NODESIG: Binary Node Embeddings via Random Walk Diffusion

Abdulkadir Çelikkanat
Technical University of Denmark
DTU Compute
Lyngby, Denmark
abdcelikkanat@gmail.com

Fragkiskos D. Malliaros
Paris-Saclay University
CentraleSupélec, Inria
Gif-sur-Yvette, France
fragkiskos.malliaros@centralesupelec.fr

Apostolos N. Papadopoulos
Aristotle University of Thessaloniki
Department of Informatics
Thessaloniki, Greece
papadopo@csd.auth.gr

***Abstract*—Graph Representation Learning (GRL) has become a key paradigm in network analysis, with a plethora of interdisciplinary applications. As the scale of networks increases, most of the widely used learning-based graph representation models also face computational challenges. While there is a recent effort toward designing algorithms that solely deal with scalability issues, most of them behave poorly in terms of accuracy on downstream tasks. In this paper, we aim to study models that balance the trade-off between efficiency and accuracy. In particular, we propose NODESIG, a scalable model that computes binary node representations. NODESIG exploits random walk diffusion probabilities via stable random projections towards efficiently computing embeddings in the Hamming space. Our extensive experimental evaluation on various networks has demonstrated that the proposed model achieves a good balance between accuracy and efficiency compared to well-known baseline models on the node classification and link prediction tasks.**

***Index Terms*—Graph representation learning, node embeddings, binary representations, node classification, link prediction**

I. INTRODUCTION

Graph-structured data is ubiquitous in many diverse disciplines and application domains, including biology, neuroscience, and applications arising from social media and networking analysis [1]. Besides being elegant models for data representation, graphs have also been proven valuable in various widely used machine learning tasks. For instance, in the case of biological networks, we are interested in predicting the function of proteins or in inferring the missing structure of the underlying protein-protein interaction network. Both of these problems can be formalized as learning tasks on graphs, with the main challenge being how to properly incorporate its structural properties and the proximity among nodes into the learning process. In this direction, *Graph Representation Learning (GRL)* has become a key paradigm for extracting information from networks and for performing various tasks such as link prediction, classification, and visualization [2], [3]. These models aim to find node representations (i.e., *node embeddings*) in a way that the desired properties and proximity among nodes are preserved in the embedding space.

Supported in part by ANR (French National Research Agency) under the JCJC project GraphIA (ANR-20-CE23-0009-01).

IEEE/ACM ASONAM 2022, November 10-13, 2022
978-1-6654-5661-6/22/$31.00 © 2022 European Union

Most of the existing GRL approaches deal with *learning-based models*, relying either on matrix factorization or on node context sampling to infer the proximities between nodes [2]. For the former, the goal is to learn embeddings by *factorizing* the matrix which has been designed for capturing and representing desired graph properties and node proximities in a lower-dimensional space. Typically, such approaches target to preserve first-order (adjacency-based) or higher-order proximity of nodes [4]–[7]. Since such models heavily rely on the expensive factorization of dense node proximity matrices, the computational cost and the high memory usage burden bring limitations for large-scale networks. Although recent studies have proposed heuristics to overcome these challenges [8], [9], they are obliged to forgo their predictive performance in most cases—hence, putting the practitioners in a dilemma between effectiveness and computational cost.

In order to address the aforementioned challenges towards developing effective and scalable algorithms for representation learning on networks, *random walk*-based models have gained considerable attention [2]. The main idea here is to generate a set of node sequences by following a random walk strategy. Node embeddings are then learned by maximizing the probability of node co-occurrences in the generated sequences. [10]–[14]. Nevertheless, a large number of random walks is required to be explicitly sampled in order to ensure the effectiveness of the embedding on downstream tasks. Furthermore, it has been shown that random walk-based embedding approaches implicitly perform factorization of a properly chosen dense transition probability matrix, leading to better performance on downstream tasks [15], [16]. Although recent studies aim to improve running time complexity via matrix sparsification techniques [17] or capitalizing on hierarchical graph representations [18], the quality of the embeddings deteriorates.

Besides the computational burden of model optimization, most of the proposed algorithms learn low-dimensional embeddings in the *Euclidean* space. A recent few studies have proposed to learn *discrete* node representations [19], [20], in which *Hamming* distance is leveraged to determine the similarity of embedding vectors. The basic idea builds upon fast sketching techniques for scalable similarity search, mainly based on data-independent or data-dependent hashing techniques [21]. Although binary embeddings speedup distance

measure computations with respect to the metrics defined in Euclidean space, the corresponding models often undergo computationally intensive learning procedures, especially in the case of learning-to-hash models [19].

Contributions. In this paper, we propose NODESIG, a scalable model for computing expressive binary node embeddings based on stable random projections. NODESIG first leverages random walk diffusions to estimate higher-order node proximity. Then, a properly defined sign random projection hashing technique is applied to obtain binary node signatures in the Hamming space, leading to an approximation of the *chi similarity* (χ) [22] between the proximity vectors in the original space. Since these vectors are constructed based on the occurrence frequencies of nodes within random walks, *chi* similarity emerges as a natural choice of similarity metric, frequently used to compare histograms in various areas including natural language processing and computer vision [23], [24].

Each component of NODESIG has been designed to ensure the scalability, while at the same time, the accuracy on downstream tasks is not compromised or even improves compared to traditional models. Figure 1 positions NODESIG regarding the accuracy and running time, providing a comparison to different models on the *PPI* network. As we observe, NODESIG's running time is comparable to that of models that focus solely on scalability (e.g., NODESKETCH, RANDNE, LOUVAINNE), with improved accuracy even higher than NODE2VEC, FREDE and HOPE in this dataset.
The main contributions can be summarized as follows:

- We introduce NODESIG, a scalable and expressive model for binary node embeddings based on stable random projection hashing of random walk diffusion probabilities.
- The distance computation between node signatures in the embedding space is provided by the Hamming distance on bit vectors, which is significantly more efficient than distance computations based on other distance measures.
- In a thorough experimental evaluation, we demonstrate that the proposed binary embeddings achieve superior performance compared to various baseline models on two downstream tasks. At the same time, the running time allows the model to scale on large graphs.

Source code. The implementation of NODESIG can be found at: https://abdcelikkanat.github.io/projects/nodesig/.

II. RELATED WORK

Graphs are rich representations of the real world that can capture different types of relationships and modalities among entities [2]. One of the first modern algorithms is the DEEPWALK algorithm [10] which uses uniform truncated random walks to represent the *context* of a node. Intuitively, nodes with similar random walks have a higher degree of similarity. The NODE2VEC method [11] is more general and manages to combine BFS and DFS search strategies, achieving significant performance improvements. LINE [12] optimizes an objective function capturing both first-order and second-order node proximities. Essentially those models constitute adaptations

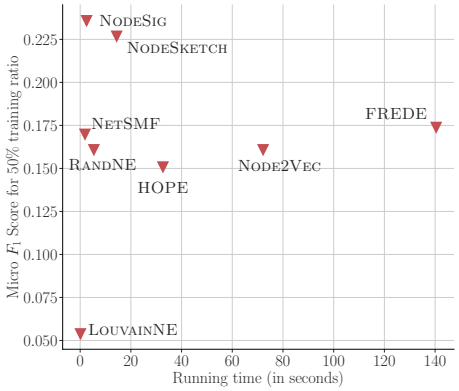

Fig. 1: Comparison of models on the *PPI* network. NODESIG balances good accuracy (Micro-F_1) and running time (in secs).

of the SKIPGRAM technique proposed for word embeddings [25]. However, they require the extensive realization of the random walks, which constitutes a computationally intensive operation. It turns out that problems related to random walk sampling can be alleviated by using matrix factorization. The main drawback of NETMF and other matrix factorization-based approaches [4], [7], however, is that in general matrix factorization is a computationally intensive operation.

The main limitation of the aforementioned embedding techniques is that they do not scale well for large networks. The main focus has been put on increasing the effectiveness of data mining tasks (e.g., classification, link prediction, network reconstruction) whereas the efficiency dimension has not received significant attention. To attack this problem, recent advances in network representation learning use random projection or hashing techniques (more specifically, variants of locality-sensitive hashing) in order to boost performance, trying to maintain effectiveness as well.

RANDNE [8] is one of the first scalable approaches which is based on iterative Gaussian random projection, being able to adapt to any desired proximity level. In the same line, FASTRP was proposed in [9] which is faster than RANDNE and also more accurate. LOUVAINNE [18] suggested learning node representations by aggregating the embeddings of nodes extracted at varying levels of the hierarchy. The NETHASH algorithm [20] expands each node of the graph into a rooted tree, and then by using a bottom-up approach encodes structural information as well as attribute values into minhash signatures in a recursive manner. FREDE [26] is a sketching-based approach relying on Personalized Page Rank (PPR) matrix, which alleviates the computation burden by applying a sketching technique. A similar approach has been used in NODESKETCH [27], where the context of every node is defined in a different way whereas the embedding vector of each node contains integer values. Although these approaches rely on fast-sketching schemes, they do not show comparable performance to the aforementioned learnable models in the downstream tasks. In this paper, we aim to introduce an approach balancing accuracy and running time.

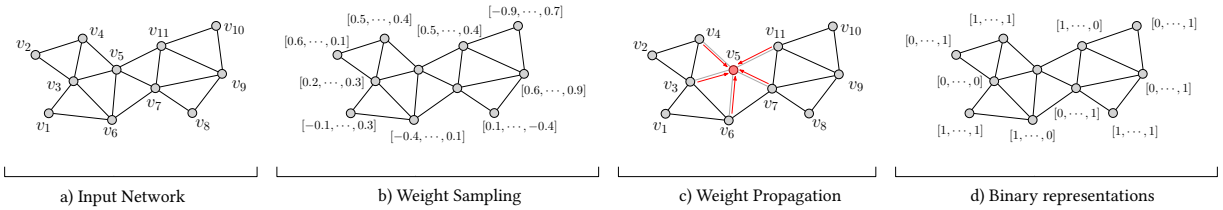

Fig. 2: Schematic representation of the NODESIG model. First, the weights of the random projection matrix are sampled and then the projection of the proximity matrix is performed via the weight propagation step. Finally, binary node representations are obtained by combining the signs of the projected values.

III. PROPOSED APPROACH

A. Random Walk Diffusion for Node Proximity Estimation

In most cases, direct links among nodes are not sufficient to grasp various inherent properties of the network that are related to node proximity. It is highly probable that the network might have missing or noisy connections, thus relying solely on first-order proximity can reduce the expressiveness of the model. To overcome this problem, we directly leverage random walk diffusions, adopting a uniform random walking strategy to extract information describing the structural roles of nodes in the network. Let $\mathbf{P}$ denote the right stochastic matrix associated with the adjacency matrix of the graph, which is obtained by normalizing the rows of the matrix. More formally, $\mathbf{P}$ can be written as $\mathbf{P}_{(i,j)} := \mathbf{A}_{(i,j)} / \sum_j \mathbf{A}_{(i,j)}$, defining the transition probabilities of the uniform random walk strategy. We use a slightly modified version of the transition matrix by adding a self-loop on each node, in case it does not exist.

Note that the probability of visiting the next node depends only on the current node that the random walk resides; therefore, node v_j can be visited starting from v_i by taking l steps with probability $\mathbf{P}^{(l)}_{(i,j)}$, if there is a path connecting them. For a given walk length L, we define the matrix $\mathbf{M}$ as

$$\mathbf{M} := \mathbf{P} + \cdots + \mathbf{P}^{(l)} + \cdots + \mathbf{P}^{(L)},$$

where $\mathbf{P}^{(l)}$ indicates the l-order proximity matrix and each entry $\mathbf{M}_{v,u}$ in fact specifies the expectation of visiting u starting from node v within L steps. By introducing an additional parameter α, $\mathbf{M}(\alpha)$ can be rewritten as follows:

$$\mathbf{M}(\alpha) := \alpha\mathbf{P} + \cdots + \alpha^{(l)}\mathbf{P}^{(l)} + \cdots + \alpha^{(L)}\mathbf{P}^{(L)}.$$

Higher order node proximities can be captured using longer walk lengths, where the impact of the walk at different steps is controlled by the *importance factor* $\alpha \in \mathbb{R}^+$. As we will present in the next paragraph, matrix $\mathbf{M}(\alpha)$ is properly exploited by a random projection hashing strategy to efficiently compute binary node representations.

B. Learning Binary Embeddings

Random projection methods [28] have been widely used in a wide range of machine learning applications dealing with large scale data. They mainly target to represent data points into a lower dimensional space by preserving the similarity in the original space. Likewise, we aim at encoding each node into a *Hamming* space $\mathbb{H}\big(d_{\mathcal{H}}, \{0,1\}^{\mathcal{D}}\big)$; we consider the normalized *Hamming* distance $d_{\mathcal{H}}$ as the distance metric [29]. The benefit of binary representations is twofold: first, they will allow us to perform efficient distance computation using bitwise operations, and secondly reduce the required disk space to store the data.

Random projections are linear mappings; the binary embeddings though require nonlinear functions to perform the discretization step, and a natural choice is to consider the signs of the values obtained by the *Johnson-Lindenstrauss (JL)* [30] transform. More formally, it can be written that

$$h_{\mathbf{W}}(\mathbf{x}) := sign(\mathbf{x}^\top \mathbf{W}),$$

where $\mathbf{W}$ is the projection matrix whose entries $\mathbf{W}_{(i,j)}$ are independently drawn from normal distribution and $sign(\mathbf{x})_j$ is equal to 1 if $\mathbf{x}_j > 0$ and 0 otherwise. The approach was first introduced in the work [31] for a rounding scheme in approximation algorithms, demonstrating that the probability of obtaining different values for a single bit quantization is proportional to the angle between vectors, as it is shown in Theorem 1. The main idea relies on sampling uniformly distributed random hyperplanes in $\mathbb{R}^{\mathcal{D}}$. Each column of the projection matrix, in fact, defines a hyperplane and the arc between vectors $\mathbf{x}$ and $\mathbf{y}$ on the unit sphere is intersected if $h_{\mathbf{W}}(\mathbf{x})_i$ and $h_{\mathbf{W}}(\mathbf{y})_i$ take different values.

Theorem 1 ([31]). *For a given pair of vectors* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,

$$\mathbb{P}\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] = \frac{1}{\pi}\cos^{-1}\left(\frac{\mathbf{x}^\top\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}\right),$$

where $\mathbf{W}_{(i,j)} \sim \mathcal{N}(0,1)$ *for* $1 \leq i,j \leq N$.

Although the signs of JL random projections allow us to approximate the angle between the vectors in the original space, in our settings, we would prefer to preserve a distance metric that can fit better the input data $\mathbf{M}(\alpha)$. Note that, the node proximity matrix $\mathbf{M}(\alpha)$ contains non-negative elements computed based on the occurrence frequencies of nodes within random walks. Hence, we will focus on estimating distance metrics capable of comparing histogram-type data by properly redesigning the projection matrix. The *stable* random projections approach [32] generalizes the aforementioned idea by using a symmetric α-stable distribution with unit scale in order

to sample the elements of the projection matrix, for $0 < \alpha \leq 2$. Li et al. [32] proposed the following upper bound

$$\mathbb{P}\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] \leq \frac{1}{\pi}\cos^{-1}\rho_\alpha \quad (1)$$

for non-negative vectors ($\mathbf{x}_i \geq 0, \mathbf{y}_i \geq 0$ for $1 \leq i \leq N$), where ρ_α is defined as

$$\rho_\alpha := \left(\frac{\sum_{i=1} x_i^{\alpha/2} y_i^{\alpha/2}}{\sqrt{\sum_{i=1} x_i^\alpha}\sqrt{\sum_{i=1} y_i^\alpha}}\right)^{2/\alpha}.$$

It is well known that the bound is exact for $\alpha = 2$, which also corresponds to the special case in which normal random projections are performed. When the vectors are chosen from the $\ell^1(\mathbb{R}^+)$ space (i.e., $\sum_{d=1} x_d = 1$, $\sum_{d=1} y_d = 1$), it is easy to see that the χ^2 similarity ρ_{χ^2} defined as $\sum_{d=1}(2x_d y_d)/(x_d + y_d)$ is always greater or equal to ρ_1, as suggested by Lemma 1.

Lemma 1. *For given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathcal{D}}$ satisfying $x_i, y_i \geq 0$ for all $1 \leq i \leq \mathcal{D}$ and $\sum_{i=1} x_i = \sum_{i=1} y_i = 1$, then $\rho_1 \leq \rho_{\chi^2}$.*

Proof.

$$\begin{aligned}
\rho_1 = \left(\frac{\sum_{i=1} x_i^{1/2} y_i^{1/2}}{\sqrt{\sum_{i=1} x_i}\sqrt{\sum_{i=1} y_i}}\right)^2 &= \left(\sum_{i=1}\sqrt{x_i y_i}\right)^2 \\
&= \left(\sum_{i=1}\frac{\sqrt{2x_i y_i}}{\sqrt{x_i + y_i}}\frac{\sqrt{x_i + y_i}}{\sqrt{2}}\right)^2 \\
&\leq \sum_{i=1}\frac{2x_i y_i}{x_i + y_i}\sum_{i=1}\frac{x_i + y_i}{2} \\
&= \sum_{i=1}\frac{2x_i y_i}{x_i + y_i} = \rho_{\chi^2},
\end{aligned}$$

where the inequality follows from the *Cauchy-Schwarz* inequality. □

Besides, it has been empirically shown [32] that the collision probability for *Cauchy* random projections with unit scale can be well estimated, especially for sparse data:

$$\mathbb{P}\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] \approx \frac{1}{\pi}\cos^{-1}\rho_{\chi^2} \leq \frac{1}{\pi}\cos^{-1}\rho_1. \quad (2)$$

Note that, the matrix, $\mathbf{M}(\alpha)$, described in the previous paragraph consists of non-negative values; its row sums are equal to $\sum_{l=1}^{L}\alpha^l$ and $\mathbf{M}(\alpha)$ is sparse enough for small walk lengths. Therefore, we design the projection matrix by sampling its entries from the *Cauchy* distribution, aiming to learn binary representations preserving the *chi-square* similarity. The *chi-square* distance is one of the measures used for histogram-based data, commonly used in the fields of computer vision and natural language processing [23], [24].

As it is shown in Figure 2, the last step of NODESIG for obtaining binary node representations is to utilize the signs of the projected data. In other words, the embedding vector $\mathbf{E}[v]$ for each node $v \in V$ is computed as follows:

$$\mathbf{E}[v] := \left[sign\left(\mathbf{M}(\alpha)_{(v,:)}\mathbf{W}_{(:,1)}\right), \ldots, sign\left(\mathbf{M}(\alpha)_{(v,:)}\mathbf{W}_{(:,\mathcal{D})}\right)\right]$$

Note that, the projection of the exact realization of $\mathbf{M}(\alpha)$ can be computationally intensive, especially for large walks. Instead, it can be computed by propagating the weights $W_{(u,d)}$ for each dimension d ($1 \leq d \leq \mathcal{D}$), using the following recursive update rule:

$$\mathcal{R}_{(v,d)}^{(l+1)}(\alpha) \leftarrow \alpha \sum_{u \in \mathcal{N}(v)} P_{(v,u)} \times \left(W_{(u,d)} + \mathcal{R}_{(u,d)}^{(l)}(\alpha)\right), \quad (3)$$

where $\mathcal{N}(v)$ refers to the set of neighbors of node $v \in V$ and $\mathcal{R}_{(v,d)}^{(l)}$ is equal to the projected data, $\mathbf{M}_{(v,:)}(\alpha) \cdot \mathbf{W}_{(:,d)}$ for the walk length l, and $\mathcal{R}_{(v,d)}^{(0)}$ is initialized to zero. By Lemma 2, it can be seen that the projection of $\mathbf{M}(\alpha)$ can be computed by applying the recursive update rules defined in Eq. 3.

Lemma 2. *Let $\mathbf{P}$ be $n \times n$ a right stochastic matrix and $\mathbf{M}^{(L)}(\alpha)$ be the matrix defined by $\alpha\mathbf{P} + \cdots + \alpha^{(l)}\mathbf{P}^{(l)} + \cdots + \alpha^{(L)}\mathbf{P}^{(L)}$. For a given $\mathbf{W} \in \mathbb{R}^{n \times \mathcal{D}}$, the term $\mathbf{M}^{(L)}(\alpha)\mathbf{W}$ is equal to $\mathcal{R}^{(L)}$ where each $R_{(v,d)}^{(l)}$ is recursively defined by $\alpha\mathbf{P}_{(v,:)}\left(\mathbf{W}_{(:,d)} + \mathcal{R}_{(:,d)}^{(l-1)}\right)$ for all $l \in \{1, \ldots, L\}$, $v \in \{1, \ldots, n\}$ and $\mathcal{R}^{(0)}$ is set to 0.*

Proof. For $l = 1$, we have that $\mathcal{R}_{(v,d)}^{(1)} = \alpha\mathbf{P}_{(v,:)}\left(\mathbf{W}_{(:,d)} + \mathcal{R}_{(:,d)}^{(0)}\right) = \alpha\mathbf{P}_{(v,:)}\left(\mathbf{W}_{(:,d)} + 0\right) = \alpha\mathbf{P}_{(v,:)}\mathbf{W}_{(:,d)}$ for all $d \in \{1, \ldots, \mathcal{D}\}$, $v \in \{1, \ldots, n\}$ so the claim holds for $n = 1$. Let us assume that it is true for $n = l \geq 1$. Then,

$$\begin{aligned}
\mathcal{R}_{(v,d)}^{(n+1)} &= \alpha\mathbf{P}_{(v,:)}\left(\mathbf{W}_{(:,d)} + \mathcal{R}_{(:,d)}^{(n)}\right) \\
&= \alpha\mathbf{P}_{(v,:)}\left(\mathbf{W}_{(:,d)} + \alpha\mathbf{P}\mathbf{W}_{(:,d)} + \cdots + \alpha^{(n)}\mathbf{P}^{(n)}\mathbf{W}_{(:,d)}\right) \\
&= \alpha\mathbf{P}_{(v,:)}\mathbf{W}_{(:,d)} + \cdots + \alpha^{(n+1)}\mathbf{P}_{(v,:)}^{(n+1)}\mathbf{W}_{(:,d)} \\
&= \left(\alpha\mathbf{P}_{(v,:)} + \cdots + \alpha^{(n+1)}\mathbf{P}_{(v,:)}^{(n+1)}\right)\mathbf{W}_{(:,d)} \\
&= \mathbf{M}^{(n+1)}(\alpha)\mathbf{W}_{(:,d)}.
\end{aligned}$$

Thus, the claim also holds for $n+1 = l$. By the principle of induction, it satisfies for all $l \in \{1, \ldots, L\}$. □

Algorithm 1 provides the pseudocode of NODESIG. We generate the projection matrix by sampling the weights from the *Cauchy* distribution with unit scale. The samples are further divided by $\sum_{l=1}\alpha^l$, because the row sums of $\mathbf{M}(\alpha)$ must be equal to 1. Then, we compute the terms $\mathcal{R}_{(v,d)}^{(l)}$ by propagating the weights in Line 9 at each walk iteration $l < L$. Note that, the term R in the pseudocode is a vector of length $\mathcal{D}$, thus we obtain the final node representation using the signs of $\mathcal{R}_{(v,d)}^{(L)}$.

C. Time and Space Complexity

At the beginning of the algorithm, we need to sample a weight matrix of size $|\mathcal{V}| \cdot \mathcal{D}$, and it can be formed in the order of $\mathcal{O}(|\mathcal{V}| \cdot \mathcal{D})$. As we observe in Algorithm 1, the main cumbersome point of NODESIG is caused by the update rule defined in Eq. (3), which corresponds to Line 9 of the pseudocode. The update rule must be repeated $|\mathcal{N}(v)|$ times for each node $v \in V$, thus it requires $2 \cdot m \cdot \mathcal{D}$ multiplication operations at the walk step l ($1 \leq l \leq L$) for a network consisting of m edges and for embedding vectors of dimension

Algorithm 1: NODESIG

Data: Graph $\mathcal{G} = (V, E)$ with the transition matrix $\mathbf{P}$; representation size $\mathcal{D}$; walk length L; importance factor α

Result: Embeddings $\mathbf{E}[v] \in \mathbb{R}^{\mathcal{D}}$ for each node $v \in V$

1 **for** ***each*** *node* $v \in V$ **do**
2 $\quad \mathcal{R}[v] \leftarrow \mathbf{0}_{\mathcal{D}} = (0, \ldots, 0)$;
3 $\quad W[v] \sim Cauchy(0,1)^{\mathcal{D}}$ / $\sum_{l=1}^{L} \alpha^l$;
4 **end**
5 **for** $l \leftarrow 1$ **to** L **do**
6 $\quad$ **for** ***each*** *node* $v \in V$ **do**
7 $\quad\quad temp[v] \leftarrow \mathbf{0}_{\mathcal{D}} = (0, \ldots, 0)$;
8 $\quad\quad$ **for** ***each*** *neighbour node* $u \in \mathcal{N}(v)$ **do**
9 $\quad\quad\quad temp[v] \leftarrow temp[v] + \big(W[u] + \mathcal{R}[u]\big) \times P[u, v]$;
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 $\quad$ **for** ***each*** *node* $v \in V$ **do**
13 $\quad\quad \mathcal{R}[v] \leftarrow \alpha \times temp[v]$;
14 $\quad$ **end**
15 **end**
16 **for** ***each*** *node* $v \in V$ **do**
17 $\quad \mathbf{E}[v] \leftarrow sign(\mathcal{R}[v])$;
18 **end**

$\mathcal{D}$. Hence, the overall time complexity of the algorithm is $\mathcal{O}\big((|\mathcal{V}| \cdot \mathcal{D} + m \cdot L \cdot \mathcal{D}\big)$. During the running course of the algorithm, we need to store a vector of size N in memory for the computation of each dimension. Assuming, in the worst case, that we aim to retain the whole projection matrix $\mathbf{W}$ in memory, we need $\mathcal{O}(N \cdot \mathcal{D})$ space in total, since each node requires $\mathcal{D}$ space for storing the $\mathcal{R}^{(l)}_{(v,d)}$ values in the update rule of Eq. (3). Note that, the performance of the algorithm can be boosted by using parallel processing for each dimension of embedding vectors or for Line 6, since the required computation for each node is completely independent.

IV. EXPERIMENTAL EVALUATION

We report empirical evaluation results demonstrating the effectiveness and efficiency of NODESIG compared to baselines. All the experiments have been performed on a server (16 Cores) with 128GB of memory.

A. Datasets and Baseline Models

Datasets. We perform experiments on networks of different scale and type. (i) *Blogcatalog* [33] social network; (ii) *Cora* [34] citation graph; (iii) *DBLP* [35] co-authorship network; (iv) *PPI* [11] is a protein-protein interaction network. (v) *Youtube* [36] is a social network in which node labels indicate categories of videos. All the networks used in the experiments are unweighted and undirected (the direction of edges are discarded), in order to be consistent in the evaluation. The characteristics of the graphs are shown in Table I.

Networks	# Nodes	# Edges	# Labels	# Density
Blogcatalog	10,312	333,983	39	6.3×10^{-3}
Cora	2,708	5,278	7	1.4×10^{-3}
DBLP	27,199	66,832	4	1.8×10^{-4}
PPI	3,890	38,739	50	5.1×10^{-3}
Youtube	1,138,499	2,990,443	47	4.6×10^{-6}

TABLE I: Characteristics of networks.

Baseline models. We have considered seven representative baseline methods in the evaluation. In particular, the first two correspond to widely used node embedding models: (i) a biased random walk-based model, NODE2VEC [11], and (ii) a matrix factorization algorithm, HOPE [7]. The remaining four baselines constitute recent models aiming to address the scalability challenge. (iii) NETSMF [17] is a sparse matrix factorization algorithm, modeling the pointwise mutual information of node co-occurrences. (iv) FREDE [26] is a matrix sketching-based approach. (v) RANDNE [8] leverages Gaussian random projections to deal with scalability. (vi) LOUVAINNE [18] constructs a hierarchical subgraph structure, aggregating the node representations learned at each level. Finally, (vii) NODESKETCH [27] learns embeddings in the *Hamming* space, using MINHASH signatures. For all methods, we learn embedding vectors of size 128.

For simplicity, we set the importance factor α to 1 in all the experiments of NODESIG, as we have observed that the algorithm shows comparable performance for values close to 1; a detailed analysis of the behaviour of NODESIG with respect to the importance factor is given in Section IV-D. The walk length is set to 3 for *Cora* and *Blogcatalog*, and to 5 for all the other networks in the classification experiment. For the link prediction task, the walk length is chosen as 15 for all networks. We set the dimension size of the embedding vectors to $8,192$ bits in order to be consistent with the experiments with the baseline methods, since modern computer architectures use 8 Bytes for storing floating point data types.

B. Multi-label Node Classification

Our goal is to correctly infer the labels of nodes chosen for the testing set, using the learned representations and the labels of nodes in the rest of the network, namely the nodes in the training set. The evaluation follows a strategy similar to the one used by baseline models [27].

1) Experimental set-up: The experiments are carried out by training an one-vs-rest *SVM* classifier with a pre-computed kernel, which is designed by computing the similarities of node embeddings. The similarity measure is chosen depending on the algorithm that we use to learn representations. More specifically, the *Hamming* similarity for NODESKETCH and the *Cosine* similarity for the rest baselines methods are chosen in order to build the kernels for the classifier. For NODESIG, we use the *chi* similarity χ, defined as $1 - \sqrt{d_{\chi^2}}$, where

$$d_{\chi^2} := \sum_{i=1}^{\mathcal{D}} \frac{(x_i - y_i)^2}{x_i + y_i} = \sum_{i=1}^{\mathcal{D}} (x_i + y_i) - \sum_{i=1}^{\mathcal{D}} \frac{4 x_i y_i}{x_i + y_i} = 2 - 2\rho_{\chi^2},$$

for the vectors satisfying $\sum_i x_i = \sum_i y_i = 1$ and $x_i \geq 0, y_i \geq 0$ for all $1 \leq i \leq \mathcal{D}$. Hence, we apply a small transformation

	Micro-F_1			Macro-F_1		
	10%	**50%**	**90%**	**10%**	**50%**	**90%**
HOPE	0.305	0.317	0.326	0.117	0.119	0.124
NODE2VEC	0.341	0.352	0.345	0.155	0.165	0.165
NETSMF	**0.360**	0.376	0.377	0.189	0.200	0.199
FREDE	0.354	0.368	0.381	0.171	0.179	0.183
LOUVAINNE	0.047	0.143	0.165	0.022	0.037	0.041
RANDNE	0.316	0.337	0.340	0.141	0.164	0.165
NODESKETCH	0.305	0.381	0.398	0.145	0.236	0.263
NODESIG	0.358	**0.408**	**0.420**	**0.191**	**0.267**	**0.286**

TABLE II: Micro-F_1 and Macro-F_1 classification scores for varying training set ratios of the *Blogcatalog* network.

	Micro-F_1			Macro-F_1		
	10%	**50%**	**90%**	**10%**	**50%**	**90%**
HOPE	0.687	0.780	0.797	0.671	0.772	0.786
NODE2VEC	0.764	0.813	0.831	0.749	0.802	0.818
NETSMF	0.763	0.824	0.831	0.751	0.815	0.821
FREDE	**0.777**	0.825	0.846	**0.766**	0.817	0.833
LOUVAINNE	0.686	0.711	0.721	0.648	0.675	0.683
RANDNE	0.583	0.676	0.693	0.557	0.668	0.686
NODESKETCH	0.648	0.825	0.872	0.632	0.818	0.866
NODESIG	0.750	**0.852**	**0.879**	0.736	**0.843**	**0.871**

TABLE III: Micro-F_1 and Macro-F_1 classification scores for varying training set ratios of the *Cora* network.

while constructing the kernel matrix of the SVM in order to approximate the *chi* similarity, instead of using $\cos^{-1}\rho_{\chi^2}/\pi$ in Eq. (2), which is estimated directly via the Hamming distance.

2) Experimental results: For the multi-label node classification task, Tables II-VI report the average Micro-F_1 and Macro-F_1 scores over 10 runs, where the experiments are performed on different training set sizes. The symbol "-" is used to indicate that the corresponding algorithm is unable to run due to excessive memory usage ($>$ 60GB) or because it requires more than one day to complete. The best and second best performing models for each training ratio (10%, 50%, and 90%) are indicated with bold and underlined text, respectively.

As we observe, NODESIG consistently outperforms the baselines for higher training ratios on the *Blogcatalog* and *Cora* networks, while the obtained Macro-F_1 score is very close to the performance of NETSMF for 10% training ratio

	Micro-F_1			Macro-F_1		
	10%	**50%**	**90%**	**10%**	**50%**	**90%**
HOPE	0.620	0.632	0.631	0.525	0.536	0.536
NODE2VEC	0.621	0.632	0.631	0.510	0.535	0.531
NETSMF	0.626	0.644	0.647	0.533	0.572	0.575
FREDE	0.648	0.661	0.661	0.567	0.586	0.588
LOUVAINNE	0.494	0.496	0.499	0.354	0.356	0.359
RANDNE	0.418	0.437	0.438	0.233	0.255	0.257
NODESKETCH	0.668	**0.847**	**0.903**	0.616	**0.831**	**0.891**
NODESIG	**0.704**	0.843	0.893	**0.660**	0.824	0.879

TABLE IV: Micro-F_1 and Macro-F_1 classification scores for varying training set ratios of the *DBLP* network.

	Micro-F_1			Macro-F_1		
	10%	**50%**	**90%**	**10%**	**50%**	**90%**
HOPE	0.134	0.151	0.146	0.083	0.085	0.077
NODE2VEC	0.141	0.161	0.138	0.084	0.087	0.070
NETSMF	0.150	0.170	0.163	0.096	0.102	0.095
FREDE	0.156	0.174	0.157	0.099	0.105	0.090
LOUVAINNE	0.042	0.054	0.056	0.023	0.025	0.021
RANDNE	0.145	0.161	0.145	0.087	0.091	0.083
NODESKETCH	0.152	0.227	0.243	0.102	0.181	**0.196**
NODESIG	**0.177**	**0.236**	**0.246**	**0.119**	**0.185**	0.191

TABLE V: Micro-F_1 and Macro-F_1 classification scores for varying training set ratios of the *PPI* network.

	Micro-F_1			Macro-F_1		
	10%	**50%**	**90%**	**10%**	**50%**	**90%**
HOPE	0.342	0.341	0.343	0.198	0.201	0.201
NODE2VEC	-	-	-	-	-	-
NETSMF	0.392	0.379	0.376	0.273	0.256	0.247
FREDE	-	-	-	-	-	-
LOUVAINNE	0.248	0.251	0.256	0.064	0.063	0.072
RANDNE	0.335	0.341	0.339	0.205	0.220	0.215
NODESKETCH	0.439	**0.467**	**0.476**	0.365	**0.412**	**0.426**
NODESIG	**0.455**	0.465	0.471	**0.387**	0.410	0.414

TABLE VI: Micro-F_1 and Macro-F_1 classification scores for varying training set ratios of the *Youtube* network.

on *Blogcatalog*. In the case of the *Cora* network which corresponds to the smallest one used in our study, FREDE shows better performance for small training ratio of 10%. For the *Youtube* and *DBLP* networks, the proposed NODESIG model along with *NodeSketch* perform equally well. This is quite surprising, since both these methods that correspond to data-independent hashing techniques offer a clear performance gain over traditional models, such as NODE2VEC and HOPE. Lastly, for the *PPI* dataset, NODESIG obtains consistently the highest scores for Micro-F_1, while its main competitor NODESKETCH has close performance for the Macro-F_1 score.

C. Link Prediction

The second downstream task used to assess the quality of node embeddings is the one of link prediction.

1) Experimental set-up: Half of the edges of a given network are removed by still keeping the residual network connected. Node embeddings are learned on the rest of the graph. The removed edges are considered as positive samples for the testing set, while the same number of node pairs which does not exist in the initial network is separately sampled for training and testing sets in order to form the negative samples. As it has been described in Section IV-B, we build the features corresponding to the node pair samples using the similarities between embedding vectors; the similarity measure is chosen depending on the algorithm that we use to extract the representations. Since *Youtube* is relatively larger than the rest of the networks, we work on 7% of its initial size. We predict edges by constructing the similarity list of edges, and we provide the *Area Under Curve* (AUC) scores in Table VII.

	Blogcatalog	Cora	DBLP	PPI	Youtube
HOPE	0.517	0.665	0.769	0.524	0.514
NODE2VEC	0.595	0.748	0.843	0.616	0.533
NETSMF	0.691	0.709	0.835	0.534	**0.542**
FREDE	0.709	**0.760**	**0.858**	0.451	0.460
LOUVAINNE	0.565	0.684	0.789	0.570	0.528
RANDNE	0.608	0.508	0.517	0.505	0.502
NODESKETCH	0.703	0.590	0.714	0.514	0.510
NODESIG	**0.822**	0.737	0.856	**0.654**	0.537

TABLE VII: Area Under Curve scores for link prediction.

2) Experimental results: For the link prediction task, NODESIG acquires the highest AUC scores on three datasets, while it is also the second-best performing model for the remaining two. In the case of the *Youtube* dataset, all baselines demonstrate comparable results. Although NODE2VEC shows good performance across most datasets in the link prediction task, it does not perform well on the *Blogcatalog* network, mainly because of its high density. On the other hand, NODESIG reaches the highest score on this dataset, with a clear difference to its main competitor, NODESKETCH.

D. Parameter Sensitivity

We concentrate on the influence of three parameters, namely walk length L, importance factor α and dimension size $\mathcal{D}$, examining their impacts on the *Cora* network.

1) Effect of walk length: In order to examine the influence of the walk length on the performance, we perform experiments for varying lengths by fixing the importance factor α to 1.0. Figure 3a depicts the Micro-F_1 scores for different training ratios. We observe a significant increase in performance when the walk length increases, particularly for small training ratios and walk lengths. Although it shows a wavy behavior for the largest training ratio, there is a logarithmic improvement depending on the walk length. NODESIG better captures the structural properties of the network in longer walks, thus the low performance observed on small training ratios can be compensated with longer walks.

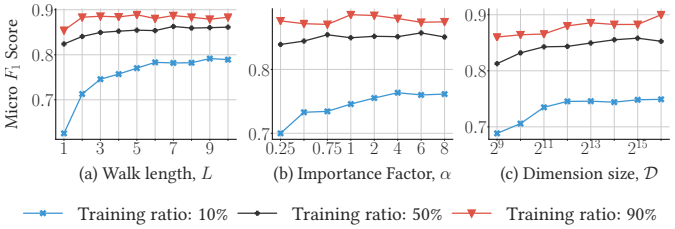

Fig. 3: Influence of various parameters in terms of Micro-F_1 score on the *Cora* network for varying training set ratios.

2) Effect of importance factor: The importance factor controls the impact of walks of different lengths: the importance of the higher levels is increasing for $\alpha > 1$, while it can be diminished choosing $\alpha < 1$. Figure 3b depicts the performance of NODESIG on the *Cora* network, fixing the walk length value to 3. Although we do not observe a steady behavior for the large training set, higher values of α, especially around 4, positively contribute to the performance; values smaller than 1 have negative impact on the performance.

3) Effect of dimension size: The dimension size is a crucial parameter affecting the performance of the algorithm, since a better approximation to the *chi* similarity measure can be obtained for larger dimension sizes, following *Hoeffding's* inequality [37]. Therefore, we perform experiments for varying dimension sizes, by fixing the walk length to 5. Figure 3c depicts the Micro-F_1 scores of the classification experiment for different dimension sizes ranging from 2^9 to 2^{17}. Although we have fluctuating scores on the large training set due to the randomized behavior of the approach, the impact of the dimension size can be observed clearly on the small training set size. On the other hand, we observe an almost stable behavior for the training ratio of 50%, encouraging the use of small embedding sizes towards reducing storage requirements.

E. Time Comparison

We have recorded the elapsed real (wall clock) time of all methods, and the results are provided in Table VIII. The *Random* network indicates the $\mathcal{G}_{n,p}$ Erdös-Renyi random graph model, using $n = 10^5$ and $p = 10^{-4}$. All the experiments have been conducted on the server whose specifications given in Section IV. We use 32 threads for each algorithm, when it is applicable. We have utilized the suggested default parameters for the baselines, and the settings described for the classification task are employed for NODESIG.

	Blogcatalog	*Cora*	*DBLP*	*PPI*	*Youtube*	*Random*	*Speedup*
HOPE	97.81	27.32	198.59	32.65	8470.33	1048.52	8.85x
NODE2VEC	1400.44	18.32	161.24	72.16	-	716.07	2.55x
NETSMF	7.78	1.32	10.91	1.90	1624.94	236.30	1.69x
FREDE	1179.79	20.46	2612.84	140.43	-	22386.98	28.33x
LOUVAINNE	0.34	0.06	0.24	0.11	6.86	1.25	0.01x
RANDNE	25.52	3.15	11.55	5.40	449.15	73.11	0.51x
NODESKETCH	64.21	13.42	19.10	14.40	1563.00	101.16	1.59x
NODESIG	17.40	0.74	9.26	2.53	1047.53	38.11	1.00x

TABLE VIII: Running time (in seconds) and average speedup.

As we observe, NODESIG runs faster than HOPE, NODE2VEC as well as FREDE. This is happening because HOPE requires an expensive matrix factorization, while NODE2VEC needs to simulate random walks to obtain their exact realizations. Although FREDE is a sketching-based approach, we have observed that the computation of the PPR matrix requires considerable time. Furthermore, although the remaining baseline methods run faster compared to NODESIG, as we have already presented, the proposed model generally outperforms them both in the classification and link prediction tasks. These experiments further support the intuition about designing NODESIG as an expressive model that balances accuracy and running time.

V. DISCUSSION FOR DYNAMIC NETWORKS

Most real-world networks undergo structural changes and evolve over time with the addition and removal of links and

nodes [38]. Therefore, designing models properly adapting to dynamic networks is an important point to investigate. As we discuss here, the proposed method allows for efficient updates of the embeddings, without requiring any costly learning procedures. More precisely, the key point in the dynamic case, is that the learned embedding vectors should be efficiently updated instead of being recalculated from scratch. If an edge is added or removed for a pair of nodes $(u, v) \in V \times V$, the terms $\mathcal{R}^{(l)}_{(w,:)}$ in Eq. (3) for node $w \in V$ are affected, for all $l > k := \min\{dist(w, u), dist(w, v)\}$—thus, it suffices to update only these affected terms. The transition probabilities for nodes u and v also change even though the remaining nodes are not affected, so all the terms $P_{(v,:)}$ must be divided by $\sum_{w \in \mathcal{N}(v)} P_{(v,w)}$ in order to normalize the transition probabilities and similarly the same procedure must also be applied to node u after each edge insertion and deletion operation.

VI. Conclusion and Future Work

We have introduced NodeSig, an efficient binary node embedding model. Its components have properly been designed to improve scalability without sacrificing effectiveness on downstream tasks. NodeSig exploits random walk diffusion probabilities via stable random projection hashing, towards efficiently computing representations in the Hamming space that approximate the *chi* similarity. The experimental results have demonstrated that NodeSig outperformed in accuracy recent highly-scalable models, being able to run within the reasonable time duration, while at the same time it shows comparable or even better accuracy with respect to widely used baseline methods in multi-label node classification and link prediction. In future work, we plan to further study the properties of the model for attributed and dynamic networks and also study the performance of parallel/distributed alternatives.

References

[1] M. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, pp. 52–74, 2017.

[3] W. L. Hamilton, *Graph Representation Learning*. Morgan and Claypool Publishers, 2020.

[4] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[5] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.

[6] A. Celikkanat, Y. Shen, and F. D. Malliaros, "Multiple kernel representation learning on networks," *IEEE Trans. Knowl. Data Eng.*, 2022.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *KDD*, 2016, pp. 1105–1114.

[8] Z. Zhang, P. Cui, H. Li, X. Wang, and W. Zhu, "Billion-scale network embedding with iterative random projection," in *ICDM*, 2018, pp. 787–796.

[9] H. Chen, S. F. Sultan, Y. Tian, M. Chen, and S. Skiena, "Fast and accurate network embeddings via very sparse random projection," in *CIKM*, 2019, pp. 399–408.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[11] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[13] D. Nguyen and F. D. Malliaros, "BiasedWalk: Biased sampling for representation learning on graphs," in *BigData*, 2018, pp. 4045–4053.

[14] A. Çelikkanat and F. D. Malliaros, "Exponential family graph embeddings," in *AAAI*, 2020, pp. 3357–3364.

[15] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *WSDM*, 2018, pp. 459–467.

[16] S. Chanpuriya and C. Musco, "Infinitewalk: Deep network embeddings as laplacian embeddings with a nonlinearity," in *KDD*, 2020, p. 1325–1333.

[17] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, "NetSMF: Large-scale network embedding as sparse matrix factorization," in *WWW*, 2019, pp. 1509–1520.

[18] A. K. Bhowmick, K. Meneni, M. Danisch, J.-L. Guillaume, and B. Mitra, "Louvainne: Hierarchical louvain method for high quality and scalable network embedding," in *WSDM*, 2020, pp. 43–51.

[19] D. Lian, K. Zheng, V. W. Zheng, Y. Ge, L. Cao, I. W. Tsang, and X. Xie, "High-order proximity preserving information network hashing," in *KDD*, 2018, pp. 1744–1753.

[20] W. Wu, B. Li, L. Chen, and C. Zhang, "Efficient attributed network embedding via recursive randomized hashing," in *IJCAI*, 2018, pp. 2861–2867.

[21] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, 2018.

[22] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *ECCV*, 2010, pp. 749–762.

[23] C. Ye, J. Wu, V. S. Sheng, S. Zhao, P. Zhao, and Z. Cui, "Multi-label active learning with chi-square statistics for image classification," in *ICMR*, 2015, pp. 583–586.

[24] V. T. L. Huong, D. Park, D. Woo, and Yunsik Lee, "Centroid neural network with chi square distance measure for texture classification," in *IJCNN*, 2009, pp. 1310–1315.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[26] A. Tsitsulin, M. Munkhoeva, D. Mottin, P. Karras, I. Oseledets, and E. Müller, "Frede: Anytime graph embeddings," *Proc. VLDB Endow.*, vol. 14, no. 6, p. 1102–1110, feb 2021.

[27] D. Yang, P. Rosso, B. Li, and P. Cudre-Mauroux, "Nodesketch: Highly-efficient graph embeddings via recursive sketching," in *KDD*, 2019, pp. 1162–1172.

[28] S. Vempala, *The random projection method*. Am. Math Soc., 2001.

[29] X. Yi, C. Caramanis, and E. Price, "Binary embedding: Fundamental limits and fast algorithm," in *ICML*, 2015, pp. 2162–2170.

[30] W. B. Johnson, J. Lindenstrauss, and G. Schechtman, "Extensions of lipschitz maps into banach spaces," *Israel Journal of Mathematics*, vol. 54, pp. 129–138, 1986.

[31] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, 1995.

[32] P. Li, G. Samorodnitsky, and J. Hopcroft, "Sign cauchy projections and chi-square kernel," in *NIPS*, 2013.

[33] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *KDD*, 2009, pp. 817–826.

[34] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, 2008.

[35] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip! online learning of multi-scale network embeddings," in *ASONAM*, 2017, pp. 258–265.

[36] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *CIKM*, 2009, pp. 1107–1116.

[37] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Am. Stat. Assoc.*, vol. 58, pp. 13–30, 1963.

[38] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, no. 70, pp. 1–73, 2020.