Billion-scale Network Embedding with Iterative Random Projection

Ziwei Zhang[1], **Peng Cui**[1], **Haoyang Li**[1], **Xiao Wang**[1], **Wenwu Zhu**[1]
[1] Department of Computer Science and Technology, Tsinghua University, China
zw-zhang16@mails.tsinghua.edu.cn, cuip@tsinghua.edu.cn, lihaoyang96@gmail.com
wangxiao007@mail.tsinghua.edu.cn, wwzhu@tsinghua.edu.cn

Abstract

Network embedding has attracted considerable research attention recently. However, the existing methods are incapable of handling billion-scale networks, because they are computationally expensive and, at the same time, difficult to be accelerated by distributed computing schemes. To address these problems, we propose RandNE, a novel and simple billion-scale network embedding method. Specifically, we propose a Gaussian random projection approach to map the network into a low-dimensional embedding space while preserving the high-order proximities between nodes. To reduce the time complexity, we design an iterative projection procedure to avoid the explicit calculation of the high-order proximities. Theoretical analysis shows that our method is extremely efficient, and friendly to distributed computing schemes without any communication cost in the calculation. We demonstrate the efficacy of RandNE over state-of-the-art methods in network reconstruction and link prediction tasks on multiple datasets with different scales, ranging from thousands to billions of nodes and edges.

1 Introduction

Network embedding is an emerging research topic in recent years, aiming to represent nodes by low-dimensional vectors while maintaining the structures and properties of the network [Cui *et al.*, 2017]. Many methods have been proposed for network embedding, such as using random walks [Perozzi *et al.*, 2014], matrix factorization [Cao *et al.*, 2015] and deep learning [Wang *et al.*, 2016]. With these methods, many network analysis tasks can be fulfilled in vector spaces and benefit from off-the-shelf machine learning models.

Despite such progress, the targeted networks of the existing methods are often in thousand or million scale. However, many real networks have billions of nodes and edges, such as social networks, e-commerce networks and the Internet. The existing methods cannot deal with billion-scale networks, because they are all learning-based methods and thus involve computationally expensive optimization procedures. For example, Stochastic Gradient Descend (SGD) is a commonly used optimization method in network embedding [Tang *et al.*, 2015], but it requires a great number of iterations to converge, which is not feasible for billion-scale networks. One way to accelerate is to resort to distributed computing solutions, but the optimization methods, like SGD, often require global embedding information for searching gradients, leading to intense communication cost. As a result, how to design an efficient and effective billion-scale network embedding method that is friendly to distributed computing is still an open problem.

Different from learning-based methods, random projection is a simple and powerful technique to form low-dimensional embedding spaces while preserving the structures of the original space. It is also friendly to distributed computing, and thus widely exploited in large-scale data scenarios [Vempala, 2005]. But the extremely sparse structures of real networks pose great challenges to applying random projection to network embedding. The existing work [Cui *et al.*, 2017] has demonstrated that high-order proximities between nodes are essential to be preserved in network embedding and can effectively address the sparsity issue. Hence, how to design a high-order proximity preserved random projection method is the key problem of billion-scale network embedding.

In this paper, we propose RandNE (Iterative Random Projection Network Embedding), a novel and simple billion-scale network embedding method based on high-order proximity preserved random projection. In order to avoid the explicit calculation of high-order proximities which induces high computational complexities, we design an iterative projection procedure, enabling arbitrary high-order proximity preserved random projection with a linear time complexity. Theoretical analysis is provided to guarantee that i) RandNE can well support distributed computing without any communication cost between different servers in the calculation, and ii) it can efficiently incorporate the dynamic changes of the networks without error aggregation. These two merits make RandNE a promising solution for billion-scale network embedding, even in dynamic environments.

Extensive experiments are conducted in network reconstruction, link prediction and classification tasks on multiple datasets with different scales, ranging from thousands to billions of nodes and edges. The results show that RandNE can boost the efficiency of network embedding by about 2 orders

over state-of-the-art methods[1] while achieving a superior or comparable accuracy. For the WeChat[2] network with 250 millions nodes and 4.8 billion edges, RandNE can produce the embeddings within 7 hours with 16 distributed servers.

The contributions of our paper are summarized as follows:

- We propose RandNE, a novel and simple random projection based network embedding method that enables billion-scale network embedding.
- We design an iterative projection procedure to realize high-order proximities preserved random projection efficiently without explicitly calculating the high-order proximities.
- We theoretically and empirically prove that RandNE can well support distributed computing without communication cost and can efficiently deal with dynamic networks without error aggregation.

The rest of this paper is organized as follows. In Section 2, we briefly review related works. We give our problem formulation in Section 3 and introduce our proposed method in Section 4. Experimental results are reported in Section 5. Finally, we summarize in Section 6.

2 Related Work

Network embedding has attracted considerable research attention in the past few years. Here, we briefly review some representative network embedding methods, and readers are referred to [Cui *et al.*, 2017] for a comprehensive survey.

The flourish of network embedding research begins when DeepWalk [Perozzi *et al.*, 2014] first proposes using truncated random walks to explore the network structure and utilizes the skip-gram model [Mikolov *et al.*, 2013] to derive the embedding vectors. LINE [Tang *et al.*, 2015] takes a similar idea with an explicit objective function by limiting the walk length as one. Node2vec [Grover and Leskovec, 2016] generalizes these two methods by taking biased random walks for more flexibility. These random walks based methods are proven equivalent to factorizing a high-order proximity matrix [Chen *et al.*, 2017; Yang *et al.*, 2017]. On the other hand, explicit matrix factorization methods for network embedding are also studied. GraRep [Cao *et al.*, 2015] directly applies SVD to preserve high-order proximity matrices. HOPE [Ou *et al.*, 2016] proposes using generalized SVD to preserve the asymmetric transitivity in directed networks. Community structure is preserved by non-negative matrix factorization in [Wang *et al.*, 2017]. Deep learning method is also studied. SDNE [Wang *et al.*, 2016] first considers the high non-linearity in network embedding and proposes a deep auto-encoder to preserve the first and the second order proximities.

Despite their remarkable performances, the targeted networks of these methods are often in thousand or million scale. In [Zhou *et al.*, 2017], a modification of DeepWalk is applied to a billion-scale network aliItemGraph. However, their method has the same time complexity as DeepWalk, which is much more computationally expensive than our method by two orders (see Figure 1). Besides, it does not address the distributed computing problem.

Another closely related topic is random projection [Vempala, 2005; Arriaga and Vempala, 2006; Shi *et al.*, 2012; Choromanski *et al.*, 2017], which is widely adopted in dimension reduction. But existing random projection methods do not consider the sparsity problem in network embedding, and thus cannot be directly applied.

3 Notation and Problem Formulation

3.1 Notations

First, we summarize the notations used in this paper. For a network G with N nodes and M edges, we use $\mathbf{A}$ to denote the adjacency matrix. In this paper, we mainly consider undirected networks, so $\mathbf{A}$ is symmetric. $\mathbf{A}(i,:)$ and $\mathbf{A}(:,j)$ denote its i^{th} row and j^{th} column respectively. $\mathbf{A}(i,j)$ is the element in the i^{th} row and j^{th} column. Throughout the paper, we use bold uppercase characters to denote matrices and bold lowercase characters to denote vectors, e.g. $\mathbf{X}$ and $\mathbf{x}$ respectively. We use dot to denote the matrix product of two matrices, e.g. $\mathbf{B} \cdot \mathbf{C}$.

3.2 Problem Formulation

To represent nodes in a network by low-dimensional vectors, one commonly adopted objective function in network embedding is matrix factorization, which decomposes a targeted similarity function of the adjacency matrix $\mathcal{F}(\mathbf{A}) \in \mathbb{R}^{N \times N}$ into the product of two low-dimensional matrices $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{N \times d}$ with the following objective function:

$$\min_{\mathbf{U},\mathbf{V}} \left\| \mathcal{F}(\mathbf{A}) - \mathbf{U} \cdot \mathbf{V}^T \right\|_p, \tag{1}$$

where p is the norm. In this paper, we only consider undirected networks and symmetric similarities, so $\mathbf{U} = \mathbf{V}$. We also focus on the spectral norm, i.e. $p = 2$, which is widely adopted [Liberty, 2013].

The previous works have shown that high-order proximities are essential to be preserved in network embedding, which can be formulated as a polynomial function of the adjacency matrix [Yang *et al.*, 2017; Chen *et al.*, 2017]. In this paper, we assume that $\mathcal{F}(\mathbf{A})$ is a positive semi-definite function, so it can be formulated as $\mathcal{F}(\mathbf{A}) = \mathbf{S} \cdot \mathbf{S}^T$. Then, we can rewrite Eq. (1) as:

$$\begin{aligned} &\min_{\mathbf{U}} \left\| \mathbf{S} \cdot \mathbf{S}^T - \mathbf{U} \cdot \mathbf{U}^T \right\|_2 \\ &\mathbf{S} = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + ... + \alpha_q \mathbf{A}^q, \end{aligned} \tag{2}$$

where $\mathbf{S}$ is the high-order proximity matrix, $\alpha_0, \alpha_1, ..., \alpha_q$ are pre-defined weights and q is the order.

From Eckart–Young theorem [Eckart and Young, 1936], Singular Value Decomposition (SVD) can lead to the optimal solution of Eq. (2). However, SVD is computationally expensive and thus not suitable for large-scale networks.

4 RandNE: the Proposed Method

4.1 Gaussian Random Projection Embedding

To minimize the objective function in Eq. (2), an extremely simple yet effective method is random projection, and Gaussian random projection is widely used [Vempala, 2005]. Formally, let $\mathbf{R} \in \mathbb{R}^{N \times d}$ and each element of $\mathbf{R}$ follows an i.i.d

[1] These algorithms are tested using the source code published by their authors.

[2] One of the largest social network platforms in China.

Gaussian distribution $\mathbf{R}(i,j) \sim \mathcal{N}\left(0, \frac{1}{d}\right)$. Then, the embeddings $\mathbf{U}$ can be obtained by performing a matrix product:

$$\mathbf{U} = \mathbf{S} \cdot \mathbf{R} = \left(\alpha_0 \mathbf{I} + \alpha_1 \mathbf{A} + \alpha_2 \mathbf{A}^2 + ... + \alpha_q \mathbf{A}^q\right) \mathbf{R}, \quad (3)$$

i.e. we randomly project the proximity matrix $\mathbf{S}$ into a low-dimensional subspace. Gaussian random projection has an theoretical guarantee, as we specific in the following theorem.

Theorem 1. *For any similarity matrix $\mathbf{S}$, denote its rank as $r_{\mathbf{S}}$. Then, for any $\epsilon \in \left(0, \frac{1}{2}\right)$, the following equation holds:*

$$P\left[\left\|\mathbf{S} \cdot \mathbf{S}^T - \mathbf{U} \cdot \mathbf{U}^T\right\|_2 > \epsilon \left\|\mathbf{S}^T \cdot \mathbf{S}\right\|_2\right] \leq 2 r_{\mathbf{S}} e^{-\frac{\left(\epsilon^2 - \epsilon^3\right) d}{4}}, \quad (4)$$

where $\mathbf{U} = \mathbf{S} \cdot \mathbf{R}$ and $\mathbf{R}$ is a Gaussian random matrix.

Proof. From the Johnson-Lindenstrauss property [Arriaga and Vempala, 2006], for any $\mathbf{s} \in \mathbb{R}^N$, we have

$$P\left[(1-\epsilon)\|\mathbf{s}\|_2^2 \leq \|\mathbf{s} \cdot \mathbf{R}\|_2^2 \leq (1+\epsilon)\|\mathbf{s}\|_2^2\right] \geq 1 - 2e^{-\frac{\left(\epsilon^2 - \epsilon^3\right) d}{4}}. \quad (5)$$

Then, we can get:

$$\begin{aligned}
&P\left[\left\|\mathbf{S} \cdot \mathbf{S}^T - \mathbf{U} \cdot \mathbf{U}^T\right\|_2 > \epsilon \left\|\mathbf{S}^T \cdot \mathbf{S}\right\|_2\right] \\
&= P\left[\sup_{\|\mathbf{x}\|_2 = 1} \left|\|\mathbf{x} \cdot \mathbf{S}\|_2^2 - \|\mathbf{x} \cdot \mathbf{S} \cdot \mathbf{R}\|_2^2\right| > \epsilon \sup_{\|\mathbf{x}\|_2 = 1} \|\mathbf{x} \cdot \mathbf{S}\|_2^2\right] \\
&\leq P\left[\exists\, \mathbf{s} \in rowspan\left\{\mathbf{S}\right\}, \left|\|\mathbf{s}\|_2^2 - \|\mathbf{s} \cdot \mathbf{R}\|_2^2\right| > \epsilon \|\mathbf{s}\|_2^2\right] \\
&\leq r_{\mathbf{S}} P\left[\mathbf{s}, \left|\|\mathbf{s}\|_2^2 - \|\mathbf{s} \cdot \mathbf{R}\|_2^2\right| > \epsilon \|\mathbf{s}\|_2^2\right] = 2 r_{\mathbf{S}} e^{-\frac{\left(\epsilon^2 - \epsilon^3\right) d}{4}}.
\end{aligned} \quad (6)$$

The last line is resulted from the union bound and Eq. (5). □

The theorem basically shows that performing a Gaussian random projection can effectively solve the objective function in Eq. (2). Actually, Gaussian random projection is also known to have other merits, such as preserving the margin for classification [Shi *et al.*, 2012], which we omit for brevity.

For the projection matrix, it is proven that orthogonal Gaussian random matrix can further improve the accuracy, which can be easily obtained by performing a Gram Schmidt process on the Gaussian random matrix [Choromanski *et al.*, 2017]. In this paper, we use the orthogonal Gaussian random matrix as the projection matrix.

However, since $\mathbf{S}$ may not be a sparse matrix, directly calculating $\mathbf{S}$ and performing the projection is time consuming and not scalable to large-scale networks.

4.2 Iterative Projection

To address the efficiency problem, we design an iterative projection procedure to avoid the explicit calculation of the high-order proximity matrix $\mathbf{S}$. Specifically, from Eq. (3), we can decompose $\mathbf{U}$ into matrices of different orders:

$$\mathbf{U} = \alpha_0 \mathbf{U}_0 + \alpha_1 \mathbf{U}_1 + ... + \alpha_q \mathbf{U}_q, \quad (7)$$

where $\mathbf{U}_i = \mathbf{A}^i \cdot \mathbf{R}, 0 \leq i \leq q$. Then, the decomposed parts, $\mathbf{U}_1 ... \mathbf{U}_q$, can be calculated iteratively:

$$\mathbf{U}_i = \mathbf{A} \cdot \mathbf{U}_{i-1}\ ,\ \forall 1 \leq i \leq q. \quad (8)$$

Note that in Eq. (8), we only need to calculate the matrix product of the adjacency matrix and a low-dimensional matrix. Since the adjacency matrix is sparse, we can use sparse matrix products, which are highly scalable and efficient.

Algorithm 1 RandNE: Iterative Random Projection Network Embedding

Require: Adjacency Matrix $\mathbf{A}$, Dimensionality d, Order q, Weights $\alpha_0, \alpha_1, ..., \alpha_q$
Ensure: Embedding Results $\mathbf{U}$
1: Generate $\mathbf{R} \in \mathbb{R}^{N \times d} \sim \mathcal{N}(0, \frac{1}{d})$
2: Perform a Gram Schmidt process on $\mathbf{R}$ to obtain the orthogonal projection matrix $\mathbf{U}_0$
3: **for** i in 1:q **do**
4: Calculate $\mathbf{U}_i = \mathbf{A} \cdot \mathbf{U}_{i-1}$
5: **end for**
6: Calculate $\mathbf{U} = \alpha_0 \mathbf{U}_0 + \alpha_1 \mathbf{U}_1 + ... + \alpha_q \mathbf{U}_q$

4.3 Discussion

We show our algorithm framework in Algorithm 1. For the time complexity, according to Algorithm 1, the complexity of line 1 is $O(N \cdot d)$, the complexity of line 2 is $O(N \cdot d^2)$, the complexity of each iteration from line 3 to line 5 is $O(M \cdot d)$ and the complexity of line 6 is $O(q \cdot N \cdot d)$, where N and M are the number of nodes and edges in the network respectively, q is the preset order and d is the dimensionality of the embedding space. As a result, we have an overall time complexity $O\left(N \cdot d^2 + M \cdot q \cdot d\right)$, i.e. our method is linear with the network size.

From the above analysis, we can also see that our method is extremely efficient because it only needs to iterate q times, and within each iteration, only a simple matrix product needs to be calculated. In contrast, although some existing network embedding methods are also proven to have linear time complexities, such as the embedding methods based on SGD [Tang *et al.*, 2015] or SVD [Ou *et al.*, 2016], they inevitably need dozens or hundreds of iterations in the optimization. As a result, our method is more efficient than these methods by orders of magnitude, and is thus more suitable for billion-scale network embedding.

In addition, according to the property of matrix products, each dimension (i.e. column) of $\mathbf{U}_i$ can be calculated separately without any information from other dimensions. We formalize this property in the following theorem.

Theorem 2. *For any $j \neq l$, the calculation of $\mathbf{U}_i(:, j)$ and $\mathbf{U}_i(:, l), 1 \leq i \leq q$ from line 3 to line 5 in Algorithm 1 are independent, if $\mathbf{A}$ is known to all the servers.*

Proof. Straightforward from the property of matrix products. □

The theorem shows that our method naturally supports distributed computing by allocating the calculation of different dimensions into distributed servers, and no communication is needed during the calculation process, if $\mathbf{A}$ is known to all the servers. For networks that cannot be stored in the memory or when the number of servers exceeds the dimensionality, we can use more advanced distributed matrix multiplication algorithms, such as [Vastenhouw and Bisseling, 2005; Boman *et al.*, 2013], which we leave as the future work. This is in sharp contrast with all the existing methods which can only be parallelizable within one server but are hard to be distributed because of intensive communication cost. This merit

lays another foundation for applying our method to billion-scale networks.

4.4 Dynamic Updating

As many real networks are dynamic, we next show how to efficiently update RandNE to incorporate the dynamic changes.

First, we focus on the changes of edges. From Algorithm 1, to update the final embedding vectors, we only need to update the decomposed parts $\mathbf{U}_i, 1 \leq i \leq q$. Formally, we denote the changes in the adjacency matrix as $\Delta\mathbf{A}$ and the changes in $\mathbf{U}_i$ as $\Delta\mathbf{U}_i, 1 \leq i \leq q$. From Eq. (8), we have:

$$\begin{aligned} &\mathbf{U}_i + \Delta\mathbf{U}_i = (\mathbf{A} + \Delta\mathbf{A}) \cdot (\mathbf{U}_{i-1} + \Delta\mathbf{U}_{i-1}) \\ \Rightarrow\ &\Delta\mathbf{U}_i = \mathbf{A} \cdot \Delta\mathbf{U}_{i-1} + \Delta\mathbf{A} \cdot \mathbf{U}_{i-1} + \Delta\mathbf{A} \cdot \Delta\mathbf{U}_{i-1}. \end{aligned} \tag{9}$$

Then, we can iteratively calculate $\Delta\mathbf{U}_i$ using Eq. (9).

Besides, nodes in the network may also be added or deleted. For deleted nodes, their embeddings can be directly set as all zeros. For newly added nodes, we first add some empty nodes (i.e. without any edge) to make the dimensionality of the matrices match, and then treat them equivalently as the changes of edges. Specifically, we denote N' as the number of added nodes. For the projection matrix $\mathbf{U}_0$, we can generate an additional orthogonal Gaussian random matrix $\hat{\mathbf{U}}_0 \in \mathbb{R}^{N' \times d}$ and concatenate it with the current matrix to form the new projection matrix. For other $\mathbf{U}_i, 1 \leq i \leq q$, we can easily find that they have all-zero elements for the empty nodes, i.e. we only need to add N' all-zero rows to adjust the dimensionality.

We show the algorithm framework of dynamic updating in Algorithm 2. As the updating only involves local changes, the dynamic updating is computationally efficient, as we specific in the following theorem.

Theorem 3. *The time complexity of dynamic updating is linear with the number of changed nodes and number of changed edges respectively.*

Proof. Omitted for the lack of space. □

Another merit of our updating method is that it has no error aggregation, i.e. the dynamic updating algorithm leads to the identical results as rerunning Algorithm 1. So our method can effectively incorporate the dynamic changes of networks with high computational efficiency.

Algorithm 2 Dynamic Updating of RandNE

Require: Adjacency Matrix $\mathbf{A}$, Dynamic Changes $\Delta\mathbf{A}$, Previous Projection Results $\mathbf{U}_0, \mathbf{U}_1, ..., \mathbf{U}_q$
Ensure: Updated Projection Results $\mathbf{U}'_0, \mathbf{U}'_1, ..., \mathbf{U}'_q$
1: **if** $\Delta\mathbf{A}$ includes N' new nodes **then**
2: Generate an orthogonal projection $\hat{\mathbf{U}}_0 \in \mathbb{R}^{N' \times d}$
3: Concatenate $\hat{\mathbf{U}}_0$ with $\mathbf{U}_0$ to obtain $\mathbf{U}'_0$
4: Add N' all-zero rows in $\mathbf{U}_1 ... \mathbf{U}_q$
5: **end if**
6: Set $\Delta\mathbf{U}_0 = 0$
7: **for** i in 1:q **do**
8: Calculate $\Delta\mathbf{U}_i$ using Eq. (9)
9: Calculate $\mathbf{U}'_i = \mathbf{U}_i + \Delta\mathbf{U}_i$
10: **end for**

5 Experiments

5.1 Experimental Setting

To comprehensively evaluate the efficacy of RandNE, we first conduct experiments on 3 commonly used networks[3]: BlogCatalog, Flickr, Youtube, which contain 10,312 nodes and 667,966 edges, 80,513 nodes and 11,799,764 edges, 1,138,499 nodes and 5,980,886 edges respectively. Furthermore, we evaluate our method on WeChat network with 250 million nodes and 4.8 billion edges.

We compare our method with the following baselines:

- DeepWalk [Perozzi *et al.*, 2014][4] uses random walks and the skip-gram model to learn embeddings. We use two parameter settings: one suggested in the paper and one used in the implementation of the authors, and report the best results.
- LINE [Tang *et al.*, 2015][5] explicitly preserves the first two order proximities, denoted as LINE_{1st} and LINE_{2nd} respectively. We exclude the results of concatenating them because no obvious improvement is observed. We use the default parameter settings except the number of training samples, which we conduct a line search for the optimal value.
- Node2vec [Grover and Leskovec, 2016][6] generalizes DeepWalk and LINE by using biased random walks. We use the default settings for all parameters except the bias parameters p, q, which we conduct a grid search from $\{0.5, 1, 2\}$.
- SDNE [Wang *et al.*, 2016][7] proposes a deep auto-encoder to preserve the first and the second order proximities simultaneously. We use the default parameter settings and the auto-encoder structure in the implementation of the authors.

There are also other methods like GraRep [Cao *et al.*, 2015] and M-NMF [Wang *et al.*, 2017], but we exclude them here for their scalability issues. We also exclude the results of SDNE on Youtube because it fails to terminate in one week. On WeChat network, as all these baselines cannot terminate within acceptable time, we mainly compare our method with other simpler graph-based methods.

For our method RandNE, we set the order $q = 3$ with a grid search for the weights. For all the methods, we uniformly set the dimensionality as $d = 128$ unless stated otherwise. All our experiments are conducted in a single PC with 2 I7 processors and 48GB memory, except for Section 5.3, where we run our method in a distributed cluster.

5.2 Moderate-scale Networks

Running Time Comparison

To compare the efficiency of different methods, we first report the running time of all the methods in Figure 1. The

[3] http://socialcomputing.asu.edu/pages/datasets

[4] https://github.com/phanein/deepwalk

[5] https://github.com/tangjianpku/LINE

[6] https://github.com/snap-stanford/snap

[7] https://github.com/suanrong/SDNE

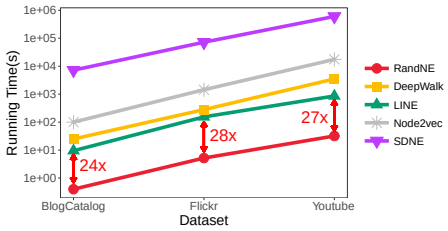

Figure 1: The running time comparison of different methods. Our method RandNE can boost the efficiency by more than 24 times over state-of-the-art methods on all networks.

Table 1: AUC scores of Link Prediction.

Dataset	BlogCatalog	Flickr	Youtube
RandNE	**0.944**	**0.940**	0.887
DeepWalk	0.760	0.938	0.909
LINE$_{1st}$	0.667	0.909	0.847
LINE$_{2nd}$	0.762	0.932	**0.959**
Node2vec	0.650	0.865	0.778
SDNE	0.940	0.926	-

results show that our proposed method RandNE can boost the efficiency by more than 24 times over the baselines on all networks. Note that the baselines are tested using the source code published by their authors. We realize that there might be slight differences in the programming languages and implementation details, but the effect of these factors can be safely ignored considering the improvement of 24 times. So we directly report their results for reproducibility concerns. The extreme efficiency lays the foundation for applying RandNE to billion-scale networks.

Link Prediction

Link prediction, aiming to predict future links using the current network structure, is a typical task of network embedding. In our experiments, we randomly hide 30% of the edges for testing. After training embedding vectors on the rest of the network, we rank pairs of nodes according to their inner product similarities and evaluate the results on the testing network. For the evaluation metrics, we use Area Under Curve (AUC) [Fawcett, 2006] and Precision@K [Wang *et al.*, 2016]. On Youtube, the number of possible pairs of nodes $\frac{N(N-1)}{2}$ is too large to evaluate, so we sample 1% for evaluation, as in [Ou *et al.*, 2016]. The process is repeated 5 times and the average results are reported.

From Table 1 and Figure 2, we can see that our proposed method consistently outperforms the baselines on the metric Precision@K. On AUC, our method achieves the best performance on BlogCatalog and Flickr, and has a comparable performance on Youtube. Considering the significant improvement in efficiency and the simplicity of our model, we regard the performances of RandNE in accuracy aspect to be satisfactory and somewhat beyond expectation.

We also conduct experiments on network reconstruction, i.e. train embedding vectors on the whole network and reconstruct the observed links. The results are consistent with link prediction and we omit the results here due to space limit.

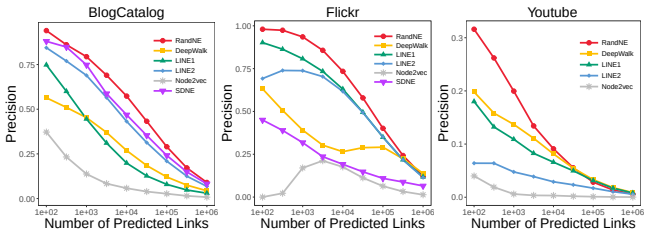

Figure 2: The Precision@K of link prediction. The results show that our proposed method outperforms the baselines in link prediction.

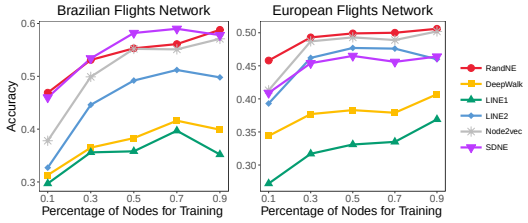

Figure 3: The accuracy of node classification. The results show that RandNE effectively captures the structural role of nodes.

Node Classification

To validate the effectiveness of our method in node classification, we conduct experiments on two air-traffic networks[8] from Brazilian and European as in [Ribeiro *et al.*, 2017], where the networks have 131 nodes and 2006 edges, 399 nodes and 11986 edges respectively. The networks are constructed by assigning airports as nodes and airlines as edges. Each node is assigned a ground-truth label ranging from 1 to 4 to indicate the level of activities of the airports.

We randomly split the nodes into a training set and a testing set. Then, an one-vs-all logistic regression with L2 regularization [Fan *et al.*, 2008] is trained using the embeddings on the training set, and tested on the testing set. We use accuracy, i.e. the percentage of nodes whose labels are correctly predicted, as the measurement. We uniformly set the dimensionality of embedding as 16 since the networks have small sizes. The average results of 20 runs are plotted in Figure 3.

From the figure, we can see that RandNE consistently achieves the best results on European Flights Network. On Brazilian Flights Network, RandNE is only second to SDNE with tiny differences. However, RandNE is much more efficient than SDNE by about 4 orders (see Figure 1).

Parameter and Scalability

In RandNE, we use iterative random projection to preserve high-order proximities. Here we analyze the effect of the proximity order, or equivalently, the number of iterations q. We report the results of varying q from 1 to 3 with the same experimental settings. For the lack of space, we only report AUC scores of link prediction in Figure 4 (A). The results show that iterative random projection ($q > 1$) greatly and consistently outperforms the simple random projection ($q = 1$), demonstrating the importance of preserving high-order proximities in network embedding.

To verify the scalability of RandNE, we conduct experiments on random networks (i.e. the Erdos Renyi model

[8] https://github.com/leoribeiro/struc2vec

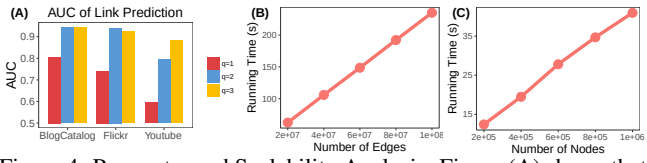

Figure 4: Parameter and Scalability Analysis. Figure (A) shows that the high-order proximity ($q > 1$) greatly outperforms the simple random projection ($q = 1$). Figures (B)(C) demonstrate the linear time complexity of our method.

Table 2: AUC scores of network reconstruction on WeChat network.

Method	AUC
RandNE	**0.989**
Common Neighbors	0.783
Adamic Adar	0.783
Random	0.500

[Erdos and Rényi, 1960]). We record the running time when fixing the number of nodes (as one million) or fixing the number of edges (as ten million) while varying the other. Figure 4 (B)(C) show that the running time grows linearly with the number of nodes and number of edges respectively.

5.3 A Billion-scale Network

WeChat[9] is one of the largest online social networks in China. We use the friendships data provided by WeChat from January 21, 2011 (the launch day of WeChat) to January 20, 2013, which contains 250 million nodes and 4.8 billion edges in total. The data is strictly anonymized for privacy purposes. Since no node label information is available, we mainly conduct experiments on network reconstruction and link prediction. As none of the aforementioned network embedding method can be applied to network of such a scale, we compare our method with two widely used graph-based measures: Common Neighbors and Adamic Adar [Liben-Nowell and Kleinberg, 2007]. The experiments are conducted in a distributed cluster with 16 computing servers, where each server has 2 Xeon E5 CPU and 128GB memory. For our method, we set the dimensionality of the embedding as $d = 512$.

Network Reconstruction

We train embedding vectors on the whole network and rank pairs of nodes according to their inner product similarities. Then, the top ranking pairs are used to reconstruct the network. We report the AUC scores in Table 2.

The results show that our proposed method greatly outperforms Common Neighbor and Adamic Adar. A plausible reason is that our method preserves the high-order proximity information in the embedding vectors by performing the iterative random projection, while the baselines only count local proximities. Adamic Adar, as a frequency-weighted modification of Common Neighbors, has the same accuracy as Common Neighbors because on the billion-scale network, AUC score mainly depends on whether two nodes have neighbors instead of the weights. Here we omit the other metric, Precision@K, because the number of possible node pairs $\frac{N(N-1)}{2} \approx 10^{16}$ is so large that even sampling is infeasible.

[9]http://www.wechat.com/en/

Table 3: AUC scores of dynamic link prediction on WeChat.

Observed Edges	30%	40%	50%	60%	70%
RandNE-D	**0.646**	**0.689**	**0.726**	**0.756**	**0.780**
RandNE-R	**0.646**	**0.689**	**0.726**	**0.756**	**0.780**
Common Neighbors	0.575	0.611	0.647	0.681	0.712
Adamic Adar	0.575	0.611	0.647	0.681	0.712
Random	0.500	0.500	0.500	0.500	0.500

Table 4: The running time of our method via distributed computing.

Number of Sub-clusters	1	2	3	4
Running Time(s)	82157	46029	33965	24757

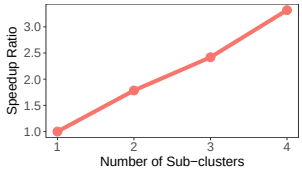

Figure 5: The speedup ratio of RandNE via distributed computing.

Dynamic Link Prediction

To simulate the evolving scenarios of real networks, we first randomly hide 70% of the edges, and gradually add the hidden edges into the network. In the process, we train embedding vectors on all the observed links, and evaluate the link prediction results on the rest of the network. We adopt two versions of our method: one by dynamic updating as the network evolves (RandNE-D), and one by re-running the algorithm at each time (RandNE-R). Table 3 shows that our proposed method again consistently outperforms the baselines on the AUC scores. Besides, RandNE-D shows identical results as RandNE-R, verifying that our dynamic updating has no error aggregation. All methods have larger AUC scores as training data increases because more information is provided.

Speedup via Distributed Computing

We evaluate the performance of our method in distributed computing by reporting the speedup ratio. We divide the 16 servers into 4 sub-clusters, with each sub-cluster containing 4 servers. We vary the number of sub-clusters used for distributed computing and record the running time and speedup ratio. Figure 5 shows that our method has a linear speedup ratio with a slope of approximately 0.8. The slope is slightly less than 1 because of subtle differences in the servers and some extra costs, e.g. reading data. We also report the running time in Table 4. It shows that RandNE can learn all the node embeddings of WeChat within 7 hours with 16 normal servers, which is promising for real network applications.

6 Conclusion

In this paper, we study the problem of embedding billion-scale networks while preserving high-order proximities. We propose RandNE, a novel and simple network embedding method based on random projection and design an iterative projection procedure to efficiently preserve the high-order proximities. Theoretical analysis shows that i) RandNE can well support distributed computing without any communication cost, and ii) it can efficiently incorporate the dynamic changes of the networks without error aggregation. Extensive experimental results on multiple datasets with different scales demonstrate the efficacy of our proposed method.

References

[Arriaga and Vempala, 2006] Rosa I Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.

[Boman *et al.*, 2013] Erik G Boman, Karen D Devine, and Sivasankaran Rajamanickam. Scalable matrix computations on large scale-free graphs using 2d graph partitioning. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 50. ACM, 2013.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[Chen *et al.*, 2017] Siheng Chen, Sufeng Niu, Leman Akoglu, Jelena Kovačević, and Christos Faloutsos. Fast, warped graph embedding: Unifying framework and one-click algorithm. *arXiv preprint arXiv:1702.05764*, 2017.

[Choromanski *et al.*, 2017] Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems*, pages 218–227, 2017.

[Cui *et al.*, 2017] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *arXiv preprint arXiv:1711.08752*, 2017.

[Eckart and Young, 1936] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[Erdos and Rényi, 1960] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(Aug):1871–1874, 2008.

[Fawcett, 2006] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[Liben-Nowell and Kleinberg, 2007] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[Liberty, 2013] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[Ou *et al.*, 2016] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 701–710. ACM, 2014.

[Ribeiro *et al.*, 2017] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[Shi *et al.*, 2012] Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton Hengel. Is margin preserved after random projection? In *Proceedings of the 29th International Conference on Machine Learning*, pages 591–598, 2012.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. WWW, 2015.

[Vastenhouw and Bisseling, 2005] Brendan Vastenhouw and Rob H Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM review*, 47(1):67–95, 2005.

[Vempala, 2005] Santosh S Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234. ACM, 2016.

[Wang *et al.*, 2017] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[Yang *et al.*, 2017] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[Zhou *et al.*, 2017] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2942–2948, 2017.