Journal of Artificial Intelligence Research 71 (2021) 237-263 Submitted 12/2020; published 06/2021

RWNE: A Scalable Random-Walk-Based Network Embedding Framework with Personalized Higher-Order Proximity Preserved

Jianxin Li LIJX@ACT.BUAA.EDU.CN
Cheng Ji JICHENG@ACT.BUAA.EDU.CN
Hao Peng PENGHAO@ACT.BUAA.EDU.CN
Yu He HEYU@ACT.BUAA.EDU.CN
Beijing Advanced Innovation Center for Big Data and Brain Computing
Beihang University, China

Yangqiu Song YQSONG@CSE.UST.HK
Department CSE, HKUST, Hong Kong

Xinmiao Zhang ZHANGXINMIAO@BUAA.EDU.CN
Fanzhang Peng PENGFANZHANG@BUAA.EDU.CN
Beihang University, China

Abstract

Higher-order proximity preserved network embedding has attracted increasing attention. In particular, due to the superior scalability, random-walk-based network embedding has also been well developed, which could efficiently explore higher-order neighborhoods via multi-hop random walks. However, despite the success of current random-walk-based methods, most of them are usually not expressive enough to preserve the personalized higher-order proximity and lack a straightforward objective to theoretically articulate what and how network proximity is preserved. In this paper, to address the above issues, we present a general scalable random-walk-based network embedding framework, in which random walk is explicitly incorporated into a sound objective designed theoretically to preserve arbitrary higher-order proximity. Further, we introduce the random walk with restart process into the framework to naturally and effectively achieve personalized-weighted preservation of proximities of different orders. We conduct extensive experiments on several real-world networks and demonstrate that our proposed method consistently and substantially outperforms the state-of-the-art network embedding methods.

1. Introduction

Network embedding, which has recently attracted increasing attention in both academia and industry, is a general and fundamental technique for representing nodes of the real-word network as vectors in a low-dimensional space while preserving the inherent properties and structures of the network (Cui et al., 2019; Goyal & Ferrara, 2018; Cai et al., 2018). Such embedding vectors can then be used for a variety of network mining tasks, such as node profiling (classification and clustering) (Sen et al., 2008; Wang et al., 2017), link prediction (Shi et al., 2015; Wei et al., 2017), similarity search (Sun et al., 2011, 2012; Zhou et al., 2017), etc.

One basic requirement of network embedding is that the learned vectors of nodes should preserve the network structures. Along with this direction, many network embedding meth-

ods are proposed to preserve the first-order proximity which expresses the local pairwise structure indicated by the observed edges between nodes (e.g., Roweis & Saul, 2000; Belkin & Niyogi, 2001; Tang & Liu, 2011; Ahmed et al., 2013), or further to preserve the second-order proximity between a pair of nodes which implies the similarity between their neighborhood structures (e.g., Tang et al., 2015; Wang et al., 2016).

Despite their success, in recent years, more and more works (Cao et al., 2015; Ou et al., 2016; Yang et al., 2017; Zhang et al., 2018) have demonstrated that, besides the first- and second- order proximity directly indicated by pairwise edges, the higher-order proximity is also one of tremendous importance in capturing the underlying structures of the network.

- **Different-order proximities describe the network structures from different levels of scope, which give us much valuable information with different granularities.** Thus, embeddings with the lower-order proximity alone or even any certain-order proximity do not necessarily perform best on all networks and target applications (Perozzi et al., 2017; Zhang et al., 2018). For example, in classification tasks with coarse-grained classes, the higher-order proximity is likely to be more helpful than lower-order proximity.

- **Real-world networks are usually very sparse, with only a small number of edges observed.** That is, the observed first-order proximity and even second-order proximity may not be sufficient to reflect the underlying relations between nodes (Tang et al., 2015). Therefore, to address the network sparsity issue, it is also very important to incorporate higher-order proximity to capture more available information.

Although many works have been proposed to preserve the higher-order proximity in network embedding, most of them are developed to explicitly exploit the higher-order proximity matrix by the technique of matrix factorization (e.g., Cao et al., 2015; Ou et al., 2016; Yang et al., 2017) or deep learning (e.g., Wang et al., 2016; Cao et al., 2016), which are known to have scalability issues when dealing with large-scale networks (Yang et al., 2017).

To be more efficient, inspired by the Skip-gram algorithm (Mikolov et al., 2013), random-walk-based network embedding algorithms have also been well developed (Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016; Perozzi et al., 2017; He et al., 2019). Although these random-walk-based algorithms are known for having superior scalability for large-scale networks and having the ability for exploring higher-order neighborhood via multi-hop random walks, there are still some issues:

- They either treat different-order neighborhood equivalently (Perozzi et al., 2014) which may not be expressive enough to incorporate a personalized combination of proximities of different orders, or use a somewhat complex second-order biased random walk (Grover & Leskovec, 2016) which can be costly when pre-computing its third-order transition probability hypermatrix (Zhang et al., 2018).

- In essence, these algorithms convert the network embedding problem as the word embedding problem by treating a node as a word and pre-generating the node "corpus" via random walks. As a consequence, they have no specific objective to articulate what and how network proximity is preserved and have no sound theory to estimate the essential role of random walk playing in network embedding, which also limits their superiority.

In this paper, to address the above issues, we present a general scalable **R**andom-**W**alk-based **N**etwork **E**mbedding framework (called ***RWNE***), in which arbitrary higher-order proximity of the network can be explicitly preserved with a sound objective carefully designed to simultaneously capture both the local pairwise similarity and the global listwise equivalence between nodes. More than that, to make the framework efficient and practical for large-scale networks, we theoretically show that the above objective can be equivalently optimized by sampling the nodes via the random walk process with the probability proportional to the proximity, which clarifies why and how we can use random walks to preserve arbitrary user-specified network proximity and reversely explains what and how network proximity is preserved for an arbitrary user-specified random walk. Further, we introduce the random walk with restart process to naturally and effectively achieve personalized-weighted preservation of different-order proximities with an elegant attenuation function controlled by a personalized teleport probability. Finally, we conduct extensive experiments on six real-world networks over three classical network mining tasks: multi-label node classification, node clustering, and link reconstruction. The experimental results demonstrate that our proposed method consistently and substantially outperforms the state-of-the-art network embedding methods. To summarize, the main contributions of our work are as follows:

1. We systematically present a general scalable random-walk-based network embedding framework *RWNE*[*], in which random walk is efficiently and explicitly incorporated into a sound objective designed theoretically to preserve arbitrary higher-order proximity.
2. We further introduce the random walk with restart process to practically preserve the personalized higher-order proximity which naturally weights different-order proximities with an elegant attenuation function controlled by a personalized teleport probability.
3. We conduct extensive experiments on several real-world networks and demonstrate that our proposed model consistently and considerably outperforms the state-of-the-art network embedding methods.

2. Related Work

Network embedding has aroused lots of research interest for a long time (Cui et al., 2019; Cai et al., 2018; Wang et al., 2019). The earlier network embedding algorithms, also called graph embedding, are studied as a dimension reduction problem, such as LLE (Roweis & Saul, 2000), Laplacian eigenmaps (Belkin & Niyogi, 2003), etc. These methods focus on the first-order proximity which captures the local structure information of the network.

Matrix Factorization. Recently, to sufficiently explore the network structure from different levels of scope, a bunch of methods has been proposed to preserve the higher-order proximity in network embedding. Most of these methods are proposed to explicitly factorize a higher-order proximity matrix, such as GraRep (Cao et al., 2015), HOPE (Ou et al., 2016), M-NMF (Wang et al., 2017), NetMF (Qiu et al., 2018), AROPE (Zhang et al., 2018), etc. However, in principle, as the computation and storage of higher-order proximity

*. The datasets and code are released at `https://github.com/RingBDStack/RWNE`.

matrices are generally at least $O(|V|^2)$ complexity, these matrix-factorization methods often have efficiency issues when dealing with large-scale networks.

Deep Learning. Besides, deep learning is also studied in preserving higher-order proximity. For example, SDNE (Wang et al., 2016) first applies a deep auto-encoder to preserve both 1st- and 2nd-order proximity. DNGR (Cao et al., 2016) further uses a stacked denoising auto-encoder to preserve higher-order proximity. Unfortunately, same as matrix-factorization methods, these methods also confront efficiency issues. Specially, deep convolution networks are popularly applied on graphs in very recent years (e.g., Kipf & Welling, 2017; Velickovic et al., 2018; Peng et al., 2019), which are studied as supervised/semi-supervised feature learning models with node attributes/features incorporated. We also compare with these methods, but it is noteworthy that in this paper we focus on the most fundamental case that only the network structure information is available.

Random Walk. On the other hand, due to the superior scalability, random-walk-based network embedding algorithms have also been well developed. A two-step framework is applied for generating the node embeddings in these methods. First, they perform random walks on a network to generate node sequences. Then they run the Skip-gram algorithm over these sequences to generate node embeddings. For example, DeepWalk (Perozzi et al., 2014) uses uniform random walks to generate node sequences and then runs the Skip-gram algorithm. The major drawback of DeepWalk is that it treats different-order neighborhoods equivalently and thus maybe not expressive enough to incorporate a personalized combination of proximities of different orders. node2vec (Grover & Leskovec, 2016) further generalizes a second-order biased random walk to seek a trade-off between breadth-first and depth-first graph searches. Although node2vec can incorporate a biased combination of different-order proximities, it is usually costly for computing the second-order transition probability hypermatrix of the proposed second-order random walk. Moreover, despite their success, as introduced in Section 1, there are still some issues. In this paper, instead of the above two-step framework, we focus on a straightforward framework by explicitly incorporating random walk into a sound objective designed theoretically to preserve personalized higher-order proximity.

3. Definitions and Preliminaries

In this section, we first introduce the problem of network embedding, and then formally define the measures of higher-order proximity to characterize network structures. We first formalize a network as follows.

Network. *A network is defined as a directed graph $G=(V,E)$, where $V=\{v_1,v_2,\cdots,v_n\}$ is the set of nodes and n is its size, $E=\{e_{i,j}\}_{i,j=1}^{n}$ is the set of edges. Each edge $e_{i,j}$ is equally as a linked node pair (v_i,v_j) and is associated with a weight $w_{i,j}\geqslant 0$, which indicates the strength of the edge. If there is no observed edge from v_i to v_j, then $w_{i,j}=0$. Specially, for an undirected graph, we have $w_{i,j}\equiv w_{j,i}$. For an unweighted graph, we have $w_{i,j}\in\{0,1\}$.*

We further denote $A\in\mathbb{R}^{n\times n}$ as the adjacency matrix of the network, where each entry $A_{i,j}=w_{i,j}$ represents the weight of the edge from v_i to v_j; denote $D\in\mathbb{R}^{n\times n}$ as the weight matrix where each diagonal entry $D_{i,i}=\sum_{j=1}^{n}A_{i,j}$ represents the summed "out-weight" of v_i and other entries are zero. Then, $\Lambda=D^{-1}A$ is the normalized adjacency matrix, i.e., the

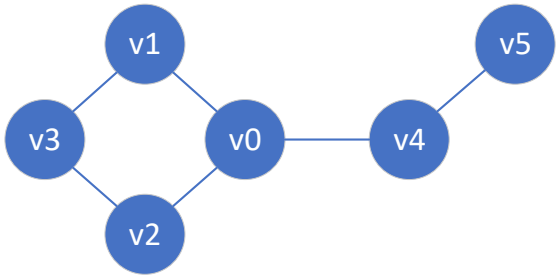


Figure 1: An illustrative network G_e with six nodes.

one-step transition probability matrix for random walks on the network, where the sum of each row is equal to one.

Example 1 *Take a simple graph G_e=(V, E) shown in Figure 1 as an example, with $w_{i,j} \equiv w_{j,i}$=1. As defined above, we have*

$\mathcal{V} = \{v_0, v_1, v_2, v_3, v_4, v_5\}$,

$\mathcal{E} = \{e_{0,1}, e_{0,2}, e_{0,4}, e_{1,0}, e_{1,3}, e_{2,0}, e_{2,3}, e_{3,1}, e_{3,2}, e_{4,0}, e_{4,5}, e_{5,4}\}$.

The adjacency matrix and weight matrix are

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thus, the normalized adjacency matrix can be calculated as

$$\Lambda = D^{-1}A = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Network Embedding. *Given a network G=(V, E), the problem of network embedding aims to embed the network into a low-dimensional space while preserving the network structures, i.e., learn a mapping function $f_G : v_i \in V \to r_i \in \mathbb{R}^d$, where $d \ll |V|$ and f_G preserves the relations between nodes.*

As defined above, to conduct the embedding, the network structures, i.e., the relations between nodes, must be preserved as much as possible. In practice, we adopt the following proximity measures to quantify the network structure information to be preserved in the embedded space. We first define the first-order proximity as follows.

First-Order Proximity. *The first-order proximity is defined to measure the adjacent structures of a network. For each pair of nodes (v_i, v_j), if they are linked by an edge, there*

exists positive first-order proximity between them, and the weight $w_{i,j}$ on that edge indicates the strength of the proximity. If no edge is observed between v_i and v_j, their first-order proximity is zero. Further, for all pairs of nodes, there is naturally a first-order proximity matrix indicated by the adjacency matrix of the network.

In practice, since the weights of edges in a network can diverge over a very wide range, we use the normalized adjacency matrix (i.e., the one-step transition probability matrix) Λ as the formal first-order proximity matrix, where each normalized entry $\Lambda_{i,j}$ is the first-order proximity of node pair (v_i, v_j), which also represents the transition probability of one-step random walk from v_i to v_j. It is necessary to consider the structural characteristics of the network from local and global perspectives, which has been proved by many researches, such as feature selection (Liu et al., 2014), semi-supervised classification (Kang et al., 2020b, 2021), clustering (Ren & Sun, 2020; Kang et al., 2020a), etc. The defined first-order proximity can measure the adjacent structures in both local and global aspects:

- In local aspect, the first-order proximity implies that two nodes are similar if they are linked by an observed edge. For example, bloggers following each other in a social network tend to share similar interests; papers citing to each other in a citation network tend to talk about similar topics.

- In global aspect, the relation between two nodes is also determined by their common neighbors. For example, people sharing many common friends in a social network are likely to share similar interest and become friends. That is, even if two nodes are not directly connected, we can capture their relation through their neighbors.

Specially, we redefine such local similarity between each two nodes (v_i, v_j) directly determined by their first-order proximity entry $\Lambda_{i,j}$ as the first-order local proximity. By preserving the first-order local proximity, we are able to characterize the local adjacent structure of the network.

For further convenience, we here define two vectors: $\Lambda_i = (\Lambda_{i,1}, \Lambda_{i,2}, \cdots, \Lambda_{i,|V|})$, and $\Lambda_{\cdot,i} = (\Lambda_{1,i}, \Lambda_{2,i}, \cdots, \Lambda_{|V|,i})$. More formally, for three nodes v_i, v_{j_1}, v_{j_2}, if their first-order proximities satisfy $\Lambda_{i,j_1}{=}\Lambda_{i,j_2}$, i.e., v_i randomly walks to v_{j_1} and v_{j_2} with the same probability, then v_{j_1} and v_{j_2} share the equivalent role for v_i. Further, if for each one of the entire node set, v_{j_1} and v_{j_2} always share the equivalent role, i.e., $\Lambda_{\cdot,j_1} \equiv \Lambda_{\cdot,j_2}$, then v_{j_1} and v_{j_2} have an equivalent global structure role in the network. For example, if there are such two papers that all the papers in the citation network who cite one of them will also cite the other, the two papers are very probably to talk about the same topic and have the equivalent significance in academia. Therefore, we can capture the global adjacent structure by preserving such global equivalence between nodes. Specially, we refer to such global equivalence between each two nodes (v_{j_1}, v_{j_2}) determined by the similarity of the two vectors $\Lambda_{\cdot,j_1}$ and $\Lambda_{\cdot,j_2}$ as the first-order global proximity.

Example 2 *For the node v_1 and node v_2 in G_e, although the two nodes are not directly connected, they should have stronger similarities because they have multiple common neighbors node v_0 and node v_3, which can be judged by the first-order global proximity:*

$$\Lambda_{\cdot,v_1} = \Lambda_{\cdot,v_2} = \begin{bmatrix} 1/3 & 0 & 0 & 1/2 & 0 & 0 \end{bmatrix}.$$

Intuitively, it is necessary for network embedding to preserve both the first-order local proximity and the first-order global proximity. The two proximities characterize the adjacent structures of the network in local and global aspects, respectively. However, there are more non-adjacent structures in the network that cannot be described by first-order proximity. For example, two nodes can be similar even if they neither have a local edge nor a common neighbor. Actually, in many real-world networks, the adjacent structures observed are only a small proportion, with many others missing (Tang et al., 2015). That is, the first-order proximity matrix is usually sparse and thus is not sufficient to capture network structures. To address the sparsity, the higher-order proximity must be preserved. In fact, even if two nodes have no edge or neighbor, they can also be related if there is a path (i.e., a sequence of directed edges) between them, which could be regarded as a long-distance "edge". Therefore, we define the higher-order proximity as follows.

Higher-Order Proximity. *The higher-order proximity is defined to measure the long-distance structures of a network. For each pair of nodes (v_i, v_j), if they are linked by a path, i.e., we can walk from v_i to v_j along several edges, they have a higher-order proximity. Specially, if the length of the path is k, they have a k-th order proximity; if there is no k-length path between them, their k-th order proximity is* 0.

Similar to the first-order proximity which represents the one-step transition probability, we can use the probability of walking from v_i to v_j along paths to represent the strength of the higher-order proximity. Specially, the k-th order proximity matrix can be computed as

$$\Lambda^k = \underbrace{\Lambda \cdot \Lambda \dots \Lambda}_{k}, \quad k = 2, 3, \cdots, \tag{1}$$

where each entry $\Lambda^k_{i,j}$ is the k-th order proximity of node pair (v_i, v_j), which also represents the k-step transition probability from v_i to v_j. Also similar to the first-order proximity, the k-th order proximity can be further derived as k-th order local proximity and k-th order global proximity to characterize "k-th order" network structures (i.e., the non-adjacent relations between nodes linked by k-length paths) in local and global aspects, respectively. For each pair of nodes (v_i, v_j), the k-th order local proximity represents their local similarity directly determined by the entry $\Lambda^k_{i,j}$, and the k-th order global proximity represents their global equivalence determined by the similarity of $\Lambda^k_{\cdot,i}$ and $\Lambda^k_{\cdot,j}$.

Example 3 *The higher-order proximity matrices of G_e are given here when k equals to 3 and 5, with the case when k equals to 1 shown in Example 1:*

$$\Lambda^3 = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \\ 1/2 & 0 & 0 & 5/12 & 0 & 1/12 \\ 1/2 & 0 & 0 & 5/12 & 0 & 1/12 \\ 0 & 5/12 & 5/12 & 0 & 1/6 & 0 \\ 1/2 & 0 & 0 & 1/6 & 0 & 1/3 \\ 0 & 1/6 & 1/6 & 0 & 2/3 & 0 \end{bmatrix}, \Lambda^5 = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \\ 1/2 & 0 & 0 & 3/8 & 0 & 1/8 \\ 1/2 & 0 & 0 & 3/8 & 0 & 1/8 \\ 0 & 3/8 & 3/8 & 0 & 1/4 & 0 \\ 1/2 & 0 & 0 & 1/4 & 0 & 1/4 \\ 0 & 1/4 & 1/4 & 0 & 1/2 & 0 \end{bmatrix}.$$

For instance, it is directly observed that the node v_1 and node v_5 are related under the view of 3-rd order proximity and 5-th order proximity, which cannot be captured by the first-order proximity (adjacency matrix) for the two nodes are not actually connected in the graph.

Moreover, it can be noticed that the relation is different under Λ^3 and Λ^5, which indicates that different higher-order proximity holds different structural semantics.

Note that first-order proximity is the special case of k-th order proximity when $k=1$. In this paper, without loss of generality, we use "the higher-order proximity" to represent an unspecified k-th order proximity with $k>1$, and use "high-order proximities" to represent the combination of 1st-, 2nd-, $\cdots$, and k-th order proximities. By preserving these high-order proximities, we are able to sufficiently characterize both adjacent and long-distance network structures in both local and global aspects, which is very effective in solving the sparsity problem and achieving high-quality embeddings.

4. The RWNE Framework

In this section, we present a general scalable embedding framework *RWNE* for network embedding and further introduce the optimization method with random walk simulation.

4.1 Problem Formulation

We formulate the normalized adjacency matrix (denoted as A in this section) as the first-order proximity matrix, which captures the direct neighbor relations between nodes (Tang et al., 2015; Yang et al., 2017). Specially, such first-order proximity can be alternatively viewed as the transition probability of a single step of the random walk over the network. Then, in the probabilistic setting based on random walk, we can easily generalize it to k-th order proximity A^k (Cao et al., 2015): the transition probability of a random walk with exactly k steps, which represents the k-hop relations between nodes. As proximities of different orders explore the relations from different levels of scope, which all can provide valuable information to guide the embedding, a desirable embedding model for real-world networks must be capable of preserving a delicate integrated higher-order proximity which combining the proximities of different orders as follows:

$$D = \beta_1 A + \beta_2 A^2 + \cdots + \beta_k A^k, \quad k = 1, 2, \cdots, \infty, \tag{2}$$

where β_k is the weight to control the prestige of k-th order proximity A^k, and the sum of all weights $\sum_{i=1}^{k} \beta_i$=1.

Example 4 *In the case where $k = 5$ and $\beta_1 = \cdots = \beta_5 = 0.2$, the integrated higher-order proximity matrix of G_e is*

$$D = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.13 & 0.2 & 0.07 \\ 0.3 & 0.16 & 0.16 & 0.26 & 0.08 & 0.04 \\ 0.3 & 0.16 & 0.16 & 0.26 & 0.08 & 0.04 \\ 0.2 & 0.26 & 0.26 & 0.18 & 0.08 & 0.02 \\ 0.3 & 0.08 & 0.08 & 0.08 & 0.23 & 0.22 \\ 0.2 & 0.08 & 0.08 & 0.03 & 0.43 & 0.17 \end{bmatrix}.$$

In the rest of this section, we will systematically present a general scalable random-walk-based network embedding framework, called *RWNE*, which is able to effectively and

efficiently preserve the above integrated higher-order proximity. Without loss of generality, we first describe the *RWNE* model to directly preserve a general form of higher-order proximity D with arbitrary weights. Then, we show a superior optimization with random-walk simulation, which makes the model computationally efficient and scalable for large-scale networks. We further introduce the random walk with restart process to naturally adjust the prestige of different-order proximities by a personalized teleport probability. Finally, we briefly analyze the time complexity of *RWNE*.

4.2 Preserving Higher-Order Proximity

The higher-order proximity is divided into two parts in our work: the local pairwise similarity and the global listwise equivalence. To preserve both two aspects, we further design a joint objective function for our model.

4.2.1 Local Pairwise Similarity

Given a higher-order proximity matrix D integrating all orders from the 1-st to the k-th as defined in Eq. (2), it is straightforward that each entry D_{ij} implies the local pairwise similarity between each pair of nodes (v_i, v_j) in view of the integrated relation from different levels of scope (from 1-hop to k-hop). If $D_{ij} > 0$, there is a similarity between v_i and v_j, and the larger D_{ij} is, the more similar v_i and v_j are; If $D_{ij} = 0$, they have no similarity. Therefore, we can directly use each entry D_{ij} to constrain the similarity of the embedding vectors of each pair of nodes (v_i, v_j). Before that, we first define a normalized cosine distance as follows, which is used to measure the similarity of the embedding vectors:

$$sim(i,j) = \frac{1}{2}(1 + \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\parallel \mathbf{v}_i \parallel \parallel \mathbf{v}_j \parallel}), \tag{3}$$

where $\mathbf{v}_i$ is the embedding vector of node v_i. Then, instead of a naive treatment which rigidly sets D_{ij} as the target similarity and minimizes the error loss between D_{ij} and $sim(i,j)$, we propose a new loss to measure the similarity cost of each pair of nodes (v_i, v_j) in the embedding space:

$$l(i,j) = \begin{cases} -D_{ij} \log sim(i,j), & D_{ij} > 0 \\ -\log(1 - sim(i,j)), & D_{ij} = 0 \end{cases}. \tag{4}$$

There are two subtleties in our careful design of Eq. (4). First, though under normalization, one proximity may have a different weight from one similarity score, which means rather than treating proximity as the exact target similarity, it can only be concluded that the larger proximity is, the more similarity is. And that is why we abandon the aforementioned naive treatment. In our design, we impose a penalty to push v_i and v_j embedded similarly and set D_{ij} as the penalty coefficient to guarantee that larger proximity will incur more penalty and thus generate a stronger push to be similar. Second, there is an exception that zero proximity expresses the dissimilarity, which is essentially different from positive proximity. Thus we separate out the zero proximity and set an opposite penalty for it, i.e., impose a penalty to push v_i and v_j embedded dissimilarly.

Then, by using the loss of Eq. (4) for all pairs of nodes, the objective function to preserve the local similarity is defined as follows:

$$\mathcal{L}_l = \sum_{i,j\in\{D_{ij}>0\}} \left(-D_{ij}\log sim(i,j)\right) + \sum_{i,j\in\{D_{ij}=0\}} \left(-\log(1-sim(i,j))\right). \tag{5}$$

Minimizing Eq. (5) pushes the similarity of the embedding vectors of v_i and v_j towards 1 if proximity D_{ij} is large (the larger the proximity, the stronger the push) and pushes it towards 0 if $D_{ij}=0$. As a result, we preserve the local pairwise similarity between nodes.

4.2.2 Global Listwise Equivalence

In addition to the local similarity, we can extract more information by considering the relative equivalence. For instance, for three nodes v_i, v_j, v_k with $D_{ij}=D_{ik}=D_{jk}=0.1$, the local similarity between v_i and v_j may be very weak in view of the strength of D_{ij}. However, in a relative view, we can conclude that v_i and v_j is equivalent in v_k's viewpoint, because they distribute the equal proximity with v_k. Further, if for the listwise proximity distribution $D_i=(D_{i,1}, D_{i,2}, \cdots, D_{i,|V|})$ and $D_j=(D_{j,1}, D_{j,2}, \cdots, D_{j,|V|})$, v_i and v_j always have the above equivalence (i.e., $D_{ik}\equiv D_{jk}$ for $\forall k=1,\cdots,|V|$), we can conclude that v_i and v_j play an equivalent global structure role in the whole network. We refer to the above information as the global listwise equivalence. Note that some works (Tang et al., 2015) extract similar information from first-order proximity. We differentiate them in the aspects where we provide a more definite meaning and generalize it as a basic attribute of arbitrary higher-order proximity.

To preserve the global listwise equivalence, we propose a self-supervised component: the proximity predictor $\Phi(\cdot)$, which is a deep architecture composed of multiple nonlinear functions to predict the proximity distribution of an input node. As shown in Figure 2, for each node v_i, the predictor $\Phi(\cdot)$ uses the embedding vector $\mathbf{v}_i$ as the input and output a $|V|$-dimensional distribution $\Phi(i)\in\mathbb{R}^{1\times|V|}$, which is the predicted approximation of proximity distribution D_i. Then, supervised by D_i, the predictor $\Phi(\cdot)$ is trained to make the output $\Phi(i)$ be close to D_i, which means that if we pick two nodes v_i and v_j with similar proximity distributions (i.e. $D_i=D_j$), the predictor $\Phi(\cdot)$ will be trained to learn similar outputs (i.e. $\Phi(i)=\Phi(j)$) and thus to push the input embedding vector $\mathbf{v}_i$ and $\mathbf{v}_j$ to be similar. Therefore, by modeling the proximity in this way, we can learn similar embeddings for nodes with similar proximity distributions. That is, we capture the global listwise equivalence between nodes.

To train the predictor $\Phi(\cdot)$, we follow the inspiration of the design in Eq. (4) which sets the target proximity as a penalty coefficient and separates zero proximity from positive proximity, and design the loss to measure the prediction cost for each node v_i as follows:

$$\begin{aligned} g(i) &= -D_i\left[D_i>0\right]\log\Phi(i) - \mathbf{1}\cdot\left[D_i=0\right]\cdot\log(1-\Phi(i)) \\ &= \sum_{j\in\{D_{ij}>0\}} \left(-D_{ij}\log\Phi(i)_j\right) + \sum_{j\in\{D_{ij}=0\}} \left(-\log(1-\Phi(i)_j)\right). \end{aligned} \tag{6}$$

Note that in principle, $\Phi(\cdot)$ can be an arbitrarily deep neural network, but as we focus on the effort of random walk in this paper, we only use a simple single-layer architecture

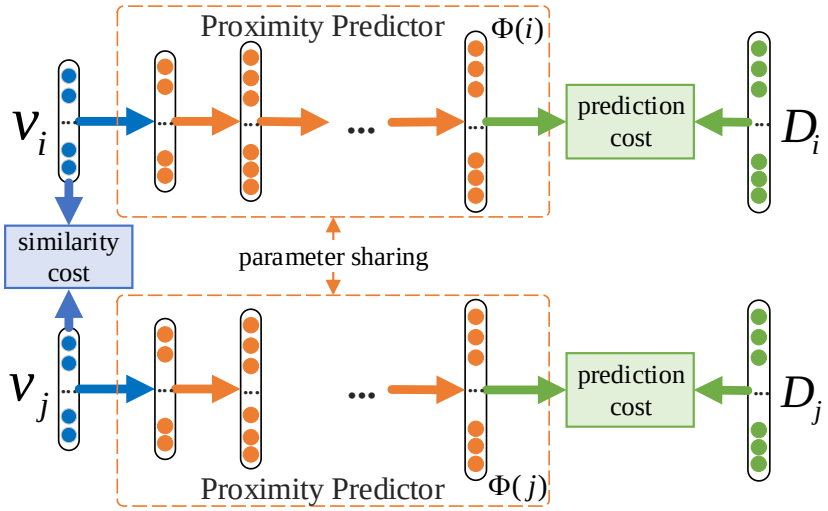


Figure 2: An illustration of the framework of *RWNE* without random walk simulation.

and leave a superior trying as future work. In this setting, the proximity predictor $\Phi(\cdot)$ is defined as

$$\Phi(i) = \sigma(\mathbf{W}\mathbf{v}_i), \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{|V|\times d}$, V is the nodes set, d is the embedding dimensionality, and $\sigma(\cdot)$ is the logistic function.

Finally, by using the loss of Eq. (6) for all nodes, the objective function to preserve the global equivalence is defined as follows:

$$\mathcal{L}_g = \sum_{i}^{|V|} \left(\sum_{j\in\{D_{ij}>0\}} (-D_{ij} \log \Phi(i)_j) + \sum_{j\in\{D_{ij}=0\}} (-\log(1-\Phi(i)_j)) \right). \tag{8}$$

4.2.3 The Joint Objective

To simultaneously preserve both the local pairwise similarity and the global listwise equivalence provided by the higher-order proximity D, we jointly minimize the following objective function, which combines Eq. (5) and Eq. (8):

$$\mathcal{L} = \gamma\mathcal{L}_l + \lambda\mathcal{L}_g = \sum_{i}^{|V|} \left(\sum_{j\in\{D_{ij}>0\}} D_{ij}\ell(i,j) + \sum_{j\in\{D_{ij}=0\}} \zeta(i,j) \right), \tag{9}$$

where $\ell(i,j) = -(\gamma \log sim(i,j) + \lambda \log \Phi(i)_j)$ is the loss for positive proximity, $\zeta(i,j) = -(\gamma \log(1-sim(i,j)) + \lambda \log(1-\Phi(i)_j)$ is the loss for zero proximity; γ and λ are hyper-parameters to reflect user's emphasis.

4.3 Optimization with Random Walk Simulation

In practice, an accurate computation of Eq. (2) to get the higher-order proximity matrix D with $k \geq 2$ is both time and space consuming. Thus it is undesirable to directly solve the objective of Eq. (9) due to the efficiency issue, especially for large-scale networks.

To address the above issue, we highlight that D is essentially an integrated probability matrix combining the transition probabilities of a random walk in 1-st, 2-nd, $\cdots$, k-th step with corresponding weights $\beta_1, \beta_2, \cdots, \beta_k$, where each entry $D_{ij} = \beta_1 A_{ij} + \beta_2 A_{ij}^2 + \cdots + \beta_k A_{ij}^k$ represents the weighted average hitting probability from v_i to v_j within a k-steps random walk. Then, without calculating Eq. (2), we can invent a k-steps "drop-out" random walk to simulate the probability matrix D: in l-th step (for $\forall l = 1, 2, \cdots, k$), the walker first randomly moves from the current node to an adjacent node as a normal random walk, and then randomly drops out the current-step hitting node with the dropping probability $1-\beta_l$. By this means, the expected hitting probability from v_i to v_j in k steps is exactly D_{ij}. That is to say, for each node v_i, the above k-steps "drop-out" random walker starting from v_i hits/samples a paired node v_j with the probability D_{ij}.

In our carefully designed objective of Eq. (9), the probability D_{ij} is delicately and theoretically arranged as a weight coefficient. Then, in the probabilistic setting, we can equivalently treat the probabilistic real weight as a binary weight by node-sampling treatment, with the sampling probability proportional to the original real weight. That is, we can eliminate the D_{ij} in the first term of Eq. (9) by sampling nodes with the probability D_{ij}, which can be simulated by the aforementioned k-steps "drop-out" random walks starting from v_i. Furthermore, for the second term $\sum_{j \in \{D_{ij}=0\}} \zeta(i,j)$ in Eq. (9), as the matrix D is usually very sparse with many zero entries, we also leverage an uniform-sampling treatment to optimize it. Overall, with the node-sampling treatment, the objective function of Eq. (9) is optimized as

$$\begin{aligned} \mathcal{L} &= \sum_{i}^{|V|} \left(\sum_{t=1}^{T} \mathbb{E}_{j \sim \{D_{i,j}\}_1^{|V|}} \ell(i,j) + \sum_{t=1}^{T} \mathbb{E}_{j' \sim \mathbf{I}/\{D_{i,j'}>0\}} \zeta(i,j') \right) \\ &= \sum_{i}^{|V|} \sum_{t=1}^{T} \left(\sum_{j \in p_i^{1 \to k}} \ell(i,j) + \sum_{m=1}^{|p_i^{1 \to k}|} \mathbb{E}_{j' \sim \mathbf{I}/\{p_i^{1 \to k}\}} \zeta(i,j') \right), \end{aligned} \tag{10}$$

where T is the sampling-frequency/walk-times which can be set as the iteration epochs when solving the objective by an iterative algorithm (e.g. SGD); $p_i^{1 \to k}$ is the hitting nodes set in a k-steps "drop-out" random walk starting from v_i; $\mathbb{E}_{j' \sim \mathbf{I}/\{p_i^{1 \to k}\}}$ means an uniform-sampling with nodes $\{p_i^{1 \to k}\}$ excluded. Note that in each walk, as the random walker samples $|p_i^{1 \to k}|$ nodes, we also operate uniform-sampling for the equal times to ensure fairness.

The optimized objective of Eq. (10) replaces the expensive computation of higher-order proximity with random walk simulation, and thus has superior efficiency and scalability, and can be easily mini-batch-minimized by applying an iterative algorithm. So far, we have theoretically described a general scalable random-walk-based embedding framework. For arbitrary user-specified weights $\{\beta_1, \beta_2, \cdots, \beta_k, \cdots\}$, the framework is able to effectively and efficiently preserve the weighted combination of different-order proximities by deploying the aforementioned "drop-out" random walk. Reversely, for an arbitrary user-specified random walk, the framework can also be used by directly minimizing the Eq. (10), However,

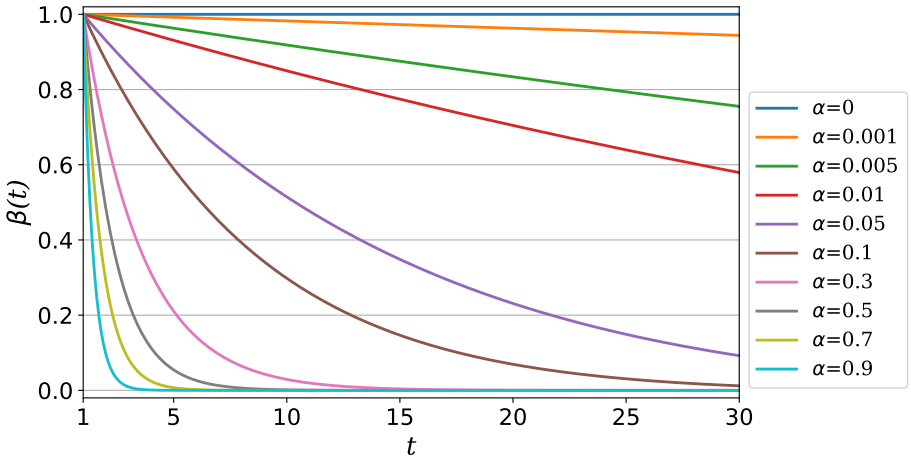


Figure 3: The decline curves of $\beta(t) = \frac{(1+\alpha(k-t))(1-\alpha)^t}{k}$ in different α. The k is set as 100 and each curve is normalized by $\beta(1)$.

superior to other random-walk models, the framework explicitly clarifies what and how network proximity is preserved in random-walk structure, that is the average hitting/sampling probability in k steps of the user-specified random walk is just the integrated higher-order proximity which is theoretically preserved by the objective of Eq. (9).

4.3.1 Random Walk with Restart

Although we have theoretically described the *RWNE* framework, it is still limited because there is no principled way to determine desirable weights $\{\beta_1, \beta_2, \cdots, \beta_k, \cdots\}$ or random walks. In general, it is intuitive that the relation/influence over a very long distance can be very weak, i.e., the prestige of different-order proximity is likely to decay with the distance. Therefore, instead of a normal random walk widely used in the existing models, we introduce the random walk with restart (referred to as RWR) process: in each step of a random walk, the walker can return back to the root with a personalized teleport probability α. The transition probability of RWR is recurrently formalized as

$$P^k = \alpha I + (1-\alpha)P^{k-1}A, \ k = 1, 2, \cdots, \tag{11}$$

where P^k is the k-th step transition probability matrix of RWR, and $P^0 = I$ is an identity matrix; A is the single-step transition probability matrix of normal random walk, i.e., the first-order proximity matrix. Equivalently, we have

$$P^k = (1-\alpha)^k A^k + \sum_{t=1}^{k} \alpha(1-\alpha)^{k-t} A^{k-t}, \ k = 1, 2, \cdots. \tag{12}$$

Thus, by deploying the RWR process, the integrated higher-order proximity matrix to be preserved (i.e. the average hitting/sampling probability matrix) in k steps is derived as

$$D = \frac{1}{k}\sum_{t=1}^{k} P^t = \sum_{t=1}^{k} \frac{(1+\alpha(k-t))(1-\alpha)^t}{k} A^t, \tag{13}$$

in which the proximities of different-orders are weighted with a function $\beta(t)=\frac{(1+\alpha(k-t))(1-\alpha)^t}{k}$. As Figure 3 shows, $\beta(t)$ is approximately an exponentially decreasing function of t, since $\alpha<1$. Compared with other common decay functions, $\beta(t)$ holds the following advantages: (1) The decay rate is first rapid and then slow down as the order t increases, which makes the influence of extreme-high-order proximity smaller. Considering the natural property that the higher-order proximity has less influence in most cases, it is thus reasonable to use $\beta(t)$ instead of the linear or Gaussian decay functions. (2) The decay rate is naturally applied to the RWR process and can be freely adjusted by the personalized teleport probability α, while other exponential decay functions are hard to adapt to the random walk process with sufficient theoretical basis.

Finally, by deploying the RWR process, *RWNE* is able to effectively and efficiently preserve personalized higher-order proximity, as shown in Eq. (13) in which the proximities of different orders are naturally weighted with an elegant attenuation function controlled by a personalized teleport probability. The final objective of *RWNE* by applying the RWR process is as follows:

$$\mathcal{L}=\sum_{i}^{|V|}\sum_{t=1}^{T}\left(\sum_{j\in rwr_i^{1\to k}}\ell(i,j)+\sum_{m=1}^{|rwr_i^{1\to k}|}\mathbb{E}_{j'\sim\mathbf{I}/\{rwr_i^{1\to k}\}}\zeta(i,j')\right), \tag{14}$$

where $rwr_i^{1\to k}$ is the hitting nodes set in a k-steps random walk with restart process starting from v_i; $\mathbb{E}_{j'\sim\mathbf{I}/\{rwr_i^{1\to k}\}}$ means an uniform-sampling with nodes $rwr_i^{1\to k}$ excluded.

4.4 Complexity Analysis

In this section, we discuss the time complexity of *RWNE*. We use iterative algorithms (e.g. mini-batch SGD) to minimize the objective function (as shown in Eq. (14)). In each iteration, we only consider a single root node and deploy a k-steps random walk (with restart) starting from it to sample k paired nodes. As random walk takes only constant time in each step (even in a weighted network, we can use Alias sampling (Walker, 1977) to perform a random walk in $O(1)$ time) and the computation of the loss for each pair of nodes also takes constant time, we can see that the complexity of each iteration (i.e., the complexity of the inner-body in Eq. (14)) is $O(k)$. Then, given the network size $|V|$ (i.e., the number of nodes) and the iteration epochs T, we can extract that the overall time complexity of *RWNE* is $O(k\cdot T\cdot|V|)$, which is linear to the number of nodes in the network. Therefore, the proposed *RWNE* model is computationally efficient and scalable for large-scale networks.

4.5 Model Generalization Analysis

In this section, we extend *RWNE* model as a general framework, in which many popular network embedding approaches are unified. Under our general framework, it is straightforward to analyze these methods' essences or drawbacks.

SpectralClustering (Tang & Liu, 2011). This method computes the d-smallest eigenvectors of normalized Laplacian matrix L as d-dimensional network representations, which essentially is a dimensionality reduction based on Laplacian Eigenmaps. As we also utilize Laplacian Eigenmaps to embed the local pairwise proximity in *RWNE*, it is straightforward

that SpectralClustering is a degraded case of *RWNE* by directly decompose the first-order proximity matrix. Actually, all the network embedding approaches based on the decomposition of adjacent matrix or Laplacian matrix could be substantially regarded as special cases of *RWNE*, which only embed the local pairwise structure information of first-order or higher-order proximity matrix. Comparing with direct matrix decomposition, the *RWNE* framework provides a computationally efficient and scalable way to equivalently embed local pairwise proximity.

DeepWalk (Perozzi et al., 2014) and **node2vec** (Grover & Leskovec, 2016). DeepWalk transforms a network structure into node sequences by random walks, then treats the sequences as "corpus" and employs the Skip-gram model for representation learning. In terms of employing Skip-gram with negative sampling and node-sampling via random walk, DeepWalk is a special case of *RWNE* model which only utilizes the global adjacency proximity between nodes. Similarly, node2vec is also unified into *RWNE* framework by using a second-order random walk to adjust the prestige of different order proximities, which is highly complex and memory-consuming. However, DeepWalk and node2vec are not direct network embedding methods, as they actually transform the network embedding problem into word embedding problem by preliminarily generating offline "corpus" via random walks, which will consume plenty of space, and they do not have explicit objective functions. The proposed *RWNE* framework do not need to generate a vast "corpus", which jointly online walks and trains in each step. Moreover, *RWNE* framework provides a substantial theoretical foundation and explicit objective for DeepWalk and node2vec.

LINE (Tang et al., 2015). LINE could be regarded as the first-order case of DeepWalk but directly uses online edge-sampling in each training step instead of generating offline "corpus". Therefore, LINE is also a special case of *RWNE* model which only characterizes the first-order proximity.

5. Experiments

In this section, we report the experimental results on six real-word datasets to demonstrate the effectiveness and efficiency of our proposed model.

5.1 Datasets

Here we use the following six publicly available networks with different scales, the statistics of which are shown in Table 1.

• **Cora** (McCallum et al., 2000) is a widely used scientific publication citation network with 2,708 papers, 5,429 citation links, and 7 categories. The labels represent seven different fields in machine learning research, which are case-based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning, and theory.

• **PubMed** (Kipf & Welling, 2017) is also a citation network with 19,717 nodes, 44,338 links, and 3 categories.

• **DBLP** (Perozzi et al., 2017) is a well-known academic collaboration network with 27,199 nodes, 133,664 links, and 4 categories.

• **Blogcatalog** (Tang & Liu, 2009a) is a social network of bloggers with 10,312 nodes, 333,983 links, and 39 categories. The labels represent the topic categories provided by the authors.

Dataset	Cora	PubMed	DBLP	Blogcat.	Flickr	Youtube
# Nodes	2,780	19,717	27,199	10,312	80,513	1,138,499
# Edges	5,429	44,338	133,664	333,983	5,899,882	2,990,443
# Classes	7	3	4	39	195	47

Table 1: Statistics of Datasets.

• **Flickr** (Tang & Liu, 2009a) is a social network of the contacts between users on the Flickr website, with 80,513 nodes, 5,899,882 links, and 195 categories. The labels represent the interest categories of the users.

• **Youtube** (Tang & Liu, 2009b) is a very large interest network between users on the YouTube website, with 1,138,499 nodes, 2,990,443 links, and 47 labels. The labels represent the interest groups subscribed by users of viewers that enjoy common video genres.

5.2 Baseline Methods and Experimental Settings

We compare our *RWNE* model with the following random-walk-based (DeepWalk, LINE, and node2vec), matrix-factorization-based (GraRep and HOPE), and deep-learning-based (SDNE, GCN, and GAT) network embedding methods, with the hyper-parameters setting listed in Table 2.

• **DeepWalk** (Perozzi et al., 2014) adopts uniform random walk to capture the contextual information and Skip-gram model to learn node embeddings.

• **LINE** (Tang et al., 2015) defines loss functions to preserve 1st- and 2nd- order proximity separately. After optimizing the loss functions, it concatenates these embeddings. We use the suggested version to learn two $d/2$-dimensional vectors (one for each-order) and then concatenate them.

• **node2vec** (Grover & Leskovec, 2016) is generalized from DeepWalk by introducing a biased random walk.

• **GraRep** (Cao et al., 2015) factorizes the higher-order proximity matrix via SVD decomposition to get low-dimensional node representations.

• **HOPE** (Ou et al., 2016) also preserves higher-order proximity based on generalized SVD decomposition.

• **SDNE** (Wang et al., 2016) uses deep auto-encoders to jointly preserve 1st- and 2nd-order proximity.

• **GCN** (Kipf & Welling, 2017) is a semi-supervised feature learning model which defines a convolution operator to directly operate graph-structured data.

• **GAT** (Velickovic et al., 2018) is a novel neural network architecture that operates graph-structured data by leveraging masked self-attentional layers.

Experimental Setting. We evaluate the quality of the embedding vectors learned by different methods on three classical network mining tasks: multi-label node classification, node clustering, and link reconstruction. To ensure the significance of the results, we repeat each experiment 10 times and report their mean value and standard deviation value.

Model	Notation	Meaning	Value
Common	lr	learning rate	0.025
	d	embedding dimension	128
	m	negative samples	5
	T	iteration epochs	20,000
DeepWalk node2vec	wt	walk times	10
	wl	walk length	80
	w	window size	10
	p, q	bias parameters of node2vec	0.25
SDNE	α	loss weight	1e3
	β	non-zero elements weight	10
	ν	regularizer term weight	1e-4
RWNE	γ	loss weight	10
	λ	loss weight	1
	α	personalized teleport probability	0.3
	k	walk steps	10

Table 2: Hyperparameters setting.

As shown in Table 2, for the common hyper-parameters, we set learning-rate lr=0.025, embedding-dimension d=128, negative-samples m=5, and iteration-epochs T=20000 for all methods. Specially, for DeepWalk and node2vec, we set walk-times wt=10, walk-length wl=80, window-size k=10, and set the bias parameters of node2vec as p=q=0.25, as recommended in their papers. For SDNE, we tune its parameters of α, β, ν by using a grid-search strategy, and get α=10^3, β=10, ν=10^{-4}. For GCN and GAT, we consistently use the structural features (i.e. adjacent matrix) as the input features. For other parameters and other baselines, we use the default settings as shown in their original papers. For our *RWNE* model, we set the loss-weight γ=10 and λ=1, the personalized-teleport-probability α=0.3, the walk-steps (walk-length) k=10. Note that the walk-steps k in our model actually delimits an upper bound of the order to be preserved which shares a similar meaning with the window-size in DeepWalk and node2vec.

5.3 Multi-Label Node Classification

For the node classification task, we first learn the node embedding vectors from the full nodes on each dataset, and then use the embedding vectors as input features for a one-vs-rest logistic regression classifier, and use both Macro-F1 score and Micro-F1 score as the metrics for evaluation. We repeat each classification experiment 10 times and randomly split 50% of the nodes for training and the other 50% for testing. Due to the lack of space, we report the mean Macro/Micro-F1 scores in Table 3 and Table 4, and report the standard deviations in Appendix A. Note that we exclude the results of some models on the Youtube dataset because they either fail to terminate in one week (SDNE) or run out of memory (GraRep, HOPE, GCN, GAT).

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
DeepWalk	0.7970	0.7764	0.5916	0.2649	0.2659	0.3741
LINE	0.7493	0.7454	0.5702	0.2596	0.2582	0.3463
node2vec	0.7970	0.7817	0.5920	0.2677	0.2637	0.3766
GraRep	0.7756	0.7679	0.5683	0.2430	0.2590	–
HOPE	0.6156	0.6357	0.4555	0.2008	0.1904	–
SDNE	0.6879	0.6700	0.5227	0.2130	0.2332	–
GCN	0.7390	0.7113	0.5330	0.1429	0.1568	–
GAT	0.7541	0.7203	0.5513	0.1659	0.1812	–
RWNE	**0.8318**	**0.8067**	**0.6149**	**0.2992**	**0.2910**	**0.3993**

Table 3: The Macro-F1 scores for multi-label node classification.

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
DeepWalk	0.8085	0.8063	0.6454	0.3922	0.3981	0.4436
LINE	0.7614	0.7551	0.6157	0.3827	0.3739	0.4272
node2vec	0.8074	0.8060	0.6520	0.3965	0.3962	0.4487
GraRep	0.7873	0.7808	0.6284	0.3852	0.3801	–
HOPE	0.6736	0.6478	0.5683	0.3259	0.2935	–
SDNE	0.7311	0.7284	0.6013	0.3520	0.3611	–
GCN	0.7672	0.7591	0.6262	0.2378	0.2493	–
GAT	0.7834	0.7595	0.6387	0.2721	0.2694	–
RWNE	**0.8430**	**0.8289**	**0.6718**	**0.4247**	**0.4157**	**0.4622**

Table 4: The Micro-F1 scores for multi-label node classification.

We can observe that our proposed *RWNE* model consistently and significantly outperforms all the baselines in both metrics on all datasets. For example, on the Cora dataset, *RWNE* outperforms them by 0.03–0.22 (relatively 4%–35%) in terms of Macro-F1 score and by 0.03–0.17 (relatively 4%–25%) in terms of Micro-F1 score; on the BlogCatalog dataset with the larger size, *RWNE* also achieves the gains of 0.03–0.16 (relatively 12%–109%) in terms of Macro-F1 score and 0.03–0.19 (relatively 7%–79%) in terms of Micro-F1 score.

Specially, by only taking random-walk-based methods (DeepWalk, LINE, and node2vec) into account, we can find that *RWNE* also consistently improves the classification performance by around 0.02–0.08 in both metric scores on all datasets, which clearly demonstrates that the effectiveness of our proposed novel random-walk-based framework.

More generally, we can find that, by deploying sufficient long-distance random walks, random-walk-based algorithms (DeepWalk, node2vec, and our *RWNE*) can achieve considerable improvements than matrix-factorization-based and deep-learning-based algorithms, especially on large-scale datasets. Besides, it is worth mentioning that by using the adjacent matrix as the input features, the semi-supervised feature learning models (GCN and GAT) show worse performance than expected, which implies that these models may not be suitable to exploit a graph with poor attribute features.

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
DeepWalk	0.4440	0.2852	0.1811	0.1734	0.3328	0.3093
LINE	0.3516	0.2302	0.1558	0.1672	0.3197	0.2877
node2vec	0.4419	0.2871	0.1850	0.1786	0.3374	0.3088
GraRep	0.4166	0.2587	0.1626	0.1687	0.3202	–
HOPE	0.2986	0.2295	0.1228	0.1387	0.2410	–
SDNE	0.3215	0.1789	0.1385	0.1439	0.2801	–
GCN	0.3607	0.1956	0.1316	0.1134	0.1422	–
GAT	0.3995	0.2078	0.1451	0.1261	0.1895	–
RWNE	**0.4683**	**0.3049**	**0.1986**	**0.1953**	**0.3543**	**0.3209**

Table 5: The NMI scores for node clustering.

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
DeepWalk	0.7849	0.6033	0.7541	0.2415	0.2737	0.2739
LINE	0.7599	0.5746	0.7183	0.2275	0.2570	0.2458
node2vec	0.8192	0.6177	0.7873	0.2616	0.2883	0.2994
GraRep	0.6944	0.5853	0.7127	0.2298	0.2522	–
HOPE	0.6126	0.4594	0.5909	0.1748	0.1633	–
SDNE	0.7387	0.5511	0.6400	0.1565	0.1817	–
GCN	0.6509	0.4851	0.5581	0.1173	0.1125	–
GAT	0.6991	0.4891	0.5975	0.1243	0.1332	–
RWNE	**0.8596**	**0.6435**	**0.8209**	**0.2815**	**0.3129**	**0.3167**

Table 6: The MAP scores for link reconstruction.

5.4 Node Clustering

For the node clustering task, we use the embedding vectors as the input to a k-means cluster and evaluate the performance in terms of NMI (Normalized Mutual Information) score. And also, all the experiments are conducted 10 times, and the mean NMI scores are shown in Table 5, and the standard deviations are shown in Appendix A.

Overall, the results of node clustering are consistent with the results of node classification, and we can reach a similar conclusion as analyzed in Section 5.3. We can see that the proposed *RWNE* model consistently and clearly outperforms all the comparative baselines on all datasets. For example, on the Cora dataset, *RWNE* improves the NMI score by 0.02–0.12 (relatively 5%–33%) over random-walk-based models, and by 0.05–0.17 (relatively 12%–57%) over other baseline models; on the BlogCatalog dataset, *RWNE* improves the NMI score by around 0.02–0.03 (relatively 9%–17%) over random-walk-based models, and by around 0.03–0.08 (relatively 15%–72%) over other baseline models.

5.5 Link Reconstruction

For the link reconstruction task (Wang et al., 2016), we rank pairs of nodes according to their similarities, i.e. the inner product of two embedding vectors, and then reconstruct/predict the links for the highest-ranking pairs of nodes. We use the MAP (Mean Average Precision) metric (Goyal & Ferrara, 2018) to estimate the reconstruction precision. Also, all the experiments are conducted 10 times, and the mean MAP scores are shown in Table 6, and the standard deviations are shown in Appendix A.

We can observe that the results of link reconstruction are also consistent with the results of node classification and node clustering, and we can draw similar conclusions. Overall, in terms of the MAP scores on all the six datasets, the proposed *RWNE* consistently and substantially achieves around 0.02–0.07 improvements over DeepWalk and node2vec, and around 0.05–0.27 gains over other baselines.

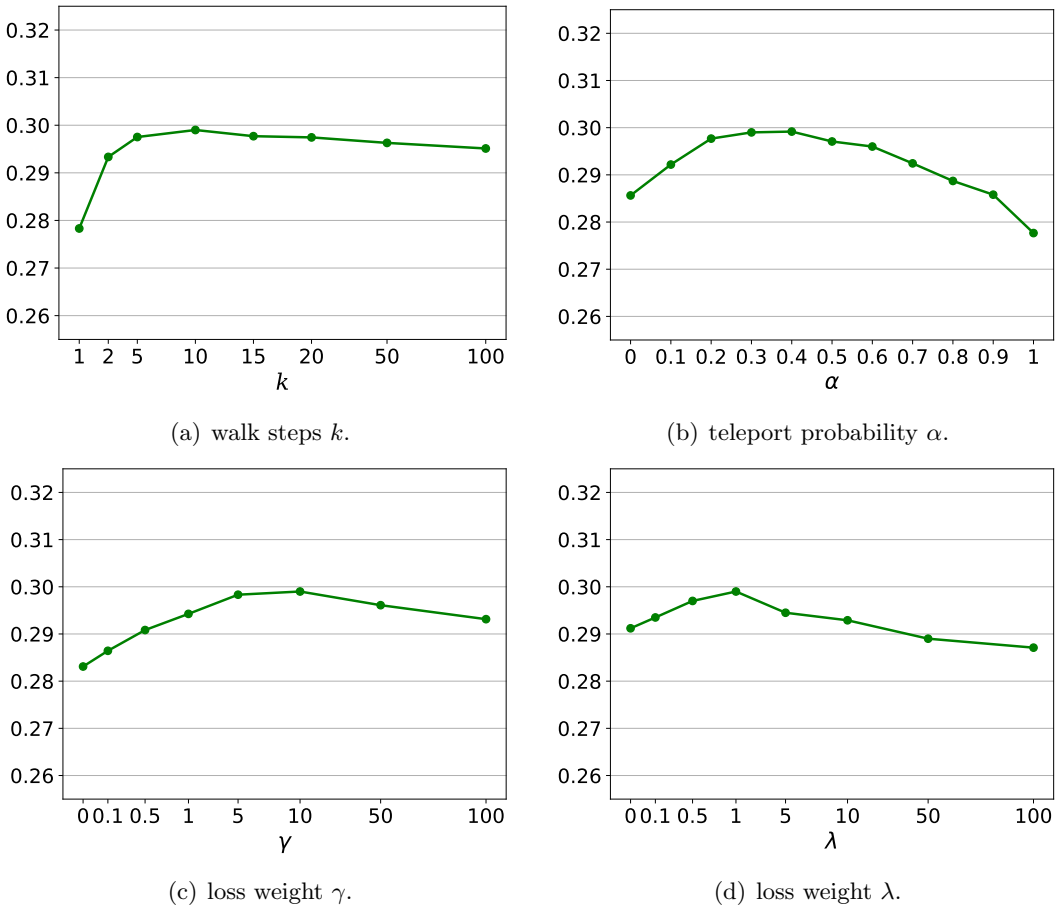
5.6 Parameter Sensitivity

In our model, there exist four important parameters: walk-steps k, personalized teleport probability α, and loss weight γ and λ. In this section, we illustrate these parameters sensitivity by the Macro-F1 scores of node classification experiments on the Blogcatalog dataset. For each experiment, we vary one parameter and fix the others as the default values (as shown in Section 5.2). The results are shown in Figure 4.

Walk-Steps k. We first examine the effects of increasing the upper bound k of the order we combine (see the solid line in Figure 4(a)). We can observe that the model will converge when we consider a large upper bound. Specially, when $k<10$, the performance increases as k increases. When $k>10$, the performance is steady even if we expand k by several times. Therefore, we can fix k as a reasonably large value in practice, and then we can easily learn a well-performed model by only tuning the restart probability. In contrast, the amount of similar hyper-parameters are three in DeepWalk and five in node2vec. The results prove that our model is very practical to achieve better performance with fewer hyper-parameters, i.e., the effectiveness and practicality of the proposed model.

Personalized Teleport Probability α. We show the effects of varying the restart probability α of RWR. As shown in Eq. (13), RWR weights different-order proximities with a decreasing function $\beta(t)=\frac{(1+\alpha(k-t))(1-\alpha)^t}{k}$. By varying α, the decreasing rapidity of $\beta(t)$ varies as shown in Figure 3. And correspondingly, the experimental results with different α are shown as the solid line in Figure 4(b). We can observe that the model achieves the optimal performance when $\alpha=0.3$. If we vary to a smaller α, the performance also falls down. Particularly, when $\alpha=0$, the model degrades to equivalently treat different-order proximities similar to DeepWalk. If we choose a bigger α, the prestige of higher-order proximities will decline.

Loss Weight γ **and** λ. In addition, Figure 4(c) and Figure 4(d) show the effects of varying the γ and λ , which are used to weight the losses designed for the pairwise similarity and the listwise equivalence information provided by higher-order proximity (as introduced in Section 4.2). We can observe that it is worthwhile to jointly preserve the local similarity and global equivalence with suitable weights (e.g., $\gamma=10$ and $\lambda=1$).



(a) walk steps k.

(b) teleport probability α.

(c) loss weight γ.

(d) loss weight λ.

Figure 4: Parameters sensitivity by the Macro-F1 scores of node classification experiments on the Blogcatalog dataset.

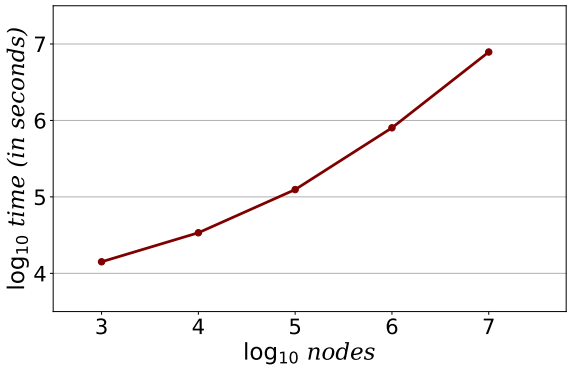


Figure 5: The scalability of *RWNE* over different network sizes.

5.7 Case Study: Scalability

As analyzed in Section 4.4, the proposed *RWNE* model has superior scalability with linear complexity. In this section, we further experimentally verify the scalability of *RWNE*. We first generate a series of random graphs with different sizes of [1k; 10k; 100k; 1000k; 10000k] from the Youtube dataset by randomly choosing several nodes and expanding from them to the fixed size while avoiding isolated nodes, and then apply our method to these synthetic networks to learn node embeddings. The time consumption is shown in Figure 5. We can see that the running time grows linearly with the number of nodes, comparable to the random-walk-based methods (e.g., DeepWalk and node2vec). In the case where the size of the graph is larger, the proposed model has obvious advantages over the methods with higher computational complexity (e.g., GCN and GAT). Thus, our method is efficient and scalable for large-scale networks.

6. Conclusions

In this paper, we present a general scalable random-walk-based network embedding framework ***RWNE*** to effectively and efficiently preserve higher-order proximity.

Distinguishing from existing random-walk-based methods, we focus more on an explicit framework to directly leverage random walk to preserve higher-order proximity with carefully designed objective instead of the current two-step approach. We first systematically design a joint objective to simultaneously capture both the local pairwise similarity and the global listwise equivalence provided by arbitrary higher-order proximity. Then we leverage a node-sampling optimization to equivalently eliminate the computation of higher-order proximity by random walk simulation.

As a result, the random walk is theoretically incorporated into the objective function, which explicitly clarifies the essential role of random walk playing in higher-order proximity preserved network embedding.

Further, we also introduce the random walk with restart process to naturally and effectively weigh the proximities of different orders by a personalized teleport probability. We conduct extensive experiments on several datasets and the results demonstrate the superiority of our model.

Acknowledgments

The authors would like to thank the anonymous reviewers of the Journal of Artificial Intelligence Research (JAIR) for their valuable advice. This work was supported by the NSFC through grants (No.U20B2053, 61872022, and 62002007), and the State Key Laboratory of Software Development Environment (SKLSDE-2020ZX-12). This work was also sponsored by CAAI-Huawei MindSpore Open Fund. Thanks for computing infrastructure provided by Huawei MindSpore platform. For any correspondence, please refer to Jianxin Li and Hao Peng.

Appendix A. More Results

As mentioned in Section 5, we evaluate the quality of the embedding vectors learned by different methods on three classical network mining tasks: multi-label node classification, node clustering, and link reconstruction. Here we report their standard deviation values which are absent due to the limitation of the page's width.

In detail, we report the standard deviations of the Micro-F1 scores and the Macro-F1 scores for multi-label node classification in Table 7 and Table 8, the NMI (Normalized Mutual Information) scores for node clustering in Table 9, and the MAP (Mean Average Precision) scores for link reconstruction in Table 10.

Dataset	**Cora**	**Pubmed**	**DBLP**	**Blogcat.**	**Flickr**	**Youtube**
DeepWalk	0.0034	0.0017	0.0029	0.0030	0.0009	0.0013
LINE	0.0098	0.0030	0.0052	0.0012	0.0078	0.0058
node2vec	0.0041	0.0016	0.0039	0.0023	0.0011	0.0018
GraRep	0.0028	0.0012	0.0018	0.0013	0.0008	–
HOPE	0.0041	0.0022	0.0054	0.0014	0.0009	–
SDNE	0.0203	0.0166	0.0164	0.0023	0.0017	–
GCN	0.0125	0.0073	0.0057	0.0028	0.0026	–
GAT	0.0183	0.0100	0.0087	0.0031	0.0044	–
RWNE	0.0038	0.0023	0.0031	0.0025	0.0012	0.0014

Table 7: The standard deviation of Macro-F1 scores for multi-label node classification.

Dataset	**Cora**	**Pubmed**	**DBLP**	**Blogcat.**	**Flickr**	**Youtube**
Deepwalk	0.0027	0.0017	0.0023	0.0019	0.0005	0.0009
LINE	0.0065	0.0036	0.0066	0.0014	0.0018	0.0036
node2vec	0.0035	0.0019	0.0034	0.0026	0.0015	0.0011
GraRep	0.0028	0.0011	0.0019	0.0014	0.0003	–
HOPE	0.0034	0.0025	0.0026	0.0010	0.0006	–
SDNE	0.0169	0.0148	0.0050	0.0048	0.0016	–
GCN	0.0120	0.0036	0.0082	0.0058	0.0021	–
GAT	0.0162	0.0082	0.0095	0.0061	0.0028	–
RWNE	0.0032	0.0017	0.0027	0.0016	0.0009	0.0008

Table 8: The standard deviation of Micro-F1 scores for multi-label node classification.

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
Deepwalk	0.0041	0.0053	0.0031	0.0014	0.0013	0.0006
LINE	0.0089	0.0127	0.0118	0.0014	0.0026	0.0017
node2vec	0.0049	0.0064	0.0052	0.0015	0.0014	0.0007
GraRep	0.0090	0.0002	0.0007	0.0014	0.0005	–
HOPE	0.0136	0.0005	0.0024	0.0024	0.0006	–
SDNE	0.0184	0.0143	0.0103	0.0047	0.0024	–
GCN	0.0103	0.0054	0.0041	0.0102	0.0067	–
GAT	0.0180	0.0088	0.0083	0.0133	0.0112	–
RWNE	0.0044	0.0062	0.0038	0.0015	0.0015	0.0008

Table 9: The standard deviation of NMI scores for node clustering.

Dataset	Cora	Pubmed	DBLP	Blogcat.	Flickr	Youtube
Deepwalk	0.0021	0.0015	0.0009	0.0008	0.0064	0.0080
LINE	0.0105	0.0046	0.0035	0.0028	0.0079	0.0085
node2vec	0.0030	0.0021	0.0011	0.0009	0.0078	0.0093
GraRep	0.0006	0.0009	0.0012	0.0009	0.0028	–
HOPE	0.0018	0.0015	0.0015	0.0015	0.0031	–
SDNE	0.0117	0.0114	0.0157	0.0036	0.0088	–
GCN	0.0135	0.0120	0.0159	0.0091	0.0056	–
GAT	0.0130	0.0152	0.0209	0.0124	0.0092	–
RWNE	0.0023	0.0016	0.0021	0.0018	0.0072	0.0078

Table 10: The standard deviation of MAP scores for link reconstruction.

References

Ahmed, A., Shervashidze, N., Narayanamurthy, S. M., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *22nd International World Wide Web Conference, WWW 2013*, pp. 37–48.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14, NIPS 2001*, pp. 585–591.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, *15*(6), 1373–1396.

Cai, H., Zheng, V. W., & Chang, K. C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, *30*(9), 1616–1637.

Cao, S., Lu, W., & Xu, Q. (2015). Grarep: Learning graph representations with global

structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015*, pp. 891–900.

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 1145–1152.

Cui, P., Wang, X., Pei, J., & Zhu, W. (2019). A survey on network embedding. *IEEE Trans. Knowl. Data Eng.*, *31*(5), 833–852.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.*, *151*, 78–94.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016*, pp. 855–864.

He, Y., Song, Y., Li, J., Ji, C., Peng, J., & Peng, H. (2019). Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*, pp. 639–648.

Kang, Z., Lu, X., Liang, J., Bai, K., & Xu, Z. (2020a). Relation-guided representation learning. *Neural Networks*, *131*, 93–102.

Kang, Z., Pan, H., Hoi, S. C. H., & Xu, Z. (2020b). Robust graph learning from noisy data. *IEEE Trans. Cybern.*, *50*(5), 1833–1843.

Kang, Z., Peng, C., Cheng, Q., Liu, X., Peng, X., Xu, Z., & Tian, L. (2021). Structured graph learning for clustering and semi-supervised classification. *Pattern Recognit.*, *110*, 107627.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*.

Liu, X., Wang, L., Zhang, J., Yin, J., & Liu, H. (2014). Global and local structure preservation for feature selection. *IEEE Trans. Neural Networks Learn. Syst.*, *25*(6), 1083–1095.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Inf. Retr.*, *3*(2), 127–163.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26, NIPS 2013*, pp. 3111–3119.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016*, pp. 1105–1114.

Peng, H., Li, J., Wang, S., Wang, L., Gong, Q., Yang, R., Li, B., Yu, P. S., & He, L. (2019). Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. *IEEE Trans. Knowl. Data Eng.*

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*, pp. 701–710.

Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017). Don't walk, skip!: Online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 258–265.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018*, pp. 459–467.

Ren, Z., & Sun, Q. (2020). Simultaneous global and local graph structure preserving for multiple kernel clustering. *IEEE Trans. Neural Networks Learn. Syst.*

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, *290*(5500), 2323–2326.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Mag.*, *29*(3), 93–106.

Shi, C., Zhang, Z., Luo, P., Yu, P. S., Yue, Y., & Wu, B. (2015). Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015*, pp. 453–462.

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. VLDB Endow.*, *4*(11), 992–1003.

Sun, Y., Norick, B., Han, J., Yan, X., Yu, P. S., & Yu, X. (2012). Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012*, pp. 1348–1356.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). LINE: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015*, pp. 1067–1077.

Tang, L., & Liu, H. (2009a). Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009*, pp. 817–826.

Tang, L., & Liu, H. (2009b). Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009*, pp. 1107–1116.

Tang, L., & Liu, H. (2011). Leveraging social media networks for classification. *Data Min. Knowl. Discov.*, *23*(3), 447–478.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018*.

Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, *3*(3), 253–256.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016*, pp. 1225–1234.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., & Yang, S. (2017). Community preserving network embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 203–209.

Wang, Y., Yao, Y., Tong, H., Xu, F., & Lu, J. (2019). A brief review of network embedding. *Big Data Min. Anal.*, *2*(1), 35–47.

Wei, X., Xu, L., Cao, B., & Yu, P. S. (2017). Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, pp. 1611–1619.

Yang, C., Sun, M., Liu, Z., & Tu, C. (2017). Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pp. 3894–3900.

Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., & Zhu, W. (2018). Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pp. 2778–2786.

Zhou, C., Liu, Y., Liu, X., Liu, Z., & Gao, J. (2017). Scalable graph embedding for asymmetric proximity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 2942–2948.