# Degree-biased random walk for large-scale network embedding

Zhang, Yunyi; Shi, Zhan; Feng, Dan; Zhan, Xiuxiu

**DOI**
10.1016/j.future.2019.05.033

**Publication date**
2019

**Document Version**
Final published version

**Published in**
Future Generation Computer Systems

Future Generation Computer Systems 100 (2019) 198–209

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Degree-biased random walk for large-scale network embedding

Yunyi Zhang [a,b], Zhan Shi [a,b,*], Dan Feng [a], Xiu-Xiu Zhan [c,**]

[a] *Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, Engineering Research Center of data storage systems and Technology, Ministry of Education of China, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China*
[b] *Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen, Gunagdong, 518000, China*
[c] *Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands*

## HIGHLIGHTS

- The current random walk based network embedding methods cannot well adapt to the scale-free feature of real-world networks.
- A degree-biased random walk method is proposed to adapt to the scale-free feature of real-world networks and extract topological information as fully as possible for representation learning.
- The space and computation overhead for network embedding can be drastically reduced by adopting by condensing the fixed-length sequences into variable-length ones based on nodes centrality.

## ARTICLE INFO

*Article history:*
Received 7 January 2019
Received in revised form 4 April 2019
Accepted 11 May 2019
Available online 15 May 2019

*Keywords:*
Network embedding
Scale-free
Random walks

## ABSTRACT

Network embedding aims at learning node representation by preserving the network topology. Previous embedding methods do not scale for large real-world networks which usually contain millions of nodes. They generally adopt a one-size-fits-all strategy to collect information, resulting in a large amount of redundancy. In this paper, we propose DiaRW, a scalable network embedding method based on a degree-biased random walk with variable length to sample context information for learning. Our walk strategy can well adapt to the scale-free feature of real-world networks and extract information from them with much less redundancy. In addition, our method can greatly reduce the size of context information, which is efficient for large-scale network embedding. Empirical experiments on node classification and link prediction prove not only the effectiveness but also the efficiency of DiaRW on a variety of real-world networks. Our algorithm is able to learn the network representations with millions of nodes and edges in hours on a single machine, which is tenfold faster than previous methods.

## 1. Introduction

In recent years, there is a growing tendency for the popularity of intelligent applications in cities. These applications could provide intelligent solutions that support millions of users, by harnessing the predictive power behind humongous volumes of data from cities and deployed sensors. Such data could involve human collaboration, interactions, communication or social behaviors, which is highly nonlinear and complicated. Network, represented by nodes and the connections between nodes as edges, is a proper tool to characterize such high dimensional data and their interactions. In fact, networks have been widely used to denote information in various areas including social sciences(social network) [1], linguistics (semantic Web) [2], Internet of Things(sensor network) [3] and biology (Protein–Protein interaction network) [4].

The scale of complex networks ranges from hundreds to billions of nodes, leading to a problem of how to analyze large networks in an efficient way. Network embedding, which maps each node to a low-dimensional vector, provides a ubiquitous way to study global and local properties of the networks [5,6]. Given a network, it is often desirable to extract latent information associated with each node in the network. The latent information can represent a variety of properties of the original network. For example, it may preserve the local neighborhood information of each node as well as global community structure of the network. Therefore, the node representations can be used as features for network analysis and network prediction tasks

[*] Corresponding author at: Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, Engineering Research Center of data storage systems and Technology, Ministry of Education of China, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China.
[**] Corresponding author.
*E-mail addresses:* ivyzhang@hust.edu.cn (Y. Zhang), zshi@hust.edu.cn (Z. Shi), dfeng@hust.edu.cn (D. Feng), X.Zhan@tudelft.nl (X.-X. Zhan).

https://doi.org/10.1016/j.future.2019.05.033
0167-739X/© 2019 Elsevier B.V. All rights reserved.

such as classification [7], clustering [8,9], link prediction [10,11], and visualization [12,13].

Despite the enormous potential of network embedding, we argue that there exists two main challenges:

**High non-linearity:** As stated in [14], network data is often sophisticated and the underlying structure of it is highly non-linear, which means it is quite difficult to design a suitable model that can well capture and preserve the network structure.

**Scalability:** With the arrival of the age of big data, the scale of real-world networks is exploding. Taking social network as an example, the Twitter network contains 175 million active users and approximately twenty billion edges in 2012 [15]. Therefore, when it comes to large-scale networks, the massive learning task may cost months to complete, or even simply fail due to insufficient memory, which is practically unfeasible for practical applications.

The general network embedding method usually takes two steps: In the first step, a sampling procedure which can explore non-linear relationship between nodes is performed to get the node 'corpus', then word embedding method such as word2vec [16] is applied to obtain the embedding vector for each node. DeepWalk [5] is the pioneer work in using random walks to learn node representations. Node2Vec [6] is an extension of DeepWalk which introduces a biased random walk procedure which combines BFS and DFS style neighborhood exploration.

Real-world networks are generally scale-free, i.e., most nodes have a low degree while only a few have very high degrees. The node degree skewness implies that the underlying information around nodes could differ a lot. However, previous network embedding methods based on random walks treated this with a 'one-size fits all' strategy which is sub-optimal to well preserve the structure of networks: One the one hand, they simply take a unified walk strategy for all nodes to sample from real-world networks which makes them unable to adapt to the different local properties; On the other hand, random walks with fixed length for each node can generate a lot of redundancy which are common in a scale-free network, restricting the scalability for large-scale networks.

To demonstrate the inference above, we use Barabási–Albert (BA) model [17] to generate a scale-free network with $2^{16}$ nodes and plot the degree distribution of nodes in Fig. 1(a). We can find that the original network follows a standard power-law distribution with a slope -2.6656. By contrast, we also plot the degree distribution of nodes in a corpus generated by the uniform random walk used in DeepWalk in Fig. 1(b). It can be seen that the degree distribution generated by the uniform random walk differs significantly from the real degree distribution. Therefore, the uniform random walk is not well enough to preserve the original network properties.

In order to effectively and efficiently preserve the network topology, we propose a high-degree biased variable-length random walk algorithm which considers the degree skewness. To be specific, we allow a walk to step back to the nodes with higher degree in a probabilistic way, which means high-degree nodes tend to be revisited more and walks starting from them could obtain richer information by traveling around the local neighborhoods. Moreover, instead of setting fixed length for all the random walks, we set the length of random walks starting from each node based on its centrality in order to solve the problem of generating redundant information. Taking degree centrality as an example, we assume that high-degree nodes should have longer walk length than low-degree nodes when served as starting nodes. In this way, we can not only better preserve the network topology to ensure the quality of node representations but also drastically reduce space and computation overhead by condensing the fixed-length sequences into variable-length ones, making our algorithm more efficient and scalable on large-scale networks.

The reminder of the paper is organized as follows. In Section 2, we first give a review for the related work. Then, we give the proposed method in Section 3. In Section 4, we empirically evaluate our method on predicting tasks, i.e., node classification and link prediction, on large-scale networks and analyze the parameter sensitivity as well as scalability of our algorithm. The paper is concluded in Section 5.

## 2. Related work

Network representation has become an important way to analyze complex network. The learning methods can be categorized into two types: matrix factorization (MF)-based and neural network-based [18].

MF-based methods are either linear [19] or nonlinear [20] in learning node embedding. The former employs the linear transformations to embed network nodes into a low dimensional embedding space, such as singular value decomposition (SVD) and multiple dimensional scaling (MDS) [19]. However, the latter maps network nodes into a low dimensional latent space by utilizing the nonlinear transformations, e.g., kernel PCA [21], spectral embedding, marginal fisher analysis (MFA) [22], and manifold learning approaches including LLE [23] and ISOMAP [20]. Generally speaking, MF-based methods have two main drawbacks: (1) Due to the eigen-decomposition operations on data matrices, they are usually computationally expensive and are difficult to be applied on large-scale network data [24,25]; (2) the performance is rather sensitive to the predefined proximity measures for calculating the affinity matrix.

Neural network-based methods are the state-of-art node representation learning techniques. The pioneer work DeepWalk [5] extended the idea of Skip-Gram [16] to model network, which is convert to a corpus of node sequences by performing truncated random walks. The Node2Vec algorithm [6] can essentially be considered as an extension of DeepWalk, introducing a biased random walk procedure which combines BFS style and DFS style neighborhood exploration. However, both of them adopted a global walk strategy which ignores individual heterogeneity. In addition, another shortcoming of Node2Vec is that its second-order random walks take too much time to compute the interconnections between neighbors of every node. There are some follow-up works exploiting both 1st-order and 2nd-order proximity between nodes to embed networks. Specifically, LINE [26] derives a joint optimization function for preserving the first and the second order proximity. It performs the optimization by stochastic gradient descent with edge sampling, aiming at efficient embedding of large-scale networks. The goal is similar as our paper, nevertheless, the performance tends to be inferior compared to ours due to its limitation and inflexibility for low order proximity. HOPE [27] defines some similarity measures between nodes which are helpful for preserving higher-order proximity as well and formulates those measures as a product of sparse matrices to efficiently find the latent representations. However, the algorithm still showed poor scalability for large-scale networks.

## 3. Proposed method

In this section, we introduce DiaRW, a network embedding method based on Skip-Gram model. In fact, the efficiency of the Skip-Gram based methods largely depends on the sampling strategies. In our method, instead of using uniform random walk for sampling, we propose a high-degree biased backtracking method to extract information from the network. In addition,

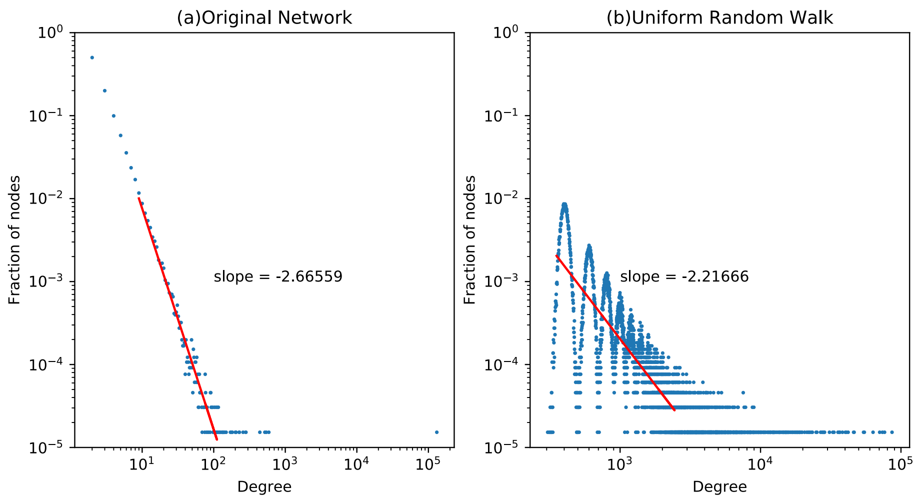

**Fig. 1.** The degree distributions of (a) the original BA network with $2^{16}$ nodes and (b) the corpus generated by uniform random walk from the original network.

**Table 1**
Table of notation.

| | |
|---|---|
| $G = (V, E)$ | An input network $G$ with set $V$ of nodes and set $E$ of edges |
| $\Phi(u)$ | Node representations of $u$ |
| $N_S(u)$ | Neighbor set of node $u$ generated by a sampling strategy $S$ |
| $P(u, v)$ | Transit probability from node $u$ to node $v$ |
| $Q(u, v)$ | Probability of selecting node $v$ as next hop of node $u$ |
| $A(u, v)$ | Probability of accepting node $v$ as next hop of node $u$ when $v$ is selected |
| $R(v, u)$ | Probability of backtracking from node $v$ to node $u$ |
| $S(u, v)$ | Similarity score of node $u$ and node $v$ |
| $Deg(u)$ | Degree of node $u$ |
| $L_{max}$ | Upper bound of walk length |
| $L(u)$ | Actual walk length starting at node $u$ |
| $k$ | Times of random walks per node |
| $d$ | Dimension of vector representations |
| $w$ | Window size of context of the Skip-Gram |

we also introduce a variable-length strategy based on the centrality of source node for the random walks, in order to reduce redundancy in the sampling process.

A network is defined as $G = (V, E)$, where $V$ is node set and $E$ is the set of edges. Table 1 includes notation used throughout the paper.

### 3.1. Network embedding framework

Network embedding aims to learn a mapping function $\Phi$: $V \rightarrow R^d (d \ll |V|)$, where we use $\Phi(u)$ to represent the embedding vector of node $u$, and $d$ is the dimension of $\Phi(u)$. The function $\Phi$ preserves network topology, such that two nodes which are similar in the original network should also be close in the embedding space.

Inspired by [5,6], we formulate network embedding as a maximum likelihood optimization problem. For every center node $u \in V$, we define the neighbor set of $u$ as $N_S(u)$, which is generated by a sampling strategy $S$. For example, $N_S(u)$ can be a set of nodes within $k$-hop distance from $u$. Therefore, we give the objective function that we need to optimize as:

$$argmax \sum_{u \in V} logPr(N_S(u)|\Phi(u)) \tag{1}$$

Skip-Gram [16] is a language model that maximizes the co-occurrence probability among the words that appear within a window size $w$, in a sentence. It can approximate the conditional probability in Eq. (1) by assuming that predicting nodes in a context set is independent of each other as:

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|\Phi(u)) \tag{2}$$

Algorithm 1 [5] shows the extension of Skip-Gram model to networks. We map each node $u$ from walk sequences to its current representation vector $\Phi(u)$. Given the representation of $u$, in order to maximize the probability of its neighbors, we use stochastic gradient descent to iteratively update it (line 3–4).

**Algorithm 1: Skip-Gram ($\Phi$, $walks$, $w$)**

```
Input: matrix of node representations $\Phi$
       walk sequences $walks$
       window size $w$
1 for each node $u \in walks$ and its index as $inx_u$ do
2     for each node $n_i \in walks[inx_u - w, inx_u + w]$ do
3         $J(\Phi) = -logPr(n_i|\Phi(u))$
4         $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5     end for
6 end for
```

And we use soft-max function to model the conditional likelihood of every center-neighbor node pair as:

$$Pr(n_i|\Phi(u)) = \frac{exp(\Phi(n_i) \cdot \Phi(u))}{\sum_{v \in V} exp(\Phi(v) \cdot \Phi(u))}, \tag{3}$$

As computing $\sum_{v \in V} exp(\Phi(v) \cdot \Phi(u))$ is very expensive, we use the negative sampling method [16] to speed up training.

### 3.2. Scale-free networks

A scale-free network is a network whose degree distribution follows a power law, or at least asymptotically. Most of real world networks are reported to be scale-free [17], from web graphs to social networks, protein networks and semantic networks.

In Fig. 2, we give a toy example of scale-free network, where most nodes have a low degree but some have a very high degree. Nodes with a number of edges that greatly exceeds

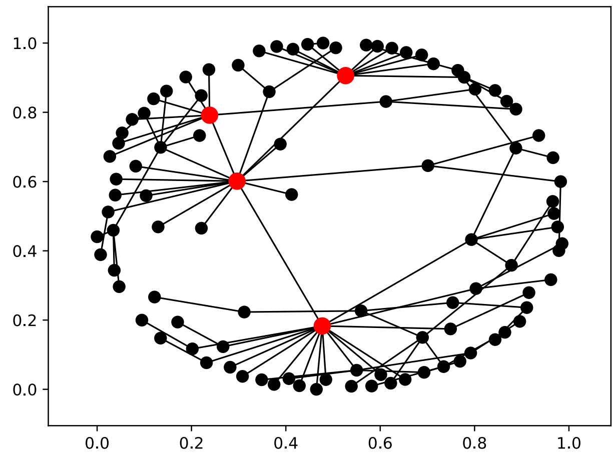

**Fig. 2.** A toy example of scale-free network(nodes colored red are hubs).

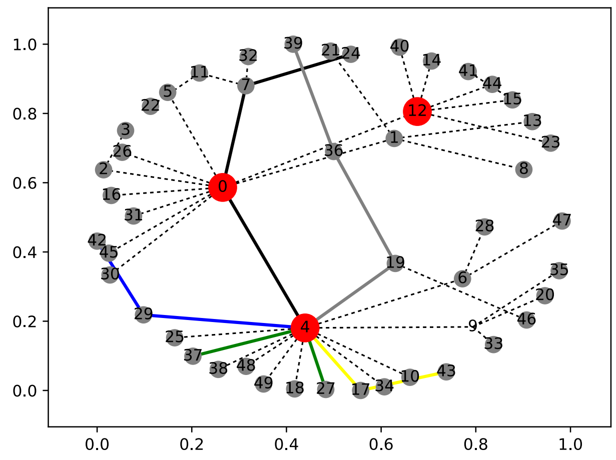

**Fig. 3.** An example of uniform random walk sampling on a scale-free network. It starts from hub node 4 with the number of walks 5 and walk length 4, generating node sequences(shown in solid line) like "4-29-42-29"(blue),"4-0-7-24"(black),"4-27-4-37"(green),"4-19-36-39"(gray) and "4-17-43-17"(yellow). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the average are called hubs in the network (red color nodes in Fig. 2). Hubs usually play important roles in the network. For example, the hubs can help information spreads to more people [28,29], and we can control the epidemic spreading by separating the infected hub individuals from the susceptible population [30]. We show how the uniform random walk fails to extract structure information for the hub nodes in Fig. 3. We set the number of walk to be 5 and the walk length to be 4. Taking the walks starting from hub node 4 as an example, we can get walks such as "4-29-42-29"(blue),"4-0-7-24"(black),"4-27-4-37"(green),"4-19-36-39"(gray) and "4-17-43-17"(yellow). On a limited budget of samples, we can see that uniform random walks from node 4 fails to completely capture the neighborhood information around it with a large part of immediate neighbors unvisited. This problem can be more serious for real world networks since there exists nodes with degrees of tens of thousands whose structure is hard to be well extracted under acceptable sample size. As the hubs and low-degree nodes can have different roles in the network, we need to treat them differently when performing the sampling procedure.

## 3.3. Sampling strategy

### 3.3.1. High-degree biased backtracking

Given a random walk on network starting from node $u$, the transition process could be decomposed into two actions of selection and acceptance or rejection, which means a neighbor $v$ of node $u$ have the probability of $Q(u,v)$ to be selected, and once it happens, node $v$ will then be accepted as next hop of the walk with probability of $A(u,v)$, otherwise it will be rejected. According to Metropolis–Hastings algorithm [31], the desired node distribution $\pi$ generated by the random walk can be associated with $A(u,v)$ and $Q(u,v)$ as:

$$A(u,v) = \left\{1, \frac{\pi(v)Q(v,u)}{\pi(u)Q(u,v)}\right\} \tag{4}$$

For a uniform random walk, the neighboring node of u has equal probability to be selected and the transition will definitely happen after the selection. Therefore, we have $Q(u,v) = \frac{1}{Deg(u)}$ and $A(u,v)$ is 1. According to Eq. (4), we have $\pi(u) = \frac{d(u)}{2|E|}$, which means the node distribution of uniform random walk is linear with the degree in this scenario. We argue that the bias for high-degree nodes is still inadequate to stress the importance of hubs, since there is a large number of low-degree nodes in scale-free networks, greatly diluting the distribution of high-degree nodes, which also theoretically accounts for the demonstration in Section 3.2 that uniform random walk fails to completely capture the neighborhood information.

To further intensify the bias for high-degree nodes, we simply replace the linear relationship with quadratic relationship and modify the desired distribution as: $\frac{\pi(u)}{\pi(v)} = \frac{d(u)^2}{d(v)^2}$. Substituting it into Eq. (4), along with $Q(u,v) = \frac{1}{Deg(u)}$, we could obtain the revised acceptance probability as $A(u,v) = \left\{1, \frac{Deg(v)}{Deg(u)}\right\}$, the corresponding rejection probability $R(u,v)$ can then be computed as: $R(u,v) = 1 - A(u,v) = max\left\{0, 1 - \frac{Deg(v)}{Deg(u)}\right\}$.

To define rejection in a walk process, we introduce a novel backtracking mechanism in random walk, specifically, when carrying out a walk from node $u$, we randomly select a neighbor $v$ of node $u$ and transit to node $v$ directly, once satisfying the rejection condition, we perform a backtracking from node $v$ to node $u$. Formally, the high-degree biased backtracking mechanism can be described with the transition and backtracking probability as:

$$\begin{aligned} P(u,v) &= \frac{1}{Deg(u)}, \\ R(v,u) &= max\left\{0, 1 - \frac{Deg(v)}{Deg(u)}\right\} \end{aligned} \tag{5}$$

From a macro perspective, by restricting search from high-degree nodes to local neighborhoods, the high-degree biased backtracking simulates a BFS-like explorations in the dense area, while in the sparse area, it tends to perform a DFS-like explorations by moving further away from low-degree nodes, thus better capturing the local and global network structure.

### 3.3.2. Variable-length walk

Random walk based methods like Node2Vec and DeepWalk set the fixed walk length for each node in networks, which is poorly adaptive to the local density of networks. On the one hand, long-length walks could sample a lot of redundant information in the sparse area such as back and forth walks between low-degree nodes, training samples generated by such invalid node sequences may increase the risk of over-fitting. In addition, setting the walk length too large would directly increase the sampling time as well as storage and computation cost for training

Skip-Gram model, and then restrict the algorithm's scalability to large-scale networks. However, short-length walks lack ability to sufficiently capture the neighborhood information in the dense area of the network such as community structures.

Considering the limitation of fixed-length strategy, we propose a variable-length walk strategy based on the centrality of nodes, since the local density of networks could be refined into the centrality of nodes, which means dense areas in a network should be constructed with important nodes. There are many ways to define node centrality in a network: such as degree [32], closeness [33], betweenness [34], PageRank and HITS. Betweenness centrality and closeness centrality involve calculating the shortest paths between all pairs of nodes on a network, which is unfeasible for practical application. Therefore, we simply abandon them and evaluate the effectiveness and efficiency of introducing the other three measures on our variable-length strategy respectively. We combine the three variable-length walk strategies with Skip-Gram to obtain the node representations for *BA* networks with increasing network sizes from $10^3$ to $10^5$ nodes. The time of centrality computation and the AUC of link prediction based on the corresponding node representations are shown in Table 2, from which we can find that these three centrality measures achieve almost equal performances for AUC. However, PageRank and HITS require considerable extra time for iterative computation. In summary, we choose degree to measure the node centrality for our variable-length strategy.

Intuitively, to respond to the high-degree biased backtracking strategy, it is more reasonable to give high-degree nodes longer walk length to better cover the backtracking process. The standard definition of degree centrality is the number of links incident upon a node without normalization, which means it can be directly used as walk length without scale factor. In this way, we can focus more on high-degree nodes by giving them larger walk length. Nevertheless, with the expanding of scale-free networks, the degree of nodes will vary by several orders of magnitude and the degree of hubs may have tens of thousands of links. We argue it unnecessary and time-consuming to set this long walk length for an individual node in a walk , because the walk may have already returned to the starting nodes and repeated multiple times. Therefore, we smooth the huge differences between degree by setting a upper bound of walk length to restrict hub nodes. Formally, for any starting node $u$, the walk length can be computed as

$$L(u) = min\{Deg(u), L_{max}\} + 1 \tag{6}$$

The actual length is increased by 1 to guarantee that walks from nodes of degree one could happen. Algorithm 2 depicts our complete sampling strategy.

We compare our DiaRW sampling strategy with the uniform random walk sampling strategy of DeepWalk in Fig. 4 for *BA* networks with increasing network sizes from $10^2$ to $10^6$ nodes. We set all the parameters to be the same except that in DeepWalk, the walk length is fixed as 80, in our work, we set $L_{max} = 80$. We find that our sampling strategy outperforms Deepwalk both in the scale of walk sequences and sampling time. Our method is able to finish sampling the *BA* network with millions of nodes in dozens of minutes while it takes several hours for Deepwalk for the same dataset. Most surprisingly, the degree distribution generated by our random walk from the *BA* network given in Fig. 1(a) shows a slope of −2.6632 (Fig. 5), which is much closer to the original network degree distribution (−2.6656) compared to the uniform random walk sampling (−2.2167 given in Fig. 1(b)).

**Table 2**
Evaluation of different centrality measure (AUC/time).

| Size | Degree | PageRank | HITS |
|---|---|---|---|
| 1 000 | 0.6099/5.4E−4 | 0.5984/0.24 | 0.6040/0.81 |
| 10 000 | 0.6399/6.0E−3 | 0.6413/2.24 | 0.6418/13.42 |
| 100 000 | 0.6395/0.06 | 0.6406/21.65 | 0.6412/280.48 |

### 3.4. The DiaRW algorithm

The pseudo-code of our entire method DiaRW is given in Algorithm 3. The algorithm consists of two main components: (1) a sampling generator and (2) a learning procedure. Algorithm 2 serves as the sampling generator, and we use algorithm 1 shown in 3.1 to train and learn the node representations.

```
Algorithm 2: DiaRW_walk($G$, $u$, $L_{max}$)
Input: Network $G(V, E)$, max walk length $L_{max}$
Output: Node sequence $walk$
1 Initialize $walk$ to $[u]$
2 $l = min\{Deg(u), L_{max}\} + 1$
3 for $i$ = 0 to $l$ do
4     $curr = walk[-1]$
5     Select a node $v$ uniformly from neighbors of curr
6     Append $v$ to $walk$
7     Generate a random value $p \in [0, 1]$
7     if $p < (1 - \frac{Deg(v)}{Deg(u)})$ then
9         Append curr to $walk$
10 end for
11 return $walk$
```

```
Algorithm 3: DiaRW($G$, $L_{max}$, $k$, $w$, $d$)
Input: Network $G(V, E)$
    max walk length $L_{max}$
    walks per node $k$
    window size $w$
    embedding dimension $d$
Output: matrix of node representations $\Phi \in R^{|V| \times d}$
1 Initialize walks to empty
2 for $iter$ = 1 to $k$ do
3     for all nodes $u \in V$ do
4         $walk$ = DiaRW_walk($G$, $u$, $L_{max}$)
5         Append $walk$ to $walks$
6     SkipGram($\Phi$, $walks$, $w$)
7     end for
8 end for
9 return $\Phi$
```

## 4. Experimental evaluation

We compare our DiaRW algorithm with four other baseline methods, i.e., DeepWalk [5], Node2Vec [6], LINE [26] and HOPE [27], for the tasks of multi-label node classification and link prediction. Our experiment environment is listed in Table 3.

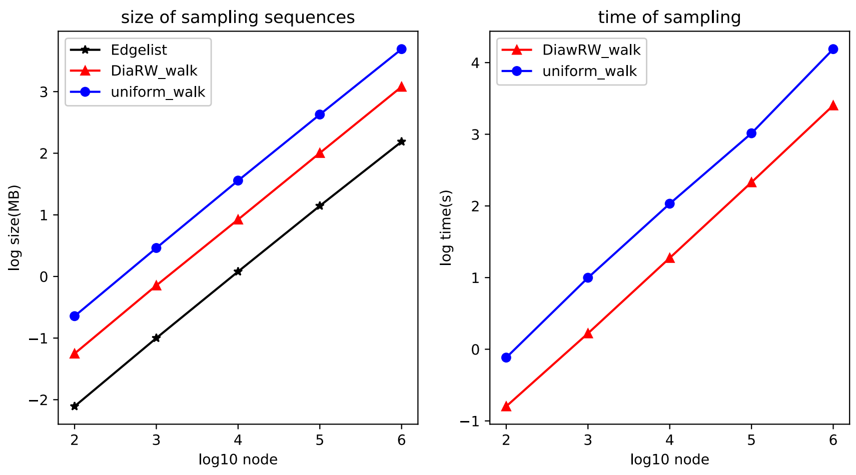

**Fig. 4.** The Space and Time cost of walk.

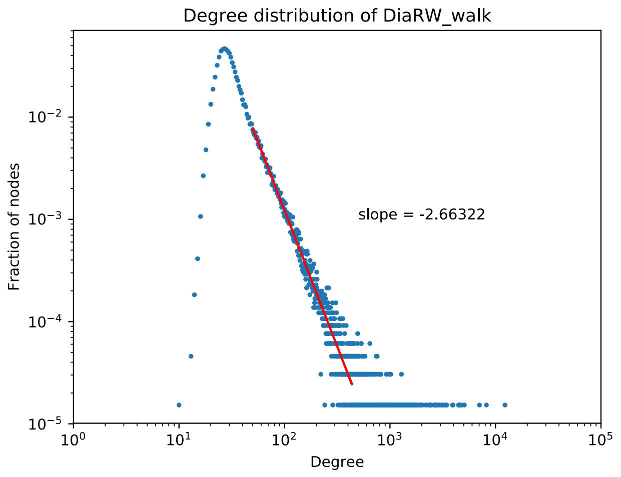

**Fig. 5.** The node frequency distribution of the corpus generated by DiaRW random walk for *BA* graph.

**Table 3**
Experiment environment.

| | |
|---|---|
| OS | CentOS 4.8.5-16 Linux 4.4.114 |
| MEMORY | 128 GB |
| DISK | 300 GB |
| CPU | Xeon(R) CPU E5-2620 v4 @ 2.10 GHz |

**Table 4**
Network datasets.

| Network | $\lvert V\rvert$ | $\lvert E\rvert$ | Avg.deg | Density | Avg.cc | Labels |
|---|---|---|---|---|---|---|
| YouTube | 1 134 890 | 2 987 624 | 5.265 | 4.6E−6 | 0.40 | 47 |
| PPI | 56 944 | 818 716 | 28.75 | 5.0E−4 | 0.18 | 121 |
| Flickr | 80 513 | 5 899 882 | 146.55 | 1.8E−2 | 0.16 | 194 |
| email-Eu-core | 1 005 | 25 571 | 33.24 | 0.03 | 0.39 | |
| Wiki-Vote | 7 115 | 103 689 | 28.32 | 0.003 | 0.14 | |
| p2p-Gnutella | 8 114 | 26 013 | 6.41 | 7.9E−4 | 7.2E−3 | |
| Astroph | 18 722 | 198 110 | 21.10 | 1.1E−3 | 0.63 | |
| Cit-HepPh | 34 546 | 421 578 | 24.36 | 7.0E−4 | 0.28 | |
| Epinions | 75 877 | 508 837 | 10.69 | 1.4e−4 | 0.13 | |
| Twitter | 11 316 811 | 85 331 846 | 11.23 | 9.9E−7 | 0.14 | |

### *4.1. Network datasets*

Table 4 gives a summary of network datasets used in our experiments. To prove the efficiency and effectiveness of our algorithm, we choose the datasets of different sizes, ranging from thousands to millions of nodes. The detailed information of network datasets for the multi-label classification task is given as follows:

- **YouTube** [35]: A social network between users on Youtube. This is a large network containing 1,157,827 nodes, 4,945,382 edges and 47 labels. The labels represent groups of users who enjoy common video genres.
- **Protein–Protein Interaction (PPI)** [36]: The network contains 56,944 nodes, 818,716 edges and 121 labels. Each of the labels corresponds to a biological function of the proteins.
- **Flickr** [35]: This is a network of the contacts between users of the photo sharing website. It contains 80,513 nodes, 5,899,882 unweighted edges and 194 labels. The labels represent the interest groups of the users such as 'black and white photos'.

The datasets used for link prediction are as follows:

- **email-Eu-core** [37]: The network was generated using email data from a large European research institution. The emails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world, it contains 1,005 nodes and 25,571 edges.
- **Wikipedia vote network(Wiki-Vote)** [38]: The network extracted all administrator election and voting history data from Wikipedia community, where node represents the users who had participated in the election or been elected and edge indicates a voting process. The network contains 7,115 nodes and 103,689 edges.
- **Gnutella peer-to-peer network (p2p-Gnutella)** [37]: A sequence of snapshots of the Gnutella peer-to-peer file sharing network from August, 2002. Nodes represent hosts in the Gnutella network and edges represent connections between the them. The network contains 8,114 nodes and 26,013 edges.
- **High-energy physics citation network(Cit-HepPh)** [39]: This is a citation network generated from papers submitted to the e-print arXiv where nodes represent papers, if a paper cites another paper, the network contains a directed edge between them. It has 34,546 nodes and 421,578 edges.
- **Astrophy collaboration** [37]: This is a collaboration network generated from papers submitted to the e-print arXiv where nodes represent scientists, and an edge is formed between two scientists if they have collaborated on one paper. The network has 18,722 nodes and 198,110 edges.
- **Epinions** [40]: The network represents who-trust-whom relationships between users of the epinions.com product review website. It has 75,877 nodes and 508,837 edges.
- **YouTube** [35]: The same dataset used in the node-classification task.

**Table 5**
Time cost on embedding for multilabel classification (seconds).

| | DiaRW | Node2Vec | DeepWalk | LINE |
|---|---|---|---|---|
| YouTube | **3391** | 581 726 | 31 668 | 272 833 |
| PPI | **290** | 1 573 | 572 | 2 538 |
| Flickr | **2166** | 179 917 | 2 487 | 11 001 |

• **Twitter** [41]: Twitter is a social news website. It can be viewed as a hybrid of email, instant messaging and SMS messaging all rolled into one neat and simple package. The nodes represent users and friends are represented using edges. This is a typical large-scale network with ten million nodes and edges, taking up 1 GB storage.

### *4.2. Baseline methods*

We use the following four methods as the baselines:

• **DeepWalk** [5]: DeepWalk adopts uniform random walk for node sampling and Skip-Gram model to generate network representation.

• **Node2Vec** [6]: This method extends DeepWalk by performing biased random walks to generate the corpus of node sequences. It contains the in–out and return hyper-parameters $p$ and $q$. We have performed a grid search over $p, q \in \{0.25, 0.5, 1, 2, 4\}$ and 10-fold cross-validation on labeled data to select the best embedding, as suggested by [6].

• **LINE** [26]: This method optimizes both the 1st-order and 2nd-order proximity in a network. We use the LINE (1st+2nd) method which has shown the best results in their paper. The original version of LINE is implemented in C++, for comparison fairness, we use an implementation of LINE in Python with TensorFlow.

• **HOPE** [27]: This method defines similarity measures between nodes which are helpful for preserving higher-order proximity and formulates these measures as a product of sparse matrices to efficiently find the latent representations. The authors experimented with different similarity measures, including Katz Index, Rooted PageRank, Common Neighbors, and Adamic–Adar score. The Katz index with decay parameter $\beta = 0.1$ is selected for HOPE's high-order proximity measurement, since this setting gave the best performance in the original article.

### *4.3. Experiments on multi-label classification*

Predicting node labels using network topology is widely applied in modern applications ranging from document classification [42] to interest prediction [43]. Among these applications, multi-label node classification is significantly challenging, especially for networks with a large number of labels. To perform this task, we use the learned node vector and an one-vs-rest logistic regression classifier (using the LIBLINEAR library with L2 regularization) [44]. When training the classifier, we randomly sample a portion of the labeled nodes as the training set and the rest as the testing set. For PPI, we randomly sample 10% to 90% of the nodes as the training samples and use the left ones to test the performance. For Flicker and YouTube, we randomly sample 1% to 10% of the nodes as the training samples and use the left nodes to test the performance, which corresponds to the fact that these two datasets have only a small part of labeled nodes for entire networks. We repeat the experiment for 5 times and report the averaged Micro-F1 and Macro-F1.

The results are shown in Fig. 6 and Table 5. Since HOPE failed to learn the embedding in our current experimental environment for all the datasets on multi-label classification, we only show the results of the remaining three methods as baselines and compare them with our DiaRW. The time cost for learning embedding is given in Table 5. From the results, we have the following observations and analysis:

• In Fig. 6, we observe that random walk based methods outperform LINE in the multi-label classification task. The main reason can be inferred from the fact that LINE simply aims to capture low-order proximities for nodes: only nodes which are at most two hops away from a center node are considered as its context. This is not enough for node classification as high-order proximity neighbors can also be classified by the same labels. In contrast, by generating random walks in the network, the neighborhoods are not restricted to just one-hop or two-hop neighbors but can have vastly different structures.

• More precisely, as shown in Fig. 6, we can see that Node2Vec gives a satisfactory performance in the multi-label classification task. As mentioned in [6], Node2Vec preserves homophily as well as structural equivalence between nodes. Results suggest this can be useful in node classification. In particular, Node2Vec is not inferior to DeepWalk for all datasets, which means biased random walks have better adaptability and accuracy for capturing network structures to generate a corpus with high fidelity than uniform random walks. However, despite the gaining for accuracy over DeepWalk, Node2Vec is far less efficient than DeepWalk (Table 5). It takes at least three times longer than DeepWalk to learn the embedding for the same dataset, which is even more noticeable for large-scale networks. This is because, Node2Vec requires a preprocess procedure to compute and store the interconnections between the neighbors of every node for 2nd-order random walks, which is pretty expensive on both time and space for large-scale networks, therefore greatly affects the efficiency and scalability of embedding.

• In addition, our method DiaRW shows a competitive performance to Node2Vec, but with much higher efficiency. Specifically, regarding to the Macro-F1 and Micro-F1 score, our method shows comparable results as Node2vec and DeepWalk in YouTube network. In PPI network, our method improves Macro-F1 score by 9.8% and Micro-F1 score by 1.5% over DeepWalk. In Flicker network, our method outperforms all the baselines, gaining 12.5% improvement on Macro-F1 score compared to Node2vec. Taking time cost showing in Table 5 together, we can conclude that DiaRW can finish embedding several times faster than DeepWalk and dozens of times faster than Node2Vec while maintaining the effectiveness for multi-label classification. The huge gains in time are mainly due to the variable-length walk strategy we adopt, which has drastically reduced the size of node sequences and accelerated walking and training as well.

• The experimental results proved that there is a lot of redundancy information sampled by uniform random walk with fixed length, which will not only be useless for accuracy but also greatly slow down the algorithm. For networks with many types of labels but short of labeled data such as Flicker [35], our method can get even better performance than prior excellent works due to better representative of network structure.

### *4.4. Experiments on link prediction*

Networks are constructed from the observed links between nodes, which may be incomplete or inaccurate. The challenge often lies in identifying spurious interactions and predicting missing links. Link prediction refers to the task of predicting either missing links or contacts that may appear in the future in an evolving network [11,45]. Link prediction can be translate into the similarity-based problem, where each pair of nodes, u and v, is assigned a similarity score $S(u, v)$ and the links connecting more similar nodes are supposed to be of higher existence likelihoods.

To perform link prediction in a network, we first randomly remove half of its edge. The node representation is then learned

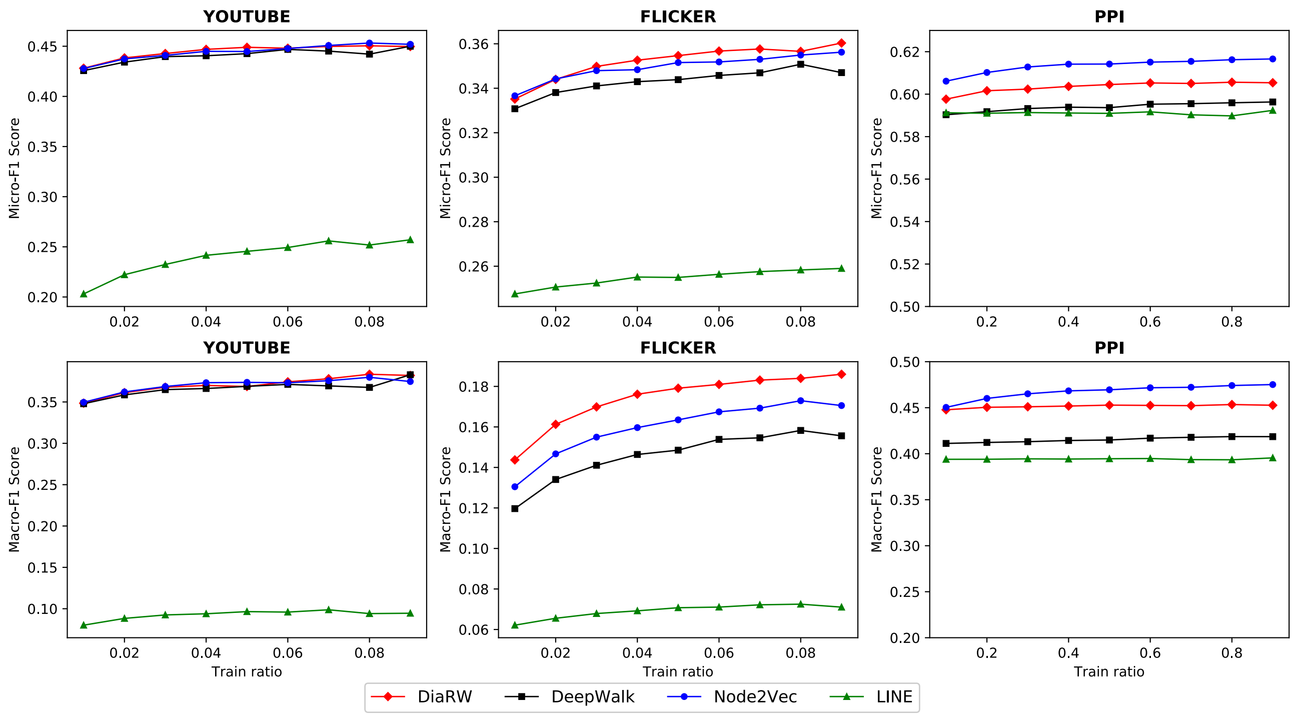

**Fig. 6.** Performance evaluation on various datasets for multi-label classification.

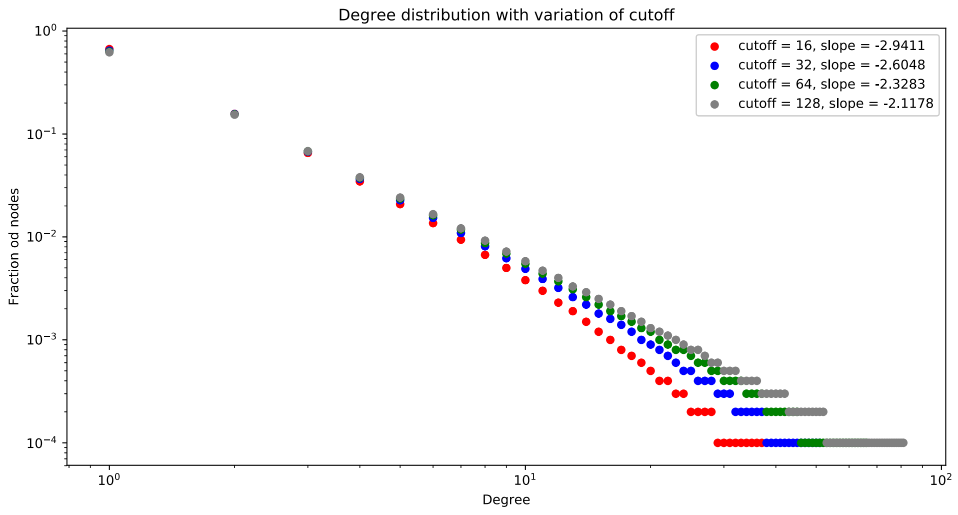

**Fig. 7.** Degree distribution of synthetic networks with different cutoff $\kappa$.

**Table 6**
Area Under Curve (AUC) scores for link prediction.

| | DiaRW | Node2Vec | DeepWalk | HOPE | LINE |
|---|---|---|---|---|---|
| email-Eu-core | 0.8572 | 0.8222 | 0.8270 | **0.8618** | 0.7870 |
| Wiki-Vote | **0.9356** | 0.7957 | 0.7946 | 0.9283 | 0.8101 |
| p2p-Gnutella | **0.7641** | 0.7025 | 0.6947 | 0.6254 | 0.6878 |
| Cit-HepPh | 0.9517 | **0.9576** | 0.9472 | 0.5125 | 0.7356 |
| Astrophy | 0.9159 | **0.9217** | 0.9088 | 0.5320 | 0.8896 |
| Epinions | **0.8972** | 0.8512 | 0.8463 | * | 0.8417 |
| YouTube | **0.7985** | 0.7726 | 0.7681 | * | 0.6463 |
| Twitter | **0.9010** | * | * | * | * |

**Table 7**
Time cost for link prediction task (seconds).

| | DiaRW | Node2Vec | DeepWalk | HOPE | LINE |
|---|---|---|---|---|---|
| email-Eu-core | 3 | 25 | 8 | **1** | 10 |
| Wiki-Vote | **10** | 113 | 28 | 180 | 29 |
| p2p-Gnutella | **7** | 111 | 59 | 25 | 87 |
| Cit-HepPh | **53** | 405 | 167 | 725 | 230 |
| Astrophy | **46** | 438 | 175 | 157 | 953 |
| Epinions | **109** | 3 466 | 1 069 | * | 4 066 |
| YouTube | **1 448** | 581 726 | 31 668 | * | 272 833 |
| Twitter | **37 167** | * | * | * | * |

from the remaining part. To create the negative labels for the prediction task, we randomly select pairs of nodes that are not connected in the original network. The number of such pairs is equal to the number of removed edges. The "negative" pairs and the pairs from edges that have been removed, are used together to form the labeled data for this task. Given embedding vector $\Phi(u)$ and $\Phi(v)$ of two nodes $u$ and $v$, we define the similarity score $S(u, v)$ as inner product $\Phi(u) \cdot \Phi(v)$, along with AUC metric to evaluate the performance. The results are given in Table 6. In addition, we also measure the time cost of representation

**Table 8**
The impact of degree heterogeneity on the performance of link prediction task.

| Cutoff $\kappa$ | $\|V\|$ | $\|E\|$ | $CV$ | $Hm$ | AUC of DiaRW | AUC of DeepWalk |
|---|---|---|---|---|---|---|
| 16 | 10 000 | 10 252 | 1.2596 | 0.0583 | 0.5059 | 0.4724 |
| 32 | 10 000 | 11 329 | 1.5261 | 0.0671 | 0.5692 | 0.4865 |
| 64 | 10 000 | 12 714 | 1.7791 | 0.0748 | 0.6279 | 0.4956 |
| 128 | 10 000 | 14 007 | 2.0099 | 0.0849 | 0.6616 | 0.5136 |

**Table 9**
Evaluations of *DiaRW_BT* and *DiaRW_VarL*.

| | DeepWalk | *DiaRW_BT* | *DiaRW_VarL* | DiaRW |
|---|---|---|---|---|
| email-Eu-core | 0.8270 | 0.8475 | 0.8229 | 0.8572 |
| Wiki-Vote | 0.7946 | 0.8477 | 0.7922 | 0.9356 |
| Cit-HepPh | 0.9472 | 0.9543 | 0.9336 | 0.9517 |
| Astrophy | 0.9088 | 0.9151 | 0.8885 | 0.9159 |
| Epinions | 0.8463 | 0.8673 | 0.8188 | 0.8852 |

learning for all the methods in Table 7, where "$*$" means the algorithm fails under the limitation of computation resources (Table 3) and time requirement (one week). From the results, we have the following observations and analyses:

• DiaRW gains the best overall performance in this task. More precisely, it achieves 8.8% improvement on AUC for p2p-Gnutella, 5.4% for Epinions and 3.95% for Youtube, compared to the best results from baseline methods. In addition, for the two small datasets: email-Eu-core and Wiki-Vote, DiaRW shows competitive performance to HOPE, which is far superior to other methods, and it performs as well as Node2Vec for Cit-HepPh and Astrophy networks. In summary, experimental results sufficiently show the advantage of our method on link prediction task, which can well adapt to networks of various sizes and fields.

• As shown in Table 6, in comparison with Node2Vec, DiaRW gains significant improvements on AUC for all the datasets except for Cit-HepPh and Astrophy networks, ranging from 4% and 18%. We observe that these datasets all exhibit such characteristics with low link density and average degree relative to the number of nodes, which means their degree distribution presents more notable heterogeneity. Therefore, we can infer that our walk strategy can better adapt to this scale-free and skew phenomenon and thus can effectively retain the network structures. As for Cit-HepPh and Astrophy networks with higher density and clustering, our walk strategy shows less obvious difference from uniform random walk, leading to a close performance with other random walk based methods.

To further verify the deduction above, we quantitatively explore the impact of degree heterogeneity on the performance of embedding. We use the model proposed in [46] to generate synthetic networks with different degree heterogeneity by multiplying the parameter exponential cutoff $\kappa$ from 32 to 128 with $\alpha = 2$. The degree distribution of these synthetic networks is shown in Fig. 7. We choose coefficient of variation($CV$), defined as the ratio of the standard deviation to the mean and a degree heterogeneity measure($Hm$) proposed in [47], to measure the heterogeneity of the synthetic networks. We also compute the AUC on link prediction task to evaluate the representations learned by DiaRW and DeepWalk for comparison. The results are given in Table 8, we find that, with the increase of heterogeneity, the performance of DiaRW achieves significant improvement on AUC. Additionally, DiaRW shows great advantages over DeepWalk, providing strong evidence that DiaRW can better adapt to heterogeneity property of real-world networks than uniform random walks.

• The performance of LINE in link prediction task is obviously better than that in the multi-label classification task, which is even comparable to that of random walk based methods on some datasets. We can infer from that low-order proximity can be helpful for link prediction task. However, real-world networks tend to be so sparse that we cannot extract enough low-order information for representation learning. In view of this, random walk based methods is more flexible and effective as they use a random walk to enrich the neighbors of nodes, which is able to introduce higher order proximities. The performance of HOPE is highly dependent on the dataset , which implies its poor adaptability to different networks.

• Results from Table 7 prove once again that our method is very scalable and efficient for large-scale networks. Taking a typical large-scale network Twitter as an example, all the methods except for DiaRW have failed to obtain the node representations. As a contrast, it takes only ten hours for DiaRW to learn the embedding for Twitter, with a superior performance for link prediction task.

## 4.5. Separate effect of backtracking and variable-length

Our walk strategy *DiaRW_walk* can be divided into two sub-strategies as high-degree biased backtracking and variable-length walk. In order to explore their effects separately, we design two variants of DiaRW based on the two sub-strategies respectively, naming *DiaRW_BT* and *DiaRW_VarL*. Taking several networks from Section 4.1 as examples, we use link prediction task to evaluate *DiaRW_BackTrack* and *DiaRW_VarL*, along with DiaRW and DeepWalk as comparisons.

As shown in Table 9, separate variable-length walk has little impact on the improvement of AUC when compared with DeepWalk, since it aims to improve the efficiency of network embedding. While high-degree biased backtracking can indeed increase the accuracy of prediction task with superior node representations, as described in Section 3.3.1, we can better capture the local and global network structures by paying more attention to high-degree nodes, bringing about a modest improvement on AUC. And notably, this improvement can be further enhanced when combined with variable-length walk, verifying the illustration from Section 3.3.2 that high-degree nodes need larger walk length to cover the loss of frequent backtracking. In summary, we can conclude that both of the sub-strategies from Section 3.3 take effects, of which the high-degree biased backtracking can directly benefit good effectiveness of node representations, and variable-length walk gains huge improvement on efficiency, meanwhile further exploiting the advantages of the former for better performance.

## 4.6. Parameter sensitivity

We explore how the different choices of parameters affect the performance of DiaRW. Fig. 8 shows the AUC gained by DiaRW on link prediction task for p2p-Gnutella network. Except for the parameter being tested, all the other parameters in the experiment are set to their default value. We find that the parameters related to the walk process(times of walk per node $k$, upper bound of walk length $L_{max}$) all have impact on the performance of embedding. The parameter $L_{max}$ has less influence since based on our random walk strategy, the actual walk length for every nodes depends not only on this parameter, but also on their degrees to a great extent. With the increase of parameter $k$ or parameter $L_{max}$, the AUC will have a significant improvement at first. This benefits from a greater overall sampling budget to learn representation. However, when these two parameters increase too much, the AUC tends to remain stable or even decrease, which means the current sampling size is sufficient to extract the network structures. This further implies that redundant information does no good for representation learning. Similarly, we

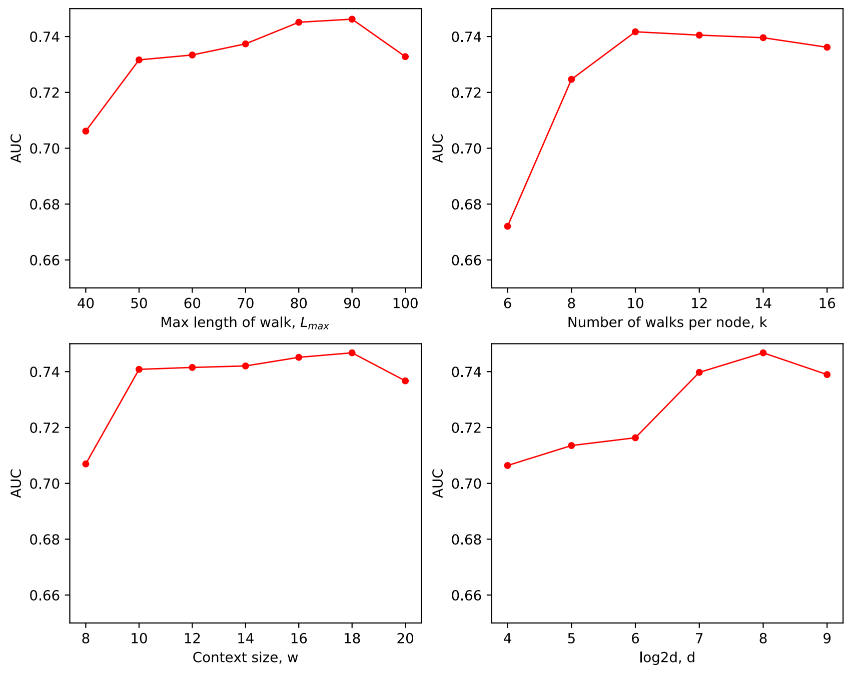

**Fig. 8.** Parameter sensitivity of DiaRW in p2p-Gnutella.

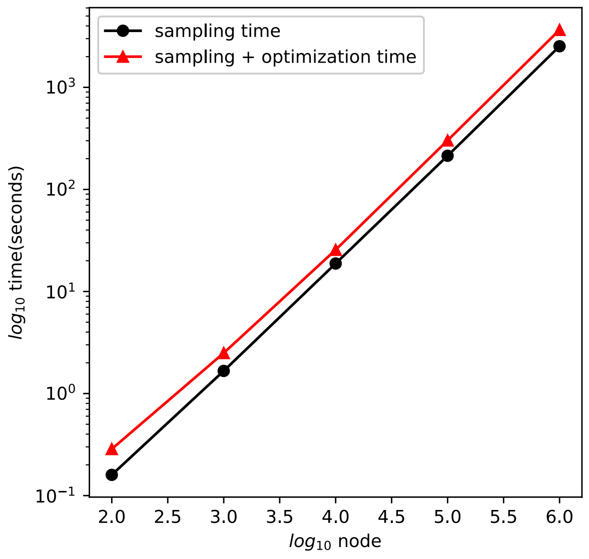

**Fig. 9.** Scalability of DiaRW on Barabási–Albert networks.

observe that increasing the context size $w$ for Skip-Gram model will also improve the AUC since larger context size could discovery higher order relationships in the network which is helpful for network inference. However, once this parameter is set too large, it will in turn introduce noise, attenuating the impact of closer neighborhoods, which accounts for the slight degradation of performance. The effectiveness of our method also depends on the dimension number $d$ of output vector representations. We can infer that vectors with too small dimensions lack expressive ability, embedding in this representation space may not be able to preserve the structure information of the networks, while continuously increasing the number of vector dimensions by adding more nodes on the hidden layer of neural network will increase the risk of over-fitting problem, which could also negatively affect the performance.

### 4.7. Scalability

To test scalability, we have learned the node representation using DiaRW with default parameter values for *BA* networks with increasing size of networks from $2^{10}$ to $2^{20}$ nodes and edges attaching from a new node to existing nodes of 10. Fig. 9 depicts the running time required for sampling and both sampling and optimization. DiaRW is able to learn embeddings for networks with millions of nodes in dozens of hours and scales linearly with respect to the size of network. Since the optimization phase is made efficient using negative sampling [48] and asynchronous SGD [49], nearly total learning time belongs to the step of sampling nodes.

## 5. Conclusions and future work

In this work, we have proposed DiaRW, an efficient method for network embedding, which can easily scale up to networks with millions of nodes and billions of edges. The core of DiaRW is a sampling procedure with biased backtracking mechanism and variable-length strategy, which can well adapt to the scale-free characteristic of real-world networks and greatly reduce the redundant information when compared with fixed-length random walks, thus ensuring its competitive performance on prediction tasks and its significant improvement on efficiency comparing with state-of-the-art baseline methods. As future work, we plan to extend our method to networks with special properties such as heterogeneous information networks, networks with explicit domain features for nodes and edges and signed-edge networks.

## Acknowledgments

This work is supported by NSFC, China No. 61772216, 61821003, U1705261, Wuhan Application Basic Research Project No. 2017010201010103, Fund from Science, Technology and Innovation Commission of Shenzhen Municipality No. JCYJ20170307172248636, and Fundamental Research Funds for the Central Universities, China. The authors declare that there is no conflict of interests regarding the publication of this manuscript.

## References

[1] M. Eirinaki, J. Gao, I. Varlamis, K. Tserpes, Recommender systems for large-scale social networks: A review of challenges and solutions, Future Gener. Comput. Syst. 78 (2018) 413–418, http://dx.doi.org/10.1016/j.future.2017.09.015.

[2] S.D. Cardoso, F.K. Amanqui, K.J.A. Serique, J.L.C. dos Santos, D.A. Moreira, SWI: A semantic web interactive gazetteer to support linked open data, Future Gener. Comput. Syst. 54 (2016) 389–398, http://dx.doi.org/10.1016/j.future.2015.05.006.

[3] J. Kamruzzaman, G. Wang, G. Karmakar, I. Ahmad, M.Z.A. Bhuiyan, Acoustic sensor networks in the internet of things applications, Future Gener. Comput. Syst. 86 (2018) 1167–1169, http://dx.doi.org/10.1016/j.future.2018.05.019.

[4] A. Theocharidis, S. van Dongen, A.J. Enright, T.C. Freeman, Network visualization and analysis of gene expression data using biolayout express(3D), Nat. Protoc. 4 (10) (2009) 1535–1550, http://dx.doi.org/10.1038/nprot.2009.177.

[5] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD'14, ACM, New York, NY, USA, 2014, pp. 701–710.

[6] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD'16, ACM, New York, NY, USA, 2016, pp. 855–864.

[7] S. Bhagat, G. Cormode, S. Muthukrishnan, Node classification in social networks, Comput. Sci. 16 (3) (2011) 115–148, http://dx.doi.org/10.1007/978-1-4419-8462-3.

[8] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, in: NIPS'01, MIT Press, Cambridge, MA, USA, 2001, pp. 849–856.

[9] A. Faroughi, R. Javidan, CANF: Clustering and anomaly detection method using nearest and farthest neighbor, Future Gener. Comput. Syst. 89 (2018) 166–177, http://dx.doi.org/10.1016/j.future.2018.06.031.

[10] D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks, in: Proceedings of the Twelfth International Conference on Information and Knowledge Management, in: CIKM'03, ACM, New York, NY, USA, 2003, pp. 556–559.

[11] S. Aslan, M. Kaya, Topic recommendation for authors as a link prediction problem, Future Gener. Comput. Syst. 89 (2018) 249–264, http://dx.doi.org/10.1016/j.future.2018.06.050.

[12] P.E. Rauber, A.X. Falcão, A.C. Telea, Visualizing time-dependent data using dynamic t-SNE, in: Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers, in: EuroVis'16, Eurographics Association, Goslar Germany, Germany, 2016, pp. 73–77.

[13] J. Gómez-Romero, M. Molina-Solana, A. Oehmichen, Y. Guo, Visualizing large knowledge graphs: A performance analysis, Future Gener. Comput. Syst. 89 (2018) 224–238, http://dx.doi.org/10.1016/j.future.2018.06.015.

[14] D. Luo, C. Ding, F. Nie, H. Huang, Cauchy Graph embedding, in: Proceedings of the 28th International Conference on Machine Learning, in: ICML'11, Omnipress, USA, 2011, pp. 553–560.

[15] S.A. Myers, A. Sharma, P. Gupta, J. Lin, Information network or social network: The structure of the twitter follow graph, in: Proceedings of the 23rd International Conference on World Wide Web, in: WWW'14, ACM, New York, NY, USA, 2014, pp. 493–498.

[16] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space, CoRR abs/1301.3781.

[17] A.L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512, http://dx.doi.org/10.1126/science.286.5439.509.

[18] W. Yuan, K. He, G. Han, D. Guan, A.M. Khattak, User behavior prediction via heterogeneous information preserving network embedding, Future Gener. Comput. Syst. 92 (2019) 52–58, http://dx.doi.org/10.1016/j.future.2018.09.036.

[19] M.E. Mugavin, Multidimensional scaling: a brief overview, Nursing Res. 57 (1) (2008) 64–68, http://dx.doi.org/10.1097/01.nnr.0000280659.88760.7c.

[20] A. Nedich, A. Ozdaglar, A geometric framework for nonconvex optimization duality using augmented lagrangian functions, J. Global Optim. 40 (4) (2008) 545–573, http://dx.doi.org/10.1007/s10898-006-9122-0.

[21] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319, http://dx.doi.org/10.1162/089976698300017467.

[22] S. Yan, D. Xu, B. Zhang, H.J. Zhang, Graph embedding: a general framework for dimensionality reduction, in: Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, in: CVPR'05, vol. 2, 2005, pp. 830–837.

[23] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326, http://dx.doi.org/10.1126/science.290.5500.2323.

[24] M. Wang, W. Fu, S. Hao, H. Liu, X. Wu, Learning on big graph: Label inference and regularization with anchor hierarchy, IEEE Trans. Knowl. Data Eng. 29 (5) (2017) 1101–1114, http://dx.doi.org/10.1109/TKDE.2017.2654445.

[25] M. Wang, W. Fu, S. Hao, D. Tao, X. Wu, Scalable semi-supervised learning by efficient anchor graph regularization, IEEE Trans. Knowl. Data Eng. 28 (7) (2016) 1864–1877, http://dx.doi.org/10.1109/TKDE.2016.2535367.

[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE:large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, in: WWW'15, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2015, pp. 1067–1077.

[27] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD'16, ACM, New York, NY, USA, 2016, pp. 1105–1114.

[28] Z.K. Zhang, C. Liu, X.X. Zhan, X. Lu, C.X. Zhang, Y.C. Zhang, Dynamics of information diffusion and its applications on complex networks, Phys. Rep. 651 (2016) 1–34, http://dx.doi.org/10.1016/j.physrep.2016.07.002.

[29] C. Liu, X.X. Zhan, Z.K. Zhang, G.-Q. Sun, P.M. Hui, How events determine spreading patterns: information transmission via internal and external influences on social networks, New J. Phys. 17 (11) (2015) 113045, http://dx.doi.org/10.1088/1367-2630/17/11/113045.

[30] R. Cohen, S. Havlin, D. Ben-Avraham, Efficient immunization strategies for computer networks and populations, Phys. Rev. Lett. 91 (24) (2003) 247901, http://dx.doi.org/10.1103/physrevlett.91.247901.

[31] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika (1970) http://dx.doi.org/10.1093/biomet/57.1.97, http://arxiv.org/abs/5744249209.

[32] L.C. Freeman, Centrality in social networks conceptual clarification, Social Networks 1 (3) (1978) 215–239, http://dx.doi.org/10.1016/0378-8733(78)90021-7.

[33] G. Sabidussi, The centrality index of a graph, Psychometrika 31 (4) (1966) 581–603, http://dx.doi.org/10.1007/BF02289527.

[34] L. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1977) 35–41, http://dx.doi.org/10.2307/3033543.

[35] L. Tang, H. Liu, Scalable learning of collective behavior based on sparse social dimensions, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, in: CIKM'09, ACM, New York, NY, USA, 2009, pp. 1107–1116.

[36] M. Livstone, B.-J. Breitkreutz, C. Stark, L. Boucher, A. Chatr-Aryamontri, R. Oughtred, J. Nixon, T. Reguly, J. Rust, A. Winter, K. Dolinski, M. Tyers, The biogrid interaction database, Nature Prec. 41 (2011) : D637—-D640., http://dx.doi.org/10.1038/npre.2011.5627.1.

[37] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data 1 (1) (2007) http://dx.doi.org/10.1145/1217299.1217301.

[38] J. Leskovec, D. Huttenlocher, J. Kleinberg, Signed networks in social media, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, in: CHI '10, ACM, New York, NY, USA, 2010, pp. 1361–1370.

[39] J. Gehrke, P. Ginsparg, J. Kleinberg, Overview of the 2003 KDD cup, ACM SIGKDD Explorations Newsletter 5 (2003) 149–151, http://dx.doi.org/10.1145/980972.980992.

[40] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: The Semantic Web - ISWC 2003, Vol. 2870, 2003, pp. 351–368.

[41] R. Zafarani, H. Liu, Social Computing Data Repository at ASU, Arizona State University, School of Computing, Informatics and Decision Systems Engineering, 2009, http://socialcomputing.asu.edu.

[42] S. Bhushan, A. Danti, Classification of compressed and uncompressed text documents, Future Gener. Comput. Syst. 88 (2018) 614–623, http://dx.doi.org/10.1016/j.future.2018.04.054.

[43] B. Xu, H. Zhuge, An angle-based interest model for text recommendation, Future Gener. Comput. Syst. 64 (2016) 211–226, http://dx.doi.org/10.1016/j.future.2016.04.011.

[44] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, C.J. Lin, LIBLINEAR: A library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874, http://dx.doi.org/10.1145/1390681.1442794.

[45] X.X. Zhan, A. Hanjalic, H. Wang, Information diffusion backbones in temporal networks, arXiv preprint arXiv:1804.09483.

[46] M. Newman, Spread of epidemic disease on networks, Phys. Rev. E 66 (1 Pt 2) (2002) 16128, http://dx.doi.org/10.1103/physreve.66.016128.

[47] R. Jacob, K. Harikrishnan, R. Misra, G. Ambika, Measure for degree heterogeneity in complex networks and its application to recurrence network analysis, R. Soc. Open Sci. 4 (1) (2017) 160757, http://dx.doi.org/10.1098/rsos.160757.

[48] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, in: NIPS'13, vol. 2, Curran Associates Inc., USA, 2013, pp. 3111–3119.

[49] N. Feng, B. Recht, C. Re, S.J. Wright, Hogwild!: A lock-free approach to parallelizing stochastic gradient descent, Adv. Neural Inf. Process. Syst. 24 (2011) 693–701.

**Yunyi Zhang**, born in 1994, M.S. candidate. Her research interest is graph analysis and data mining.

**Zhan Shi** received his B.S. degree and Master degree in Computer Science, and Ph.D. degree in Computer Engineering from Huazhong University of Science and Technology (HUST), China. He is working at the Huazhong University of Science and Technology (HUST) in China, and is an associate professor in Wuhan National Research Center for Optoelectronics. His research interests include graph processing, distributed storage system and cloud storage.

**Dan Feng** received the B.E., M.E., and Ph.D. degrees in Computer Science and Technology in 1991, 1994, and 1997, respectively, from Huazhong University of Science and Technology (HUST), China. She is a professor and the dean of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, massive storage systems, and parallel file systems. She has more than 100 publications in major journals and international conferences, including IEEETC, IEEETPDS, ACM-TOS, FAST, USENIX ATC, ICDCS, HPDC, SC, ICS, IPDPS, and ICPP. She has served as the program committees of multiple international conferences, including SC 2011, 2013 and MSST 2012, 2015. She is a member of IEEE and a member of ACM.

**Xiu-Xiu Zhan** received the B.S. and M.S. degree in Mathematics from North University of China, Shanxi, China, in 2012 and 2016, respectively. She is currently a PhD candidate working in the Multimedia Computing Group at the Delft University of Technology. Her research interests includes complex networks and recommender systems.