Directed Network Embedding with Virtual Negative Edges

Hyunsik Yoo*
Hanyang University
Seoul, Korea
hsyoo32@hanyang.ac.kr

Yeon-Chang Lee*
Hanyang University
Seoul, Korea
lyc0324@hanyang.ac.kr

Kijung Shin
KAIST
Seoul, Korea
kijungs@kaist.ac.kr

Sang-Wook Kim†
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

ABSTRACT

The *directed network embedding* problem is to represent the nodes in a given directed network as *embeddings (i.e., low-dimensional vectors)* that preserve the asymmetric relationships between nodes. While a number of approaches have been developed for this problem, we point out that existing approaches commonly face difficulties in accurately preserving asymmetric proximities between nodes in a *sparse network* containing a large number of low out- and in-degree nodes. In this paper, we focus on addressing this intrinsic difficulty caused by the lack of information. We first introduce the concept of *virtual negative edges* (VNEs), which represent latent negative relationships between nodes. Based on the concept, we propose a novel DIrected NE approach with VIrtual Negative Edges, named as **DIVINE**. DIVINE carefully decides the number and locations of VNEs to be added to the input network. Once VNEs are added, DIVINE learns embeddings by exploiting both the signs and directions of edges. Our experiments on four real-world directed networks demonstrate that adding VNEs alleviates the lack of information about low-degree nodes, thereby enabling DIVINE to yield high-quality embeddings that accurately capture asymmetric proximities between nodes. Specifically, the embeddings obtained by DIVINE lead to up to **10.16% more accurate** link prediction, compared to those obtained by state-of-the-art competitors. All DIVINE code are available at: https://github.com/hsyoo32/divine.

CCS CONCEPTS

• **Information systems** → **Social networks**.

KEYWORDS

network embedding; directed networks; virtual negative edges

ACM Reference Format:
Hyunsik Yoo, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2022. Directed Network Embedding with Virtual Negative Edges. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498470

*Two first authors have contributed equally to this work.
†Corresponding author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498470

1 INTRODUCTION

Background. *Network embedding* (NE) aims to represent the nodes in a given network by vectors in a low-dimensional embedding space so that the vectors, which are called *embeddings*, preserve structural properties, such as proximity in the network [5]. Literature [6, 25, 32] has shown that the low-dimensional vectors can be used as informative features of nodes in solving various information-retrieval tasks – *e.g.*, link prediction [21] and node classification [42]. In recent studies, additional information, such as edge directions [22, 41], edge signs [12, 37], and node attributes [20, 39], has been incorporated to improve the accuracy of NE.

In this paper, we focus on NE that utilizes *edge directions*. For example, on Instagram, even if a user i follows an influencer j, j may not follow i. This *asymmetric relationship* can be expressed as a directed edge from i to j but not from j to i. In order to accurately capture such asymmetric relationships and furthermore asymmetric proximities [41] between nodes, various *directed NE* methods [11, 22, 27, 30, 41] have been proposed. For each directed edge from i to j, most directed NE methods distinguish the *source node* i and the *target node* j according to their roles in the edge. Then, for each node, the methods learn a *source embedding* and a *target embedding*, which preserve the node's properties as sources and targets, respectively. The source and target embeddings of nodes can be used as inputs in various downstream tasks.

Most existing directed NE methods can be categorized into (1) *matrix factorization* (MF)-based methods, (2) *deep learning* (DL)-based methods, and (3) *random walk* (RW)-based methods, as suggested in [5, 38]. Specifically, MF-based [22, 30] and DL-based [27] methods first represent the asymmetric proximities between all pairs of source and target nodes in the form of a matrix. Then, they seek embeddings that approximate asymmetric proximities between nodes by using MF techniques (*e.g.*, singular value decomposition [23]) or DL techniques (*e.g.*, graph autoencoder [13]). On the other hand, for each node, RW-based methods [11, 41] sample a number of nodes visited during RWs from the node as a seed. Then, they seek embeddings that maximize relative asymmetric proximities between each seed node and the sampled nodes.

Motivation. In this paper, we show that the *sparsity* of real-world networks makes it difficult for existing directed NE methods to accurately preserve asymmetric proximities. It is well known that the *density*, which is defined as the ratio of the number of (directed) edges with respect to the maximum possible (directed) edges, is extremely low in many real-world networks. To make matters more challenging, they have *power-law degree distributions*. That is, most nodes in them have extremely low out- and in-degrees.

As a result, in MF-based and DL-based methods, many rows and columns that correspond to low out-degree and in-degree nodes,

respectively, are filled mostly with 0 values. In RW-based methods, RWs from low out-degree nodes tend to visit relatively a small number of unique nodes; and low in-degree nodes are hardly sampled during RWs. Therefore, the three NE approaches easily fail to capture the properties of low out- and in-degree nodes as sources and targets, respectively. Surprisingly, in real-world directed networks, a considerable fraction of nodes have a zero out- and in-degree. For instance, in the four datasets considered in this work, the average ratios of nodes with a zero out- and in-degree are 34.86% and 34.29%, respectively (refer to Table 2 in Section 4). Such zero out- and in-degree nodes aggravate the aforementioned challenges.

Our Ideas. The intrinsic difficulty of the directed NE problem is *the lack of information* for understanding the properties of low out- and in-degree nodes as sources and targets, respectively, in a sparse network. Regarding this difficulty, we note that literature has shown that exploiting not only positive edges (*e.g.*, trust or friend) but also negative edges (*e.g.*, distrust or foe) helps to improve the accuracy of various network analysis tasks, including NE [16, 17, 19, 31]. From these earlier studies, we can infer that exploiting both two types of relationships between nodes, which often convey different meanings and information [16], helps to address the lack of information. Unfortunately, however, most real-world directed networks do not have explicit negative edges.

In this work, we aim to answer three relevant research questions: (1) *Can we infer negative edges and use them as a source of information?* (2) *Do such inferred negative edges help accurate embedding of low-degree nodes?* (3) *How can we maximize the accuracy gain?*

To this end, we propose a novel DIrected NE approach with VIrtual Negative Edges, named as **DIVINE**, which effectively preserves asymmetric proximities between nodes based on three steps: (1) selecting the number and locations of *virtual negative edges* (VNEs), which represent latent negative relationships between nodes in the input network; (2) adding VNEs to the input network; (3) performing NE by exploiting both signs and directions of edges.

The key challenge in designing DIVINE is to selectively choose VNEs from a large number of potential ones (*i.e.*, non-existent edges) in the (original) input network. To address this challenge, we first infer the degree of (latent) negativity of i with respect to j for each pair of nodes i and j from existent (positive) edges (Section 3.2). Next, for each node i, we find a proper number of nodes with respect to which i has the highest degree of negativity; then, we add VNEs from i to such nodes to the input network (Sections 3.3 and 3.4). We here deal with the issue of determining the number of VNEs from each node based on *structural balance* [1, 4, 9], which is a well-known property of signed networks.

Once VNEs are added, we can learn the source and target embeddings of nodes by employing **any** signed NE methods [12, 18, 36, 37], which are designed to be effective for signed directed networks (Section 3.5). We show experimentally that adding VNEs, carefully as proposed, effectively alleviates the lack of information about low-degree nodes. Specifically, we show that, by exploiting information in the form of VNEs, existing signed NE methods capture asymmetric proximities between nodes in the input directed network significantly more accurately.

Why Virtual Negative Edges? Since virtual edges (VEs) are inferred only from the topology of a given input network, VEs do not bring any extra information beyond the network. However, adding VEs facilitates the utilization of information expressed in the form of VEs, and as a result, adding VEs has been proven useful for various graph mining tasks [15, 40].

Previous studies focused on positive edges (VPEs), and our approach DIVINE can easily be extended to VPEs. In our preliminary experiments, while adding VPEs improved the quality of directed NE in terms of link-prediction accuracy, the accuracy gain was larger when VNEs were added instead. The results show that VNEs provide information more useful to directed NE methods than VPEs; furthermore, we suspect that the information inherent in VNEs is more difficult for directed NE methods to utilize (than that in VPEs) unless it is explicitly provided in the form of VEs.

Contributions. Our contributions are summarized as follows:

- **Novel Solution to the Lack of Information:** We propose the idea of exploiting latent negative relationships between nodes, *i.e.*, VNEs, to mitigate the lack of information when embedding low-degree nodes in a sparse directed network.
- **Effective Algorithm:** We propose DIVINE, which carefully decides the number and locations of VNEs, which it then exploits for effective directed NE. The embeddings obtained by DIVINE lead to up to **10.16% more accurate** link prediction, compared to those obtained by state-of-the-art competitors.
- **General Methodology:** We propose a methodology that can be equipped with **any** signed NE methods. Thus, if better signed NE methods become available, they can be used without any modification to improve the performance of DIVINE.
- **Extensive Experiments:** We validate the effectiveness of DIVINE by comparing it with 9 competitors in 3 types of link prediction tasks on 4 real-world datasets. We also validate the benefit of our design choices through extensive ablation studies.

Organization. In Section 2, we review previous studies on NE. In Section 3, we present our proposed approach in detail. In Section 4, we validate the effectiveness of the proposed approach through extensive experiments. Finally, we conclude the paper in Section 5.

2 RELATED WORK

In this section, we briefly review existing approaches for NE of undirected and directed networks. These approaches can be categorized into (1) *matrix factorization* (MF)-based methods, (2) *deep learning* (DL)-based methods, and (3) *random walk* (RW)-based methods.

Undirected NE. First, MF-based methods, such as GraRep [3] and M-NMF [35], build a proximity matrix that represents the proximities of all pairs of nodes. Then, they factorize the proximity matrix into a product of low-rank matrices, which correspond to embeddings of nodes. Next, DL-based methods, such as SDNE [34] and GraphSAGE [7], employ deep neural network models (*e.g.*, *deep autoencoder* [26] and *graph convolutional networks* (GCN) [14]) as encoders that map the input network and node attributes to embeddings of nodes. Lastly, RW-based methods, such as DeepWalk [25], Node2Vec [6], and LINE [32], sample a number of nodes visited during RWs from each seed node. Then, they obtain embeddings of nodes by maximizing the proximities between each seed node and the sampled nodes while minimizing those between the seed node and randomly chosen nodes.

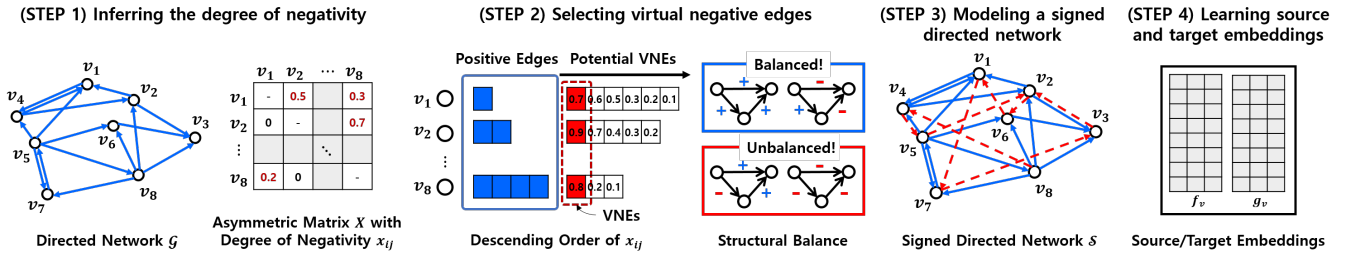


Figure 1: Overview of DIVINE, which consists of four steps: (1) inferring the degree of negativity; (2) selecting VNEs based on the degree of negativity; (3) modeling a signed directed network; (4) learning source and target embeddings of nodes.

Directed NE. However, none of the above-mentioned methods are designed to consider edge directions; thus, the embeddings provided by them do not accurately preserve asymmetric proximities between nodes. To address this limitation, a number of approaches have been proposed. For each node, they learn separately a source embedding and a target embedding, which are vectors that preserve the node's properties as sources and targets, respectively.

First, HOPE [22] and ATP [30] extend MF-based undirected NE methods; and GravityAE/VAE [27] and DiGCN [33] extend DL-based ones. They are based on different asymmetric proximity scores. Specifically, HOPE directly employs a well-known proximity measurement (*e.g.*, Katz measure [10] and Rooted PageRank [29]), while ATP is based on a new measurement that captures both the hierarchy and reachability between nodes in the network. Then, both HOPE and ATP build a proximity matrix where each (i, j)-th entry indicates an asymmetric proximity score from i to j, while GravityAE/VAE and DiGCN use the asymmetric adjacency matrix of the input network. Then, they obtain source and target embeddings of nodes, which the asymmetric proximities are approximated from, by using MF techniques (*e.g.*, singular value decomposition [23]) or DL techniques (*e.g.*, graph autoencoders [13, 34] and GCN [14]).

Next, APP [41] and NERD [11] extend the RW-based undirected NE methods. They employ different RW strategies to sample a number of positive nodes close to each seed node while taking edge directions into consideration. Specifically, APP employs an RW strategy that starts from a seed node and then follows out-going edges randomly. On the other hand, NERD proposes an alternating RW strategy that starts from a seed node and follows out-going and in-coming edges alternately. APP and NERD also sample negative nodes for each seed node uniformly at random. Then, both methods update the source embedding of each seed node and the target embeddings of the sampled positive and negative nodes so that the proximities between the seed node and the positive nodes surpass those between the seed node and the negative nodes. Note that, by the RW strategy of NERD, low out- and in-degree nodes are sampled mostly as targets and sources, respectively, but rarely as sources and targets, respectively. Thus, NERD still fails to accurately capture the properties of low out- and in-degree nodes as sources and targets, respectively, as discussed in Section 1.

All the aforementioned directed NE methods, including NERD, face difficulties in correctly preserving asymmetric proximities in a sparse network with many low out- and in-degree nodes.

3 DIVINE: PROPOSED APPROACH

In this section, we propose DIVINE, a novel directed NE approach based on virtual negative edges (VNEs). We first formulate the problem of directed NE and present an overview of DIVINE. Then, we describe each step of DIVINE in detail.

Table 1: Notations used in this paper

Notation	Description
$\mathcal{G}$	Original directed network only with positive edges
$\mathcal{S}$	Signed directed network with positive edges and VNEs
$\mathcal{V}$, $\mathcal{E}^+$, $\mathcal{E}^-$	Sets of nodes, positive edges and VNEs
$\lvert\mathcal{V}\rvert$, $\lvert\mathcal{E}^+\rvert$, $\lvert\mathcal{E}^-\rvert$	Numbers of nodes, positive edges and VNEs
θ	Parameter for determining the number of VNEs
$\mathbf{A} = (a_{ij})_{n\times n}$	Asymmetric adjacency matrix of $\mathcal{G}$
$\mathbf{X} = (x_{ij})_{n\times n}$	Asymmetric matrix whose entry x_{ij} represents the degree of negativity of v_i with respect to v_j
$\mathbf{f}_{v_i}, \mathbf{g}_{v_i} \in \mathbb{R}^d$	d-dimensional source and target embeddings of v_i

3.1 Overview

The directed NE problem is formulated as follows: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given directed network, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ denotes the set of n nodes and $\mathcal{E}$ denotes the set of directed edges. Let $e_{ij} \in \mathcal{E}$ be the directed edge from v_i (*i.e.*, source) to v_j (*i.e.*, target). Directed NE methods aim to learn source/target embedding functions $\mathbf{f}, \mathbf{g} : \mathcal{V} \to \mathbb{R}^d$ which map each node $v_i \in \mathcal{V}$ to d-dimensional source/target embeddings so that asymmetric proximities between nodes in $\mathcal{G}$ are preserved. Table 1 summarizes a list of notations.

Now, we present an overview of our DIVINE approach. As shown in Figure 1, DIVINE consists of four steps: (Step 1) inferring the degree of negativity; (Step 2) selecting VNEs based on the degree of negativity; (Step 3) modeling a signed directed network; (Step 4) learning source and target embeddings of nodes. In Step 1, from the existent (positive) edges in $\mathcal{G}$, we infer the degree of negativity x_{ij} of v_i with respect to v_j for each pair of nodes v_i and v_j. In Step 2, we select VNEs from each node v_i based on each x_{ij}, i.e., v_i's degree of negativity for each of other nodes, v_j. In Step 3, after determining the number of VNEs from each node, based on the *structural balance* [1, 4, 9] of signed directed networks, we add VNEs to $\mathcal{G}$ and model it as a signed directed network $\mathcal{S} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ denotes the set of n nodes and $\mathcal{E}^+$ and $\mathcal{E}^-$ denote the sets of directed positive edges and directed VNEs, respectively. We let $e^+_{ij} \in \mathcal{E}^+$ and $e^-_{ij} \in \mathcal{E}^-$ be a directed positive edge and a VNE from v_i (*i.e.*, source) to v_j (*i.e.*, target), respectively. Note that $\mathcal{E}^+ \cap \mathcal{E}^- = \emptyset$; that is, a node pair cannot have both a positive edge and a VNE simultaneously. In Step 4, we learn the source and target embeddings of each node v_i while exploiting both signs and directions of edges in $\mathcal{S}$.

As a demonstration of our key challenge, Figure 2 shows the number of existent edges and non-existent edges in real-world directed networks. We observe that the number of non-existent edges is quite large – *i.e.*, 39M, 50M, 37M, 534M for GNU, Wiki-Vote, JUNG, EAT, respectively. As such, it is very challenging to

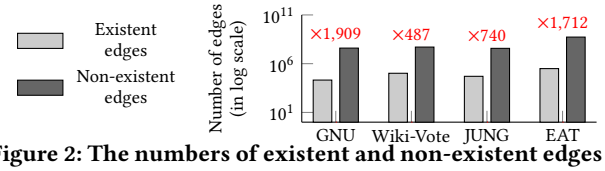


Figure 2: The numbers of existent and non-existent edges in real-world networks.

selectively create a *small number of VNEs* among a large number of non-existent edges. In DIVINE, Steps 1, 2, and 3 address this challenge. Then, in Step 4, we learn the embeddings of nodes by employing off-the-shelf NE methods designed to be effective for signed directed networks.

3.2 Inferring the degree of negativity

To quantify the degree of negativity of all pairs of nodes, we first regard each existent edge $e_{ij} \in \mathcal{E}$ in the input network as a positive edge $e^+_{ij} \in \mathcal{E}^+$. From these positive edges, we infer the degree of positivity $\hat{a}_{ij}$ of a node v_i with respect to every other node v_j. Finally, we consider that the lower $\hat{a}_{ij}$ of v_i with respect to v_j is, the higher the degree of negativity x_{ij} of v_i with respect to v_j is.

In order to infer the degree of positivity, we employ *weighted regularized matrix factorization* (WRMF) [24]. For WRMF, we use the (asymmetric) adjacency matrix $\mathbf{A} = (a_{ij})_{n\times n}$ of a given directed network $\mathcal{G}$, where each entry a_{ij} indicates whether the corresponding directed edge e_{ij} exists (*i.e.*, $a_{ij} = 1$) or not (*i.e.*, $a_{ij} = 0$). Then, we construct a weight matrix $\mathbf{W} = (w_{ij})_{n\times n}$ whose entries indicate the relative confidence in the corresponding entries of $\mathbf{A}$. For each existent edge e_{ij}, which we directly observe, we assign the highest value of 1 to the corresponding entry of $\mathbf{W}$ (i.e., if $a_{ij} = 1$, then $w_{ij} = 1$). For all non-existent edges, we assign an equal weight less than 1 to the corresponding entries of $\mathbf{W}$. [1]

Then, we factorize $\mathbf{A}$ into the product of two low-rank matrices $\mathbf{P}$ and $\mathbf{Q}$ by performing *weighted alternating least squares* using $\mathbf{W}$ as weights. Specifically, to obtain $\mathbf{P}$ and $\mathbf{Q}$, which are latent factors representing the properties of nodes as sources and targets, respectively, we aim to minimize the following objective function:

$$\mathcal{L}(\mathbf{P},\mathbf{Q}) = \sum_{i,j} w_{ij}\{(a_{ij} - \mathbf{P}_{i(\cdot)}(\mathbf{Q}_{j(\cdot)})^\top)^2 + \lambda(\|\mathbf{P}_{i(\cdot)}\|_F^2 + \|\mathbf{Q}_{j(\cdot)}\|_F^2)\},$$

where $\mathbf{P}_{i(\cdot)}$ and $\mathbf{Q}_{j(\cdot)}$ indicate the i-th row of $\mathbf{P}$, and the j-th row of $\mathbf{Q}$, respectively. In addition, $\|(\cdot)\|_F$ denotes the *Frobenius norm* and λ is a regularization parameter for preventing overfitting. In order to factorize $\mathbf{A}$, WRMF first assigns random values to the entries of $\mathbf{Q}$, and updates the entries of $\mathbf{P}$ as follows, $\forall 1 \le i \le n$:

$$\mathbf{P}_{i(\cdot)} = \mathbf{A}_{i(\cdot)}\widetilde{\mathbf{W}}_{i(\cdot)}\mathbf{Q}\{\mathbf{Q}^\top\widetilde{\mathbf{W}}_{i(\cdot)}\mathbf{Q} + \lambda(\sum\nolimits_j w_{ij})\mathbf{I}\}^{-1}, \tag{1}$$

where $\widetilde{\mathbf{W}}_{i(\cdot)}$ is a diagonal matrix with the entries of $\mathbf{W}_{i(\cdot)}$ on the diagonal, and $\mathbf{I}$ is an identity matrix. After that, WRMF updates the entries of $\mathbf{Q}$ while fixing $\mathbf{P}$ as follows, $\forall 1 \le j \le n$:

$$\mathbf{Q}_{j(\cdot)} = (\mathbf{A}_{(\cdot)j})^\top\widetilde{\mathbf{W}}_{(\cdot)j}\mathbf{P}\{\mathbf{P}^\top\widetilde{\mathbf{W}}_{(\cdot)j}\mathbf{P} + \lambda(\sum\nolimits_i w_{ij})\mathbf{I}\}^{-1}. \tag{2}$$

We reduce the objective function by repeatedly updating $\mathbf{P}$ and $\mathbf{Q}$ using Eq. (1) and Eq. (2) computing both $\mathbf{P}_{i(\cdot)}$ and $\mathbf{Q}_{j(\cdot)}$ until $\mathbf{P}$ and $\mathbf{Q}$ converge. Then, we compute the matrix $\hat{\mathbf{A}} = (\hat{a}_{ij})_{n\times n}$, which approximates $\mathbf{A}$, by multiplying $\mathbf{P}$ and $\mathbf{Q}$ (*i.e.*, $\mathbf{A} \approx \hat{\mathbf{A}} = \mathbf{P}\mathbf{Q}^\top$). Here, each entry $\hat{a}_{ij}$ of $\hat{\mathbf{A}}$ indicates the degree of positivity of v_i with respect to v_j. Lastly, for each node pair v_i and v_j, we compute the degree of negativity of v_i with respect to v_j by as follows:

$$x_{ij} = 1 - \frac{\hat{a}_{ij} - \|\hat{\mathbf{A}}\|_{\min}}{\|\hat{\mathbf{A}}\|_{\max} - \|\hat{\mathbf{A}}\|_{\min}}. \tag{3}$$

Note that the degree of negativity is not necessarily symmetric. That is, for each node pair v_i and v_j, x_{ij} and x_{ji} can be different.

To infer the degree of negativity, we can also exploit any existing methods (*e.g.*, RWR [28], directed NE methods) that are designed to consider edge directions, instead of WRMF. However, we found that, compared to five competitors, WRMF performs better or at least comparably; we will discuss the results in detail in Section 4.2.

3.3 Selecting virtual negative edges

In this subsection, we discuss selecting VNEs among a large number of non-existent edges, based on their degree of negativity. To this end, we can consider two strategies: (1) global selection and (2) local selection. The *global selection* strategy is to select VNEs with high degrees of negativity among *all* potential ones. while the *local selection* strategy is to select incident VNEs for *each* node.

First, the global selection strategy sorts all non-existent edges in descending order of their degree of negativity (*i.e.*, Eq. (3)). Then, it selects a pre-defined number of VNEs sequentially from the beginning of the sorted list. This strategy may select a large fraction of VNEs among non-existent edges between high-degree nodes, failing to address the lack of information about low-degree nodes.

On the other hand, the local selection strategy selects an equal number of VNEs per source node. Specifically, for each node v_i, the local selection strategy sorts all non-existent edges from v_i in descending order of their degree of negativity, and then it selects a pre-defined number of VNEs sequentially from the beginning of the sorted list. By selecting an equal number of VNEs from each node, the local selection strategy guarantees that every node has an equal opportunity to express its latent negative relationships.

We validate empirically in Section 4.2 that the local selection strategy addresses the sparsity problem more effectively than (1) the global selection strategy and (2) two variants of the local selection strategy where the number of VNEs from each node is proportional and inversely proportional, respectively, to its out-degree. In addition, we confirmed in our preliminary experiments that selecting an equal number of VNEs per source node, as proposed, is more effective than (1) selecting an equal number of VNEs per target node and (2) mixing the two strategies.

3.4 Modeling a signed directed network

In this subsection, we determine the total number of VNEs to be added and then build a signed directed network composed of both the VNEs and the existent postive edges.

First, we formulate an equation for determining the total number of VNEs. Since we are adding the VNEs to the given (unsigned) network $\mathcal{G}$, we determine the total number of VNEs, $|\mathcal{E}^-|$, by considering the total number of (positive) edges, $|\mathcal{E}^+|$, in $\mathcal{G}$. That is, we set $|\mathcal{E}^-|$ to be θ times the number of positive edges (*i.e.*, $|\mathcal{E}^-| = |\mathcal{E}^+|\times\theta$), and θ is a parameter that determines $|\mathcal{E}^-|$. Intuitively, it is natural to set θ to a small value since in most real-world signed networks, the number of negative edges is significantly smaller than that of

[1] This corresponds to the the uniform scheme. In [24], the authors proposed three schemes–*i.e.*, user-oriented, item-oriented, and uniform, but the differences in their accuracies are insignificant in our preliminary experiments.

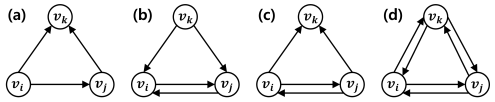


Figure 3: All four types of transitive triads [1].

positive edges (*e.g.*, in the Wiki-election dataset, the ratios of positive and negative edges are 79% and 21%, respectively). Moreover, among a large number of potential VNEs, only a small fraction of them clearly represent negative relationships. Below, we further discuss the effect of the parameter θ on *structural balance* [1, 4, 9], an important property of signed networks.

The *structural balance* indicates how well the edge signs in a given signed network follow the *balance theory*, a well-known theory in social sciences [9]. The balance theory states four rules of real-world social relationships: "a friend of my friend is my friend," "a friend of my enemy is my enemy," "an enemy of my friend is my enemy," and "an enemy of my enemy is my friend." Specifically, this theory considers triangles with an even number (*i.e.*, 0 or 2) of negative edges as *balanced* triangles and the other triangles as *unbalanced* triangles [4]. While the balance theory originated from social network analysis, it is also valid and useful in various types of signed networks, such as biological, political, and technical networks [2]. However, the structural balance has been studied primarily for signed *"undirected"* networks. Therefore, most related measures cannot take into account edge directions within triangles.

Recently, Aref et al. [1] proposed a new measure *a triadic balance* $T(\mathcal{S})$ for assessing the structural balance of the signed *"directed"* network $\mathcal{S}$. Given $\mathcal{S}$, they first collect all the transitive triads consisting of at least one or multiple triangles where the directions of three edges satisfy the transitivity (*i.e.*, if x follows y and y follows z, then x follows z.), as described in Figure 3. Then, they measure the ratio, $T(\mathcal{S})$, of balanced ones among all the collected transitive triads. A transitive triad is balanced if *all* of its transitive triangles (see Figure 3-(a) for a transitive triangle) inside are balanced, and a transitive triangle is balanced if it has an even number (*i.e.*, 0 or 2) of negative edges. The larger the value of $T(\mathcal{S})$ is, the better the edge signs of $\mathcal{S}$ follow the balance theory.

Figure 4 shows the triadic balance $T(\mathcal{S})$ in our $\mathcal{S}$ with varying θ for four *unsigned* directed networks (*i.e.*, GNU, Wiki-Vote, JUNG, and EAT). For comparison, we also show $T(\mathcal{R})$ in eight real-world *signed* directed networks $\mathcal{R}$ described in [1] (see the left side of Figure 4.). Note that $\mathcal{R}$ has true negative edges, and VNEs are not added to them. Thus, each of $\mathcal{R}$ has a single value of $T(\mathcal{R})$ regardless of θ. Overall, the values of $T(\mathcal{R})$ are high across all signed directed networks with an average of 0.78 (min=0.5, max=0.9, stdev=0.12), except for College-B [1]. The results show that the edge signs in real-world signed directed networks follow the rules of balance theory well. On the other hand, in the cases of $\mathcal{S}$, which are modeled by DIVINE, as θ increases, the values of $T(\mathcal{S})$ tend to decrease. Specifically, when $\theta < 1$, the values of $T(\mathcal{S})$ become as high as $T(\mathcal{R})$, whereas, when $\theta \geq 1$, the values of $T(\mathcal{S})$ become lower than $T(\mathcal{R})$. Recall that as θ increases, $\mathcal{S}$ contains more VNEs with lower degrees of negativity. Accordingly, if $\mathcal{S}$ includes uncertain VNEs, the edge signs in $\mathcal{S}$ do not follow well the rules of balance theory.

Based on this observation, we propose to set θ to a value around 0.25 or 0.5 where $T(\mathcal{S})$ and $T(\mathcal{R})$ in Figure 4 become close. We will also show empirically in Section 4.2 that such values of θ lead to high accuracy of DIVINE in the link-prediction task. Finally, we

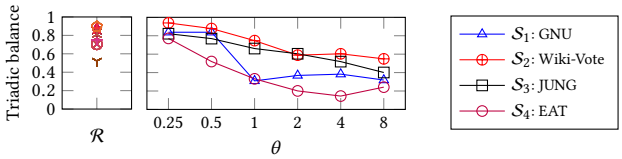


Figure 4: The triadic balance in 8 real-world signed directed networks $\mathcal{R}$ and the effect of θ on the triadic balance in 4 real-world unsigned directed networks $\mathcal{S}$. Larger θ indicates that more VNEs are added to $\mathcal{S}$.

build a signed directed network $\mathcal{S}$ composed by both the positive edges $\mathcal{E}^+$ and the VNEs $\mathcal{E}^-$. Now, we can exploit $\mathcal{E}^-$ in addition to $\mathcal{E}^+$ to better understand the properties of low-degree nodes, thereby addressing the lack of information about them.

3.5 Learning source and target embeddings

We learn the source and target embeddings of each node while preserving the asymmetric proximities between nodes in $\mathcal{S}$ modeled by DIVINE. To this end, we note recent signed NE methods [12, 18, 36, 37]. In a nutshell, they attempt to represent the node pairs with the positive edges to be close and those with the negative edges to be distant in the embedding space. Some of them further use the balance theory [4, 9] to exploit more-complex relationships between nodes by combining positive and negative edges. In this paper, we incorporate such a signed NE method into our DIVINE approach. We present the process of learning the source and target embeddings of nodes in $\mathcal{S}$ based on SIDE [12], which is designed to consider edge directions. However, we note *any* signed NE method, such as STNE [36] as shown in our evaluation, can be used instead.

First, we perform a directed random walk that starts from each node and follows out-going edges while generating a walk sequence $\{v_i \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n\}$ with edge signs. Within a window of size s, we then sample each directed node pair (v_i, v_j) where v_i (*i.e.*, source) precedes v_j (*i.e.*, target) in the sequence. For example, when there is a sequence $\{v_i \rightarrow v_1 \rightarrow v_2\}$ and s is 2, we sample three node pairs $(v_i, v_1), (v_i, v_2), (v_1, v_2)$. Here, we determine the sign of each node pair (v_i, v_j) by combining the edge signs in the sequence from v_i to v_j based on the balance theory.[2] Finally, for each node pair (v_i, v_j) with a positive (resp. negative) sign, we aim to maximize (resp. minimize) the proximity between the source embedding of v_i and the target embedding of v_j. To this end, we aim to minimize the following objective function:

$$\mathcal{L}(\mathbf{f}, \mathbf{g}) = \sum_{(v_i, v_j) \in O} [-\log \mathcal{P}(v_i, v_j) + \sum_{k=1} -\log \mathcal{P}(v_i, v_k)] + \mathcal{R}(\delta), \quad (4)$$

where O is a set of the sampled node pairs. For each pair (v_i, v_j), α noise nodes are randomly selected, and each noise node v_k is used to form a noise pair (v_i, v_k); $\mathcal{R}(\delta)$ is a regularization term for bias vectors of v_i and v_j. Lastly, $\mathcal{P}(v_i, v_j)$ corresponds to the estimated likelihood of (v_i, v_j) being sampled. Specifically, $\mathcal{P}(v_i, v_j)$ is calculated depending on the sign of (v_i, v_j): if the sign is positive, $\mathcal{P}(v_i, v_j) = (\mathbf{f}_{v_i} \cdot \mathbf{g}_{v_j})$, otherwise, $\mathcal{P}(v_i, v_j) = -(\mathbf{f}_{v_i} \cdot \mathbf{g}_{v_j})$. To minimize Eq. (4), we employ the gradient descent algorithm. Due to space limitations, we omit the details of the bias term and the optimization process; we refer to [12] for the details of SIDE.

Now, we obtain source and target embeddings $\mathbf{f}_{v_i}$ and $\mathbf{g}_{v_i}$ of each node v_i, that approximate asymmetric proximities in $\mathcal{S}$. The embeddings can be used as inputs in various downstream tasks.

[2]The sign is negative if there are an odd number of negative edges along the sequence while it is positive if there are an even number of negative edges.

4 EVALUATION

We designed our experiments, aiming at answering the following key research questions (RQs):

- **RQ1**: How should the degree of negativity be inferred in DIVINE?
- **RQ2**: How should the locations of VNEs be decided in DIVINE?
- **RQ3**: How should VNEs be distributed to nodes in DIVINE?
- **RQ4**: How many VNEs should be added in DIVINE?
- **RQ5**: Does DIVINE outperform its competitors for directed NE?
- **RQ6**: Is DIVINE effective for embedding low-degree nodes?

4.1 Experimental settings

Datasets. We used four real-world datasets of unsigned directed networks in different types: Gnutella (GNU), Wiki-Vote, JUNG, and Edinburgh Associative Thesaurus (EAT). They are all publicly available.[3] Table 2 provides some statistics of the four datasets.

- **GNU** is a peer-to-peer network for file sharing. A node represents a host, and a directed edge from a host v_i to a host v_j represents that v_i made a connection to v_j.
- **Wiki-Vote** is an online voting network. A node represents a user, and a directed edge from a user v_i to a user v_j represents that v_i voted on v_j.
- **JUNG** is a software class dependency network of JUNG 2.0.1 libraries. A node represents a Java class, and a directed edge from a class v_i to a class v_j represents that v_i is dependent on v_j.
- **EAT** is a lexical network. A node represents an English word, and a directed edge from a word v_i to a word v_j represents that v_j was given as a response to v_i in user experiments.

Competitors. We have two versions of DIVINE: one employing SIDE [12], denoted by DIVINE-I, and the other employing STNE [36], denoted by DIVINE-T. We have two families of state-of-the-art competitors: undirected NE methods (DeepWalk [25], Node2Vec [6], and LINE [32]) and directed NE methods (APP [41], ATP [30], NERD [11], GravityAE [27], GravityVAE [27], and DiGCN [33]).

For evaluation, we used the source code provided by the authors. For a fair comparison, we set the dimensionality of embeddings to 128 in all methods including DIVINE. We carefully tuned the hyperparameters of competitors and DIVINE. For DIVINE, we set its hyperparameters as follows: $\theta = 0.5$; epochs = 50 (for WRMF); learning rate = 0.025 (for SIDE and STNE); number of walks = 80 (for SIDE) / 20 (for STNE); walk length = 40 (for SIDE and STNE); window size = 5 (for SIDE) / 10 (for STNE); number of negative samples = 20 (for SIDE) / 5 (for STNE).

Evaluation Tasks. We employ a *link prediction* (LP) task to evaluate the effectiveness of DIVINE and competitors. The goal of this task is to evaluate how accurately we can predict the directed edges removed from the input directed network by using each NE method. For evaluation, we perform five-fold cross-validation, which splits the edges in the input network into training (80%) and test (20%) sets. In each of the training/test sets, we consider the existent edges as positive examples, and the same number of randomly-sampled non-existent edges as negative examples. Then, we obtain node embeddings by using each NE method on the training set.

[3] http://snap.stanford.edu/ | http://konect.cc/networks/

Table 2: Dataset statistics

Datasets	GNU	Wiki-Vote	JUNG	EAT
Nodes	6,301	7,115	6,120	23,132
0 out-degree	59.35%	15.21%	1.35%	63.54%
0 in-degree	4.11%	64.49%	66.43%	2.16%
Edges	20,777	103,689	50,535	312,320
Reciprocity	0.00%	5.64%	0.90%	9.50%
Density	0.05%	0.20%	0.13%	0.06%
Types	P2P	Election	Software	Word

Then, we concatenate[4] the embeddings of two nodes on each training example and train a *logistic regression* classifier using the concatenated embeddings as the input. Finally, we classify whether each testing example is positive or negative based on the learned classifier. We measure classification accuracy using the *area under curve* (AUC) [8], which has been widely used in many previous studies on NE [6, 11, 27, 30, 41].

We note that some directed NE methods [11, 27, 41] perform the above tasks by using the dot product, instead of the classifier employed in this paper. They predict the appearance of each testing example e_{ij} via the dot product of v_i's source embedding and v_j's target embedding; then, they classify the half of all examples with the highest predicted values as positive examples and the other half as negative examples. However, it may be *unfair to* the undirected NE methods that use a single embedding per node. Note that, by using the dot product, they cannot predict the appearances of bidirectional edges (e.g., e_{ij} and e_{ji}) *separately*.

In this sense, we claim that employing the classifier instead of the dot product alleviates the aforementioned limitation in validating undirected NE methods. Specifically, for the above example, we use v_i' embedding followed by v_j's embedding for e_{ij}; and use v_j' embedding followed by v_i's embedding for e_{ji}. We confirmed, through preliminary experiments, that employing the classifier significantly improves not only the accuracies of undirected NE methods but also the accuracies of directed NE methods, including DIVINE, compared to employing the dot product. Thus, in this work, we report the results in LP when employing the classifier.

Following [11, 27, 41], we further extend the aforementioned LP task for directed networks. Specifically, we also measure how accurately the directions of the *unidirectional* edges in the input directed network can be predicted using each NE method. Towards this end, we sample k% of the unidirectional positive examples and consider the edges with the opposite directions as negative examples, as in [11, 27, 41], and we sample the remaining $(100-k)$% of negative examples uniformly at random among non-existent edges. According to the ratio (i.e,. k%), we divide the LP task into three types: (1) $k = 0$ (Uniform LP; U-LP, in short), (2) $k = 50$ (Mixed LP; M-LP, in short), and (3) $k = 100$ (Biased LP; B-LP, in short). Note that the U-LP task is the same as the aforementioned LP task.

4.2 Results

RQ1: Comparisons of methods for inferring the degree of negativity. In Section 3.2, DIVINE infers the degree of negativity between nodes based on WRMF [24]. For RQ1, we compare the AUCs of the variants of DIVINE equipped with the following methods for inferring the degree of negativity: WRMF [24], RWR [28], ATP [30], APP [41], GravityVAE [27], and NERD [11].

[4] In our comparison experiments, we confirmed that the concatenation operator consistently outperforms several other operators (*e.g.*, element-wise product)

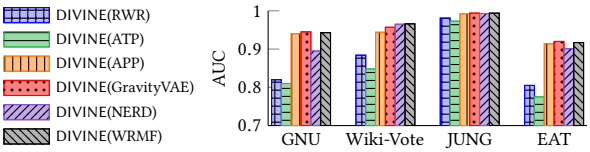


Figure 5: Comparisons of several methods for inferring the degree of negativity. When it is equipped with WRMF, DIVINE-I consistently achieves high AUC in all datasets.

Table 3: The effects of VNE-selection strategies on the accuracies of DIVINE-I. The local selection strategy is most effective, improving the accuracies of DIVINE-I most.

Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Global)	0.923	0.839	0.978	0.802
DIVINE(Local)	**0.943**	**0.966**	**0.994**	**0.917**
DIVINE(Local$_{vari}$)	0.920	0.838	0.986	0.813

Figure 5 shows the AUCs of the variants of DIVINE in B-LP. DIVINE(WRMF) consistently outperforms DIVINE(RWR) and DIVINE(ATP) in all datasets; and it shows AUCs higher than or comparable to those of DIVINE(APP), DIVINE(GravityVAE), and DIVINE(NERD). The results show that WRMF is consistently effective in inferring the degree of negativity. We also note that, among six variants, DIVINE(RWR) and DIVINE(ATP) showed the lowest AUC in most cases. Further investigation reveals that, when RWR and ATP are used, both (1) the degrees of negativity between a node v_i and each node non-reachable from v_i and (2) those between v_i and each zero in-degree node are predicted to be high. As a result, when equipped with RWR or ATP, DIVINE selects VNEs randomly among them, failing to choose VNEs effectively for accurate embedding.

RQ2: Effectiveness of the local selection strategy. In Section 3.3, we considered two strategies for selecting VNEs: global selection and local selection. For RQ2, we compare the AUCs of two variants of DIVINE: (1) DIVINE(Local), which employs the local selection strategy and (2) DIVINE(Global), which employs the global selection strategy. Table 3 shows that, in B-LP, DIVINE(Local) consistently and significantly outperforms DIVINE(Global) in all datasets. The results indicate that giving VNEs to *all* nodes (*i.e.*, the local selection strategy) is more beneficial than giving those to only a *small fraction* of nodes (*i.e.*, the global selection strategy). We found that the global selection strategy added VNEs to only 35%, 44%, 53%, and 34% of nodes in GNU, Wiki-Vote, JUNG, and EAT, respectively.

Furthermore, we examine the effectiveness of the local selection strategy in detail. Note that, in the local selection strategy, we also select VNEs from zero out-degree nodes, *i.e.*, the extreme case of low out-degree nodes. To verify whether this strategy, denoted by DIVINE(Local), contributes to improving the accuracy of directed NE, we compare it and another variant of DIVINE, denoted by DIVINE(Local$_{vari}$), which does not select VNEs from zero out-degree nodes. Table 3 shows that DIVINE(Local) consistently outperforms DIVINE(Local$_{vari}$) in all datasets. The results show that giving VNEs to all nodes *including zero out-degree nodes* effectively mitigates the lack of information when embedding low-degree nodes.

RQ3: Effectiveness of adding an equal number of VNEs to each source node. In Section 3.3, DIVINE adds an equal number of VNEs to all source nodes. For RQ3, we verify that this strategy is more effective than to add different numbers of VNEs to source nodes. Towards this end, we made two variants of DIVINE, denoted by DIVINE(Prop.) and DIVINE(InverseProp.). They set the number of VNEs from each node proportionally and inverse proportionally, respectively, to its out-degree. For comparison, we denote DIVINE based on our strategy by DIVINE(Uniform). Table 4 shows how the AUCs of DIVINE-I depend on the number of VNEs from source nodes in B-LP. We confirm that DIVINE(Uniform) consistently outperforms the other two variants in all datasets. The results show that treating all source nodes equally by adding an equal number of VNEs to them helps learn accurate embeddings most.

Table 4: The effects of the numbers of VNEs from source nodes on the accuracies of DIVINE-I. Adding an equal number of VNEs to all source nodes is most effective.

Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Prop.)	0.922	0.862	0.976	0.806
DIVINE(InverseProp.)	0.915	0.951	**0.994**	0.760
DIVINE(Uniform)	**0.943**	**0.966**	**0.994**	**0.917**

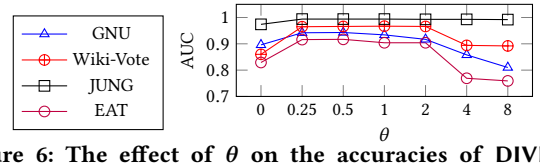


Figure 6: The effect of θ on the accuracies of DIVINE-I. Larger θ indicates that more VNEs are added to $\mathcal{S}$. It is most effective when the number of added VNEs is 0.25 times or 0.5 times the number of positive edges.

RQ4: Accuracy changes with varying θ. The total number of VNEs in DIVINE is θ times the number of positive edges. For RQ4, we show how θ affects the AUC of DIVINE-I. Figure 6 shows the results in B-LP. Note that setting $\theta = 0$ is equivalent to performing DIVINE-I on the original input directed networks *without VNEs*.

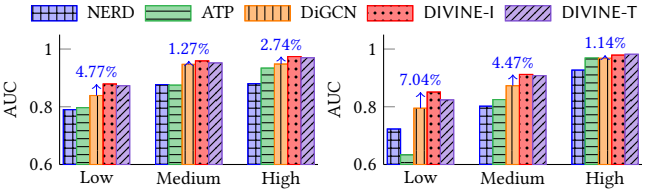
Figure 6 shows that the AUCs increase in a range between $\theta = 0$ and $\theta = 0.25$ and they remain almost the same in a range between $\theta = 0.25$ and $\theta = 1$. Then, the AUCs gradually decrease. In summary, DIVINE-I achieves the best AUC when $0.25 \leq \theta \leq 0.5$ in all datasets. This trend is similar to that of the triadic balance as shown in Figure 4 of Section 3.4. Specifically, when $0.25 \leq \theta \leq 0.5$, each signed directed network $\mathcal{S}$ (modeled by DIVINE) has a high value of triadic balance, similar to that in real-world signed directed networks $\mathcal{R}$, indicating that the edge signs of the network follow the rules of balance theory well. In this sense, we found that setting θ so that $\mathcal{S}$ follows the rules of balance theory well helps improve the AUC of DIVINE. Notably, DIVINE-I achieves significantly higher AUC when $0.25 \leq \theta \leq 0.5$ than when $\theta = 0$. This indicates that exploiting VNEs in addition to the given positive edges clearly helps to obtain more accurate embeddings.

RQ5: Comparison with nine competitors. We conducted comparative experiments with 3 LP tasks on 4 datasets to demonstrate the superiority of DIVINE-I and DIVINE-T over the following 9 competitors: DeepWalk [25], Node2Vec [6], LINE [32], APP [41], GravityAE/VAE [27], NERD [11], ATP [30], and DiGCN [33]. Table 5 shows the results. The value in boldface and underlined indicate the best AUC in each row and the AUC of the best 'competitor', respectively. Below, we summarize the results in Table 5.

First, the undirected NE methods provide AUCs comparable to or even higher than some directed NE methods (*e.g.*, APP). The

Table 5: Accuracies of nine competitors and our DIVINE. Both versions of DIVINE significantly and consistently outperform all competitors in all LP tasks on all datasets. That is, DIVINE provides most informative embeddings.

Datasets	Types	Undirected NE			Directed NE						DIVINE-I	DIVINE-T
		DeepWalk	Node2Vec	LINE	APP	GravityAE	GravityVAE	NERD	ATP	DiGCN		
GNU	U-LP	0.644±0.005	0.639±0.005	0.710±0.003	0.617±0.006	0.634±0.013	0.723±0.005	0.773±0.003	0.758±0.002	0.768±0.002	0.784±0.006	**0.798±0.002**
	M-LP	0.618±0.007	0.600±0.005	0.772±0.004	0.606±0.003	0.648±0.016	0.750±0.007	0.809±0.006	0.813±0.004	0.836±0.003	**0.858±0.010**	0.857±0.002
	B-LP	0.654±0.012	0.679±0.008	0.859±0.005	0.634±0.007	0.710±0.017	0.822±0.008	0.851±0.007	0.877±0.004	0.917±0.002	**0.943±0.008**	0.937±0.003
Wiki-Vote	U-LP	0.890±0.002	0.880±0.003	0.864±0.007	0.823±0.002	0.871±0.008	0.906±0.002	0.901±0.006	0.824±0.004	0.826±0.001	0.910±0.002	**0.929±0.001**
	M-LP	0.883±0.002	0.894±0.002	0.886±0.002	0.676±0.004	0.878±0.017	0.905±0.005	0.890±0.007	0.891±0.002	0.850±0.002	0.918±0.003	**0.933±0.001**
	B-LP	0.922±0.002	0.944±0.002	0.944±0.001	0.686±0.006	0.922±0.017	0.950±0.005	0.897±0.007	0.966±0.001	0.917±0.002	0.966±0.004	**0.971±0.001**
JUNG	U-LP	0.880±0.009	0.948±0.003	0.936±0.003	0.939±0.002	0.946±0.039	0.954±0.002	0.955±0.002	0.951±0.002	0.955±0.001	0.948±0.002	**0.960±0.002**
	M-LP	0.902±0.007	0.956±0.003	0.957±0.002	0.950±0.002	0.944±0.033	0.968±0.003	0.963±0.002	0.968±0.002	0.971±0.002	0.969±0.001	**0.976±0.001**
	B-LP	0.950±0.006	0.982±0.001	0.989±0.001	0.930±0.001	0.976±0.027	0.991±0.002	0.979±0.001	0.990±0.001	0.994±0.001	0.994±0.001	**0.996±0.001**
EAT	U-LP	0.831±0.001	0.832±0.002	0.824±0.001	0.772±0.001	0.836±0.009	0.839±0.004	0.864±0.002	0.855±0.002	0.831±0.001	0.880±0.006	**0.888±0.001**
	M-LP	0.682±0.001	0.759±0.001	0.827±0.001	0.701±0.001	0.791±0.033	0.815±0.001	0.825±0.002	0.882±0.001	0.860±0.001	0.881±0.007	**0.889±0.001**
	B-LP	0.614±0.001	0.819±0.001	0.863±0.001	0.630±0.002	0.838±0.029	0.851±0.003	0.802±0.002	0.915±0.001	0.901±0.001	0.917±0.006	**0.921±0.002**



(a) Out-degree-based (b) In-degree-based

Figure 7: Comparisons of our DIVINE and the strongest competitors in the out- and in-degree-based node groups. Both versions of DIVINE consistently achieve higher AUC than all competitors across all groups; and the performance gain is largest for the low-degree node groups.

results are different from those reported in some previous studies where LP tasks are performed using the dot product [11, 41]. By employing the logistic classifier, the undirected NE methods can now consider the directions of edges, making their AUCs improve significantly in all types of LP tasks. Second, no single competitor consistently outperforms the other competitors. Best competitors change depending on tasks and datasets: GravityVAE for U-LP and M-LP on Wiki-Vote; NERD for U-LP on GNU, JUNG, and EAT; ATP for B-LP on GNU and for M-LP and B-LP on EAT; DiGCN for all remaining cases. *To the best of our knowledge, this work made a direct comparison of the four recent NE methods (i.e., GravityAE/VAE, NERD, ATP, and DiGCN) for the first time.*

Third and most importantly, both versions of DIVINE significantly and consistently outperform *all* competitors in *all* LP tasks on *all* datasets. Specifically, DIVINE-T yields up to 14.32%, 14.89%, 12.70%, and 12.41% higher AUC than the strongest competitors GravityVAE, NERD, ATP, and DiGCN, respectively. Additionally, we note that the AUCs of DIVINE-I and DIVINE-T are highest in B-LP followed by M-LP and then U-LP on all datasets. This indicates that DIVINE is most accurate in the task of predicting the edge directions. In other words, the embeddings obtained by DIVINE accurately preserve asymmetric proximities between nodes.

RQ6: Effectiveness in embedding low-degree nodes. We verify that DIVINE is effective in embedding low out- and in-degree nodes about which we lack of information. To this end, we divide all nodes in the test set into three groups (*i.e.*, low, medium, and high) according to their out-degree. Then, we compare the AUCs of DIVINE-I and DIVINE-T with those of the strongest competitors (*i.e.*, NERD, ATP, and DiGCN), obtained from the testing examples having the nodes of each group as sources. Similarly, we divide the nodes into three groups according to their in-degree and compare the AUCs obtained from the nodes of each group as targets. Figures 7-(a) and 7-(b) show the results in B-LP on GNU, the sparsest one among the four datasets used in this paper, for out-degree-based groups and in-degree-based groups, respectively. We note that both versions of DIVINE consistently outperform all the competitors across all groups. *Notably, the performance gain is largest in the low-degree node groups.* The results indicate that DIVINE successfully addresses the lack of information about low out- and in-degree nodes, and as a result, it is particularly effective in improving the quality of embeddings of such nodes.

5 CONCLUSIONS

Observation. In this work, we pointed out that the existing directed NE methods face difficulties in accurately preserving asymmetric proximities between nodes in a sparse network with a large fraction of low out- and in-degree nodes.

Algorithm Design. To address this limitation, we designed a novel directed NE approach, named as DIVINE, which effectively captures asymmetric proximities between nodes by exploiting information in the form of VNEs. Under DIVINE, we proposed three ideas to selectively add VNEs: (1) inferring the degree of negativity; (2) using the local selection strategy to distribute VNEs to all nodes; (3) determining the number of VNEs based on the theory of structural balance. Once VNEs are added, DIVINE learns the source and target embeddings of nodes by employing *any* NE methods that are designed to be effective for signed directed networks.

Experiments. We demonstrated that DIVINE significantly and consistently outperforms its 9 state-of-the-art competitors in 3 LP tasks on 4 real-world datasets. Through extensive ablation studies, we showed clearly the effectiveness of *each* of our design choices.

ACKNOWLEDGMENT

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01373,Artificial Intelligence Graduate School Program(Hanyang University)), the research fund of Hanyang University(HY-202100000660001), and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1A2B5B03001960).

REFERENCES

[1] Samin Aref, Ly Dinh, Rezvaneh Rezapour, and Jana Diesner. 2020. Multilevel structural evaluation of signed directed social networks based on balance theory. *Scientific Reports* 10, 1 (2020), 1–12.

[2] Samin Aref and Mark C. Wilson. 2019. Balance and frustration in signed networks. *J. Complex Networks* 7, 2 (2019), 163–189.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 891–900.

[4] Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of Heider's theory. *Psychological Review* 63, 5 (1956), 277–293.

[5] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A Survey on Network Embedding. *IEEE Trans. Knowl. Data Eng.* 31, 5 (2019), 833–852.

[6] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 855–864.

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of the Annual Conference. on Neural Information Processing Systems (NeurIPS)*. 1024–1034.

[8] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.

[9] Fritz Heider. 1946. Attitudes and cognitive organization. *The Journal of psychology* 21, 1 (1946), 107–112.

[10] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[11] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. 2019. Node Representation Learning for Directed Graphs. In *Proc. of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Vol. 11906. 395–411.

[12] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U. Kang. 2018. SIDE: Representation Learning in Signed Directed Networks. In *Proc. of The Web Conference (WWW)*. 509–518.

[13] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. In *Proc. NeurIPS Workshop Bayesian Deep Learning*. 1–3.

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of the International Conference on Learning Representations (ICLR)*.

[15] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *Proc. of the Annual Conference. on Neural Information Processing Systems (NeurIPS)*. 13333–13345.

[16] Jérôme Kunegis, Julia Preusse, and Felix Schwagereit. 2013. What is the added value of negative links in online social networks?. In *Proc. of The Web Conference (WWW)*. 727–736.

[17] Wonchang Lee, Yeon-Chang Lee, Dongwon Lee, and Sang-Wook Kim. 2021. Look Before You Leap: Confirming Edge Signs in Random Walk with Restart for Personalized Node Ranking in Signed Networks. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 143–152.

[18] Yeon-Chang Lee, Nayoun Seo, Kyungsik Han, and Sang-Wook Kim. 2020. ASiNE: Adversarial Signed Network Embedding. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 609–618.

[19] Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proc. of The Web Conference (WWW)*. 641–650.

[20] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 387–396.

[21] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 556–559.

[22] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 1105–1114.

[23] Christopher C Paige and Michael A Saunders. 1981. Towards a generalized singular value decomposition. *SIAM J. Numer. Anal.* 18, 3 (1981), 398–405.

[24] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *Proc. of the IEEE International Conference on Data Mining (IEEE ICDM)*. 502–511.

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 701–710.

[26] Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reason.* 50, 7 (2009), 969–978.

[27] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet-Anh Tran, and Michalis Vazirgiannis. 2019. Gravity-Inspired Graph Autoencoders for Directed Link Prediction. In *Proc. of the ACM International Conference on Information and Knowledge Management (ACM CIKM)*. 589–598.

[28] Kijung Shin, Jinhong Jung, Lee Sael, and U Kang. 2015. BEAR: Block Elimination Approach for Random Walk with Restart on Large Graphs. In *Proc. of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD)*. 1571–1585.

[29] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. 2009. Scalable proximity estimation and link prediction in online social networks. In *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC)*. 322–335.

[30] Jiankai Sun, Bortik Bandyopadhyay, Armin Bashizade, Jiongqian Liang, P. Sadayappan, and Srinivasan Parthasarathy. 2019. ATP: Directed Graph Embedding with Asymmetric Transitivity Preservation. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 265–272.

[31] Jiliang Tang, Yi Chang, Charu C. Aggarwal, and Huan Liu. 2016. A Survey of Signed Network Mining in Social Media. *ACM Comput. Surv.* 49, 3 (2016), 42:1–42:37.

[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. of The Web Conference (WWW)*. 1067–1077.

[33] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and Andrew Lim. 2020. Digraph Inception Convolutional Networks. In *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[34] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (ACM KDD)*. 1225–1234.

[35] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 203–209.

[36] Pinghua Xu, Wenbin Hu, Jia Wu, Weiwei Liu, Bo Du, and Jian Yang. 2019. Social Trust Network Embedding. In *Proc. of the IEEE International Conference on Data Mining (IEEE ICDM)*. 678–687.

[37] Shuhan Yuan, Xintao Wu, and Yang Xiang. 2017. SNE: Signed Network Embedding. In *Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 183–195.

[38] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2020. Network Representation Learning: A Survey. *IEEE Trans. Big Data* 6, 1 (2020), 3–28.

[39] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*. 3155–3161.

[40] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data Augmentation for Graph Neural Networks. *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*.

[41] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable Graph Embedding for Asymmetric Proximity. In *Proc. of the AAAI International Conference on Artificial Intelligence (AAAI)*. 2942–2948.

[42] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. 2007. Combining content and link for classification using matrix factorization. In *Proc. of the ACM International Conference on Research & Development in Information Retrieval (ACM SIGIR)*. 487–494.