ $(i = 1, \dots, g)$ to be zero matrices, $\rho_i = 10^{-6}$ $(i = 1, \dots, g)$,
    $\max_\rho = 10^{10}$, $\tau = 1.1$, $\varepsilon = 10^{-3}$, $t = 0$.
 2: Initialize $\mathbf{Z}_i$ $(i = 1, \dots, g)$ by Eq. (12).
 3: while objective function in Eq. (2) not converge do
 4:     Update $\mathbf{U}_i^{t+1}$ $(i = 1, \dots, g)$ by Eq. (8);
 5:     Update $\widetilde{\mathbf{U}}_i^{t+1}$ $(i = 1, \dots, g)$ by Eq. (10);
 6:     Update $\mathbf{Z}_i^{t+1}$ $(i = 1, \dots, g)$ by Eq. (12);
 7:     Update Lagrange multipliers $\Lambda_i^{t+1}$ $(i = 1, \dots, g)$ via
        $\Lambda_i^{t+1} = \Lambda_i^t + \rho_i((\mathbf{U}_i^{t+1})^T - \widetilde{\mathbf{U}}_i^{t+1})$;
 8:     Update the parameters $\rho_i$ $(i = 1, \dots, g)$ by
        $\rho_i = \min(\tau\rho_i, \max_\rho)$;
 9:     $t = t + 1$;
10:     Check the convergence conditions $\|(\mathbf{U}_i^t)^T - \widetilde{\mathbf{U}}_i^t\|_\infty < \varepsilon$ $(i = 1, \dots, g)$ and
        $|\frac{\mathcal{F}^t - \mathcal{F}^{t-1}}{\mathcal{F}^{t-1}}| < \varepsilon$, where $\mathcal{F}^t$ is the value of Eq. (2) at the $t$-th iteration.
11: end while
12: Output the learned hash embedding $U$ for the HIN $G$.
```

5 Experiments

In this section, we empirically verify the effectiveness of the proposed algorithm *HIN2Hash* on two fundamental tasks on HINs, including link prediction and cross-type node similarity search [3]. In particular, we attempt to answer the following three research questions:

Q1 How effective is the proposed algorithm in predicting missing links and performing cross-type node similarity search?

Q2 Will the binary codes reduce the processing time of the task that rely on the NNS algorithms?

Q3 Will the binary codes reduce the memory storage costs?

5.1 Datasets

We perform evaluations on three datasets that have been widely used in the previous research [2, 12], including one academic network - AMINER, one infrastruc-

Table 1: Statistics of the used datasets.

	AMINER	INFRA	BIO
#node types	3	3	3
#inter-type relations	2	3	3
#nodes	17,504	8,325	35,631
#intra-type links	107,466	15,138	253,827
#inter-type links	35,229	23,897	75,456

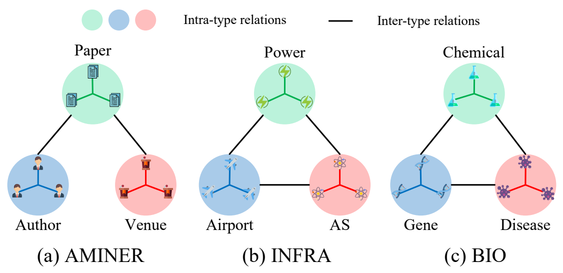


Fig. 1: Abstract network structure diagram.

ture system network - INFRA, and one comparative toxicogenomics database (CTD) network in the biological domain - BIO. The statistics of the datasets are listed in Table 1, and Figure 1 is a schematic diagram of their respective link relations.

AMINER [25] is an academic network in the domain of computer science[4] which contains three node types, *i.e.*, papers, authors and conference venues. Specifically, there are three intra-type relations: (1) a paper-paper citation network, (2) an author-author co-authorship network, and (3) a venue-venue citation network; and two inter-type relations: (1) the paper-author dependency, and (2) the paper-venue dependency. The abstract network structure diagram of AMINER is shown in Figure 1(a).

INFRA [27] involves three critical infrastructure networks: (1) an autonomous system network[5], (2) an airport network[6], and (3) a power grid networks. The above three networks record the intra-type relations, and they are functionally dependent on each other and form a triangle-shaped inter-type dependency network as shown in Figure 1(b).

BIO is a CTD network that is constructed based [19, 4]. It includes three intra-type relations which are chemical, disease and gene similarity networks. Meanwhile, as shown in Figure 1(c), interactions in the form of inter-type relations also exist among three types of nodes.

4 https://aminer.org/
5 http://snap.stanford.edu/data/
6 http://www.levmuchnik.net/Content/Networks/NetworkData.html

5.2 Compared Methods

We compare *HIN2Hash* with six state-of-the-art network embedding methods which can be roughly categorized into two classes: matrix factorization based methods and skip-gram based methods. The details of compared baseline methods are given as follow.

- **CMF** [21]: The collective matrix factorization (CMF) approach can be regarded as a variant of *HIN2Hash*. Specifically, it removes the binary constraints imposed on the objective function, and encodes nodes into a continuous low-dimensional feature space.
- **DeepWalk** [17]: DeepWalk learns low-dimensional node representations via truncated random walks and the skip-gram model.
- **LINE** [24]: LINE is a large-scale information network embedding method which preserves the 1st-order and the 2nd-order node proximity. In the experiments, we concatenate both the 1st and the 2nd order representations together.
- **Node2vec** [8]: With parameterized random walks and the skip-gram model, node2vec explores neighborhood structure around each node in a more flexible way for embedding representation learning.
- **DCF** [30]: The discrete collaborative filtering (DCF) algorithm is originally applied to the recommendation problem. In the experiments, we ignore the node and edge types, and transform the heterogeneous network G into a large flattened homogeneous network.
- **Metapath2vec** [5]: Metapath2vec learns node embeddings in HINs through meta-path-guided random walks and a heterogeneous skip-gram model.

Among them, DeepWalk, LINE and node2vec are designed for homogeneous networks. In the experiments, they are performed in a way by treating different types of nodes and edges as the same type. According to the form of derived embeddings, the above algorithms can fall into two categories: algorithms for generating binary hash codes including DCF and *HIN2Hash*, and the remaining algorithms for generating real-valued embedding representations.

5.3 Settings and Evaluation Metrics

In terms of the default parameter settings, the dimension of node vectors d is set to 128 for all approaches. For DeepWalk and node2vec, the context window size, walk length and the number of walks per node are set to 8, 80, and 10, respectively. For LINE, the size of negative samples is set to 5. The two parameters p and q for parameterized random walks in node2vec are set to 1 and 4, respectively. As for metapath2vec, the context window size, walk length, the number of walks per node and the number of negative examples are set as 5, 80, 600, and 5, separately. Regarding the regularization parameters in the proposed framework, we tune them in the range of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$.

For the link prediction task, we refer to the experimental settings and the evaluation metrics in [8]. To be more specific, given a network with a certain fraction of edges removed, the task is to predict these missing edges. In particular, we randomly remove 10%, 20% and 50% neighbors of each node and use the rest for training, while ensuring that the residual network obtained after the edge removal is fully connected. The criteria we adopted is AUC which is the area under the ROC curve. Additionally, the negative examples involved in AUC are built by randomly sampling an equal number of node pairs with no connections in the original network.

For the cross-type node similarity search task, we also remove a portion of edges with the goal of detecting the robustness of *HIN2Hash*. Since our work is on heterogeneous network hashing, we focus more on the similarity search across different types of nodes. For instance, given a particular disease as a query node, the task can be finding the genes that highly associated with this disease. In order to quantitatively evaluate the performance of our method, we leverage MAP to measure the mean average precision over all nodes in the inter-type relations and the other types of nodes that are connected to the query node are taken as the ground truth.

5.4 Experimental Results

Quantitative results Table 2 and Table 3 summarize the AUC and MAP scores of all the methods, respectively. We have the following observations from these tables:

- The AUC scores of *HIN2Hash* are comparable to the best baseline method on all datasets, regardless of the removal ratio of edges. The MAP scores of *HIN2Hash* is slightly worse than DeepWalk and node2vec on INFRA and BIO. While on the AMINER dataset, our approach achieves the best performance.
- In the link prediction task, CMF outperforms *HIN2Hash* in most cases but is not as good as ours on the AMINER dataset. In the cross-type node similarity search task, *HIN2Hash* outperforms CMF on all datasets. Hence, it is reasonable to conclude that even though transforming the node embeddings from real-valued vectors to binary hash codes will result in certain information loss, but the sacrifice is not much. In some certain cases, it may even be helpful to filter out the noise in the embedding such that a number of tasks that heavily reply on the NNS algorithms can be even slightly improved.
- The DCF algorithm, the other algorithm which learns the binary hash codes for nodes in the network, is almost the worst among all the methods. Their inferior performance can be attributed to the following two reasons: (1) it does not differentiate different inter-type and intra-type connections among nodes, and (2) the optimization algorithm used in DCF loses a lot of information as its subproblems are NP-hard to solve, while the subproblems in our algorithm have closed optimal solutions.
- For metapath2vec, it often requires prior human knowledge to define the most appropriate metapaths. Even though the high-quality matapaths have

Table 2: AUC scores for link prediction on three datasets.

ALGORITHM	AMINER			INFRA			BIO		
	10%	20%	50%	10%	20%	50%	10%	20%	50%
CMF	0.9216	0.9074	0.8807	**0.8899**	0.8651	0.7612	**0.9452**	**0.9401**	**0.9314**
DeepWalk	0.9624	**0.9575**	0.9238	0.8764	**0.8676**	**0.8306**	0.9241	0.9200	0.9051
LINE	0.9424	0.9348	0.9035	0.8122	0.7244	0.5332	0.6048	0.5963	0.5789
Node2vec	**0.9684**	0.9568	**0.9354**	0.8753	0.8662	0.8281	0.9165	0.9076	0.9048
DCF	0.8495	0.8431	0.8055	0.6960	0.6850	0.5555	0.5681	0.5536	0.5337
Metapath2vec	0.9110	0.9008	0.8735	0.8615	0.7820	0.6762	0.8206	0.8027	0.7728
HIN2Hash	**0.9426**	**0.9316**	**0.9260**	**0.8313**	**0.8258**	**0.7950**	**0.9104**	**0.9101**	**0.8871**

The percentage denotes the removal ratio of edges.

Table 3: MAP scores for cross-type node similarity search on three datasets.

ALGORITHM	AMINER			INFRA			BIO		
	10%	20%	50%	10%	20%	50%	10%	20%	50%
CMF	0.6480	0.5941	0.5431	0.5914	0.5783	0.5080	0.6446	0.6158	0.5124
DeepWalk	0.5260	0.4949	0.4660	**0.8711**	**0.8436**	**0.8039**	**0.7544**	0.7439	**0.7248**
LINE	0.4370	0.4098	0.3875	0.4549	0.4271	0.3867	0.3678	0.3766	0.3601
Node2vec	0.6147	0.5704	0.5285	0.8579	0.8384	0.7927	0.7538	**0.7517**	0.7223
DCF	0.4083	0.3875	0.3624	0.4016	0.3921	0.3701	0.3445	0.3368	0.3154
Metapath2vec	0.4906	0.4780	0.4666	0.4665	0.4342	0.3980	0.5676	0.5493	0.5152
HIN2Hash	**0.7809**	**0.6978**	**0.6303**	**0.8147**	**0.7760**	**0.7138**	**0.7056**	**0.7006**	**0.6947**

The percentage denotes the removal ratio of edges.

been well discovered in the academic datasets such as AMINR, it is difficult to achieve desirable results on INFRA and BIO with limited human experiences of the optimal metapaths.

Time Cost Since the time cost of similarity calculation for CMF, DeepWalk, LINE, node2vec and metapath2vec (algorithms for generating real-valued embeddings) are quite close, Table 4 only illustrates the results of *HIN2Hash* and node2vec in node similarity computation. As can be seen, the similarity calculation on binary codes is significantly faster than that on real-valued vectors in all cases, which substantiates our motivation. We can attribute this superiority to the fact that similarity calculation in the real-valued vector space is replaced by the bit operations in a low-dimensional Hamming space. It is well known that computers are better at performing bit operations than floating-point arithmetic. Therefore, hashing is a promising and impactful technique in accelerating computing on large-scale networks.

Memory Usage This part is carried out on the MATLAB platform. As algorithms for generating real-valued embeddings included in baselines use *double*

Table 4: Time cost for similarity calculation of node2vec and *HIN2Hash* .

Metric	Dataset	*HIN2Hash* Time(s)	Node2vec Time(s)	Speedup
AUC	AMINER	0.013±0.001	0.079±0.004	6.08×
	INFRA	0.011±0.001	0.074±0.001	6.73×
	BIO	0.021±0.001	0.149±0.013	7.10×
MAP	AMINER	49.342±0.196	404.118±1.617	8.19×
	INFRA	22.241±0.114	205.192±0.791	9.23×
	BIO	355.383±2.834	3077.695±11.079	8.66×

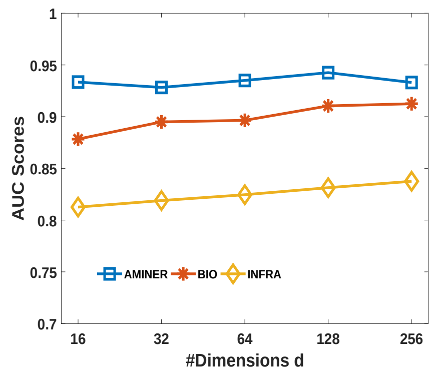
Table 5: Memory usage of embeddings derived by node2vec and *HIN2Hash* .

Computer Memory	Dataset	*HIN2Hash* Size(KB)	Node2vec Size(MB)	Reduction
Internal Memory	AMINER	273.50	17.09	64×
	INFRA	130.08	8.13	64×
	BIO	556.73	34.80	64×
External Memory	AMINER	274.03	15.76	59×
	INFRA	130.43	7.52	59×
	BIO	557.61	32.27	59×

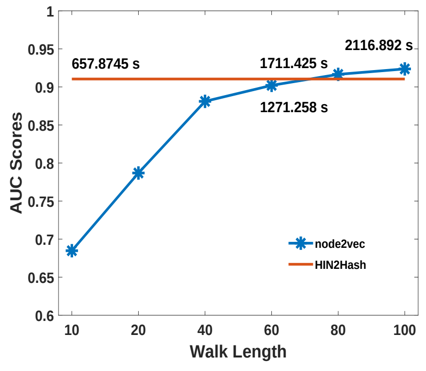
data type to store vector representations, occupying the same storage space, we show the memory usage of node2vec and ours for storing network embedding/hashing results in Table 5. In theory, only 1 bit is enough to represent binary numbers. Hence, we design an algorithm to compress the binary codes and store them with *uint8* data type instead of *double*. As for external memory, we save the above embeddings as MAT files, where MATLAB compresses the data using HDF5-variant format. It is obvious to see that compared with node2vec, *HIN2Hash* remarkably reduces the memory usage of embeddings with the same dimensionality. Besides, *HIN2Hash* only needs to store less than 1 MB of binary codes for all three datasets, which further demonstrates the superiority of hashing in reducing storage costs.

5.5 Parameter Analysis

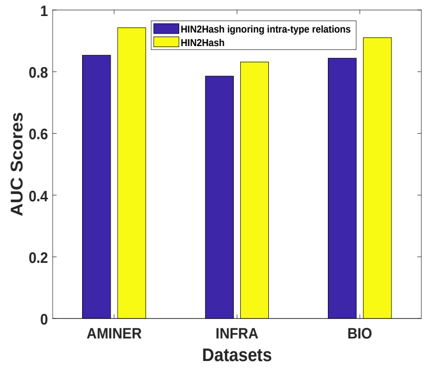
Dimensionality and Training Time Due to space limitation, only the results of link prediction are given here. Figure 2(a) shows how the AUC scores vary with different dimensionality d. In AMINER, the best performance is achieved when d is 128. In INFRA and BIO, the performance enhances with the increasing d up to 256, but only a little. Therefore, setting d to 128 is reasonable as it is sufficient to capture the node proximity among nodes, and too large d cannot gain desired benefits but greatly increases the burden of storage. Also, we have an interesting observation on BIO as shown in Figure 2(b). *HIN2Hash* requires much less training time (657s) to obtain competitive results with node2vec, while



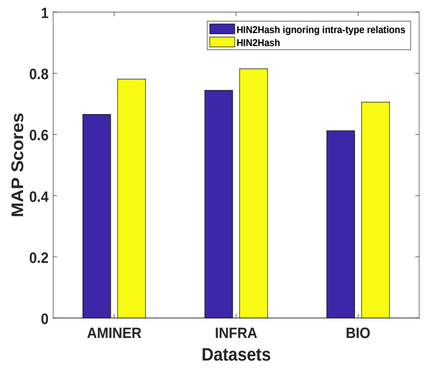
(a) AUC $w.r.t$ dimensionality



(b) AUC v.s. training time



(c) AUC for link prediction



(d) MAP for cross-type node similarity search

Fig. 2: (a) AUC scores in different dimensions. (b) AUC scores v.s. training time on BIO dataset. (c) Effects of intra-type relations on link prediction. (d) Effects of intra-type relations on cross-type node similarity search.

node2vec takes about three times (1,711s). The main reason is that *HIN2Hash* benefits from the characteristics of ADMM that produces acceptable results for practical use within tens of iterations. The reduction of walk length in node2vec can accelerate training but has the side effect of reduced performance since walk length has a relatively high impact on the performance of skip-gram based network embedding algorithms.

Effects of Intra-type Relations Among the parameters appeared in Eq. (2), α_i and $\beta_i (i = 1, \ldots, g)$ are relatively more important since they control the participation of inter-type and intra-type relations for binary hash codes learning, respectively. In this section, we attempt to explore the impact of intra-type relations on the quality of heterogeneous network hashing. By setting β_i to be zero, only inter-type relations are taken into consideration in the learning process, while two kinds of relations are both used when β_i does not equal to zero. Figure 2(c)(d) shows the performance of link prediction and cross-type node similarity search in these two scenarios. As can be observed, when the intra-type relations are ignored, the AUC and MAP scores are lower on all the three datasets. Hence, we can conclude that the intra-type relations play a vital role in heterogeneous network hashing and it is necessary to consider them in our framework.

6 Conclusions and Future Work

In this work, we study the issue of learning to hash HINs and propose a novel CMF based heterogeneous information network hashing algorithm *HIN2Hash* that projects nodes into a low-dimensional Hamming space instead of Euclidean space. Experiments on three real-world datasets indicate that *HIN2Hash* can achieve comparable results in most cases *w.r.t.* the link prediction and the cross-type node similarity search tasks, and even outperforms other prevalent network embedding techniques in certain cases. Meanwhile, *HIN2Hash* is superior to other embedding methods in both computation time and memory usage, due to the characteristics of the learned discrete binary hash codes. Future work will concentrate on attributed network hashing and task-oriented network hashing.

Acknowledgements. This work is supported by National Key Research and Development Program of China (2016YFB1000903), National Nature Science Foundation of China (61872287, 61532015 and 61672418), Innovative Research Group of the National Natural Science Foundation of China (61721002), Innovation Research Team of Ministry of Education (IRT_17R86), Project of China Knowledge Center for Engineering Science and Technology.

References

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends® in Machine Learning **3**(1), 1–122 (2011)
2. Chen, C., Tong, H., Xie, L., Ying, L., He, Q.: FASCINATE: Fast Cross-Layer Dependency Inference on Multi-Layered Networks. In: KDD (2016)
3. Cui, P., Wang, X., Pei, J., Zhu, W.: A Survey on Network Embedding. IEEE Transactions on Knowledge and Data Engineering (2018)
4. Davis, A.P., Grondin, C.J., Lennon-Hopkins, K., Saraceni-Richards, C., Sciaky, D., King, B.L., Wiegers, T.C., Mattingly, C.J.: The Comparative Toxicogenomics Database's 10th Year Anniversary: Update 2015. Nucleic Acids Research **43**(D1), D914–D920 (2014)
5. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In: KDD (2017)
6. Eldén, L., Park, H.: A Procrustes Problem on the Stiefel Manifold. Numerische Mathematik **82**(4), 599–619 (1999)

7. Fu, T.y., Lee, W.C., Lei, Z.: HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In: CIKM (2017)
8. Grover, A., Leskovec, J.: node2vec: Scalable Feature Learning for Networks. In: KDD (2016)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Representation Learning on Graphs: Methods and Applications. arXiv preprint arXiv:1709.05584 (2017)
10. Håstad, J.: Some Optimal Inapproximability Results. Journal of the ACM **48**(4), 798–859 (2001)
11. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (1990)
12. Li, J., Chen, C., Tong, H., Liu, H.: Multi-Layered Network Embedding. In: SDM (2018)
13. Lian, D., Zheng, K., Zheng, V.W., Ge, Y., Cao, L., Tsang, I.W., Xie, X.: High-order Proximity Preserving Information Network Hashing. In: KDD (2018)
14. Liben-Nowell, D., Kleinberg, J.: The Link-Prediction Problem for Social Networks. Journal of the American Society for Information Science and Technology **58**(7), 1019–1031 (2007)
15. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender Systems with Social Regularization. In: WSDM (2011)
16. Opsahl, T., Panzarasa, P.: Clustering in Weighted Networks. Social Networks **31**(2), 155–163 (2009)
17. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online Learning of Social Representations. In: KDD (2014)
18. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In: WSDM (2018)
19. Razick, S., Magklaras, G., Donaldson, I.M.: iRefIndex: A Consolidated Protein Interaction Database with Provenance. BMC Bioinformatics **9**(1), 405 (2008)
20. Shen, X., Pan, S., Liu, W., Ong, Y.S., Sun, Q.S.: Discrete Network Embedding. In: IJCAI (2018)
21. Singh, A.P., Gordon, G.J.: Relational Learning via Collective Matrix Factorization. In: KDD (2008)
22. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. Proceedings of the VLDB Endowment **4**(11), 992–1003 (2011)
23. Tang, J., Qu, M., Mei, Q.: PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. In: KDD (2015)
24. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: Large-scale Information Network Embedding. In: WWW (2015)
25. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: Extraction and Mining of Academic Social Networks. In: KDD (2008)
26. Wang, J., Zhang, T., Sebe, N., Shen, H.T., et al.: A Survey on Learning to Hash. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(4), 769–790 (2018)
27. Watts, D.J., Strogatz, S.H.: Collective Dynamics of Small-World Networks. Nature **393**(6684), 440 (1998)
28. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: NIPS (2009)
29. Willett, P., Barnard, J.M., Downs, G.M.: Chemical Similarity Searching. Journal of Chemical Information and Computer Sciences **38**(6), 983–996 (1998)
30. Zhang, H., Shen, F., Liu, W., He, X., Luan, H., Chua, T.S.: Discrete Collaborative Filtering. In: SIGIR (2016)