SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

October 2020, Vol. 63 202103:1–202103:19
https://doi.org/10.1007/s11432-018-9943-9

Dynamic network embedding via incremental skip-gram with negative sampling

Hao PENG[1,2], Jianxin LI[1,2*], Hao YAN[1,2], Qiran GONG[2], Senzhang WANG[3], Lin LIU[2], Lihong WANG[4] & Xiang REN[5]

[1]*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing* 100083, *China;*
[2]*State Key Laboratory of Software Development Environment, Beihang University, Beijing* 100083, *China;*
[3]*Collage of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing* 211106, *China;*
[4]*National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing* 100029, *China;*
[5]*Department of Computer Science, University of Southern California, Los Angeles,* 90089, *USA*

Received 11 October 2018/Revised 14 March 2019/Accepted 10 June 2019/Published online 18 September 2020

Abstract Network representation learning, as an approach to learn low dimensional representations of vertices, has attracted considerable research attention recently. It has been proven extremely useful in many machine learning tasks over large graph. Most existing methods focus on learning the structural representations of vertices in a static network, but cannot guarantee an accurate and efficient embedding in a dynamic network scenario. The fundamental problem of continuously capturing the dynamic properties in an efficient way for a dynamic network remains unsolved. To address this issue, we present an efficient incremental skip-gram algorithm with negative sampling for dynamic network embedding, and provide a set of theoretical analyses to characterize the performance guarantee. Specifically, we first partition a dynamic network into the updated, including addition/deletion of links and vertices, and the retained networks over time. Then we factorize the objective function of network embedding into the added, vanished and retained parts of the network. Next we provide a new stochastic gradient-based method, guided by the partitions of the network, to update the nodes and the parameter vectors. The proposed algorithm is proven to yield an objective function value with a bounded difference to that of the original objective function. The first order moment of the objective difference converges in order of $\mathbb{O}(\frac{1}{n^2})$, and the second order moment of the objective difference can be stabilized in order of $\mathbb{O}(1)$. Experimental results show that our proposal can significantly reduce the training time while preserving the comparable performance. We also demonstrate the correctness of the theoretical analysis and the practical usefulness of the dynamic network embedding. We perform extensive experiments on multiple real-world large network datasets over multi-label classification and link prediction tasks to evaluate the effectiveness and efficiency of the proposed framework, and up to 22 times speedup has been achieved.

Keywords dynamic network embedding, bound and convergence analysis, multi-label classification, link prediction

Citation Peng H, Li J X, Yan H, et al. Dynamic network embedding via incremental skip-gram with negative sampling. Sci China Inf Sci, 2020, 63(10): 202103, https://doi.org/10.1007/s11432-018-9943-9

1 Introduction

Recently network representation learning, also known as network embedding, has received considerable research attention. That is due to the fact that many real-world problems in complex systems, such as

* Corresponding author (email: lijx@act.buaa.edu.cn)

recommended systems, social networks and biology networks, can be modelled as machine learning tasks over large network. The idea of network embedding is to learn a mapping that projects each vertex in a network to a low dimensional and continuous distributed vector space, where each vertex is represented as a dense vector. The mapping is learned with the objective of preserving the structural information of the original network in the geometric relationships among vertices' vector representations [1]. Network representation learning has been proven to be a useful tool for various real-world network mining tasks such as vertex community detection [2], recommended system [3], anomaly detection [4], multi-label classification [5–9], link prediction [5–7], and knowledge representation [10].

Previous studies have proposed several prominent network embedding methods. DeepWalk and node2vec capture higher-order proximities in embeddings by maximizing the conditional probability of observing the neighbourhood of vertices of a vertex given the mapped point of the vertex. Here the neighbourhood vertices are obtained from vertices traversed in a random walk. The crucial difference between DeepWalk [6] and node2vec [5] is that node2vec employs a biased random walk procedure to provide a trade-off between breadth-first search (BFS) and depth-first search (DFS) in a network, which might lead to a better mapping function. LINE [7] and SDNE [11] learn graph embeddings by preserving the first- and second-order proximities in the embedded space, where the former refers to the pairwise neighborhood relationship and the latter is determined by the similarity of nodes' neighbors. The difference is that the SDNE uses highly non-linear functions to represent the mapping function.

Most existing network embedding methods [12–15] focus on learning the node representations in static network where no temporal information is associated with the nodes and edges. However, the majority of real-world networks are dynamical and continuously growing over time (i.e., nodes occur and disappear, and edges are added and vanish as time goes), such as the friendship network in Facebook, the citation network in DBLP, and the web-pages hyperlink dataset updating in Wikipedia. There are a lot of scenarios, such as real-time social network node classification and knowledge graph link processing, requiring dynamic update of the node representation given the fact that the working domains are fast evolving. Unfortunately, the above methods ignore the dynamic nature and are unable to efficiently update the vertices' representations in accordance with networks' evolution. However, prior studies have demonstrated that, besides the dynamic edge and vertex modeling, the negative sampling or hierarchical softmax optimizing for representation learning is tremendous importance in capturing the evolution patterns of the dynamic network [16, 17]. When the difference between the updated network and the old network is relatively small, it is inefficient to obtain the new node embeddings through retraining the entire new network. Indeed, the few very recent studies [16–19] adapted from the above methods required either prior knowledge of new vertices' attributes or retrained on new graphs with uncertain convergence time. It is a challenge for many high-throughput production machine learning systems that need generating the representations of new vertices promptly.

In this paper, we study the problem of efficiently learning the node embedding for dynamic networks by proposing an incremental skip-gram with negative sampling model. In particular, we adopt the popular and fundamental network representation models, such as DeepWalk and node2vec, due to their simplicity, interpretability, time efficiency, and comparable performance to other complex network embedding technologies [7, 11, 20–22]. The two models make use of skip-gram which is initially proposed in natural language processing (NLP) to train vertice representations through generating the sequences of the vertex by random walk. To speed up the training process, unsupervised neural network based language learning models employ two techniques called hierarchical softmax and negative sampling [23, 24]. Hierarchical softmax was first proposed by Morin and Bengio [25] where a hierarchical tree is constructed to index all the words in a corpus as leaves. Negative sampling is developed based on noise contrastive estimation [26] and randomly samples the words not in the context to distinguish the observed data from the artificially generated random noise. The fixed number of the negative samples replaces the variable layers of hierarchy. Although the original DeepWalk employs hierarchical softmax [6], it can be also implemented using the negative sampling like node2vec and LINE. Considering the interpretability, popularity and good performance of skip-gram and negative sampling on various representation learning models [5–7, 22–24, 27], we investigate the problem of learning dynamic network embeddings with a focus

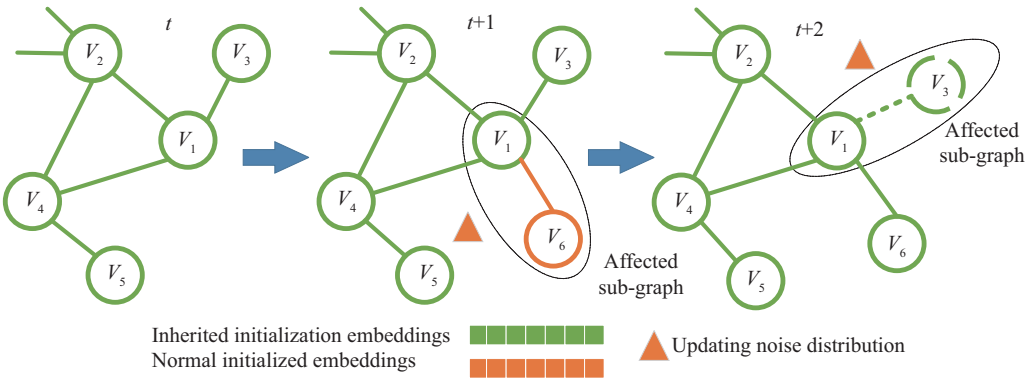

Figure 1 (Color online) An illustration of the temporal evolving of the dynamic network. The green vertices and edges constitute the initial network in time t. The vertex V_6 (marked in orange color) and the corresponding edge (V_1, V_6) (marked in orange color) emerge in time $t+1$. The vertex V_3 (marked in dotted line) and the corresponding edge (V_1, V_3) (marked in dotted line) vanish in time $t+2$.

on designing an incremental skip-gram model with negative sampling (ISGNS).

When applying skip-gram with negative sampling to network representation learning, the first problem is to investigate the structure proximities and compute the noise distributions for negative sampling [5–7, 22–24, 27, 28]. When the vertices and edges of a network evolve over time, as shown in Figure 1, the proximities and noise distributions will update automatically to reflect the change of the network structure. For example, in DeepWalk and node2vec, they use edges to construct the sequences of the vertices and the noise distribution over the vocabulary, and result in the faster training process. In the dynamic scenario, when the edges, the edge weights and the vertices change, the sequences of vertices, the structure proximities and the noise distribution should be updated correspondingly. To address this issue, we first partition the network into the updated part (new added/vanished links and nodes) and the retained part. Then, we employ random walk and sliding window [5, 6] to extract the sequences of the nodes or subgraphs, namely affected sequences of subgraphs in the network. To speed up model training, our model inherits all the retained nodes and parameter vectors and implements a new stochastic gradient-based method to update the changed nodes and parameter vectors, by comparing the old and updated networks. When updating the vectors for the updated part of the subgraphs, we make use of stochastic gradient descent and ascent methods based on the latest noise distributions to optimize the model. In this way, we only need to update vectors in affected subgraphs. Our theoretical analyses reveal that, under a mild assumption, the objective difference can be bounded by the scale of the old network, and the convergence of objective difference can also be bounded. So the optimal solution of the dynamic network embedding by ISGNS agrees with the original network SGNS when the network scale is infinitely large. Because the update process is independent of all the shared vectors, we also present the techniques for an efficient parallel implementation of dynamic network embedding with ISGNS. In the experiments, we show that the proposed model can significantly reduce the training time while preserving comparable performances with state-of-the-art models on static networks. The code of this study is publicly available at the web[1)].

Our main contributions are summarized as follows.

• A dynamic network embedding framework based on an approximately optimal solution of incremental skip-gram with negative sampling is proposed, which can be directly applied in existing network embedding models such as DeepWalk and node2vec.

• The solid theoretical analyses show that our proposal guarantees the boundness of the objective difference and the convergence when the training network scale is infinitely large. The empirical study also verifies the boundary and moments of the network dynamic change.

• Extensive experiments on multiple large real-world network datasets show both the efficiency and effectiveness of the proposed ISGNS on multi-label classification and link prediction tasks. ISGNS achieves

1) https://github.com/RingBDStack/dynamic_network_embedding.

up to 22 times speedup while preserving comparable performance with global re-training methods.

The remainder of the paper is organized as follows. We first review the related work in Section 2. Then we introduce the proposed ISGNS model in detail in Section 3. Section 4 provides the mathematical details of the dynamic objective difference, corresponding bound analysis and convergence analysis for both first-order and second-order moments. We evaluate our model in Section 5, and finally conclude this study in Section 6.

2 Related work

In this section, we briefly review related work on network embedding models, including static network embedding and dynamic network embedding technologies.

Static network embedding. DeepWalk [6] is the first work that utilizes a truncated random walk to transform a static network into a collection of node sequences. Then the skip-gram on hierarchical softmax function[2)] is utilized to learn the vertex representations. Node2vec [5] further generalizes DeepWalk with breadth-first search (BFS) and depth-first search (DFS) on random walks, and employs the popular skip-gram with negative sampling to learn the vertex representations. LINE [7] and SDNE [11] model the first-order and second-order proximities between vertices, and employ the skip-gram with negative sampling to deal with the limitation of stochastic gradient descent on weighted edges without compromising the efficiency. Struct2vec [29] proposes to preserve the structural identity between nodes in the representation. To achieve this goal, it first creates a new graph based on the structural identity similarity between nodes and then follows a similar method to DeepWalk on the created graph. A very recent method GraphWave [30] makes use of wavelet diffusion patterns by treating the wavelets from the heat wavelet diffusion process as distributions. Overall, those methods of generalized network embeddings are typically designed to go through the entire network multiple times. It means that they cannot perform online learning of the node representation in a dynamic scenario when the vertices, edges and edge weights change over time.

Dynamic network embedding. DANE [31] leverages a matrix perturbation theory to update the dynamic attributed network spectral embeddings. Zhu et al. [18] proposed a temporal latent space learning model BCGD via non-negative matrix factorization to target the link prediction task in dynamic social networks. But it belongs to especial embedding method for the purpose of link prediction. Jian et al. [32] designed an online embedding representation learning method OLSN based on spectral embedding used for node classification. However, the proposed unsupervised dynamic network embedding models are more generalized. Trivedi et al. [16] proposed a deep recurrent architecture Know-Evolve modeling the historical evolution of entity representations in a specific relationship space. Compared to the proposed unsupervised dynamic network embedding method, the Know-Evolve model consumes lots of memory and computational time. Xu et al. [33] proposed a statistical model Dynamic SBM for dynamic networks that utilized a set of unobserved time-varying states to characterize the dynamics of the network. Zhou et al. [34] proposed a triadic closure process based semi-supervised algorithm Dynamic Triad to learn the structural information and evolution pattern in dynamic networks. Du et al. [35] proposed a heuristic dynamic network embedding method DNE, which employed a decomposable objective based on the skip-gram objective, and gave the objective function difference minimization. Zuo et al. [17] proposed a Hawkes process based temporal network embedding method HTNE which captured the influence of the historical neighbors on the current neighbor formation simultaneously. Inspired by the unsupervised neural network representation learning, previous incremental word embedding models [36–39] proposed the incremental hierarchical softmax function, the small adaptive unigram table based negative sampling for incremental word embeddings, and Gaussian random walk based dynamic word embedding. For existing dynamic network embedding and analysis models [16–18, 31–33, 35], these models belong to heuristic methods, and cannot theoretically guarantee the equivalence and optimality of generalized network embedding objective function. Even, the computational cost or memory cost linearly increases with the assumes and

2) However, an alternative to the hierarchical softmax is noise contrastive estimation (NCE) [26, 28].

learning time. For the bright dynamic network embedding model [34], it cannot handle the addition of vertices, and the scalability of the model and the hypothetical process are the bottlenecks when applied in real large scale networks. In addition, the above discussed dynamic network representation learning models have not strictly followed the original object in the sampling optimization. Therefore, different from existing studies, we study the popular neural network based dynamic network embedding from the perspectives of the objective function and the sampling strategy.

3 Dynamic network representation learning

As we discussed above, network representation learning is sensitive to the network structure and the objective proximities among vertices. When the edges and vertices evolve over time, we depict the structural and proximity differences of the network snapshots in different time slots by metabolic sub-graphs, which directly reflect the changes in edges, vertices, and noise distributions. Inspired by the principle of network embedding approximating the adjacency matrix [22], we assume that the influence of structural changes on the representation learning is partial in neighborhoods/sub-graphs for limited adjacency matrix float. We firstly locate the metabolic sub-graphs by local random walk in re-training. Then, for newly added nodes or edges, we implement the random walk only on the sub-graphs to generate sequences of vertices. Note that one vertex sequence contains at least one new node or edge. For the vanished nodes or edges, we also implement the random walk on the sub-graphs to generate sequences of vertices following the same rule. Then for each vertex in the above sequences, we re-calculate its frequency and add/subtract the result to/from its frequency in the previous network. Thus, we obtain the latest noise distributions. For a fast training in dynamic network scenario, we adopt a strategy that inherits the vertexes and the parameter vectors through changes in the network structures. If part of the network remains the same, we can retain the vectors as well as the structure associated to the nodes, and distinguish the updated sub-graphs between the old network and the new network structures.

3.1 Node initialization and inheritance

Given a network $\mathcal{W}$ in time t, we formulate the updated network $\mathcal{W}'$ in time $t+1$ as

$$\mathcal{W}' = \mathcal{W} + \Delta\mathcal{W}_{\text{inc}} - \Delta\mathcal{W}_{\text{dis}}, \tag{1}$$

where $\Delta\mathcal{W}_{\text{inc}}$ and $\Delta\mathcal{W}_{\text{dis}}$ refer to the newly added and vanished sub-graphs, respectively. We re-calculate the noise distributions for each vertex on new network $\mathcal{W}'$ in time $t+1$ with random walk [5, 6] on the above sub-graphs.

We first preserve all the vertices and the corresponding parameter vectors in the old network. Then we inherit the reserved vertex and parameter vectors as initialization in the new network. If a vertex is newly added, we initialize it as a random vector with the same dimension as the existing vertexes, and the related parameter vector is initialized as a zero vector. It can be formally defined as follows:

$$v'(u) = \begin{cases} v(u), & u \in \mathcal{W}, \\ \text{random}, & u \notin \mathcal{W}, \end{cases} \tag{2}$$

and

$$\widetilde{v}'(u) = \begin{cases} \widetilde{v}(u), & u \in \mathcal{W}, \\ 0, & u \notin \mathcal{W}, \end{cases} \tag{3}$$

where $v(u)$ and $v'(u)$ are the representation vectors of node u for the old and new networks, and $\widetilde{v}(u)$ and $\widetilde{v}'(u)$ are the parameter vectors of u in the old and new networks, respectively.

3.2 Model updating

In a dynamic network, we assume the updated nodes and edges only affect the representations of the local nodes and edges, and perform an approximate stochastic gradient method to update the related

vectors. In detail, after generating the sequence of vertices, given the size of sliding window $2c$, we can build local sub-graphs, named as affected sub-graphs, for the newly added and vanished vertices and edges. The approximate stochastic gradient method can be described as following two steps. Firstly, for the vanished vertices and edges, we perform a stochastic gradient descent method to update the old network representations with the updated noise distributions. Secondly, after inheriting and initializing the vertexes and the parameter vectors, for the newly added vertexes and edges, we perform a stochastic gradient ascent method to update the new network representations. More specific, we extend the widely used skip-gram with negative sampling method to dynamic sampling scenarios for network embeddings. In terms of vertex representation updating, this yields to such an optimization problem:

$$\max_{f} \quad \sum_{u\in\mathcal{W}'} \log \quad \Pr(N_{S'}(u)|f(u)), \tag{4}$$

where f is the mapping function from the nodes to the feature representations, and $N_{S'}(u)$ refers to the neighborhood or context nodes of node u generated through a sampling optimization strategy S'.

We aim to optimize the above objective function, which maximizes the log-probability of observing a network partitioning. The objective in (4) can be approximatively simplified to

$$\begin{aligned}\max_{f} \quad & \sum_{u\in\mathcal{W}} \left[-\log Z_u + \sum_{n_i\in N_{S(u)}} f(n_i)\cdot f(u)\right] \\ & + \left(\sum_{u\in\Delta\mathcal{W}_{\rm inc}} - \sum_{u\in\Delta\mathcal{W}_{\rm dis}}\right)\left[-\log Z_u + \sum_{n_i\in N_{S'(u)}} f(n_i)\cdot f(u)\right],\end{aligned} \tag{5}$$

where $Z_u = \sum_{u\in\mathcal{W}'} \exp(f(u)\cdot f(v))$ is expensive to compute for dynamic and large networks. So we approximate it by negative sampling. S is the sampling optimization strategy in the old network $\mathcal{W}$. We factorize log-likelihood function for skip-gram with negative sampling model based on the network partitioning. Here, we firstly retain the log-likelihood function and the inherited vectors from the old network. Secondly, we re-calculate the sampling optimization strategy S'. Then we employ the sub-graph compensation strategy to increase or decrease the holistic log-likelihood function, respectively. Note that the difference between (4) and (5) is that the sampling optimization strategies S and S' are different.

Our goal is to speed up the training of dynamic network representation learning. To train the node representations for the updated parts of a network, existing methods need to re-scan and re-train the whole proximities based on skip-gram with negative sampling and stochastic gradient methods. Given the above factorization analysis of the objective function, we find that for the old network $\mathcal{W}$, we can apply the initialization and inheritance of vertices and parameters trick in (2) and (3) to significantly save the training time. We just need to update the related vertex and parameter vectors following the new sampling optimization strategy S'. Finally, we release the disappearance of nodes and the corresponding parameters vector.

4 Theoretical analysis

Although the extension from the batch global training to the dynamic network training is simple and intuitive, it is not clear whether the incremental skip-gram with negative sampling technology based dynamic network embedding method can learn the represented vectors of the nodes as good as that learned by the batch global learning counterpart. To answer this question, in this section we examine the dynamic network representation learning from a theoretical point of view.

We will firstly show the difference between the objectives optimized by the approximative incremental skip-gram with negative sampling (ISGNS) and batch skip-gram with negative sampling (SGNS) models in Subsection 4.1. Secondly, we will prove that the objective difference is bounded by the scale of the network in Subsection 4.2. Then, we will investigate the probabilistic properties of the objective difference

to demonstrate the equivalent relationship between batch SGNS and ISGNS in Subsection 4.3. Finally, we will analyze the time and memory complexity of ISGNS in Subsection 4.4.

4.1 Objective difference

As discussed in Subsection 3.1, the network updates from $\mathcal{W}$ to $\mathcal{W}'$ and the size of vertex sequences growing from n to N. We denote the sizes of the local sequences that contain the newly added and vanished vertices as n_{add} and n_{van}. Following the studies [5, 6], the size of vertex sequences can be updated as $N = n + n_{\text{add}} - n_{\text{van}}$. The ISGNS optimizes the following objective function:

$$\begin{aligned}\mathcal{L}_{\text{ISGNS}}(\theta) = -\Bigg\{ & \frac{1}{n}\sum_{i=1}^{n}\sum_{|j|<c, j\neq 0}\psi^{+}_{w_i,w_{i+j}} + k\mathrm{E}_{v\sim q_n(v)}[\psi^{-}_{w_i,v}] \\ & + \left(\frac{1}{n_{\text{add}}} - \frac{1}{n_{\text{van}}}\right)\sum_{i=1}^{N}\sum_{|j|<c, j\neq 0}\psi^{+}_{w_i,w_{i+j}} + k\mathrm{E}_{v\sim q_N(v)}[\psi^{-}_{w_i,v}]\Bigg\},\end{aligned} \tag{6}$$

where $\theta = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{\mathcal{W}'}, \boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{\mathcal{W}'})$ collectively represents model parameters, including both target and context vertex embeddings. The function $q_n(v)$ represents the old noise distribution, and it is defined as

$$q_n(v) = \frac{f_n(v)^{\frac{3}{4}}}{\sum_{v'\in\mathcal{W}} f_n(v')^{\frac{3}{4}}},$$

where $f_n(v)$ represents the frequency of vertex v in the sequences of vertices. Note that the noise distribution in the first term of the objective is $q_n(v)$ rather than $q_N(v)$. Because we employ the parameter initialization strategy and it can be seen as a simple approximation of the gradient. In detail, $\psi^{+}_{w,v} = \log\sigma(t_w \cdot c_v)$, $\psi^{-}_{w,v} = \log\sigma(-t_w \cdot c_v)$, and $\sigma(x)$ is the sigmoid function. Given a target-context vertex pair (w_i, w_{i+j}) and k negative samples $(v_1, v_2, \ldots, v_k)$ sampled from the latest noise distribution $q_N(v)$, the gradient of $-\psi^{+}_{w_i,w_{i+j}} - k\mathrm{E}_{v\sim q_N(v)}[\psi^{-}_{w_i,v}]$ is computed at each step.

In contrast, the original objective function of re-training the network embedding model based on SGNS can be given as

$$\mathcal{L}_{\text{SGNS}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{|j|<c, j\neq 0}\psi^{+}_{w_i,w_{i+j}} + k\mathrm{E}_{v\sim q_N(v)}[\psi^{-}_{w_i,v}], \tag{7}$$

which can be interpreted as the re-training procedure with SGD. Because the expectation terms in the objectives can be rewritten as $\mathrm{E}_{v\sim q_N(v)}[\psi^{-}_{w_i,v}] = \sum_{v\in\mathcal{W}'} q_N(v)\psi^{-}_{w_i,v}$, the difference between the two objectives can be formalized as follows:

$$\begin{aligned}\Delta\mathcal{L}_{\text{DI}}(\theta) =& \mathcal{L}_{\text{SGNS}}(\theta) - \mathcal{L}_{\text{ISGNS}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{|j|<c, j\neq 0} k\sum_{v\in\mathcal{W}'}(q_N(v) - q_n(v))\psi^{-}_{w_i,v} \\ =& \frac{2ck}{n}\sum_{i=1}^{n}\sum_{w,v\in\mathcal{W}'}\delta_{w_i,w}(q_N(v) - q_n(v))\psi^{-}_{w,v},\end{aligned} \tag{8}$$

where δ is the delta function.

4.2 Boundness analysis of $\Delta\mathcal{L}_{\text{DI}}(\theta)$

To verify the correctness of our dynamic network embedding framework, we present the boundness analysis of the objective difference $\Delta\mathcal{L}_{\text{DI}}(\theta)$ in this subsection.

We first give a theorem as follows.

Theorem 1. The objective difference $\Delta\mathcal{L}_{\text{DI}}(\theta)$ can be directly bounded by the scale of the old network as follows:

$$\Delta\mathcal{L}_{\text{DI}}(\theta) < \frac{2ck}{n}\epsilon = \frac{2ck}{N - (\dot{n} - \ddot{n})}\epsilon. \tag{9}$$

Sketch of Proof. The delta function can be loosely considered as a function on the real line which is zero everywhere except at the origin, where it is infinite,

$$\delta_{w_i,w}(q_N(v)-q_n(v))=\begin{cases}+\infty, & q_N(v)\neq q_n(v),\\ 0, & q_N(v)=q_n(v),\end{cases} \tag{10}$$

and it is also constrained to satisfy the identity:

$$\sum_{w,v\in\mathcal{W}'}\delta_{w_i,w}(q_N(v)-q_n(v))\leqslant 1. \tag{11}$$

Because $\psi^-_{w,v}=\log\sigma(-t_w\cdot c_v)$ is bounded in practice, we assume $\psi^-_{w,v}<\epsilon$. It supports the very intuitive understanding that the less updated network nodes lead to a lower upper bound.

4.3 Convergence analysis of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$

It shows that the first order of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ has an analytical form.

Definition 1. Let $X_{i,w}$ be a random variable that represents $\delta_{w_i,w}$. It is assigned value 1 when the i-th node in the sampled data is $w\in\mathcal{W}'$. For any i and j, remind that $\mathrm{E}[X_{i,w}]=\mu_w$ and $\mathbb{V}[X_{i,w},X_{j',w}]=\rho_{w,v}$.

Definition 2. Let $Y_{j,v}$ be a random variable that represents $q_N(v)$.

Theorem 2. The first-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ is given as

$$\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]=\frac{2ck}{n}\left(\frac{1}{N}-\frac{1}{n}\right)\sum_{w,v\in\mathcal{W}'}\rho_{w,v}\psi^-_{w,v}, \tag{12}$$

where $\rho_{w,v}$ is the covariance of $X_{i,w}$ and $X_{j,v}$.

Sketch of Proof. Here, for any i and j such that $i<j$, we have

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}]&=\mathrm{E}\left[X_{i,w}\frac{1}{j}\sum_{j'=1}^{j}X_{j',v}\right]=\frac{1}{j}\sum_{j'=1}^{j}\mathrm{E}[X_{i,w}X_{j',w}]\\&=\frac{1}{j}\sum_{j'=1}^{j}(\mathrm{E}[X_{i,w}]\mathrm{E}[X_{j',v}]+\mathbb{V}[X_{i,w},X_{j',w}])=\mu_w\mu_v+\frac{1}{j}\rho_{w,v}.\end{aligned} \tag{13}$$

Therefore, $\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]$ can be written as

$$\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]=\frac{2ck}{n}\sum_{w,v\in\mathcal{W}'}\left(\mu_w\mu_v+\frac{1}{N}\rho_{w,v}-\mu_w\mu_v-\frac{1}{n}\rho_{w,v}\right)\psi^-_{w_r,v}=\frac{2ck}{n}\left(\frac{1}{N}-\frac{1}{n}\right)\sum_{w,v\in\mathcal{W}'}\rho_{w,v}\psi^-_{w,v}. \tag{14}$$

Theorem 3. The first-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ decreases in the order of $\mathcal{O}(\frac{1}{n^2})$:

$$\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]=\mathcal{O}\left(\frac{1}{n^2}\right), \tag{15}$$

and thus converges to zero in the limit of infinity:

$$\lim_{n\to\infty}\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]=0. \tag{16}$$

Proof. We assume that N and n are in the same order of magnitude and thus Theorem 2 gives the proof.

Theorem 4. The second-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ can be bounded as

$$\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)]<\sum_{w,v\in\mathcal{W}'}\left[\frac{24c^2k^2}{L^2T^2}+\mathcal{O}\left(\frac{1}{n}\right)\right](\psi^-_{w,v})^2, \tag{17}$$

and thus decreases in the order of $\mathcal{O}(1)$:

$$\lim_{n\to\infty} \mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)] = \mathcal{O}(1), \tag{18}$$

where L refers to the random walk steps in one round, and T regers to the random walk times for each vertex.

Proof. A similar result to first-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ can be proved for the second order moment of objective difference as well. The upper-bound of $\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)]$ is examined to prove the theorem. Let $\Psi_{i,N,n,w,v} = \delta_{w_i,w}(q_N(v) - q_n(v))\psi^-_{w,v}$. Making use of Jensen's inequality, we have

$$\begin{aligned}\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)] =& \mathrm{E}\left[\frac{4c^2k^2}{n^2}\left(\sum_{i=1}^{n}\sum_{w,v\in\mathcal{W}'}\Psi_{i,N,n,w,v}\right)^2\right] = \mathrm{E}\left[\frac{4c^2k^2}{n^2}|\mathcal{W}'|^4n^2\left(\sum_{w,v\in\mathcal{W}'}\sum_{i=1}^{n}\frac{1}{|\mathcal{W}'|^2n}\Psi_{i,N,n,w,v}\right)^2\right]\\ \leqslant& \mathrm{E}\left[\frac{4c^2k^2}{n^2}|\mathcal{W}'|^4n^2\sum_{w,v\in\mathcal{W}'}\sum_{i=1}^{n}\frac{1}{|\mathcal{W}'|^2n}\Psi^2_{i,N,n,w,v}\right] = \frac{4c^2k^2|\mathcal{W}'|^2}{n}\sum_{w,v\in\mathcal{W}'}\sum_{i=1}^{N}\mathrm{E}[\Psi^2_{i,N,n,w,v}].\end{aligned} \tag{19}$$

To prove Theorem 4, we begin by examining the upper- and lower-bounds of $\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}]$ in the following lemma, and then make use of the bounds to evaluate the order of the second order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$.

Lemma 1. For any j and k such that $j \leqslant k$, we have

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &\leqslant \frac{(jk-2j-k+2)\mu_w\mu_v^2 + 2j + k - 2}{jk},\\ \mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &\geqslant \frac{(jk-2j-k+2)\mu_w\mu_v^2}{jk}.\end{aligned} \tag{20}$$

See Appendix A for detailed proof. Furthermore, the term $\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)]$ is upper-bounded as

$$\mathrm{E}[\Psi^2_{i,N,n,w,v}] = \mathrm{E}[\delta_{w_i,w}(q_N(v) - q_n(v))^2(\psi^-_{w,v})^2] < \sum_{w,v\in\mathcal{W}}\left[\frac{3}{N} + \frac{3}{n} + \left(\frac{2}{N^2} + \frac{2}{n^2}\right)\mu_w\mu_v^2\right](\psi^-_{w,v})^2. \tag{21}$$

Because the sequence of vertices is generated by random walk technologies, the mathematical relationship between set of vertices $\mathcal{W}'$ and set of sequences N can be formalized as

$$N = \mathcal{W}' \cdot L \cdot T. \tag{22}$$

So, the upper-bounded of $\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)]$ can be written as

$$\begin{aligned}\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)] &< \sum_{w,v\in\mathcal{W}'}\frac{4c^2k^2|\mathcal{W}'|^2}{n}\left[\frac{3}{N} + \frac{3}{n} + \left(\frac{2}{N^2} + \frac{2}{n^2}\right)\mu_w\mu_v^2\right](\psi^-_{w,v})^2\\ &< \sum_{w,v\in\mathcal{W}'}\left[\frac{24c^2k^2}{L^2T^2} + \mathcal{O}\left(\frac{1}{n}\right)\right](\psi^-_{w,v})^2.\end{aligned} \tag{23}$$

Therefore, we have the second-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ decreases in the order of $\mathcal{O}(1)$:

$$\lim_{n\to\infty} \mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)] = \mathcal{O}(1), \tag{24}$$

and thus converges to constant influenced by $\sum_{w,v\in\mathcal{W}'}(\psi^-_{w,v})^2$ in the limit of infinity.

Table 1 Statistics of the dynamic network datasets

Name	$\|V\|$	$\|E\|$	Label	Time step
Wikipedia	1985098	1000924086	7	16
BlogCatalog	10312	333983	39	20
Flickr	80513	5899882	195	100
Facebook	1715256	22613981	–	24
ArXiv	18722	198110	195	16
DBLP	524061	20580238	100	730

4.4 Complexity analysis

The computational cost of each operation in dynamic embedding model (6) is the same as that of model (7). Thus, the total computational cost is $\mathcal{O}((\Delta\mathcal{W}_{\rm inc} + \Delta\mathcal{W}_{\rm dis})k)$, where k is the number of the negative samples. In practice, we can use the size of the affected sub-graphs to evaluate the main computation complexity of network embedding learning. According to the random walk and sliding window, the size of the affected nodes is $n_{\rm add} + n_{\rm van}$. Therefore, the computation complexity of our dynamic network embedding framework is bounded by $\mathcal{O}((n_{\rm add} + n_{\rm van})k)$. Similarly, the memory cost is bounded by $\mathcal{O}(n + n_{\rm add} + n_{\rm van})$. Note that the memory cost of non-negative matrix factorization based temporal latent space network analysis approach [18] linearly grows over time.

5 Experiments

We apply ISGNS to various large-scale real-world dynamic networks including a language network, three social networks and two citation networks (Subsection 5.1). We empirically evaluate the time efficiency (Subsection 5.2), the theoretical reliability (Subsection 5.3) and the quality of network representation (Subsection 5.4) of the proposed ISGNS.

5.1 Datasets

The datasets used in this paper are Wikipedia, BlogCatalog, Flickr, Facebook, ArXiv, and DBLP networks. A summarization of the statistics of the six datasets is shown in Table 1.

• **Wikipedia.** This is a word co-occurrence network in the webpages of Wikipedia. We set an edge between two words if they co-occurr within the 5-words sliding window in the English Wikipedia pages. The labels represent the part-of-speech (POS) tags inferred using the Stanford POS-Tagger. In total the network has 1985098 nodes, 1000924086 edges, and 7 different labels from the year 2001 to 2016.

• **BlogCatalog [40].** This is a social blog directory which manages the bloggers and their blogs. It contains 10312 bloggers as nodes and 333983 relationships as edges in 20 days. The labels represent the topic interests provided by the bloggers. The network has 39 labels and a blogger may have multiple labels.

• **Flickr [41].** This dataset is constructed with the images and the links among them collected from Flickr in 100 days. The links between images represent they share common meta-data. In this data, edges are formed between two images that are taken from the same location. This data has 80513 nodes and 5899882 edges.

• **Facebook [42].** This is a social network dataset. Nodes represent users, and edges are friendship relation between them. The network has 4039 nodes and 20580238 edges.

• **ArXiv [43].** This is a collaboration network generated from the e-print arXiv and covers scientific collaborations between authors. Nodes represent scholars, and an edge represents two scholars have collaboration in a paper. The network has 18722 nodes and 198110 edges.

• **DBLP [44].** This is also a co-author network. Nodes represent authors, and edges represent the co-author relation among them. The network has 524061 nodes and 970742 edges.

Table 2 The evolving procedure of the dynamic DBLP network

Window sliding rate (%)	0.10	0.50	1.0	1.50	2.00
Edge (+)	2018	4547	7266	9937	13498
Edge (−)	1527	3381	6464	9884	12860
Window sliding rate (%)	2.50	3.00	3.50	4.00	–
Edge (+)	17054	22842	26340	29217	–
Edge (−)	16515	19353	22852	25860	–

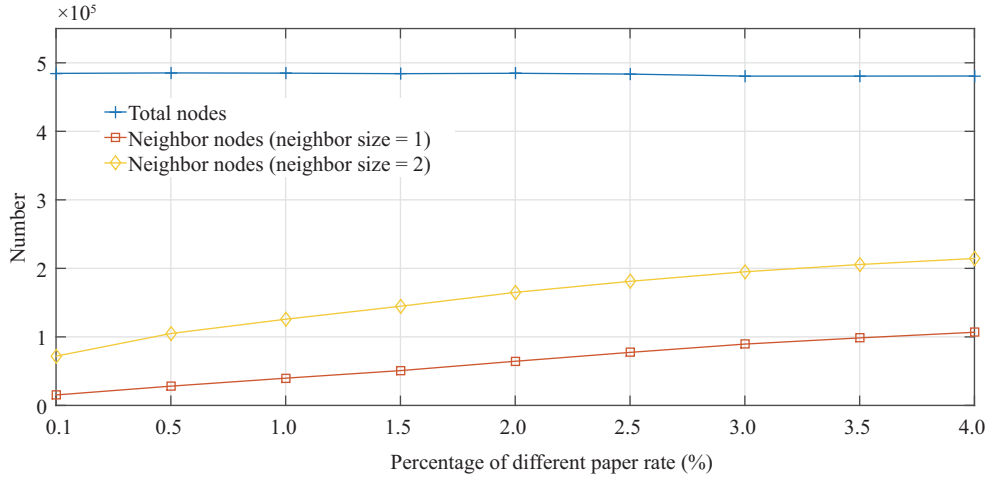

Figure 2 (Color online) Number of the neighbor nodes with different settings.

5.2 Training time and speedup

We use the DBLP dataset to evaluate the training time and speedup performance of ISGNS. In order to simulate the addition and deletion of nodes and edges, we build the dynamic co-author network. We use 200000 papers published earlier as the initial network, which contains 484095 nodes and 970742 edges. Then we choose different sliding windows to move the the initial network along papers' publishing time. More specific, we choose 0.10%, 0.50%, 1.0%, 1.50%, 2.00%, 2.50%, 3.00%, 3.50% and 4.00% of time-series of window sliding rate, respectively, to add and delete related vertexes and edges to update the initial network as dynamic network. Note that we consider both the addition and deletion of vertices and edges as shown in Table 2. We apply ISGNS to DeepWalk and node2vec models on the initial DBLP network with 200000 papers, and run our algorithms (2)–(5) to update the noise distributions, the node vectors and the corresponding parameter vectors. For comparisons, we also re-train the network embedding and run SGNS for the updated new network. In the experiments, we run with 10 CPU threads, and the dimension of the generate network embedding is set to 128.

We first check the updating network structure including the added and vanished vertices and edges by comparing the generated sequences of vertices before and after the random walk. We retain the vectors related to the old vertices and update the sequences of vertices for more convenient gradient iterations in (5). For the sequence n, if there are sliding windows of vertices containing the newly vanished nodes, we update the vectors of the vertices in the sliding windows with stochastic gradient descent method. For the sequence N, if there are sliding windows of vertices containing the newly added nodes, we update the vectors of the vertices in the sliding windows with stochastic gradient ascent method. We count the rates of the nodes in the affected neighbor sub-graphs, as shown in Figure 2. The blue line refers to the total number of nodes in the dynamic scenario. Note that the total number of nodes is changing. The orange and yellow lines are the number of the affected nodes when the sliding window arises from 3 to 5.

We apply ISGNS to DeepWalk and node2vec, and check the training time and the achieved speedup. The results are shown in Figures 3(a) and (b). It is shown that the time consumption of ISGNS linearly increases with the increase of the size of the updated nodes and edges in the dynamic network. Although

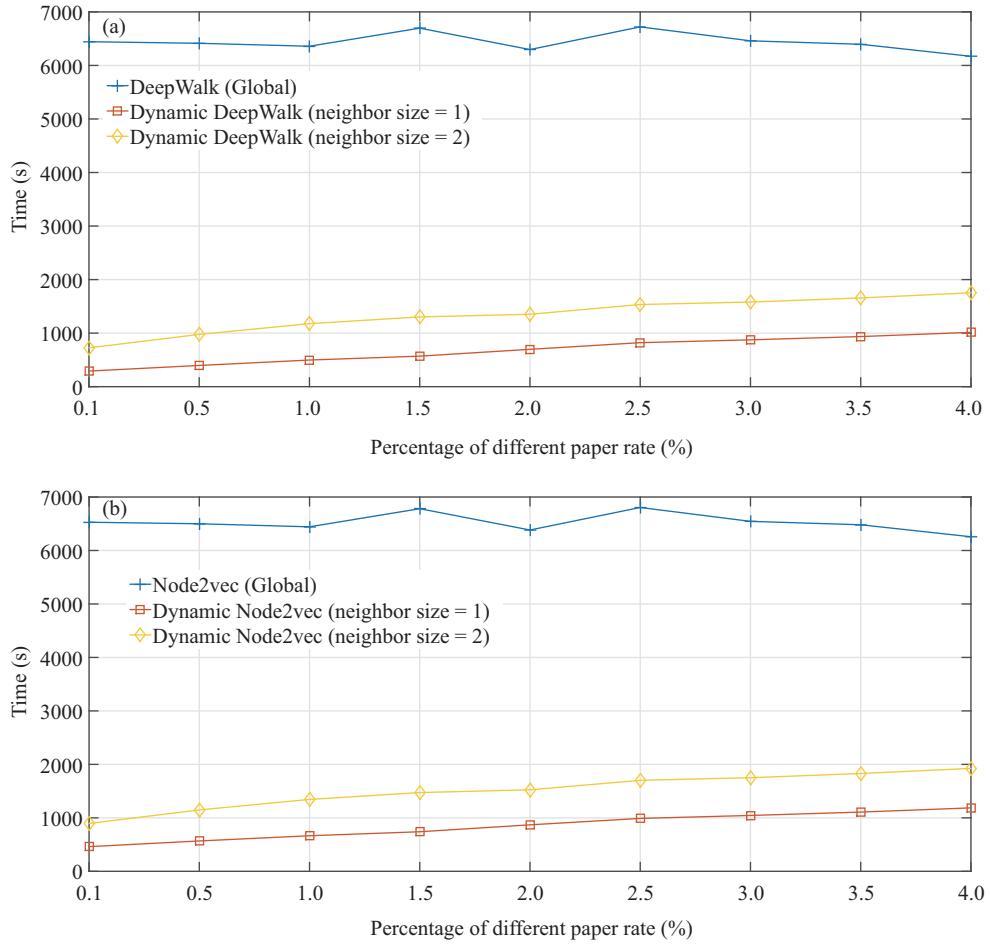

Figure 3 (Color online) Training time of different methods under different settings. (a) DeepWalk; (b) node2vec.

the sliding rate from 0.10% to 4.00% is relatively small compared to the original network, the proportion of affected neighbor nodes is relatively large rising from about 14.83% to 44.60% due to the dense connectivity among the nodes. We employ stochastic gradient method to update the vectors among sliding windows of the sequence of vertices following (5). The time consumptions of the batch global network embedding models depend on the overall scale of the networks. It costs about 6500 and 6800 s when the size of the nodes is 484000, while our models take much less time than batch global re-training. One can also see that the time consumptions in DeepWalk and node2vec both linearly increase with the change rate increasing for dynamic networks.

The speedup results are shown in Figures 4(a) and (b). One can see that for the smaller change in the dynamic network, the speedup is more significant. DeepWalk with ISGNS achieves up to 22 times speedup, while node2vec with ISGNS has up to 14 times speedup.

5.3 Validation of theoretical analysis

Now we give an empirical experiment to validate our theoretical analysis in Section 4. Because it is difficult to assess the node vector value generated by stochastic gradient optimization models between (6) and (7) directly, we focus on verifying the boundness analysis and give the limit of first-order and second-order moments of objective difference. As we proved the limits of infinity in (16) and (24), the first-order and second-order moments of objective difference are affected by the old sequence of vertices n. We measure the first and second order of moments on Facebook dataset with various network sizes $\mathbb{W}$ varying over $\{10^2, 2\times10^2, 2^2\times10^2, 2^3\times10^2, \ldots, 2^{14}\times10^2\}$. We also choose $1\%, 5\%, 10\%, 15\%$ of the network nodes change rates for the above dynamic Facebook network. Because the second-order moment

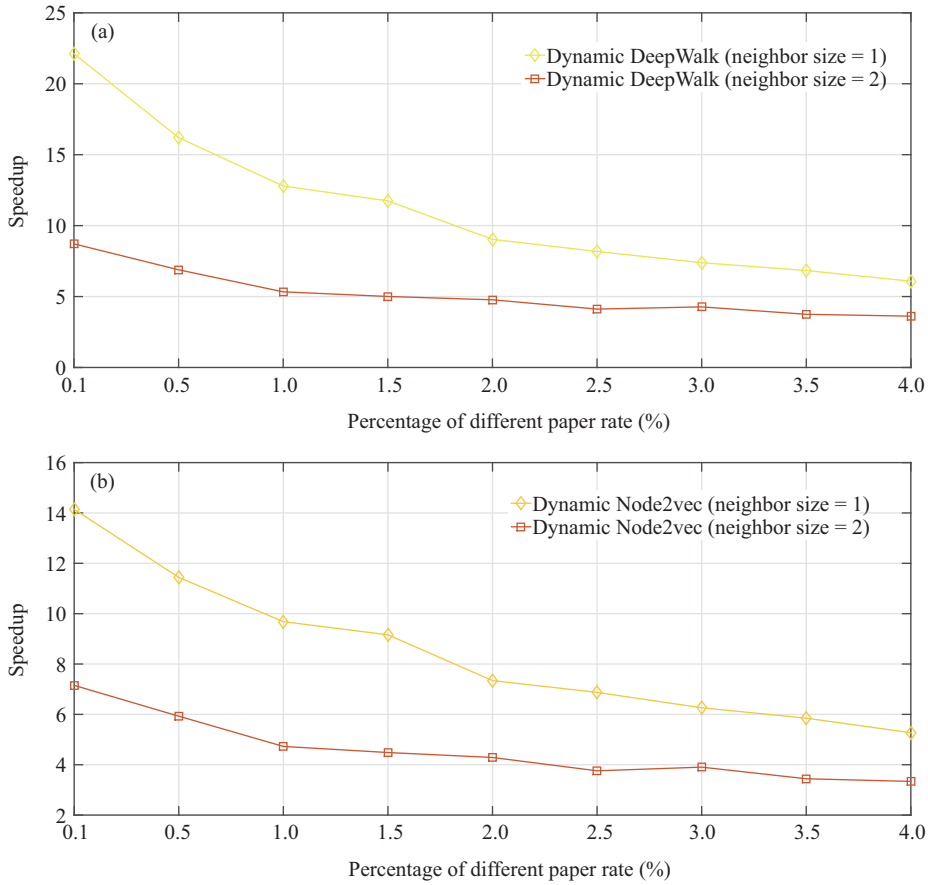

Figure 4 (Color online) Speedup performances in dynamic DeepWalk (a) and dynamic node2vec (b) with different settings.

can be affected by the times and length of random walk for each node. For all the experiments, we set the length and times of random walk as 80 and 100, respectively.

Figure 5(a) shows the first-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ computed on the different sizes of the training data and different network change rates. Because Eq. (16) suggests that the first-order moment decreases in the order of $\mathcal{O}(\frac{1}{n^2})$. The expectation of $\mathrm{E}[\Delta\mathcal{L}_{\mathrm{DI}}(\theta)]$ converges to zero when the network size tends to be infinitely great. In Figure 5(a), the four lines are close to each other, which demonstrates the smaller the change rate of the network, the smaller the expectation of objective difference. Note that the x-axis is log scale, and the first-order moments of magnitudes decrease from 10^{-8} to 10^{-16}.

Figure 5(b) shows the second-order moment of $\Delta\mathcal{L}_{\mathrm{DI}}(\theta)$ computed on different sizes of datasets and different network change rates. Different from the first-order moments, We can see that the second order moments slowly increase with the same exponentially growing network scales. However, the limits of infinity of $\mathrm{E}[\Delta\mathcal{L}^2_{\mathrm{DI}}(\theta)]$ is $\mathcal{O}(1)$. Similar to the trend in first-order moments, a smaller change rate of network leads to a smaller second order moments of the objective difference. Although the second order moments do not converge to zero, the real values are small and change in a relatively small range from 1.52×10^{-2} to 1.87×10^{-2}.

5.4 Quality of network embeddings

This experiment aims to investigate the quality of the network embeddings learned by our dynamic network embeddings through comparison with the batch global re-training based counterparts and other dynamic network embedding approach [31]. We employ the multi-label classification and link predication tasks [5, 6, 31] to evaluate the quality of the learned network embeddings.

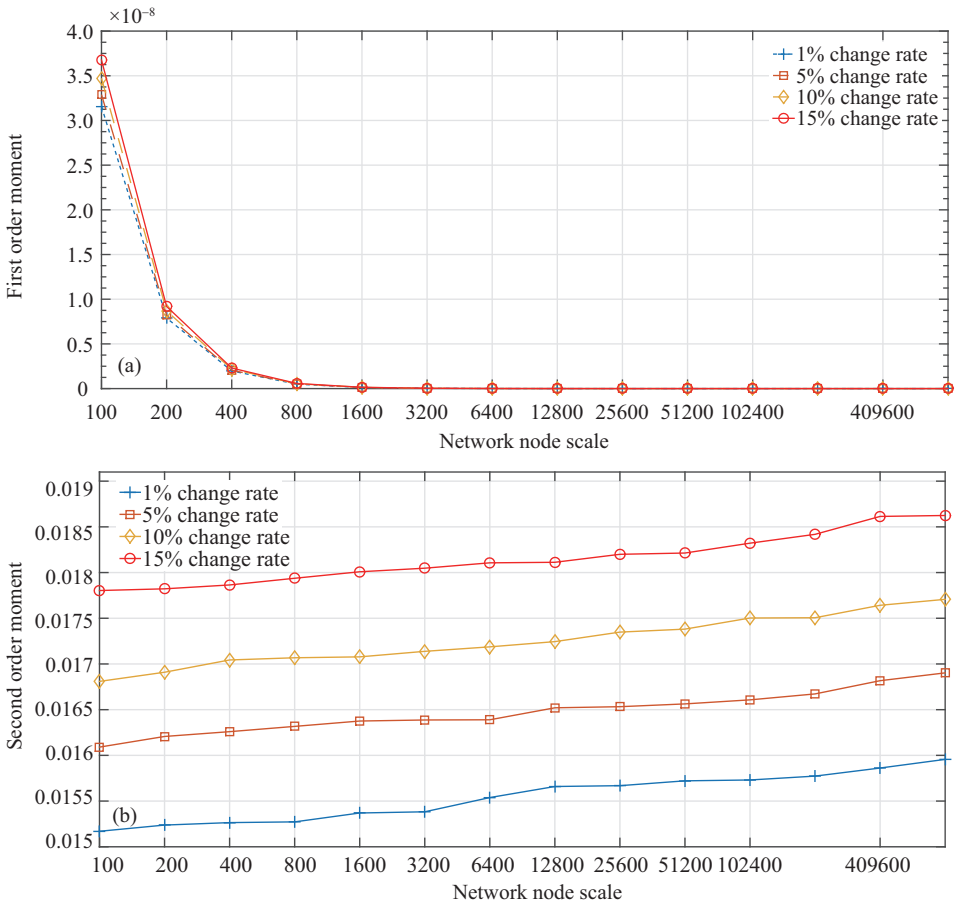

Figure 5 (Color online) First-order moment (a) and second-order moment (b) with different change rates.

5.4.1 *Multi-label classification*

We firstly evaluate the quality of the learned dynamic network embeddings through the multi-label classification task on the Wikipedia (Wp), BlogCatalog (BC) and Flickr (Fl) datasets. In order to construct a dynamic network, we take full advantage of the time attributes of nodes and edges on the three networks. We adopt two kinds of training modes, including dynamic training (Dy) over time and batch global training (Gl), to extract the dynamic network features. For the dynamic training, we adopt the time sequence batch network structures. BlogCatalog, Flickr and Wikipedia are the time-series accumulative social relationship networks, as shown in Table 1. We use an online learning method to train the three dynamic network datasets over time. However, in addition to the dynamic time factor, we also inherit the original random walk strategies and the dimension of the vector configurations. We inherit the parameter pairs $\{(0.25, 0.25), (0.25, 0.25), (4, 0.5)\}$ for BlogCatalog, Flickr and Wikipedia from node2vec [5].

We train the node feature representations, including dynamic network embeddings considering time factor in (6) and one-round global network embeddings without consideration of time factor in (7), as the input to a one-vs-rest logistic regression classifier with L2 regularization implemented by LibLinear [45]. Specifically, we randomly sample a portion of the labeled nodes and use them as training data. The rest of the nodes are used as the test data. We also randomly and equally split the train and test data over 10 parts, perform a 10-fold cross validation, and report the average Micro-F1 (Mi-F1) and Macro-F1 (Ma-F1). From the results given in Tables 3 and 4, one can see that the dynamic network embedding results are comparable to and sometimes better than the global training results. Both the Micro-F1 and Macro-F1 are reported when the labeled node percentage increases from 10% to 90%. Overall, the dynamic network embedding performs better than one round of global training. This may be because

Table 3 Multi-label classification results by DeepWalk (%)[a)]

Item	DS	Metric	10%	20%	30%	40%	50%	60%	70%	80%	90%
Dy	BC	Mi-F1	**36.02**	**36.21**	**39.61**	40.28	**41.11**	41.29	**41.51**	41.47	**42.05**
Gl	BC	Mi-F1	36.00	36.20	39.60	**40.30**	41.00	**41.30**	41.50	**41.50**	42.00
Dy	BC	Ma-F1	**21.31**	**23.81**	**25.31**	26.29	27.33	**27.60**	**27.90**	28.18	**28.92**
Gl	BC	Ma-F1	21.30	23.80	25.30	**26.30**	**27.30**	**27.60**	**27.90**	**28.20**	28.90
Dy	Fl	Mi-F1	32.30	34.59	36.11	**36.88**	**37.21**	37.77	38.05	**38.43**	**38.88**
Gl	Fl	Mi-F1	**32.44**	**34.61**	35.94	36.79	**37.21**	**37.79**	**38.13**	38.41	38.76
Dy	Fl	Ma-F1	13,91	17.14	**19.74**	**21.16**	**22.07**	22.75	23.55	**24.11**	**24.78**
Gl	Fl	Ma-F1	**14.06**	**17.17**	19.69	21.11	22.05	**22.78**	**23.62**	24.10	24.72
Dy	Wp	Mi-F1	**78.87**	79.91	**80.42**	**80.72**	**80.93**	81.15	**81.27**	81.33	81.42
Gl	Wp	Mi-F1	78.86	**79.93**	80.41	80.69	**80.93**	**81.16**	81.25	**81.35**	**81.43**
Dy	Wp	Ma-F1	**78.72**	79.74	**80.33**	**80.56**	**80.81**	80.93	**81.11**	81.21	81.21
Gl	Wp	Ma-F1	78.71	**79.76**	80.32	80.50	**80.81**	**80.94**	81.10	**81.23**	**81.31**

a) We show the best results with boldface.

Table 4 Multi-label classification results by node2vec (%)[a)]

Item	DS	Metric	10%	20%	30%	40%	50%	60%	70%	80%	90%
Dy	BC	Mi-F1	**36.71**	**37.19**	**39.99**	**40.30**	**41.29**	**42.06**	41.44	42.57	**42.87**
Gl	BC	Mi-F1	36.70	37.17	39.98	**40.30**	41.27	**42.06**	**41.46**	**42.58**	42.86
Dy	BC	Ma-F1	**21.40**	**23.97**	**25.37**	**26.39**	**27.51**	27.69	27.96	**28.21**	**28.97**
Gl	BC	Ma-F1	**21.40**	23.96	**25.37**	26.38	27.50	**27.70**	**27.97**	**28.21**	28.96
Dy	Fl	Mi-F1	**33.59**	35.15	**37.11**	**37.93**	38.26	**38.91**	**38.99**	39.14	**39.44**
Gl	Fl	Mi-F1	33.57	**35.16**	36.96	37.84	**38.27**	38.90	38.95	**39.17**	39.42
Dy	Fl	Ma-F1	14.11	**18.21**	20.41	**22.25**	23.27	23.26	**24.72**	**25.81**	25.91
Gl	Fl	Ma-F1	**14.12**	18.18	**20.43**	22.24	**23.30**	**23.28**	24.68	25.79	**25.94**
Dy	Wp	Mi-F1	**79.05**	**80.05**	**80.75**	**80.89**	**81.39**	**81.33**	**81.55**	81.62	**81.69**
Gl	Wp	Mi-F1	79.04	**80.05**	80.74	80.87	81.38	81.31	**81.55**	**81.63**	**81.69**
Dy	Wp	Ma-F1	**78.97**	79.82	**80.60**	**80.71**	**81.28**	**80.26**	**81.11**	81.47	**81.57**
Gl	Wp	Ma-F1	78.96	**79.83**	80.59	80.70	81.27	80.25	81.10	**81.48**	81.56

a) We show the best results with boldface.

Table 5 Multi-label classification results comparison of different embedding methods

Dataset	Algorithms	Micro-F1	Macro-F1
BC	Dynamic DeepWalk	42.05%	28.92%
BC	Dynamic node2vec	42.87%	28.97%
BC	DANE [31]	**43.27%**	**29.12%**
BC	Dynamic SBM [33]	39.41%	22.63%
BC	DNE [35]	40.75%	26.13%
Fl	Dynamic DeepWalk	38.88%	24.78%
Fl	Dynamic node2vec	**39.44%**	**25.91%**
Fl	DANE [31]	32.81%	20.75%
Fl	Dynamic SBM [33]	36.52%	23.87%
Fl	DNE [35]	37.87%	24.28%

some important dynamic structure patterns are sufficiently trained and captured. Moreover, the node2vec based embedding models performs better than DeepWalk in experiments.

We further compare with the state-of-the-art dynamic network embedding methods including DANE [31], Dynamic SBM [33] and DNE [35] on the multi-label classification task. Because the vertices of our networks do not have attribute information, we adopt the DANE with only network information for fairness. As shown in Table 5, the neural network based network embeddings is better than spectral embedding based DANE and statistical model based Dynamic SBM models. The experiments demonstrate the generality of ISGNS model. The experimental results also show that the convergent ISGNS

Table 6 Area under curve (AUC) scores for link prediction

Dataset	Algorithm	Average	Hadamard	Weighted-L1	Weighted-L2
Fb	Dynamic DeepWalk	**0.7268**	0.9548	0.9474	**0.9536**
Fb	Global DeepWalk	0.7261	0.9544	0.9461	0.9535
Fb	Dynamic node2vec	0.7266	**0.9555**	**0.9504**	0.9526
Fb	Global node2vec	0.7264	0.9554	0.9503	0.9524
Ax	Dynamic DeepWalk	0.7058	0.9275	0.8186	0.8278
Ax	Global Deepwalk	0.7056	0.9274	0.8183	0.8276
Ax	Dynamic node2vec	**0.7204**	**0.9305**	**0.8371**	**0.8474**
Ax	Global node2vec	0.7203	0.9305	0.8371	0.8474

has better learning ability for dynamic network embedding than the heuristic models.

5.4.2 *Link prediction*

In the link prediction task, we are given a network with a certain fraction of missing edges, and we need to predict the missing edges. To facilitate the comparison between the dynamic embedding method and the baselines, we use the same datasets and experiment setting as in [5]. We generate the labeled dataset of edges as follows. We randomly remove 50% of edges from the network as the positive samples. We randomly sample an equal number of node pairs from the network which actually have no edges connecting them as the negative examples. We conduct the experiment on Facebook (Fb) and ArXiv (Ax) datasets, and use the area under curve (AUC) scores for link prediction with four different binary operators, including Average, Hadamard, Weighted-L1, and Weight-L2, for learning edge features [5].

From the results in Table 6, one can see that similar to the multi-label classification task, the performances of dynamic network embedding in link prediction are comparable with and sometimes better than the global training results on the two datasets. The four binary operators which generate edge features are reported from dynamic embeddings to global embeddings. On the whole, the dynamic embeddings results are better than one round of global training for networks of sequential growth. This again demonstrates the generality of our dynamic skip-gram with negative sampling framework.

6 Conclusion

This paper proposed a dynamic network embedding framework based on the incremental skip-gram with negative sampling from both practical and theoretical perspectives. Theoretical analysis showed that the objective difference can be bounded by a function of the number of changed nodes and links, and the first-order moment of objective difference can be convergent in order of $\mathbb{O}(\frac{1}{n^2})$, and the second-order moment of objective difference can be stabilized in order of $\mathbb{O}(1)$. The results of the systematic evaluations on multi-label classification task and link prediction tasks over multi real-world dynamic network datasets show that our dynamic network embedding framework is significantly faster than global training, and achieve comparable network embedding performance. The success of this study proves the scalability and robustness of the incremental skip-gram with negative sampling algorithm. A potential future work is to extend our approach to other advanced network representation learning models [7,11,17,21,22,34,46,47].

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2016YFB1000103) and National Natural Science Foundation of China (Grant Nos. 61872022, 61772151, 61421003, SKLSDE-2018ZX-16).

References

1 Hamilton W L, Ying R, Leskovec J. Representation learning on graphs: methods and applications. In: Proceedings of IEEE Data Engineering Bulletin, 2017

2 Cavallari S, Zheng V W, Cai H Y, et al. Learning community embedding with community detection and node embedding on graphs. In: Proceedings of ACM International Conference on Information and Knowledge Management, 2017. 377–386

3 Shi C, Hu B B, Zhao W X, et al. Heterogeneous information network embedding for recommendation. IEEE Trans Knowl Data Eng, 2019, 31: 357–370
4 Hu R J, Aggarwal C C, Ma S, et al. An embedding approach to anomaly detection. In: Proceedings of 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, 2016. 385–396
5 Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2016. 855–864
6 Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2014. 701–710
7 Tang J, Qu M, Wang M Z, et al. Line: large-scale information network embedding. In: Proceedings of International World Wide Web Conference, 2015. 1067–1077
8 Tang J, Qu M, Mei Q Z. Pte: predictive text embedding through large-scale heterogeneous text networks. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2015. 1165–1174
9 He Y, Li J X, Song Y Q, et al. Time-evolving text classification with deep neural networks. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 2241–2247
10 Ren X, He W Q, Qu M, et al. Label noise reduction in entity typing by heterogeneous partial-label embedding. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2016. 1825–1834
11 Wang D X, Cui P, Zhu W W. Structural deep network embedding. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2016. 1225–1234
12 Cui P, Wang X, Pei J, et al. A survey on network embedding. IEEE Trans Knowl Data Eng, 2019, 31: 833–852
13 Li C Z, Wang S Z, Yang D J, et al. PPNE: property preserving network embedding. In: Proceedings of International Conference on Database Systems for Advanced Applications. Berlin: Springer, 2017. 163–179
14 Yang D J, Wang S Z, Li C Z, et al. From properties to links: deep network embedding on incomplete graphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. New York: ACM, 2017. 367–376
15 Zhang H M, Qiu L W, Yi L L, et al. Scalable multiplex network embedding. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 3082–3088
16 Trivedi R, Dai H J, Wang Y C, et al. Know-evolve: deep temporal reasoning for dynamic knowledge graphs. In: Proceedings of International Conference on Machine Learning, 2017. 3462–3471
17 Zuo Y, Liu G N, Lin H, et al. Embedding temporal network via neighborhood formation. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2018. 2857–2866
18 Zhu L H, Guo D, Yin J M, et al. Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Trans Knowl Data Eng, 2016, 28: 2765–2777
19 Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2017
20 Chen J F, Zhang Q, Huang X J. Incorporate group information to enhance network embedding. In: Proceedings of ACM International Conference on Information and Knowledge Management, 2016. 1901–1904
21 Cao S S, Lu W, Xu Q K. Grarep: learning graph representations with global structural information. In: Proceedings of ACM International Conference on Information and Knowledge Management, 2015. 891–900
22 Yang C, Sun M S, Liu Z Y, et al. Fast network embedding enhancement via high order proximity approximation. In: Proceedings of International Joint Conference on Artificial Intelligence, 2017
23 Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2013. 1–9
24 Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. 2013. arXiv: 1301.3781
25 Morin F, Bengio Y. Hierarchical probabilistic neural network language model. In: Proceedings of International Conference on Artificial Intelligence and Statistics, 2005. 5: 246–252
26 Gutmann M U, Hyvärinen A. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. J Mach Learn Res, 2012, 13: 307–361
27 Levy O, Goldberg Y. Neural word embedding as implicit matrix factorization. In: Proceedings of Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014
28 Mnih A, Teh Y W. A fast and simple algorithm for training neural probabilistic language models. In: Proceedings of the 29th International Coference on Machine Learning, 2012. 419–426
29 Ribeiro L F R, Saverese P H P, Figueiredo D R. struc2vec: learning node representations from structural identity. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2017. 385–394
30 Donnat C, Zitnik M, Hallac D, et al. Learning structural node embeddings via diffusion wavelets. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2018. 1320–1329
31 Li J D, Dani H, Hu X, et al. Attributed network embedding for learning in a dynamic environment. In: Proceedings of ACM International Conference on Information and Knowledge Management, 2017. 387–396
32 Jian L, Li J D, Liu H. Toward online node classification on streaming networks. In: Proceedings of International Conference on Data Mining and Knowledge Discovery, 2018. 231–257
33 Xu K S, Hero A O. Dynamic stochastic blockmodels: statistical models for time-evolving networks. In: Proceedings of International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction and Behavior Representation in Modeling and Simulation, 2013. 201–210
34 Zhou L, Yang Y, Ren X, et al. Dynamic network embedding by modelling triadic closure process. In: Proceedings of

AAAI Conference on Artificial Intelligence, 2018
35 Du L, Wang Y, Song G J, et al. Dynamic network embedding: an extended approach for skip-gram based network embedding. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2086–2092
36 Peng H, Li J X, Song Y Q, et al. Incrementally learning the hierarchical softmax function for neural language models. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017
37 Kaji N, Kobayashi H. Incremental skip-gram model with negative sampling. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, 2017
38 Rudolph M, Blei D. Dynamic embeddings for language evolution. In: Proceedings of International World Wide Web Conferences Steering Committee, 2018. 1003–1011
39 Peng H, Bao M J, Li J X, et al. Incremental term representation learning for social network analysis. Future Generation Comput Syst, 2018, 86: 1503–1512
40 Barbier G, Liu H. Data mining in social media. In: Social network data analytics. Boston: Springer, 2011. 327–352
41 Tang L, Liu H. Relational learning via latent social dimensions. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2009. 817–826
42 Leskovec J, Mcauley J J. Learning to discover social circles in ego networks. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2012
43 Leskovec J, Kleinberg J, Faloutsos C. Graph evolution: densification and shrinking diameters. ACM Trans Knowl Discov Data, 2007, 1: 2
44 Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. In: Proceedings of ACM SIGKDD Workshop on Mining Data Semantics, 2012. 181–213
45 Fan R E, Chang K W, Hsieh C J, et al. Liblinear: a library for large linear classification. J Mach Learn Res, 2008, 9: 1871–1874
46 Dong Y X, Chawla N V, Swami A. metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2017. 135–144
47 Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: a survey. Knowledge-Based Syst, 2018, 151: 78–94

Appendix A

To prove the Lemma 1, we begin by examining the upper- and lower-bounds of $\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}]$.

Lemma 1. For any j and k such that $j \leqslant k$, we have

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &\leqslant \frac{(jk-2j-k+2)\mu_w\mu_v^2+2j+k-2}{jk},\\ \mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &\geqslant \frac{(jk-2j-k+2)\mu_w\mu_v^2}{jk}.\end{aligned} \tag{A1}$$

Proof. We have

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &= \mathrm{E}\left[X_{i,w}\left(\frac{1}{j}\sum_{l=1}^{j}X_{l,v}\right)\left(\frac{1}{k}\sum_{m=1}^{k}X_{m,v}\right)\right]\\ &= \sum_{l=1}^{j}\sum_{m=1}^{k}\frac{\mathrm{E}[X_{i,w}X_{l,v}X_{m,v}]}{jk}.\end{aligned} \tag{A2}$$

To prove the lemma, we rewrite the expression by splitting the set of (l,m) into two subsets. Let $S_i^{(j,k)}(j \leqslant k)$ be a set of (l,m) such that $X_{i,w}$, $X_{l,v}$, and $X_{m,v}$ are independent from each other (i.e., i,l and m are all different), and let $\bar{S}_i^{(j,k)}$ be its complementary set,

$$\begin{aligned}S_i^{(j,k)} &= \{(l,m) \in \{1,2,3,\ldots,j\}\times\{1,2,3,\ldots,k\} | i \neq l \wedge l \neq m \wedge m \neq i\},\\ \bar{S}_i^{(j,k)} &= \{1,2,3,\ldots,j\}\times\{1,2,3,\ldots,k\}/S_i^{(j,k)}.\end{aligned} \tag{A3}$$

Then $\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}]$ is upper-bounded as

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &= \sum_{(l,m)\in S_i^{(j,k)}}\frac{\mathrm{E}[X_{i,w}]\mathrm{E}[X_{l,v}]\mathrm{E}[X_{m,v}]}{jk} + \sum_{(l,m)\in \bar{S}_i^{(j,k)}}\frac{\mathrm{E}[X_{i,w}X_{l,v}X_{m,v}]}{jk}\\ &\leqslant \sum_{(l,m)\in S_i^{(j,k)}}\frac{\mu_w\mu_v^2}{jk} + \sum_{(l,m)\in \bar{S}_i^{(j,k)}}\frac{1}{jk}\\ &= \frac{|S_i^{(j,k)}|\mu_w\mu_v^2 + |\bar{S}_i^{(j,k)}|}{jk},\end{aligned} \tag{A4}$$

where the inequality holds because $X_{i,w}$, $Y_{l,v}$, and $Y_{m,v}$ are binary random variables and thus $\mathrm{E}[X_{i,w}Y_{l,v}Y_{m,v}] \leqslant 1$. Here, we have $\bar{S}_i^{(j,k)} = 2j+k-2$, because $\bar{S}_i^{(j,k)}$ includes j elements such that $l=m$ and also includes $k-1$ and $j-1$ elements

such that $i = l \neq m$ and $i = m \neq l$, respectively. And we consequently have $|S_i^{(j,k)}| = jk - |\bar{S}_i^{(j,k)}| = jk - 2j - k + 2$. Therefore, the upper-bound can be rewritten as

$$\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] \leqslant \frac{(jk-2j-k+2)\mu_w\mu_v^2 + 2j + k - 2}{jk}. \tag{A5}$$

Similarly, by making use of $0 \leqslant \mathrm{E}[X_{i,w}Y_{l,v}Y_{m,v}]$, the lower-bound can be derived:

$$\begin{aligned}\mathrm{E}[X_{i,w}Y_{j,v}Y_{k,v}] &= \sum_{(l,m)\in S_i^{(j,k)}} \frac{\mathrm{E}[X_{i,w}]\mathrm{E}[X_{l,v}]\mathrm{E}[X_{m,v}]}{jk} + \sum_{(l,m)\in \bar{S}_i^{(j,k)}} \frac{\mathrm{E}[X_{i,w}X_{l,v}X_{m,v}]}{jk} \\ &\geqslant \sum_{(l,m)\in S_i^{(j,k)}} \frac{\mu_w\mu_v^2}{jk} + \sum_{(l,m)\in \bar{S}_i^{(j,k)}} \frac{0}{jk} \\ &= \frac{|S_i^{(j,k)}|\mu_w\mu_v^2}{jk} = \frac{(jk-2j-k+2)\mu_w\mu_v^2}{jk}.\end{aligned} \tag{A6}$$