SNEGAN: Signed Network Embedding by Using Generative Adversarial Nets

Lijia Ma, Yuchun Ma, Qiuzhen Lin, *Member, IEEE*, Junkai Ji, Carlos A. Coello Coello, *Fellow, IEEE,* and Maoguo Gong, *Senior Member, IEEE*

***Abstract*—Network embedding (NE) aims to learn low-dimensional node representations of networks while preserving essential node structures and properties. Existing NE methods mainly preserve simple link structures in unsigned networks, neglecting conflicting relationships that widely exist in social media and Internet of things. In this paper, we propose a novel generative adversarial nets learning framework (called SNEGAN) for signed network embedding, which tries to preserve link structures signed by positive or negative labels. SNEGAN combines a generator with a signed random walker technique and a graph softmax function to generate fake links that are used to deceive discriminator. Moreover, it combines a discriminator with a tanh function to discriminate truths and signs of links sampled from the generator and real network structures. The generator and discriminator play a two-player minimax game, and they will eventually generate low-dimensional node representations of signed networks. Extensive experiments on both LFR benchmark and real-world signed networks show the superiority of SNEGAN over the state-of-the-art NE methods in tackling both link (sign) prediction and reconstruction tasks.**

***Index Terms*—Network embedding, Generative Adversarial Nets, Signed networks, Link prediction**

I. Introduction

With the advance of graph signal processing, networks have become the most popular models to represent various complex systems in the fields of society, biology, ecology and economic [1]. In networks, nodes denote entities while links represent communications between entities. However, the rapid developments of Internet make it more difficult for networks to represent features of real-world biological and social systems due to exponentially increased data. Network embedding (NE) is devoted to revealing these features by learning low-dimensional node representations of networks [2]. Studies on NE have received much attention due to wide applications in graph tasks such as link prediction, network reconstruction, community detection, robustness optimization, node ranking, privacy protection and item recommendation [3]–[7].

Previous NE methods mainly work on unsigned networks that have simple links. These methods try to preserve essential network structures such as link and community structures using a dimensionality reduction learning architecture (RLA) [2], [3], [8]–[11]. Generally, RLA first embeds network structures A into low-dimensional representations Θ, and then transforms geometric relations in Θ into high-dimensional network structures $\widetilde{\mathrm{A}}$. The embedding performance of RLA is usually determined by the Euclidean distance between A and $\widetilde{\mathrm{A}}$. Classical NE methods include matrix factorization (DNR [8]), random walk (DeepWalk [12] and Node2vec [2]), and deep learning (SDNE [13] and Line [14]). Moreover, several representative methods such as inductive and dynamic representation learning [15]–[17] were proposed to embed hierarchal, uncertain and dynamic link structures. A systematic review of RLA for NEs of unsigned networks could be found in [3], [4], and related works for the applications of learning methods on data representation, extensive objective optimization, classification, approximation, reconstruction and prediction could be found in [18]–[20].

In recent years, generative adversarial nets (GANs) have quickly attracted much attention in NEs of unsigned networks due to their good performance in learning potential link distribution of the networks [21]. GANs mainly consist of two components: a generator ($\mathbf{G}$) that tries to approximate underlying prior link distribution of networks and a discriminator ($\mathbf{D}$) that tries to discriminate whether tested data are generated from real link structures or the generator. $\mathbf{G}$ and $\mathbf{D}$ play a two-player minimax game, and their learning performance will be iteratively improved by using a competitive learning. GANs have been successfully applied to many real-world applications [22], such as image generation [23], target intrusion sensing [24], representation disentanglement [25], evolutionary optimization [26], multiobjective optimization [27] and 3D objects generation [28]. One representative work of GANs for NEs is GraphGAN [21]. GraphGAN uses $\mathbf{G}$ to learn potential link distribution of unsigned networks, and adopts $\mathbf{D}$ to discriminate the truths of links generated by $\mathbf{G}$ and real network structures. GraphGAN was further generalized by CommunityGAN [29], MEGAN [30], GraphSGAN [31], HeGAN [32] and DynGraphGAN [33] to learn communities, biased random walks, dense subgraphs, heterogeneous information and dynamic links in unsigned networks, respectively.

However, these aforementioned NE methods are hardly

This work was supported by the National Natural Science Foundation of China under Grants 61803269, 61672358, 61572330, 61772393 and 61836009, in part by the Natural Science Foundation of Guangdong Province under Grant 2020A1515010790, and in part by the Technology Research Project of Shenzhen City under Grant by JCYJ20190808174801673. The work of C. A. Coello Coello was supported in part by CONACyT under Project 1920 (Fronteras de la Ciencia) and in part by SEP-Cinvestav 2018 Project (application no. 4).

L. Ma, Y. Ma, Q. Lin and J. Ji are with the School of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China.

C. A. Coello Coello is with the Department of Computer Science, CINVESTAV-IPN, México, D.F., 07360, México.

M. Gong is with the School of Electronic Engineering, the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China.

*Corresponding authors: M. Gong (E-mail: gong@ieee.org).

applied to signed networks, as they neglect conflicting relationships that widely exist in social media and Internet of things, such as friend/enemy, cooperation/competition and supporter/opponent [1], [34]. To preserve these conflicting relationships, some signed NE methods have been proposed by using a deep learning framework. One representative work is SiNE [35] which tries to preserve the structural balance of sampled triplets. The main idea behind SiNE is that a node is closer to its positively linked nodes than negatively linked nodes in the embedded space. Moreover, SIGNET [36] and DNE-SBP [37] were proposed to learn low-dimensional node vector representations, while preserving structural balance of signed networks. In addition, SNEA [38] was proposed to preserve node properties of signed networks. Note that, these aforementioned signed NE methods [35]–[38] generally have many hyper-parameters that need to be manually set in advance. Accordingly, they may obtain unsatisfactory performance for NEs of signed networks in a limited number of experimental trials.

Inspired by the two-player minimax game of GANs, in this paper, we propose a novel GAN framework (called SNEGAN) that unifies a generator and discriminator to learn low-dimensional node representation of signed networks while preserving both link structures and signs. More specifically, SNEGAN uses a generator to generate fake links that are used to deceive discriminator, while it adopts a discriminator to find truths and signs of these generated links and real ones. The generator and discriminator are gradually and iteratively optimized by minimaxing the representation difference between fake and real links, and they will eventually generate low-dimensional node representations of signed networks by using their potential model parameters. The main contributions of this paper are shown as follows:

- We proposes a novel GAN framework (SNEGAN) to learn low-dimensional node representations of signed networks while preserving both link structures and signs.
- In SNEGAN, we combine a generator with a graph softmax function and signed random walker technique to approximate underlying true connectivity distribution of signed networks. The generator enables to generate fake links and low-dimensional node representations to deceive discriminator and predict missing links, respectively. Moreover, the signed graph softmax function and random walk technique enable the generator to approximate real link distribution.
- We combine a discriminator with a tanh function to discriminate truths and signs of links sampled from the generator and real network structures. The discriminator enables to discriminate fake and real links and generate low-dimensional node representations to predict missing signs.
- Systematic experiments on nine LFR benchmark and six real-world signed networks show the superiority of the proposed SNEGAN over several state-of-the-art NE methods in embedding signed networks for tackling link (sign) prediction and reconstruction tasks.

The rest of the paper is organized as follows. Section II gives the preliminaries of signed NEs and GANs. Section III provides the proposed SNEGAN for NEs in signed networks. Experimental analyses are given in Section IV, and concluding remarks and possible future work are given in Section V.

II. Preliminaries

A. Signed network embedding

A signed network $\mathcal{G}$ is used to represent complex systems in social media and Internet of things that have conflicting relationships, where nodes denote entities of systems while positive/negative links represent friendly/hostile relationships between entities. $\mathcal{G}$ can be mathematically expressed as follows:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{S}\},$$

where $\mathcal{V}$ and $\mathcal{E}$ denote sets of nodes and links, respectively, while $\mathcal{S}$ represents signs of $\mathcal{E}$. We let $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ be the numbers of nodes and links of $\mathcal{G}$, respectively, and let the operator $|\mathcal{V}|$ denote the number of elements in $\mathcal{V}$. Moreover, we let $\mathrm{S}_{ij} \in \{+, -, 0\}$ denote the sign of a link e_{ij}, where $\mathrm{S}_{ij} = +$ ($\mathrm{S}_{ij} = -$) denotes a positive (negative) link between nodes i and j (i.e., $e_{ij} \in \mathcal{E}$) while $\mathrm{S}_{ij} = 0$ represents a fake link (i.e., $e_{ij} \notin \mathcal{E}$). In addition, we use a symmetric adjacency matrix $\mathrm{A} = [\mathrm{A}_{ij}] \in \mathbb{R}^{n \times n}$ to mathematically express $\mathcal{E}$ and $\mathcal{S}$ of $\mathcal{G}$, where each element $\mathrm{A}_{ij} \in \{+1, -1, 0\}$ represents link and sign states of e_{ij}. More specifically, A_{ij} is expressed as follows:

$$\mathrm{A}_{ij} = \begin{cases} +1 & \text{if } e_{ij} \in \mathcal{E} \text{ and } \mathrm{S}_{ij} = + \\ -1 & \text{if } e_{ij} \in \mathcal{E} \text{ and } \mathrm{S}_{ij} = - \\ 0 & \text{otherwise} \end{cases}.$$

We let $\mathrm{A}_{i\cdot}$ denote all elements of A at i-th row.

Signed network embedding aims to learn a mapping function $\mathbf{F} : \mathrm{A} \mapsto \Theta \in \mathbb{R}^{n \times d}$, which embeds A of $\mathcal{G}$ into low-dimensional node representations Θ in $\mathbb{R}^{n \times d}$, while preserving both link structures $\mathcal{E}$ and signs $\mathcal{S}$ of $\mathcal{G}$, where $d \ll n$ is the number of node representation dimensions.

Unlike unsigned networks, signed networks have conflicting links and structural balance properties. To obtain the underlying features of signed networks, certain signed NE methods such as SiNE [35], SIGNET [36], SNEA [38] and DNE-SBP [37] have been proposed. These methods try to obtain low-dimensional node representations of signed networks while preserving structural balance properties or node attributes. More specifically, SiNE and SIGNET try to preserve structural balance properties of triplets and paths, respectively, while DNE-SBP aims to preserve structural link balance properties. Moreover, SNEA mainly preserves both node attributes and link balance properties. However, these methods generally have many hyper-parameters that need to be manually set in advance. Accordingly, they may obtain unsatisfactory performance for NEs of signed networks in a limited number of experimental trials.

B. Generative Adversarial Nets

GANs [39] play a two-player minimax game between a generator $\mathbf{G}$ and a discriminator $\mathbf{D}$. $\mathbf{G}$ learns a data generation model $G(z; \Theta_G)$ to approximate the underlying true

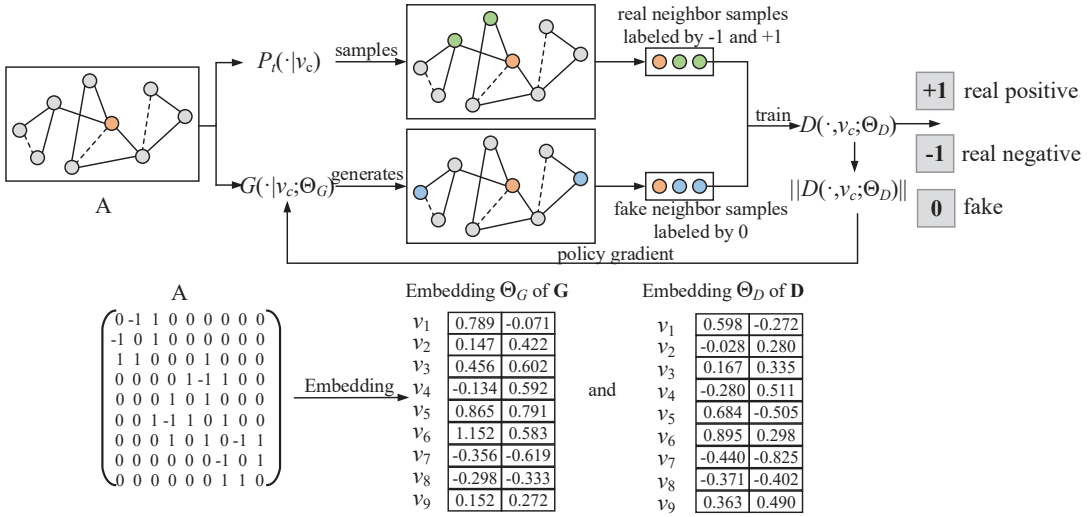


Fig. 1. Framework of SNEGAN. The solid edges are positive links and the dashed edges are negative links. The green vertices are sampled from the underlying true connectivity distribution $P_t(\cdot|v_c)$, while the blue vertices are generated by $G(\cdot|v_c;\Theta_G)$.

distributions P_t of data, and generates fake samples z to deceive $\mathbf{D}$, where Θ_G is the parameters of $\mathbf{G}$. $\mathbf{D}$ learns a discrimination model $D(x;\Theta_D)$ to discriminate real data x from fake samples, where Θ_D is the parameters of $\mathbf{D}$. $\mathbf{G}$ and $\mathbf{D}$ compete with each other, and they will be gradually evolved by playing a two-player minimax game with an objective $\mathbf{O}(G,D)$. This minimax game is mathematically expressed as follows:

$$\begin{aligned}\min_{\Theta_G}\max_{\Theta_D}\ \mathbf{O}(G,D) &= \mathbb{E}_{x\sim P_t}\left[\log D\big(x;\Theta_D\big)\right] \\ &+ \mathbb{E}_{z\sim P_z}\left[\log\big(1-D(G(z;\Theta_G);\Theta_D)\big)\right],\end{aligned} \tag{1}$$

where P_z denotes the input data distribution for $\mathbf{G}$ and $\mathbb{E}$ denotes an operator of mathematical expectation.

Inspired by GANs, many NE methods have been proposed for unsigned networks, such as GraphGAN [21], Community-GAN [29], MEGAN [30], GraphSGAN [31], HeGAN [32] and DynGraphGAN [33], and they have been successfully applied to various graph tasks such as link prediction, community detection and node classification.

III. The Proposed Method: SNEGAN

In this section, we first introduce the framework and optimization model of SNEGAN for NEs in signed networks, and then give the implementation and optimization of the designed generator and discriminator.

A. Framework and optimization model in SNEGAN

Given a signed network $\mathcal{G}=\{\mathcal{V},\mathcal{E},\mathcal{S}\}$, SNEGAN tries to learn the low-dimensional representation of nodes $\mathcal{V}$ while preserving both link structures $\mathcal{E}$ and signs $\mathcal{S}$. For a node v_c, we define its neighbors $\mathcal{L}_c$ as a set of nodes that have a positive or negative link with v_c. Moreover, we let $P_t(v|v_c)$ denote the underlying true connectivity distribution of v_c. In this case, $\mathcal{L}_c$ can be considered as a set of real neighbor samples extracted from $\mathbf{G}$ under $P_t(v|v_c)$. In addition, we let $P_t^p(v|v_c)$ and $P_t^n(v|v_c)$ denote the true positive and negative connectivity distribution of v_c, respectively.

SNEGAN adopts a GAN framework that has a generator $\mathbf{G}$ and a discriminator $\mathbf{D}$. $\mathbf{G}$ tries to learn a data generation model $G(v|v_c;\Theta_G)$ to approximate $P_t(v|v_c)$ for each node $v_c\in\mathcal{V}$, and attempts to generate (or select) sets of nodes from $\mathcal{V}$ as fake neighbors $\widetilde{\mathcal{L}}_c$ of v_c, where Θ_G is the learned low-dimensional node representations in $\mathbf{G}$. $\mathbf{D}$ aims to learn a discrimination model $D(v,v_c;\Theta_D)$ for each node $v_c\in\mathcal{V}$ to detect whether a node $v\in\{\mathcal{L}_c\bigcup\widetilde{\mathcal{L}}_c\}$ is sampled from real neighbors $\mathcal{L}_c$ or fake neighbors $\widetilde{\mathcal{L}}_c$ of v_c, where Θ_D is the learned low-dimensional node representations in $\mathbf{D}$. Moreover, $D(v,v_c;\Theta_D)$ tries to detect link signs (i.e., positive, negative, or unobserved) for each tested pair of nodes.

In SNEGAN, $\mathbf{G}$ and $\mathbf{D}$ play a two-player minimax game. More specifically, for each node $v_c\in\mathcal{V}$, $\mathbf{G}$ generates fake neighbors $\widetilde{\mathcal{L}}_c$ based on $G(v|v_c;\Theta_G)$ to deceive $\mathbf{D}$, while $\mathbf{D}$ finds the truths and signs of links sampled from $\widetilde{\mathcal{L}}_c$ and $\mathcal{L}_c$ by using $D(v,v_c;\Theta_D)$. For each test fake or real neighbor sample v, $\mathbf{D}$ will return a discrimination value $D(v,v_c;\Theta_D)\in[-1\ 1]$ which determines the probability of positive and negative link existence between v and v_c, and discriminates the truths and signs of e_{vc} as follows:

$$e_{vc}=\begin{cases}\text{real positive} & \text{if } D(v,v_c;\Theta_D)>\xi_p>0\\ \text{real negative} & \text{if } D(v,v_c;\Theta_D)<\xi_n<0\\ \text{fake} & \text{if } 0>\xi_n\le D(v,v_c;\Theta_D)\le\xi_p>0\end{cases}, \tag{2}$$

where ξ_p and ξ_n are probability thresholds of a positive and negative link, respectively. In SNEGAN, we need not set ξ_p and ξ_n values in advance as the optimization of $\mathbf{D}$ is to maximize the log-probability of assigning correct labels {+1, -1, 0} to both real and fake samples (see Section III. C and Section III. D).

The optimization model of SNEGAN for the two-player minimax game with a value function $\mathbf{O}(G,D)$ can be mathe-

matically formulated as follows:

$$\begin{aligned}&\min_{\Theta_G}\max_{\Theta_D}\ \mathbf{O}\left(G,D\right)\\&=\sum_{c=1}^{n}\Big(\mathbb{E}_{v\sim P_t^p(\cdot|v_c)}\left[\log\max\left(\tau,D(v,v_c;\Theta_D)\right)\right]\\&+\mathbb{E}_{v\sim P_t^n(\cdot|v_c)}\left[\log\max\left(\tau,-D(v,v_c;\Theta_D)\right)\right]\\&+\mathbb{E}_{v\sim G(\cdot|v_c;\Theta_G)}\left[\log\left(1-||D\left(v,v_c;\Theta_D\right)||\right)\right]\Big),\end{aligned}\tag{3}$$

where τ is a small positive constant value (we set it as 10^{-6}). Moreover, $||D\left(v,v_c;\Theta_D\right)||$ evaluates the absolute value of $D\left(v,v_c;\Theta_D\right)$ while $\sum_{c=1}^{n}(\cdot)$ enables to tackle link structures of all nodes in signed networks. As known from (3), $\mathbf{O}(G,D)$ mainly consists of three parts. The first and second parts are mainly used for $\mathbf{D}$ to discriminate truths of positive and negative link samples, respectively, while the final part is mainly used for $\mathbf{G}$ to generate fake neighbors to deceive $\mathbf{D}$.

The optimal $\mathbf{G}$ and $\mathbf{D}$ can be learned by alternately and iteratively minimizing and maximizing $\mathbf{O}\left(G,D\right)$. The main framework of SNEGAN is given as follows:

Step 1: $\mathbf{G}$ optimization: $G\left(\cdot|v_c;\Theta_G\right)$ is optimized to deceive $\mathbf{D}$ by minimizing the difference between sampled real and fake neighbors.

Step 2: $\mathbf{D}$ optimization: $D\left(v,v_c;\Theta_D\right)$ is trained with sampled real neighbors from $P_t\left(\cdot|v_c\right)$ and sampled fake neighbors from $G\left(\cdot|v_c;\Theta_G\right)$, which tries to maximize the log-probability of assigning correct labels {+1, -1, 0} to both real and fake samples.

Step 3: Step 1 and Step 2 are alternately and iteratively executed until estimated Θ_G and Θ_D converge or the current iteration reaches the preset maximum number of generations t_{max}. The competition between $\mathbf{G}$ and $\mathbf{D}$ would drive them to improve their performance.

Step 4: The optimal $\mathbf{G}$ and $\mathbf{D}$ return d-dimensional node representations Θ_G and Θ_D in the embedding space, respectively. Θ_G and Θ_D can be used for link and sign prediction (or reconstruction) tasks, respectively.

Fig. 1 shows the framework of SNEGAN. In the following, the details of the designed $\mathbf{G}$ and $\mathbf{D}$ and their optimization at each iteration are given.

B. The designed generator $\mathbf{G}$

The generator $\mathbf{G}$ aims to learn $G\left(v|v_c;\Theta_G\right)$ to approximate the underlying true connectivity distribution $P_t\left(v|v_c\right)$, and attempts to generate fake neighbors $\widetilde{\mathcal{L}}_c$ for each node $v_c\in\mathcal{V}$ in signed networks. Given the low-dimensional node representations Θ_G in a signed network $\mathcal{G}$, a straightforward way for learning $G\left(v|v_c;\Theta_G\right)$ is to compute the conventional softmax function over all other nodes [40] which evaluates the relevance probability between v and v_c. Formally, $G\left(v|v_c;\Theta_G\right)$ is computed as follows:

$$G\left(v|v_c;\Theta_G\right)=\frac{\exp\left(\Theta_G^{v\cdot}\cdot(\Theta_G^{c\cdot})^{\mathrm{T}}\right)}{\sum_{v\in\mathcal{V}}\exp\left(\Theta_G^{v\cdot}\cdot(\Theta_G^{c\cdot})^{\mathrm{T}}\right)},\tag{4}$$

where $\Theta_G^{c\cdot}$ denotes all elements of c-th row in Θ_G and $(\Theta_G^{c\cdot})^{\mathrm{T}}$ is the transposition of $\Theta_G^{c\cdot}$. In this case, fake neighbors $\widetilde{\mathcal{L}}_c$ for v_c are generated by choosing the nodes with the k_c top $G\left(v|v_c;\Theta_G\right)$ values, where k_c is the degree of node v_c in $\mathcal{G}$. The main idea behind this generation way is that two linked nodes generally have similar properties, and therefore they should be close to each other in the embedding space.

This softmax function provides a possibly effective way to compute $G\left(v|v_c;\Theta_G\right)$. However, its computation and updating are time-consuming as they should compute the gradients of $G\left(v|v_c;\Theta_G\right)$ to Θ_G and update $G\left(v|v_c;\Theta_G\right)$ for all nodes in $\mathcal{G}$ (see (4)). Moreover, it neglects the link structures of $\mathcal{G}$. Accordingly, it is difficult to reveal the potential link proximity of nodes.

To solve these above-mentioned issues, we propose a signed graph softmax function for computing $G\left(\cdot|v_c;\Theta_G\right)$. This signed graph softmax function, which is an extended version of the graph softmax function [21], enables to efficiently compute $G\left(\cdot|v_c;\Theta_G\right)$ by taking links and signs of $\mathcal{G}$ into consideration. It has certain desirable properties such as signed structure awareness, computational efficiency and normalization.

The signed graph softmax function works as follows. For each node v_c, we first construct a BFS-tree $\mathcal{T}_c$ rooted at v_c on $\mathcal{G}$ by using a breadth first search (BFS) algorithm, and then compute the relevance probability $p_c(v_i|v_c)$ between v_c and v_i by using a softmax function multiplied by a signed transition probability coefficient H_{ic} over all v_c' neighbors in $\mathcal{T}_c$. Formally, $p_c(v_i|v_c)$ is evaluated as follows:

$$p_c\left(v_i|v_c\right)=\frac{\mathrm{H}_{ic}\cdot\exp\left(\Theta_G^{i\cdot}\cdot(\Theta_G^{c\cdot})^{\mathrm{T}}\right)}{\sum_{v_i\in\mathcal{T}_c}\mathrm{H}_{ic}\cdot\exp\left(\Theta_G^{i\cdot}\cdot(\Theta_G^{c\cdot})^{\mathrm{T}}\right)}.\tag{5}$$

Here, H_{ic} records the probability of a walk from v_c to v_i on $\mathcal{G}$, and it is computed as follows:

$$\mathrm{H}_{ic}=\begin{cases}\dfrac{k_c^2-1}{k_c^2}\cdot\dfrac{1}{k_c^+} & \mathrm{A}_{ic}=+1\\ 0 & \mathrm{A}_{ic}=0\\ \dfrac{1}{k_c^2}\cdot\dfrac{1}{k_c^-} & \mathrm{A}_{ic}=-1\end{cases},\tag{6}$$

where k_c denotes the degree of v_c, while k_c^+ and k_c^- represent the numbers of positive and negative edges linked with v_c in $\mathcal{T}_c$, respectively. From (6), we can see that this transition probability setting allows to walk along negative links with a smaller probability than positive links, and $\sum_i\mathrm{H}_{ic}=1$. The intuition behind this setting is to proximate the distribution of potential balanced clusters of signed networks, where a balanced cluster is composed of nodes that have more positive links than negative links. In signed networks, the balance properties of clusters will affect link distributions. More specifically, nodes in the same clusters have many links while those in different clusters have few links.

For a special case that v_c has only positive or negative links, we set its signed transition probability H_{ic} as follows:

$$\mathrm{H}_{ic}=\frac{1}{k_c}.\tag{7}$$

Fig. 2 shows the difference of H_{ic} evaluation under the signed and regular random walks.

Next, given a BFS tree $\mathcal{T}_c$ rooted at v_c, each node $v\in\mathcal{T}_c$ has a unique path $P_{v_c\to v}=(v_c,v_{r_1},v_{r_2},\ldots,v_{r_{l^c-1}},v)$ from v_c to v, where l^c is the length of the path. Moreover, any two

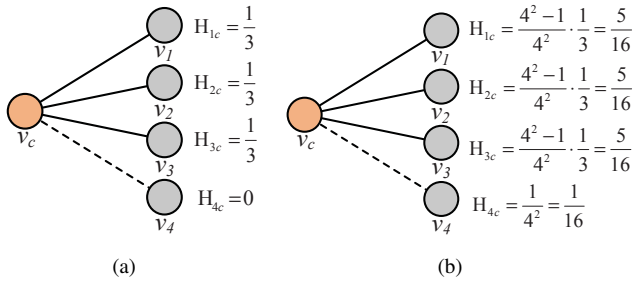


Fig. 2. Different random walks on signed networks. The solid and dashed edges are positive and negative links, respectively. (a) unsigned random walk: it evenly walks along positive links, and (b) signed random walk: it can walk along a negative link with a smaller probability than a positive link.

Algorithm 1 Signed random walk sampling technique on v_c for the generator

Input: Signed BFS tree $\mathcal{T}_c$ and representation vectors Θ_G in $\mathbf{G}$.
Output: Fake neighbors $\widetilde{\mathcal{L}}_c$.
while ($|\widetilde{\mathcal{T}}_c| < k_c$) **do**
  Set $\mathcal{V}_c \leftarrow \varnothing$ and $v \leftarrow v_c$.
  while ($v \notin \mathcal{V}_c$) **do**
    Set $v_t \leftarrow v$.
    Set $\mathcal{V}_c \leftarrow \mathcal{V}_c \bigcup \{v\}$.
    Randomly choose a node v_i proportionally to $p_c(v_i|v)$ from the nodes linked with v in $\mathcal{T}_c$.
    Set $v \leftarrow v_i$.
  end while
  Set $\widetilde{\mathcal{L}}_c \leftarrow \widetilde{\mathcal{L}}_c \bigcup \{v_t\}$.
end while

adjacent nodes in $\mathcal{T}_c$ have a tree link (i.e., a direct path with $l^c = 1$). Accordingly, the probability of a link from v_c to v is approximated as follows:

$$G\left(v|v_c;\Theta_G\right) = p_c\left(v_{r_1}|v_c\right)\cdot\prod_{i=2}^{l^c-1} p_c\left(v_{r_i}|v_{r_{i-1}}\right)\cdot p_c\left(v|v_{r_{l^c-1}}\right). \tag{8}$$

We refer to this approximation probability index G as the signed graph softmax function.

Based on (8), we can approximate the connectivity probability of v_c by computing $G\left(v|v_c;\Theta_G\right)$ over all $v \in \mathcal{V}$. However, computing $G\left(v|v_c;\Theta_G\right)$ is time-consuming as all nodes in $\mathcal{G}$ should be traversed for evaluating p_c.

To reduce the computational complexity, for each node v_c, we choose k_c nodes by using a signed random walk sampling technique as fake neighbors $\widetilde{\mathcal{L}}_c$. The signed random walk sampling technique independently executes k_c samplings, each of which tries to choose a node as a fake neighbor of v_c. More specifically, for each sampling, a random walker starts at v_c, and then randomly walks on $\mathcal{T}_c$ based on $p_c(\cdot|v_c)$. Subsequently, a node is chosen as the neighbor of v_c when its previous visited node has been walked two times. Algorithm 1 gives the details of this signed random walk sampling technique on v_c. Fig. 3 gives an illustration example of a sampling process for generating a fake neighbor of v_c.

C. The designed discriminator $\mathbf{D}$

The discriminator $\mathbf{D}$ aims to learn $D\left(v, v_c;\Theta_D\right)$ for each node $v_c \in \mathcal{V}$ to maximally detect the truths and signs of links between v_c and all nodes in $\mathcal{L}_c \bigcup \widetilde{\mathcal{L}}_c$. More specifically, it tries to discriminate whether a node $v \in \{\mathcal{L}_c \bigcup \widetilde{\mathcal{L}}_c\}$ is sampled from real neighbors $\mathcal{L}_c$ or fake neighbors $\widetilde{\mathcal{L}}_c$, and detect the signs (i.e., positive, negative, or unobserved) of true links.

Given the low-dimensional representations Θ_D of nodes in $\mathbf{D}$, a straightforward way for learning $D\left(v|v_c;\Theta_D\right)$ is to compute the sigmoid function [21] under the inner product between Θ_D^v and $(\Theta_D^c)^{\mathrm{T}}$. However, this sigmoid function only predicts the connectivity between v and v_c, while neglecting the sign of the connectivity. To solve this issue, in SNEGAN, we adopt a tanh function to define D, which evaluates the similarity scores between predicted and real relationships and signs. More specifically, for each pair of nodes v and v_c, the tanh function returns a value $\hat{\mathrm{A}}_{vc}$ in the range of [-1 1]. In this case, we can use the absolute value and sign of $\hat{\mathrm{A}}_{vc}$ to predict the connectivity and sign of a possible link between v and v_c, respectively. Formally, the tanh function is computed as follows:

$$\begin{aligned} D\left(v, v_c;\Theta_D\right) &= \tanh\left(\Theta_D^v\cdot\left(\Theta_D^c\right)^{\mathrm{T}}\right) = \hat{\mathrm{A}}_{vc} \\ &= \frac{\exp\left(\Theta_D^v\cdot\left(\Theta_D^c\right)^{\mathrm{T}}\right) - \exp\left(-\Theta_D^v\cdot\left(\Theta_D^c\right)^{\mathrm{T}}\right)}{\exp\left(\Theta_D^v\cdot\left(\Theta_D^c\right)^{\mathrm{T}}\right) + \exp\left(-\Theta_D^v\cdot\left(\Theta_D^c\right)^{\mathrm{T}}\right)}, \end{aligned} \tag{9}$$

where Θ_D^v is the d-dimensional representation vectors of node v and v_c in $\mathbf{D}$ while Θ_D is the union of all d-dimensional node representations in $\mathbf{D}$.

Subsequently, the truths and signs of tested links can be discriminated by (2). Note that, during the training of $\mathbf{D}$, the aim of $\mathbf{D}$ is to maximize the log-probability of assigning correct labels {+1, -1, 0} to both real and fake samples.

$\mathbf{D}$ iteratively evolves the node representations until $\hat{\mathrm{A}}$ is obviously distinguishable from A.

D. Optimization of the generator $\mathbf{G}$ and the discriminator $\mathbf{D}$

As known from (3), $\mathbf{G}$ tries to minimize the statistical probability that the given discriminator $\mathbf{D}$ correctly detects the truths and signs of links sampled from $P_t\left(\cdot|v_c\right)$ and $G\left(\cdot|v_c;\Theta_G\right)$. Here, when considering the discreteness of the sampling of v, we adopt the policy gradient [21], [41], [42] to compute the gradient of $\mathbf{O}(G, D)$ under a given $\mathbf{D}$ with respect to the parameters Θ_G in $\mathbf{G}$. Formally,

$$\begin{aligned} &\nabla_{\Theta_G}\mathbf{O}\left(G, D\right) \\ =&\nabla_{\Theta_G}\sum_{c=1}^{n}\mathbb{E}_{v\sim G(\cdot|v_c)}\left[\log\left(1-||D\left(v, v_c\right)||\right)\right] \\ =&\sum_{c=1}^{n}\mathbb{E}_{v\sim G(\cdot|v_c)}\left[\nabla_{\Theta_G}\log G\left(v|v_c\right)\log\left(1-||D\left(v, v_c\right)||\right)\right]. \end{aligned} \tag{10}$$

Similarly, as known from (3), $\mathbf{D}$ aims to maximize the statistical probability of discriminating true labels of links sampled from $P_t\left(\cdot|v_c\right)$ and $G\left(\cdot|v_c;\Theta_G\right)$. Here, when considering a good tradeoff between efficiency and performance, we adopt a well-known stochastic gradient ascent method to

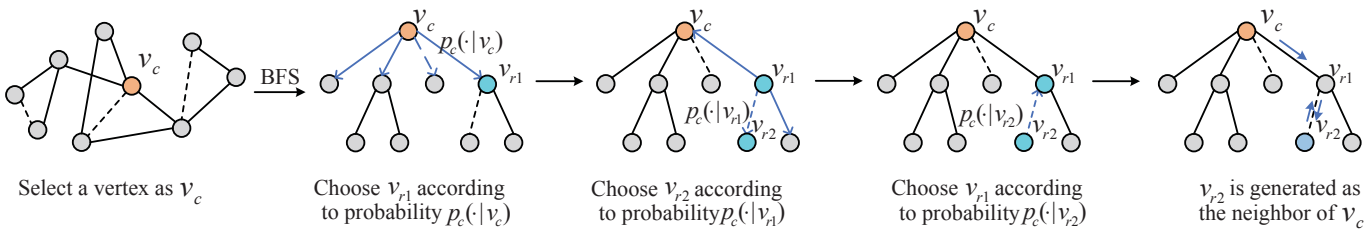


Fig. 3. Sampling process of generator $\mathbf{G}$ for generating a possible neighbor of v_c.

solve the maximization problem. More specifically, for each iteration, we update $\mathbf{D}$ by ascending the gradient of $\mathbf{O}(G, D)$ with respect to its parameters Θ_D. Formally,

$$\begin{aligned}\nabla_{\Theta_D}\mathbf{O}(G,D) = \\ \sum_{c=1}^{n}\Big(&\mathbb{E}_{v\sim P_t^p(\cdot|v_c)}\left[\nabla_{\Theta_D}\log\max\left(\tau, D(v,v_c;\Theta_D)\right)\right] \\ &+\mathbb{E}_{v\sim P_t^n(\cdot|v_c)}\left[\nabla_{\Theta_D}\log\max\left(\tau, -D(v,v_c;\Theta_D)\right)\right] \\ &+\mathbb{E}_{v\sim G(\cdot|v_c;\Theta_G)}\left[\nabla_{\Theta_D}\log\left(1-||D(v,v_c;\Theta_D)||\right)\right]\Big).\end{aligned} \tag{11}$$

E. Algorithm Framework and Computational Complexity of SNEGAN

Given the aforementioned optimization model in (3), $\mathbf{G}$ and $\mathbf{D}$, Algorithm 2 gives the algorithm framework of SNEGAN for NEs. Subsequently, the computational complexity of SNEGAN in **Algorithm 2** is analyzed.

Initialization and pre-train (line 3): SNEGAN adopts a random strategy to initialize Θ_G and Θ_D. Specifically, $n \cdot d$ values within [-1 1] are randomly generated for Θ_G and Θ_D. Therefore, this step has a computational complexity $O(n \cdot d)$.

Construct signed BFS tree (line 4): The complexity of constructing signed BFS trees for all nodes is $O(n \cdot m)$ [43].

G optimization (lines 6-12): SNEGAN executes the loop $t_g \cdot n$ times in lines 8-10. Within this loop,

- Line 8: It executes Algorithm 1 $\bar{k}$ times, where $\bar{k}$ is the averaged node degree of the network. Algorithm 1 takes $(\log n + 1)$ steps at most to visit a node two times on $\mathcal{T}_c$, each of which takes $O(d \cdot \bar{k})$ to compute $p_c(v_i|v_c)$ for all $v_i \in \mathcal{L}_c$.
- Line 9: It evaluates $G(v|v_c;\Theta_G^t)$ for each node $v \in \mathcal{T}_c$, and each evaluation has a computational complexity $d \cdot \bar{k} \cdot \log n$.
- Line 10: It takes $O(\bar{k}^2 \cdot d \cdot \log n)$ to update Θ_G.

Therefore, $\mathbf{G}$ optimization has an overall computational complexity $O(t_g \cdot n \cdot \bar{k}^2 \cdot d \cdot \log n)$.

D optimization (lines 13-19): SNEGAN executes the loop $t_g \cdot n$ times in lines 15-17. Within this loop,

- Line 15: It constructs test samples for node v_c, which has a computational complexity $O(1)$.
- Line 16: It computes $D(v, v_c;\Theta_D^t)$ $2 \cdot \bar{k}$ times, and each computation takes d basic operations.
- Line 17: It takes $O(n \cdot \bar{k} \cdot d)$ to update Θ_D^t.

Therefore, $\mathbf{D}$ optimization has an overall computational complexity $O(t_g \cdot n \cdot \bar{k} \cdot d)$.

Algorithm 2 Signed SNEGAN framework

1: **Input**: A signed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, dimension of embedding d, number of training t_g for $\mathbf{G}$ ($\mathbf{D}$) at each iteration, and maximum number of iterations t_{max}.
2: **Output**: The low-dimensional representations Θ_G and Θ_D of nodes in $\mathbf{G}$ and $\mathbf{D}$, respectively.
3: Initialization and pre-train $G(v|v_c;\Theta_G)$ and $D(v, v_c;\Theta_D)$.
4: Construct signed BFS-tree $\mathcal{T}_c$ for all $v_c \in \mathcal{V}$.
5: **for** $(t = 1 : t_{max})$ **do**
6: **for** $q = 1 : t_g$ **do** // **G optimization**
7: **for** (Each node $v_c \in \mathcal{V}$) **do**
8: Generate k_c nodes from $\mathcal{T}_c$ as the neighbors $\widetilde{\mathcal{L}}_c$ of v_c by using Algorithm 1.
9: Compute $G(v|v_c;\Theta_G^t)$ for each node $v \in \mathcal{T}_c$ by using (5), (6), (7) and (8)
10: Update Θ_G^t to Θ_G^{t+1} by using (10).
11: **end for**
12: **end for**
13: **for** $q = 1 : t_g$ **do** // **D optimization**
14: **for** (Each node $v_c \in \mathcal{V}$) **do**
15: Set test samples $\leftarrow$ {real neighbor samples $\mathcal{L}_c$ $\bigcup$ fake neighbor samples $\widetilde{\mathcal{L}}_c$}.
16: Compute $D(v, v_c;\Theta_D^t)$ for each node $v \in$ test samples by using (9).
17: Update Θ_D^t to Θ_D^{t+1} by using (11).
18: **end for**
19: **end for**
20: **end for**

In **Algorithm 2**, SNEGAN executes steps 6-19 with t_{max} times. Therefore, SNEGAN has a computational complexity $O\left(t_{max} \cdot t_g \cdot n \cdot \bar{k}^2 \cdot d \cdot \log n\right)$. Generally, $\bar{k}$ and d are small constant values.

IV. Experimental Results

In this section, we test SNEGAN on nine simulated LFR and six real-world signed networks, and compare it with four classical NE methods in tackling link (sign) prediction and reconstruction tasks. In the following, the experimental settings are first given, and then the experimental results are analyzed. Finally, the sensitivity of some vital parameters in SNEGAN is analyzed.

TABLE I
BASIC PROPERTIES OF THE TESTED REAL-WORLD NETWORKS. m_+ AND m_- DENOTE THE NUMBERS OF POSITIVE AND NEGATIVE LINKS, RESPECTIVELY.

Networks	n	m	m_+	m_-	Fields
GGS	16	58	29	29	Social
War	166	1,433	1,295	138	Social
Yeast	690	1,080	860	220	Biological
E.coli	1,461	3,215	1,879	1,336	Biological
Soc-alpha	3,783	21,486	22,650	1,536	Social
Wiki	7,115	100,693	78,603	22,090	Social

TABLE II
PARAMETER SETTINGS OF ALL TEST ALGORITHMS.

Algorithm	Parameter	Meaning	Setting
SNEGAN	d	Embedding dimension	20
	l_r	Learning rate	$1\mathrm{E}^{-3}$
	t_{max}	Maximum number of iterations	15 or 20
	t_g	Number of training	5 or 20
DNE-SBP	d	Embedding dimension	20
	n_l	Number of layers	3
	$\alpha_1, \ldots, \alpha_{n_l}$	Constraint weights	Setting as [37]
	$\lambda_1, \ldots, \lambda_{n_l}$	Regularization weights	Setting as [37]
	β	Reconstruction weight	25
	l_r	Learning rate	$1\mathrm{E}^{-4}$
	t_{max}	Maximum number of iterations	100
GraphGAN	d	Embedding dimension	20
	l_r	Learning rate	$1\mathrm{E}^{-3}$
	t_{max}	Maximum number of iterations	15 or 20
	t_g	Number of training	5 or 20
Node2vec	d	Embedding dimension	20
	k	Size of window	5
	ζ	Number of walk	5
	l	Length of walk	10
	l_p	Enter parameter	1
	l_q	In-out parameter	0.5
DeepWalk	d	Embedding dimension	20
	k	Size of window	5
	ζ	Number of walk	5
	l	Length of walk	10

A. Experimental Settings

1) Experimental Networks: LFR benchmark signed networks [44], [45]: They are generalized from LFR benchmark networks [44], which considers conflicting, scale-free and clustering properties of signed networks. They use a mixing parameter r to control the ratio of positive links within communities, and adopt mixing parameters γ and β to control power-law distribution of node degrees and cluster sizes, respectively. Here, nine LFR benchmark undirected signed networks are tested with r ranging from 0.1 to 0.5 in an interval 0.05, $n = 500$, $\beta = 1$, $\gamma = 2$ and $\bar{k} = 10$. These networks are tested to validate the performance of SNEGAN on signed networks with heterogeneous clusters and conflicting link distribution.

Real-world signed networks: Six real-world undirected signed networks (see Table I) are tested, including two small-scale social networks, two medium-scale biological networks and two large-scale social networks.

GGS network: It depicts alliances/opponents of 16 Gahuku-Gama subtribes in the cultures of highland New Guinea [46] (http://mrvar.fdv.uni-lj.si/sola/info4/andrej/prpart5.htm).

War network: It shows alliances/opponents of 166 countries in the period from 2000 to 2010. It was extracted from the Correlates of War project [47] that contain datasets "Formal Alliances (v4.1)" and "Militarized Interstate Disputes (v4.01)" (http://www.correlatesofwar.org/datasets.htm).

Yeast network: It represents 1,080 binding site inter-actions of 690 transcription factors in the gene regulatory network of S.cerevisiae [48].

E.coli network: It is the gene regulatory network of Escherichia coli [7] with 1,461 operon and 3,215 transcriptional interactions. It can be downloaded from RegulonDB database (http://regulondb.ccg.unam.mx).

Soc-alpha network: It describes trust/distrust relationships among the anonymous people who use Bitcoin for trade on the Bitcoin Alpha platform. Here, the unweighted version of the network in http://snap.standord.edu/data is chosen.

Wiki network: It describes support/opponent votes from electors to candidates. It is composed of 7,115 electors and 100,693 candidates. A detailed description of this network is given in [1], [49].

2) Baseline Algorithms: Four classical NE algorithms are chosen as baseline methods, including DeepWalk [12], Node2vec [2], GraphGAN [21] and DNE-SBP [37]. The reason for choosing DeepWalk and Node2vec as baseline methods is to show the limitation of classical NE algorithms for signed networks. A comparison of SNEGAN with GraphGAN is made to illustrate the effectiveness of SNEGAN in solving NEs for signed networks, while a comparison of SNEGAN with DNE-SBP tries to validate the superiority of the GAN-based NE framework over dimensional reduction frameworks.

DeepWalk [12]: It uses a random walk to sample nodes in graphs, and employs a Skip-Gram model for node embeddings.

Node2vec [2]: It extends the idea of DeepWalk by using a biased random walk for node embeddings of unsigned networks.

GraphGAN [21]: It first unifies a generative and discriminative model for node embedding of unsigned networks.

DNE-SBP [37]: It employs a deep autoencoder to learn low-dimensional node vector representations while preserving the structural balance of signed networks.

3) Criteria: To validate the performance of all algorithms, the learned low-dimensional node representations are used for sign (link) prediction and reconstruction tasks in signed networks.

Sign Prediction and Reconstruction: These tasks reflect the performance of all algorithms in predicting and reconstructing link signs (positive and negative) of signed networks. For the sign prediction task, we first hide 10% of link signs, and then use the rest of link signs to train a logistic regression model for predicting link signs. For the sign reconstruction task, we use all link signs to train a logistic regression model for reconstructing link signs. Then, we built the *Had* feature vector representations for each tested link based on Θ_D of $\mathbf{D}$. More specifically, the *Had* feature vector representation of an link e_{ij} is evaluated as $\Theta_D^{i\cdot} \odot \Theta_D^{j\cdot}$, where $\odot$ represents the hadamard product operator. To evaluate an overall sign

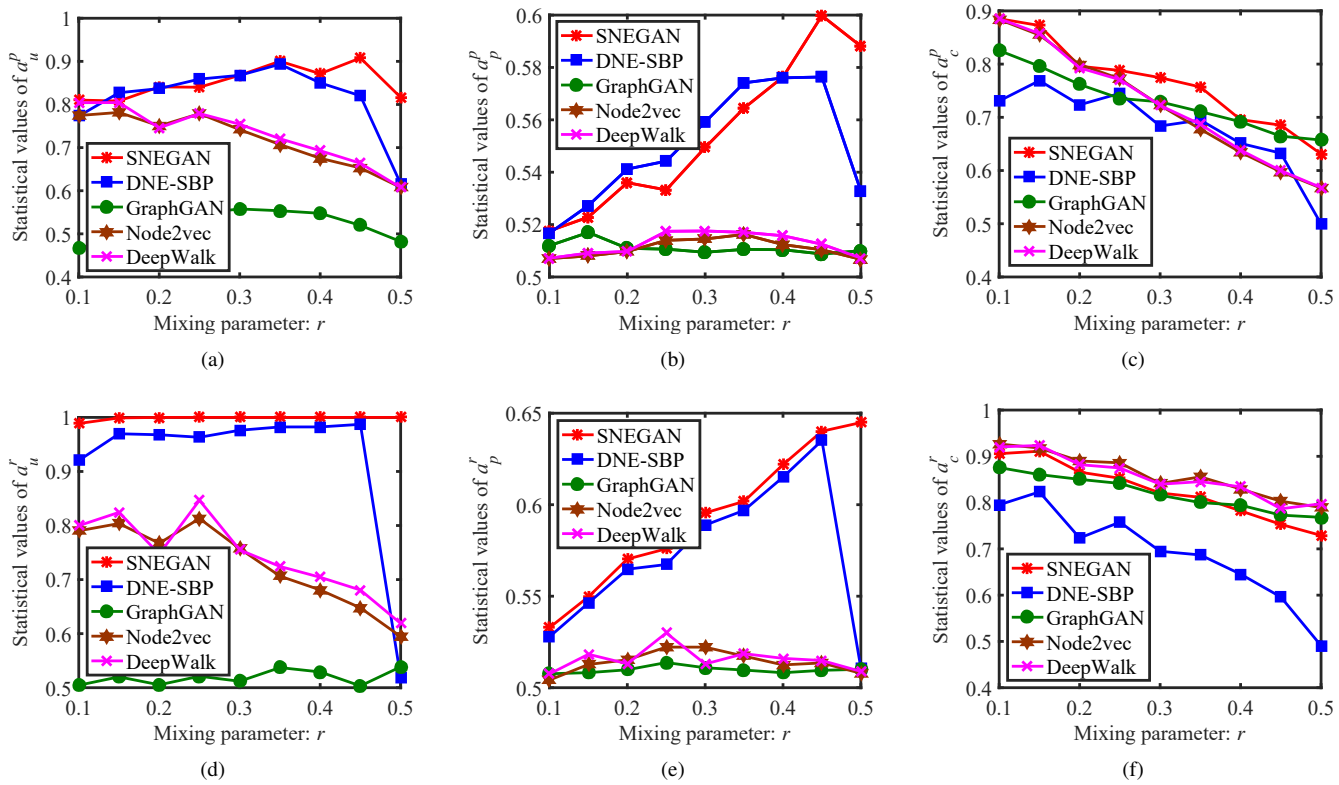


Fig. 4. Statistical results of all algorithms on the nine LFR benchmark signed networks with different mixing parameters r. (a) a_u^p, (b) a_p^p, (c) a_c^p, (d) a_u^r, (e) a_p^r and (f) a_c^r

.

prediction and reconstruction performance, we adopt the area under a ROC curve (a_u^b, $b \in \{p, r\}$) and average precision (a_p^b, $b \in \{p, r\}$) as metrics, where the superscript letters p and r correspond to the criteria for sign prediction and reconstruction tasks, respectively. Here, a_u^b evaluates the probability that a real neighbor sample has a higher ranking than a fake neighbor sample [50], and it is computed as follows:

$$a_u^b = \frac{n_s - n_0(n_0+1)/2}{n_0 \cdot n_1}, \tag{12}$$

where n_0 and n_1 are the numbers of tested real and fake neighbor samples, respectively. Here, $n_s = \sum r_i$, where r_i is the ranking of the i-th real neighbor sample in the list. A higher a_u^b denotes a better performance in sign prediction or reconstruction. For each $b \in \{p, r\}$, a_p^b evaluates the sum of a precision-recall curve as the weighted mean of precision under each threshold $q \in \mathcal{Q}$ [51], and it is computed as follows:

$$a_p^b = \sum_{q=1}^{|\mathcal{Q}|} (a_r^q - a_r^{(q-1)}) a_p^q / |\mathcal{Q}|, \tag{13}$$

where a_r^q and a_p^q are the evaluated recall and precision under threshold q, respectively.

Link Prediction and Reconstruction: These tasks evaluate the accuracy of all algorithms in predicting and reconstructing links. For the link prediction task, we first randomly hide 10% of links, and then generate 10% of links on the disconnected node pairs. Next, we learn node representations based on the rest of the network and adopt the inner product representation to predict the probability of the existence of the tested edges. For the link reconstruction task, we use all links to learn node representations for reconstructing links. The inner product presentation of an link e_{ij} is evaluated as $\Theta_G^i \cdot \Theta_G^j$. To evaluate an overall performance in link prediction and reconstruction, we adopt the prediction accuracy (a_c^b, $b \in \{p, r\}$) as a criterion. For each $b \in \{p, r\}$, a_c^b evaluates the probability that a predicted link is the same as the real one [21], [51], and it is computed as follows:

$$a_c^b = \frac{m_t^p + m_t^n}{m_t^p + m_f^p + m_f^n + m_t^n}, \tag{14}$$

where m_t^p and m_f^p denote the numbers of true and false real samples, respectively, while m_f^n and m_t^n represent the numbers of true and false fake samples, respectively.

4) Simulation Settings: GraphGAN, SNEGAN, DeepWalk and Node2vec are simulated by Python on a PC with Intel (R), Core (TM), i7-6700 CPU and 3.41 GHZ, while DNE-SBP is coded by Matlab. For each network, all algorithms are independently tested for 30 trials with key parameter settings in Table II.

B. Experimental Results on Signed LFR Benchmark Networks

In this part, SNEGAN and all baseline algorithms are tested on nine LFR benchmark signed networks, and their statistical performance in prediction and reconstruction tasks over 30 independent trials is recorded in Fig. 4.

TABLE III
STATISTICS PERFORMANCE OF ALL ALGORITHM ON THE REAL-WORLD SIGNED NETWORKS. THE BEST PERFORMANCE IS MARKED IN BOLDFACE.

Algorithms	Criteria	GGS	War	Yeast	E. coli	Soc-alpha	Wiki	best/all
DeepWalk	a_u^p	0.8414	0.9245	0.7414	0.6794	0.6789	0.6073	0/36
	a_p^p	0.5000	0.5044	0.5010	0.5086	0.5000	0.5000	
	a_c^p	0.5480	0.8723	0.6052	0.5862	0.6390	0.6195	
	a_u^r	0.7412	0.9571	0.8307	0.6728	0.6783	0.6036	
	a_p^r	0.5000	0.5151	0.5130	0.5068	0.5000	0.5000	
	a_c^r	0.5360	0.8775	0.8515	0.7229	0.6660	0.6484	
Node2vec	a_u^p	0.9000	0.9024	0.7662	0.6797	0.6644	0.5985	0/36
	a_p^p	0.5000	0.5048	0.5022	0.5074	0.5000	0.5000	
	a_c^p	0.5520	0.8620	0.6034	0.5841	0.6352	0.6128	
	a_u^r	0.7268	0.9171	0.8539	0.6542	0.6692	0.5988	
	a_p^r	0.5000	0.5103	0.5147	0.5058	0.5000	0.5000	
	a_c^r	0.5200	0.8624	0.8378	0.7112	0.6617	0.6402	
GraphGAN	a_u^p	0.7167	0.8459	0.6335	0.5665	0.6416	0.5345	1/36
	a_p^p	0.6361	0.5203	0.5235	0.5096	0.6416	0.50122	
	a_c^p	0.5600	0.7636	0.6055	0.6061	0.6533	0.5871	
	a_u^r	0.5000	0.8381	0.7380	0.5880	0.6199	0.5603	
	a_p^r	0.6472	0.5213	0.5312	0.5148	0.5026	0.5021	
	a_c^r	0.4400	0.7902	**0.9093**	0.7726	0.7207	0.6983	
SNEGAN	a_u^p	**1**	**0.9655**	0.7975	**0.8188**	**0.8613**	0.8022	27/36
	a_p^p	**0.6833**	**0.5259**	**0.5355**	**0.5672**	**0.5107**	0.5316	
	a_c^p	0.6000	**0.8797**	**0.6204**	**0.6181**	**0.7093**	0.6779	
	a_u^r	**0.9333**	0.9554	**0.9731**	**0.9933**	**0.9445**	**0.9096**	
	a_p^r	0.7083	**0.5288**	**0.5537**	**0.6221**	**0.5143**	**0.5453**	
	a_c^r	**0.6400**	**0.8895**	0.9037	**0.7782**	**0.7212**	0.7118	
DNE-SBP	a_u^p	**1**	0.9645	**0.8550**	0.7878	0.8099	**0.8583**	10/36
	a_p^p	**0.6833**	0.5244	0.5299	0.5508	0.5083	**0.5391**	
	a_c^p	**0.6400**	0.8322	0.4833	0.5162	0.5490	**0.7001**	
	a_u^r	0.9218	**0.9744**	0.9449	0.9629	0.9054	0.8740	
	a_p^r	**0.7236**	0.5275	0.5492	0.6099	0.5125	0.5408	
	a_c^r	0.3600	0.8517	0.4759	0.5969	0.6031	**0.7157**	

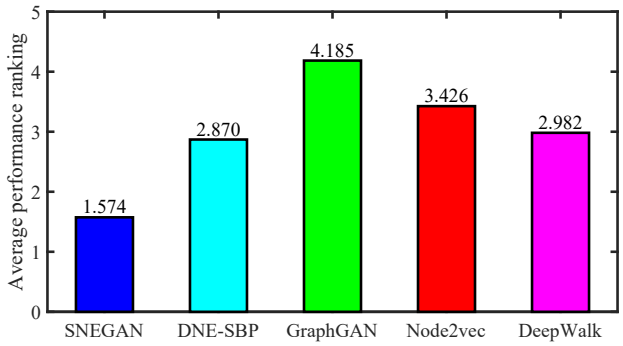


Fig. 5. Average performance ranking of the criteria for all baseline algorithms on the nine LFR benchmark signed networks.

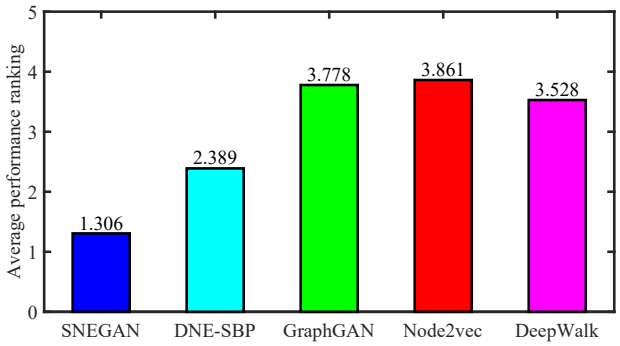


Fig. 6. Average performance ranking of the criteria for all baseline algorithms on the six real-world signed networks.

From Fig. 4, we can see that SNEGAN obtains the highest a_c^p, a_u^r and a_p^r for all LFR benchmark signed networks. This validates the superiority of SNEGAN over all baseline algorithms in the link prediction and sign reconstruction for the LFR benchmark signed networks. Moreover, for most networks, SNEGAN only has lower a_u^p and a_p^p values than DNE-SBP, and only has lower a_c^r values than Node2vec and DeepWalk. This demonstrates the competitive performance of SNEGAN in sign prediction and link reconstruction.

Fig. 4 presents that the signed NE method (DNE-SBP) has a good performance in sign prediction and reconstruction. This is because DNE-SBP uses a structural balance theory with some control parameters to further increase the distance heterogeneity between positively and negatively linked nodes. However, DNE-SBP has a worse performance in link prediction and reconstruction than unsigned NE methods (Deep-

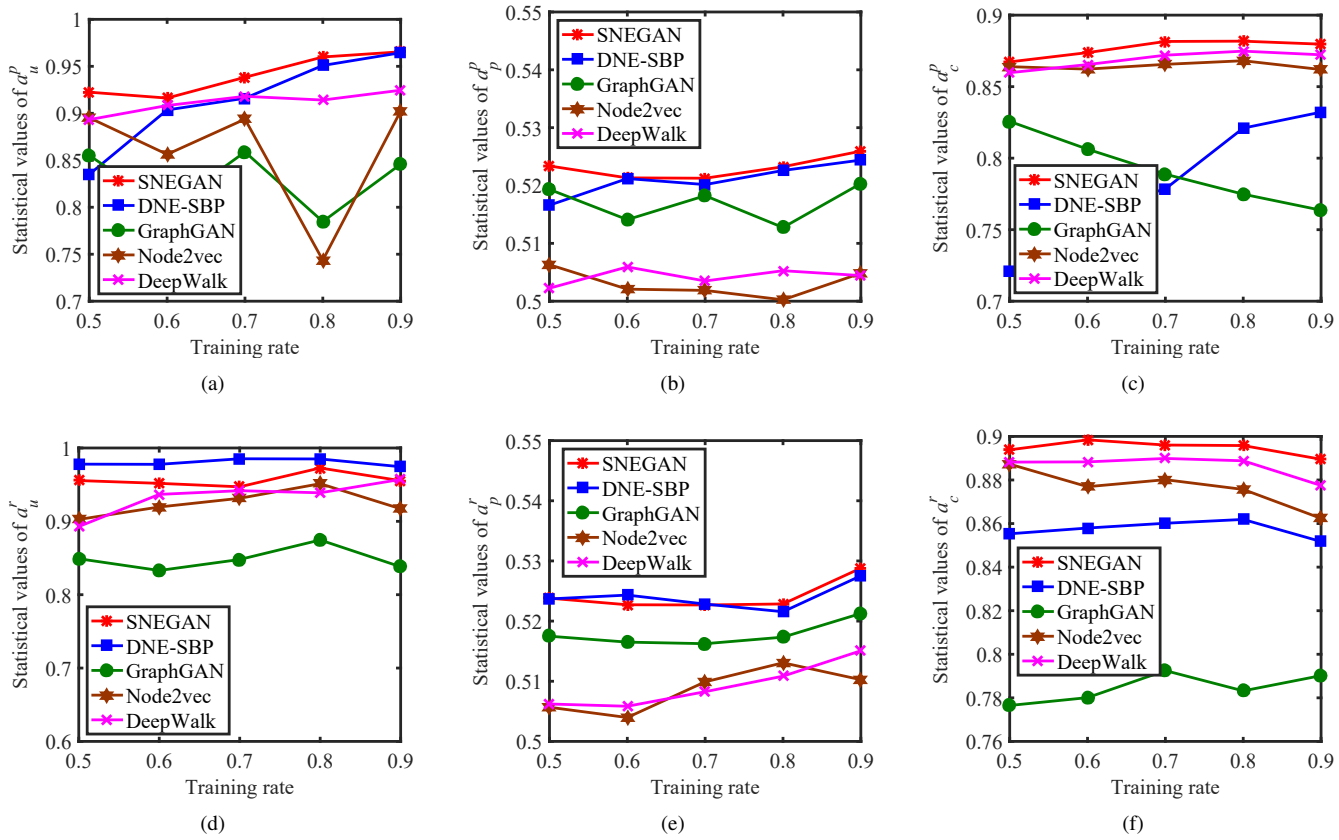


Fig. 7. Statistical results of all algorithms on the real-world War networks VS. different training ratios. (a) a_u^p, (b) a_p^p, (c) a_c^p, (d) a_u^r, (e) a_p^r and (f) a_c^r

.

Walk, Node2vec and GraphGAN). The proposed NE method (SNEGAN) performs well in both link (sign) prediction and reconstruction.

Fig. 4 also shows that all unsigned NE methods (DeepWalk, Node2vec and GraphGAN) have low values of a_u^p, a_p^p, a_u^r and a_p^r for all LFR benchmark signed networks. This validates that when neglecting conflicting relationships, classical unsigned NE methods (DeepWalk, Node2vec and GraphGAN) are difficult to tackle sign prediction and reconstruction in signed networks.

As known from Fig. 4, a_u^p, a_p^p, a_u^r and a_p^r values of the signed NE methods (SNEGAN and DNE-SBP) increase with the increase of r. This is because the number of positive (or negative) links decreases (or increases) with the increase of r, and positively linked nodes are generally closer than negatively linked nodes in the embedded low-dimensional space. In this case, a link sign S_{ij} is easily predicted and reconstructed based on the distance between nodes v_i and v_j in the learned low-dimensional space.

Fig. 5 shows the overall performance ranks of all algorithms on the LFR benchmark signed networks in terms of all criteria. For each network, all algorithms are ranked from 1 to 5 based on their performance, and the algorithm with the best performance is ranked as 1. The results show that SNEGAN has an overall performance rank (1.574) much smaller than that of DNE-SBP (2.870), GrapGAN (4.185), Node2vec (3.426) and DeepWalk (2.982). This further validates the superiority of SNEGAN over the other baseline methods when we comprehensively consider all criteria for link (sign) prediction and reconstruction tasks.

C. Experimental Results on Real-world Signed Networks

To validate the performance of SNEGAN for real systems, we test all methods on six real-world signed networks, and record their statistical results over 30 independent trials into Table III. For each network and criterion, the best result of all algorithms is marked in boldface. The results show that SNEGAN obtains the best results in 27 out of all 36 statistical values (75.00%), while DNE-SBP, GrapGAN, Node2vec and DeepWalk perform best in 10, 1, 0 and 0 cases, respectively. To quantitatively show the overall performance of SNEGAN, we also record the average performance ranks (see Fig. 6) of all algorithms in terms of a_u^p, a_p^p, a_c^p, a_u^r, a_p^r and a_c^r over all the real-world networks, in which a lower rank value corresponds to a better overall performance. The results in Fig. 6 show that the average performance rank value (1.306) of SNEGAN is much smaller than that of DNE-SBP (2.389), GrapGAN (3.778), Node2vec (3.861) and DeepWalk (3.528). Those comparisons validate the superiority of SNEGAN over all baseline methods in solving both prediction and reconstruction tasks, especially for sign prediction and reconstruction.

From Table III, we can see that for most networks, the signed NE method (DNE-SBP) and the unsigned NE methods

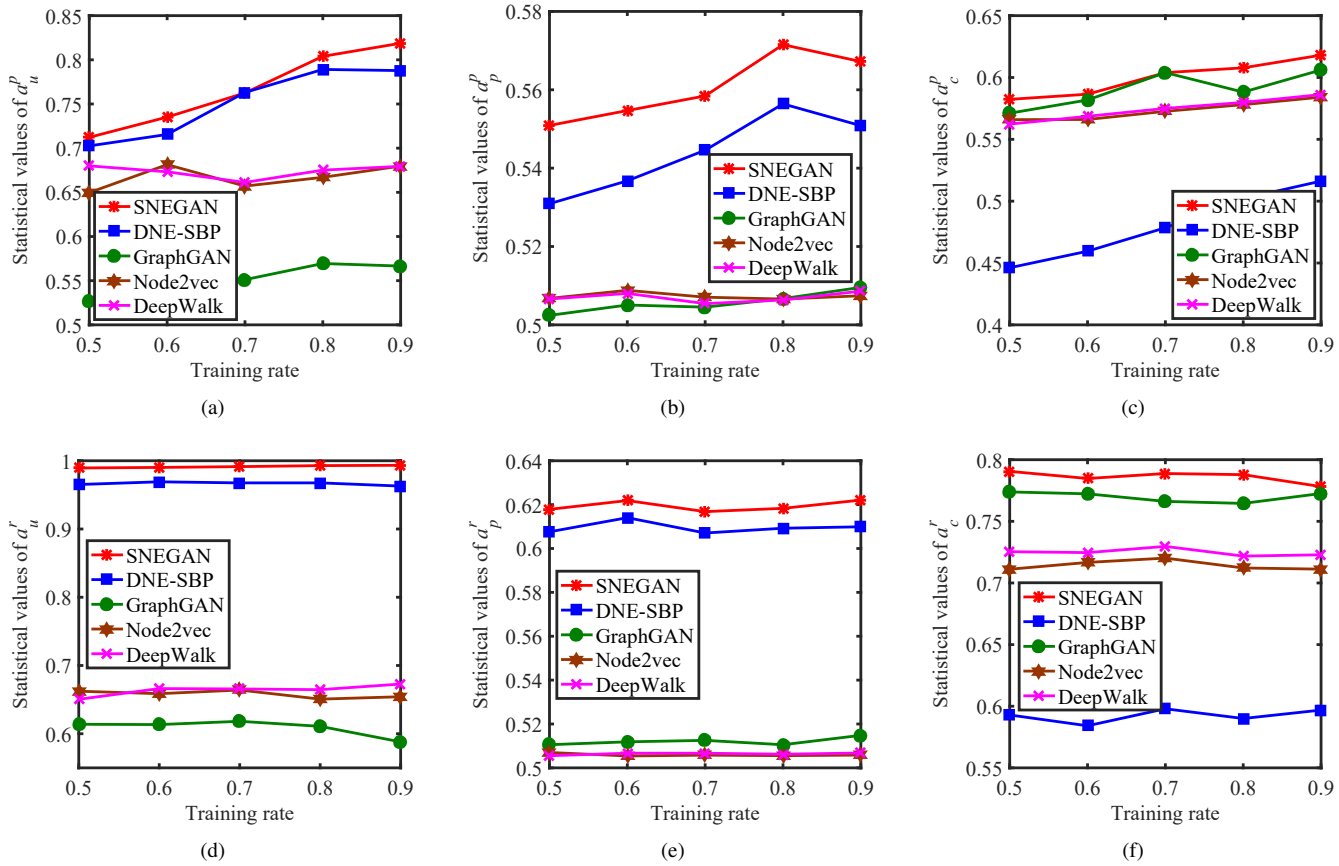


Fig. 8. Statistical results of all algorithms on the real-world E. coli network VS. different training ratios. (a) a_u^p, (b) a_p^p, (c) a_c^p, (d) a_u^r, (e) a_p^r and (f) a_c^r

.

(GraphGAN, Node2vec and DeepWalk) are hard to obtain a good performance tradeoff between sign and link prediction. More specifically, DNE-SBP facilitates sign prediction, but it has a poor performance in link prediction. Although GraphGAN, Node2vec and DeepWalk show a competitive performance in link prediction, they cannot tackle sign prediction. By combining conflicting properties with a GAN framework, SNEGAN has a good performance in sign and link prediction.

Table III also shows some similar results as Fig. 4. For the GGS, War, Yeast, Ecoli and Soc-alpha signed networks, SNEGAN outperforms DNE-SBP for sign prediction and reconstruction. This further validates the superiority of the adversarial training process of GANs over the classical dimensionality reduction techniques. For the large-scale networks, SNEGAN has a slightly worse performance than DNE-SBP for sign prediction. This is probably because DNE-SBP avoids reconstruction of many useless unlinked edges in large-scale networks by adding penalties on the reconstruction errors of nonzero elements whereas SNEGAN needs to take extra resources to detect whether there exist unlinked edges.

The results in Table III present that although they extract high-level node proximity information, the dimensionality reduction based methods (Node2vec and DeepWalk) have a worse performance than SNEGAN. This further validates the effectiveness of the GAN framework based NE methods for link prediction and reconstruction. Note that, for most networks, GraphGAN has a low perform for link prediction and reconstruction. This is because GraphGAN is sensitive to its parameter settings, and its performance will degrade after a certain number of iterations.

To further validate the performance of all algorithms, Figs. 7 and 8 record the performance of all algorithm on the real-world War and E. coli singed networks with varied training percentages. The results in Figs. 7 and 8 show that for most cases, SNEGAN obtains the best performance. Moreover, DNE-SBP facilitates sign prediction and reconstruction, but it is difficult to tackle link prediction and reconstruction. Node2vec and DeepWalk show a good performance for link prediction and reconstruction, but they have a poor performance for sign prediction and reconstruction. GraphGAN shows a poor performance for both prediction and reconstruction tasks.

Figs. 7 and 8 also shows that the sign prediction performance of SNEGAN and DNE-SBP increases with the increase of training ratios, while the link prediction of SNEGAN, Node2vec and DeepWalk increases with the increase of training ratios. The sign and link reconstruction performance of all algorithms are robust to the training ratios. This is to be expected for the reconstruction tasks, all signs and links are used for training logistic regression models that are used for reconstructing and predicting links (signs).

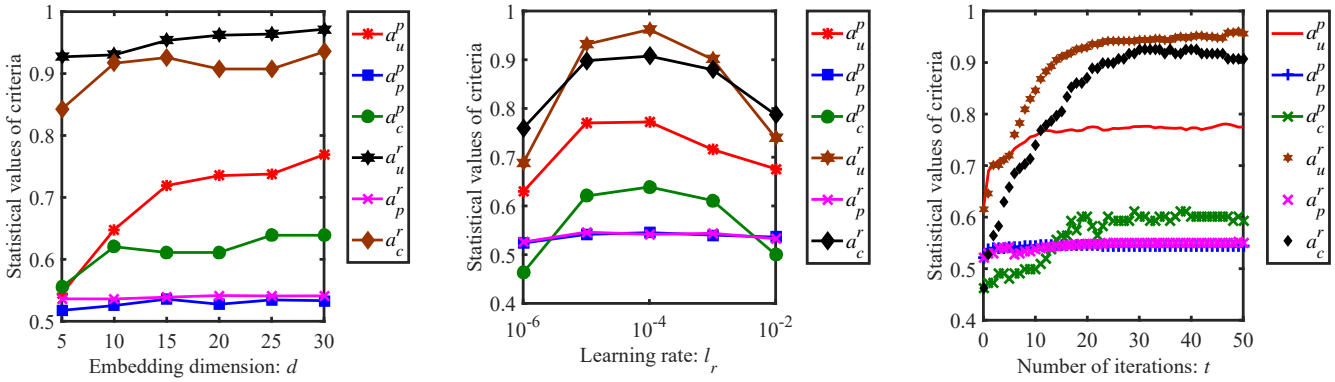


Fig. 9. Statistic values of all criteria obtained by SNEGAN on the real-world Yeast network VS. The settings of different parameters. (a) the number of embedding dimension d, (b) the learning rate l_r, and (c) the number of iterations t.

D. Effects of Parameter Settings

In this part, effects of certain parameter settings (d, l_r and t_{max}) on prediction and reconstruction performance of SNEGAN are analyzed. To show these effects, we test SNEGAN on the medium-scale biological network (Yeast) with different d, l_r and t_{max} values, and record the statistical results of all criteria over 30 independent trials in Fig. 9.

d denotes the number of embedding dimensions in the low-dimensional space, which affects the search space, convergence speed and potentially learned features of SNEGAN. Fig. 9(a) shows the variations of the prediction and reconstruction performance of SNEGAN with different d values. The results illustrate that SNEGAN obtains the best performance in both sign (link) prediction and reconstruction when $d = 20$. This is reasonable a small d setting would result in the loss of features while a large d setting would cause a huge search space. A proper d setting ($d = 20$) can facilitate both feature preservation and search space reduction.

l_r is the learning rate of the stochastic gradient ascent optimization method for the generator and discriminator, which affects both the convergence and exploration of SNEGAN. Fig. 9(b) records the performance of SNEGAN varying with different $l_r \in \{0.01, 0.001, 0.0001, 0.00001, 0.000001\}$ values. The results show that a proper l_r setting such as $l_r = 0.001, 0.0001$ and 0.00001 makes SNEGAN quickly converge to a good solution, whereas a small or large l_r setting such as $l_r = 0.01$ and 0.000001 makes SNEGAN converge to a poor solution.

To show the convergence, Fig. 9(c) records the prediction and reconstruction performance obtained by SNEGAN varying with number t of iterations. The results show that SNEGAN can quickly converge to a good solution with high values of a_u^p, a_p^p, a_c^p, a_u^r, a_p^r and a_c^r for the Yeast network in 20 iterations. This validates the convergence of SNEGAN and the proper setting $t_{max} = 20$.

V. Conclusions

In this paper, we have studied the embedding of signed networks, and have proposed a novel generative adversarial nets learning framework (called SNEGAN) for learning low-dimensional node representations and preserving links and signs of signed networks. SNEGAN first models the embedding problem in signed networks as a two-player minimax game, and then uses a generative adversarial net framework with a generator and discriminator to find an optimal solution for the game. The generator plays a minimization game, which tries to generate fake links to deceive the discriminator by using a signed random walker technique and graph softmax function, while the discriminator plays a maximization game, which aims to discriminates the truths and signs of generated links by using a tanh function. The generator and discriminator are gradually and iteratively evolved by the adversarial training with a stochastic gradient ascent optimization. Experimental results on nine LFR benchmark and six real-world signed networks have shown that SNEGAN outperforms several state-of-the-art NE algorithms (DNE-SBP, GraphGAN, Node2vec and DeepWalk) in learning low-dimensional node representations for link (sign) prediction and reconstruction tasks.

In the future, we will study NEs of signed networks with element properties such as node, edge, dynamic attributes. Moreover, inspired by the work [18]–[20], [52], [53], we will consider an evolutionary multitasking of GANs for learning low-dimensional node representations under multiple dimensions, and study the applications of SNEGAN to data representation, extensive objective optimization, classification, approximation, reconstruction and prediction. In addition, inspired by the works in [54]–[56], we will extend the proposed SNEGAN for node embeddings of sparse, noise and directed signed networks. Finally, we will consider the balance heterogeneity of the minimax game in SNEGAN.

References

[1] G. Facchetti, G. Iacono, and C. Altafini, "Computing global structural balance in large-scale signed social networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 52, pp. 20 953–20 958, 2011.

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[4] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2018.

[5] A. Chaturvedi and A. Tiwari, "System network complexity: Network evolution subgraphs of system state series," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 130–139, 2018.

[6] S. Wang, J. Liu, and Y. Jin, "Surrogate-assisted robust optimization of large-scale networks based on graph embedding," *IEEE Transactions on Evolutionary Computation*, vol. doi:10.1109/TEVC.2019.2950935, 2019.

[7] L. Ma, X. Huang, J. Li, Q. Lin, Z. You, M. Gong, and V. C. Leung, "Privacy-preserving global structural balance computation in signed networks," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 164–177, 2019.

[8] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning." in *Twenty-fivth International Joint Conference on Artificial Intelligence*, 2016, pp. 2252–2258.

[9] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-first AAAI Conference on Artificial Intelligence*, 2017, pp. 203–209.

[10] M. Gong, C. Chen, Y. Xie, and S. Wang, "Community preserving network embedding based on memetic algorithm," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 108–118, 2018.

[11] M. Li, J. Liu, P. Wu, and X. Teng, "Evolutionary network embedding preserving both local proximity and community structure," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 523–535, 2020.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[13] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[14] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2017, pp. 1024–1034.

[16] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, "High-order proximity preserved embedding for dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, pp. 2134–2144, 2018.

[17] L. Ma, Y. Zhang, J. Li, Q. Lin, Q. Bao, S. Wang, and M. Gong, "Community-aware dynamic network embedding by using deep autoencoder," *Information Sciences*, vol. 519, pp. 22–42, 2020.

[18] C. Leng, H. Zhang, G. Cai, I. Cheng, and A. Basu, "Graph regularized lp smooth non-negative matrix factorization for data representation," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 584–595, 2019.

[19] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, "Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, 2019.

[20] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2018.

[21] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Thirty-second AAAI Conference on Artificial Intelligence*, 2018, pp. 2508–2515.

[22] Z. Pan, W. Yu, B. Wang, H. Xie, V. S. Sheng, J. Lei, and S. Kwong, "Loss functions of generative adversarial networks (gans): Opportunities and challenges," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. doi:10.1109/TETCI.2020.2991774, 2020.

[23] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Proceedings of the 29th International Conference on Neural Information Processing Systems*, 2015, pp. 1486–1494.

[24] M. Zhou, Y. Lin, N. Zhao, Q. Jiang, X. Yang, and Z. Tian, "Indoor wlan intelligent target intrusion sensing using ray-aided generative adversarial network," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, pp. 61–73, 2020.

[25] W.-C. Huang, H. Luo, H.-T. Hwang, C.-C. Lo, Y.-H. Peng, Y. Tsao, and H.-M. Wang, "Unsupervised representation disentanglement using cross domain features and adversarial learning in variational autoencoder based voice conversion," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. doi:10.1109/TETCI.2020.2977678, 2020.

[26] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 921–934, 2019.

[27] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (gans)," *IEEE Transactions on Cybernetics*, vol. doi:10.1109/TCYB.2020.2985081, 2020.

[28] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Proceedings of the 29th International Conference on Neural Information Processing Systems*, 2016, pp. 82–90.

[29] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "Communitygan: Community detection with generative adversarial nets," in *proceedings of the 28th International Conference on World Wide Web*, 2019, pp. 784–794.

[30] Y. Sun, S. Wang, T.-Y. Hsieh, X. Tang, and V. Honavar, "Megan: A generative adversarial network for multi-view network embedding," *arXiv preprint arXiv:1909.01084*, 2019.

[31] M. Ding, J. Tang, and J. Zhang, "Semi-supervised learning on graphs with generative adversarial nets," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 913–922.

[32] B. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 120–129.

[33] Y. Xiong, Y. Zhang, H. Fu, W. Wang, Y. Zhu, and S. Y. Philip, "Dyngraphgan: Dynamic graph embedding via generative adversarial networks," in *24th International Conference on Database Systems for Advanced Applications*, 2019, pp. 536–552.

[34] L. Ma, J. Li, Q. Lin, M. Gong, C. A. C. Coello, and Z. Ming, "Cost-aware robust control of signed networks by using a memetic algorithm," *IEEE Transactions on Cybernetics*, vol. doi:10.1109/TCYB.2019.2932996, 2020.

[35] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017, pp. 327–335.

[36] M. R. Islam, B. A. Prakash, and N. Ramakrishnan, "Signet: Scalable embeddings for signed networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2018, pp. 157–169.

[37] X. Shen and F.-L. Chung, "Deep network embedding for graph representation learning in signed networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1556–1568, 2020.

[38] S. Wang, C. Aggarwal, J. Tang, and H. Liu, "Attributed signed network embedding," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 137–146.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[40] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 515–524.

[41] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015, pp. 3528–3536.

[42] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 2852–2858.

[43] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[44] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016118, 2009.

[45] C. Liu, J. Liu, and Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social network-

s," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2274–2287, 2014.
[46] K. E. Read, "Cultures of the central highlands, new guinea," *Southwestern Journal of Anthropology*, vol. 10, no. 1, pp. 1–43, 1954.
[47] F. Ghosn, G. Palmer, and S. A. Bremer, "The mid3 data set, 1993–2001: Procedures, coding rules, and description," *Conflict Management and Peace Science*, vol. 21, no. 2, pp. 133–154, 2004.
[48] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
[49] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 1361–1370.
[50] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.
[51] M. D. Power, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.
[52] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.
[53] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.
[54] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 524–537, 2015.
[55] X. Shi, Q. He, X. Luo, Y. Bai, and M. Shang, "Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems," *IEEE Transactions on Big Data*, p. doi10.1109/TBDATA.2020.2973141, 2020.
[56] X. Luo, M. Zhou, S. Li, L. Hu, and M. Shang, "Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 1844–1855, 2019.

Lijia Ma received the B.S. degree in communication engineering from Hunan Normal University, Changsha, China, and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2010 and 2015, respectively.

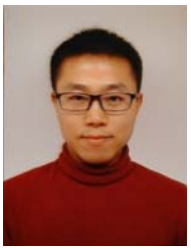
From 2015 to 2016, he was a Postdoctoral Fellow with Hong Kong Baptist University, Hong Kong, and with Nanyang Technological University, Singapore, from 2016 to 2017. He is an assistant professor at the College of Computer and Software Engineering of Shenzhen University. His research interests mainly include evolutionary computation, machine learning and complex networks.

Yuchun Ma received the B.S. degree in internet of things engineering from Shenzhen University, Shenzhen, China, in 2020. He is currently pursuing the M.S. degree in information technology with the Department of Computing, Hong Kong Polytechnic University, Hong Kong.

His current research interests include graph representation learning, machine learning and complex networks.

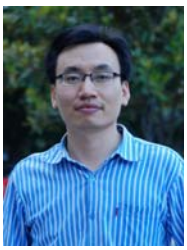
Qiuzhen Lin (M'18) received the B.S. degree from Zhaoqing University and the M.S. degree from Shenzhen University, China, in 2007 and 2010, respectively. He received the Ph.D. degree from Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong, in 2014.

He is currently an associate professor in College of Computer Science and Software Engineering, Shenzhen University. His current research interests include artificial immune system, multi-objective optimization and dynamic system.



Junkai received the B.S. degree from Hefei University of technology, Anhui, China, in 2013, and an M.S. degree and a D.E. degree from University of Toyama, Toyama, Japan, in 2016 and 2018, respectively. In 2019, he joined Shenzhen University, Shenzhen, China, where he is currently a Research Associate in the College of Computer Science and Software Engineering. His current research interests include neural networks, evolutionary computation and bioinformatics.

Carlos A. Coello Coello (M'98-SM'04-F'11) received PhD degree in computer science from Tulane University, USA, in 1996. He is currently Professor (CINVESTAV-3F Researcher) at the Computer Science Department of CINVESTAV-IPN, in Mexico City, Mxico. Dr. Coello has authored and co-authored over 450 technical papers and book chapters. He has also co-authored the book *Evolutionary Algorithms for Solving Multi-Objective Problems* (Second Edition, Springer, 2007). His publications report over 52,000 citations in Google Scholar (his h-index is 92). Currently, he is associate editor of the *IEEE Transactions on Evolutionary Computation* and serves in the editorial board of 12 other international journals. His major research interests are: evolutionary multi-objective optimization and constraint-handling techniques for evolutionary algorithms. He received the *2007 National Research Award* from the Mexican Academy of Sciences in the area of Exact Sciences, the *2013 IEEE Kiyo Tomiyasu Award* and the *2012 National Medal of Science and Arts* in the area of *Physical, Mathematical and Natural Sciences*. He is a Fellow of the IEEE, and a member of the ACM, Sigma Xi, and the Mexican Academy of Science.



Maoguo Gong (M'07-SM'14) received the B.S. degree and Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2003 and 2009, respectively.

Since 2006, he has been a Teacher with Xidian University. In 2008 and 2010, he was promoted as an Associate Professor and as a Full Professor, respectively, both with exceptive admission. His research interests are in the area of computational intelligence with applications to optimization, learning, data mining and image understanding.

Dr. Gong received the prestigious National Program for the support of Top-Notch Young Professionals from the Central Organization Department of China, the Excellent Young Scientist Foundation from the National Natural Science Foundation of China, and the New Century Excellent Talent in University from the Ministry of Education of China. He is the Vice Chair of the IEEE Computational Intelligence Society Task Force on Memetic Computing, an Executive Committee Member of the Chinese Association for Artificial Intelligence, and a Senior Member of the Chinese Computer Federation. He is also the associate editor of *IEEE Trans. Evolutionary Computation*.