

PSNE: Efficient Spectral Sparsification Algorithms for Scaling Network Embedding

Longlong Lin
College of Computer and
Information Science,
Southwest University
Chongqing, China
longlonglin@swu.edu.cn

Yunfeng Yu
College of Computer and
Information Science,
Southwest University
Chongqing, China
YunfengYu817@outlook.com

Zihao Wang
College of Computer and
Information Science,
Southwest University
Chongqing, China
zihaoawang@outlook.com

Zeli Wang
Chongqing University of
Posts and
Telecommunications
Chongqing, China
zlwang@cqupt.edu.cn

Yuying Zhao
Vanderbilt University
Nashville, USA
yuying.zhao@vanderbilt.edu

Jin Zhao
Huazhong University of
Science and Technology
Wuhan, China
zjin@hust.edu.cn

Tao Jia*
College of Computer and
Information Science,
Southwest University
Chongqing, China
tjia@swu.edu.cn

Abstract

Network embedding has numerous practical applications and has received extensive attention in graph learning, which aims at mapping vertices into a low-dimensional and continuous dense vector space by preserving the underlying structural properties of the graph. Many network embedding methods have been proposed, among which factorization of the Personalized PageRank (PPR for short) matrix has been empirically and theoretically well supported recently. However, several fundamental issues cannot be addressed. (1) Existing methods invoke a seminal Local Push subroutine to approximate a single row or column of the PPR matrix. Thus, they have to execute n (n is the number of nodes) Local Push subroutines to obtain a provable PPR matrix, resulting in prohibitively high computational costs for large n . (2) The PPR matrix has limited power in capturing the structural similarity between vertices, leading to performance degradation. To overcome these dilemmas, we propose PSNE, an efficient spectral sParsification method for Scaling Network Embedding, which can fast obtain the embedding vectors that retain strong structural similarities. Specifically, PSNE first designs a matrix polynomial sparser to accelerate the calculation of the PPR matrix, which has a theoretical guarantee in terms of the Frobenius norm. Subsequently, PSNE proposes a simple but effective multiple-perspective strategy to enhance further the representation power of the obtained approximate PPR matrix. Finally, PSNE applies a randomized singular value decomposition algorithm on the sparse and multiple-perspective PPR matrix to get the target

embedding vectors. Experimental evaluation of real-world and synthetic datasets shows that our solutions are indeed more efficient, effective, and scalable compared with ten competitors.

CCS Concepts

• **Computing methodologies** → *Learning latent representations.*

Keywords

Network Embedding; Spectral Graph Theory

ACM Reference Format:

Longlong Lin, Yunfeng Yu, Zihao Wang, Zeli Wang, Yuying Zhao, Jin Zhao, and Tao Jia. 2024. PSNE: Efficient Spectral Sparsification Algorithms for Scaling Network Embedding. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*, October 21–25, 2024, Boise, ID, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3627673.3679540>

1 Introduction

Graphs are ubiquitous for modeling real-world complex systems, including financial networks, biological networks, social networks, etc. Analyzing and understanding the semantic information behind these graphs is a fundamental problem in graph analysis [2, 17, 24, 25, 49]. Thus, numerous graph analysis tasks are arising, such as node classification, link prediction, and graph clustering. Due to their exceptional performance, network embedding is recognized as an effective tool for solving these tasks. Specifically, given an input graph G with n nodes, network embedding methods aim at mapping any node $v \in G$ to a low-dimensional and continuous dense vector space $x_v \in \mathbb{R}^k$ (k is the dimension size and $k \ll n$) such that the embedding vectors can unfold the underlying structural properties of graphs. Thus, the obtained embedding vectors can be effectively applied to these downstream tasks mentioned above [16, 48].

Many network embedding methods have been proposed in the literature, among which matrix factorization has been empirically and theoretically shown to be superior to Skip-Gram based random walk methods and deep learning based methods [33, 44, 46], as stated in Section 2. Specifically, matrix factorization based solutions

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).
CIKM '24, October 21–25, 2024, Boise, ID, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0436-9/24/10
<https://doi.org/10.1145/3627673.3679540>

first construct a proximity matrix S according to their corresponding applications, in which $S(i, j)$ represents the relative importance of node j with respect to (w.r.t.) node i . Then, the traditional singular value decomposition algorithm is executed on S or some variants of S to obtain the target embedding vectors. Thus, different proximity matrices were designed and the Personalized PageRank (PPR) matrix emerges as a superior choice due to its good performance [46, 47, 51]. In particular, given two nodes $i, j \in G$, the PPR value $\Pi(i, j)$ is the probability that a random walk starts from i and stops at j using k steps state transition, in which k follows the geometric distribution. Therefore, the PPR values can be regarded as the concise summary of an infinite number of random walks, possessing nice structural properties and strong interpretability for network embedding.

Despite their success, most existing methods suffer from the following several fundamental issues: (1) **They either have high costs to achieve provable performance, or hard to obtain empirical satisfactory embedding vectors for downstream tasks.** Specifically, the state-of-the-art (SOTA) methods, such as STRAP [47] and Lemane [51], applied the seminal Local Push subroutine [3] to approximate a *single* row or column of the PPR matrix, resulting in that have high overheads, especially for massive graphs. For example, Ref. [23] reported that the current fastest Local Push algorithm takes about 2 seconds on a million-node YouTube graph. Thus, the Local Push algorithm takes over 23 days to compute the PPR matrix, which is infeasible even for a powerful computing cluster. For reducing computational costs, NRP [46] integrated the calculation and factorization of the PPR matrix in an iterative framework. However, NRP has unsatisfactory theoretical approximation errors and poor empirical performance because it loses the nonlinear capability, as stated in our experiments. (2) Existing methods are highly dependent on the assumption that the PPR matrix can reflect the structural similarity between two nodes [30]. Namely, larger PPR values, denoted as $\Pi(i, j)$, correspond to the higher structural similarity between nodes i and j . **However, the original PPR matrix has limited power in capturing the structural similarity between vertices, leading to performance degradation.** Take Figure 1 as an example, we can see that $\Pi(v_1, v_3)$ (= 0.054) is almost three times as many as $\Pi(v_1, v_7)$ (= 0.140). Thus, v_1 is more similar to v_7 than v_3 according to the PPR metric. However, v_1 and v_7 have no common neighbors, and their walking trajectories are not similar. On the contrary, v_1 and v_3 share their only neighbor v_2 and have almost the same walking trajectories, which is a key property used to characterize whether nodes are in the same cluster [6, 45]. Specifically, let $\mathcal{P}(u)$ be all walking trajectories starting with u , we have $\mathcal{P}(v_1) = \{\{v_1, p(v_2)\} | p(v_2) \in \mathcal{P}(v_2)\}$ and $\mathcal{P}(v_3) = \{\{v_3, p(v_2)\} | p(v_2) \in \mathcal{P}(v_2)\}$. Thus, $\mathcal{P}(v_1)$ and $\mathcal{P}(v_3)$ are identical except for the starting vertex. As a result, by the PPR metric, the classifier tends to mistakenly assign v_1 and v_7 to the same class even though v_1 and v_3 belong to the same community and have stronger structural similarities. So, exploring alternative methods for overcoming these limitations remains a huge challenge.

To this end, we propose a novel and efficient spectral sParsification method for Scaling Network Embedding (PSNE). Specifically, PSNE first non-trivially utilizes the theories of spectral sparsification and random-walk matrix polynomials [9, 35] to *directly* construct a sparse PPR matrix with a theoretical guarantee in terms of the

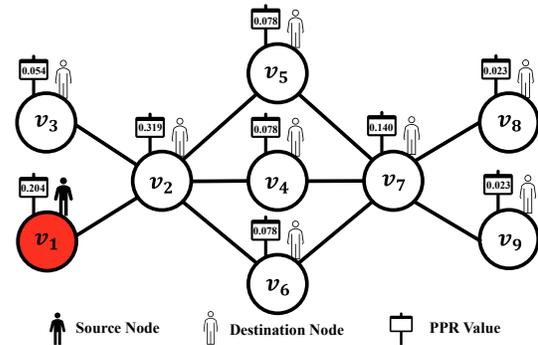


Figure 1: v_1 is the source node and $\alpha = 0.15$ is the decay parameter in PPR. The PPR values $\Pi(v_1, v_3)$ and $\Pi(v_1, v_7)$ are 0.054 and 0.140, respectively. The proposed multiple-perspective PPR values $M(v_1, v_3)$ and $M(v_1, v_7)$ are 0.142 and 0.095, respectively (see Table 1 for details).

Frobenius norm (Theorem 4.6), which avoids repeatedly computing each row or column of the PPR matrix, reducing greatly the computational overheads. Then, a simple yet effective multiple-perspective strategy (Section 4.2) is proposed to further enhance the representation power of the approximate PPR matrix, which can alleviate the inherent defects of the original PPR metric. Finally, PSNE employs a randomized singular value decomposition algorithm to efficiently factorize the sparse and multiple-perspective PPR matrix and obtain high-quality target embedding vectors. In a nutshell, we highlight our contributions as follows.

- We are the first to adopt the spectral sparsification theory to *directly* approximate the whole PPR matrix, circumventing the expensive costs for a single row or column of the PPR matrix in existing push-based methods.
- We devise a simple but effective multiple-perspective strategy to further enhance the representation power of the approximate PPR matrix. A striking feature of the strategy is that it also can be generalized to improve the qualities of SOTA baselines.
- Empirical results on real-world and synthetic datasets show that our proposed PSNE outperforms the quality by at least 2% than ten competitors in most cases. Besides, PSNE is also more efficient than existing PPR-based methods without sacrificing accuracy, showing a better trade-off between efficiency and accuracy.

2 Related Work

2.1 Random Walk Based Network Embedding

Random walk based methods are inspired by the Skip-Gram model [27]. The high-level idea is to obtain embedding vectors by keeping the co-occurrence probability of the vertices on the random walks. The main difference between these methods is how to generate positive samples by different random walk strategies. For example, *DeepWalk* [31] utilized truncated random walks. *Line* [36] and *Node2vec* [14] extended *DeepWalk* with more complicated higher-order random walks or DFS/BFS search schemes. *APP* [53] and *VERSE* [38] adopted the α -discounted random walks to obtain positive samples. However, a shared challenge of these methods is high computational costs due to the training of the Skip-Gram model.

2.2 Deep Learning Based Network Embedding

Deep learning provides an alternative solution to generate embedding vectors. For example, *SDNE* [41] utilized multi-layer auto-encoders to generate embedding vectors. *DNGR* [8] combined random walk and deep auto-encoder for network embedding. *PRUNE* [21] applied the Siamese Neural Network to retain both the point-wise mutual information and the PageRank distribution. *GraphGAN* [42] and *DWNS* [11] employed generative adversarial networks [10] to capture the probability of node connectivity in a precise manner. *AW* [1] proposed an attention model that operates on the power series of the transition matrix. Note that although the Graph Neural Network (GNN) with feature information [20] has achieved great success in many tasks, network embedding, which only uses the graph topology like our paper, is still irreplaceable. Specifically, (1) obtaining the rich node feature information is very expensive and even is not always available for downstream tasks, resulting in limited applications [12]. (2) Existing GNNs are typically end-to-end and need different training processes for different downstream tasks, leading to inflexibility. On the contrary, by focusing on the graph topology, network embedding provides a structure feature for each node, which is independent of downstream tasks [37]. Thus, network embedding provides a trade-off between the accuracy of downstream tasks and the training cost. In short, the main bottlenecks of these deep learning methods are high computational costs and labeling costs, which fail to deal with massive graphs.

2.3 Matrix Based Network Embedding

Other popular methods are to factorize a pre-defined proximity matrix that reflects the structural properties of the graph. For example, *GraRep*[7] performs SVD on the k -th order transition matrix. *NetMF* [33] demonstrated the equivalence between random walk based methods and matrix factorization based methods. *NetSMF* [32] combined *NetMF* with sparsification techniques to further improve efficiency of *NetMF*. However, *NetSMF* cannot effectively capture the non-uniform higher-order topological information because *NetSMF* inherits the defects of *DeepWalk*, resulting in poor practical performance in most cases, as stated in our empirical results. *ProNE* [50] utilized matrix factorization and spectral propagation to obtain embedding vectors. *STRAP* [47] adopted the PPR matrix as the proximity matrix for improving the performances of *NetMF* and *NetSMF*. However, *STRAP* applied the seminal Local Push subroutine [3] to approximate a single row or column of the PPR matrix, resulting in prohibitively high time&space overheads. *HOPE* [29], *AROPE* [52], *NRP* [46], *FREDE*[39], and *SketchNE*[44] derived embedding vectors by implicitly computing the proximity matrix. Thus, they abandoned nonlinear operations on proximity matrices, which limits their representation powers. *Lemane* [51] considered the decay factor α in PPR should not be fixed but learnable, resulting in more flexibility. However, this learning process brings high overheads for *Lemane*.

3 Preliminaries

We use $G(V, E)$ to denote an undirected graph, in which V and E are the vertex set and the edge set of G , respectively. Let $n = |V|$ (resp., $m = |E|$) be the number of vertices (resp., edges). A is the adjacency matrix with A_{ij} as the element of i -th row and j -th column of A ,

$D = \text{diag}(d_1, \dots, d_n)$ is the degree matrix with $d_i = \sum_j A_{ij}$, $L = D - A$ be the Laplacian matrix, $P = D^{-1}A$ be the state transition matrix.

Personalized PageRank (PPR) is the state-of-the-art proximity metric, which can measure the relative importance of nodes [3, 26]. The PPR value $\Pi(u, v)$ is the probability that an α -decay random walk from u stops at node v , in which an α -decay random walk has α probability to stop at the current node, or $(1 - \alpha)$ probability to randomly jump to one of its neighbors. Thus, the length of α -decay random walk follows the geometric distribution with success probability α . The PPR matrix Π are formulated as follows:

$$\Pi = \sum_{r=0}^{\infty} \alpha(1 - \alpha)^r \cdot P^r \quad (1)$$

Problem Statement. Given an undirected graph $G(V, E)$, the network embedding problem aims to obtain a mapping function $f : V \rightarrow \mathbb{R}^k$, in which k is a positive integer representing the embedding dimension size and $k \ll n$. An effective network embedding function f unfolds the underlying structural properties of graphs.

4 PSNE: Our Proposed Solution

Here, we introduce a novel and efficient spectral sParsification algorithm PSNE for Scaling Network Embedding. PSNE first applies non-trivially the spectral graph theories to sparse the PPR matrix with theoretical guarantees. Subsequently, PSNE devises multiple-perspective strategies to further enhance the representation power of the sparse PPR matrix. Finally, a random singular value decomposition algorithm is executed on the refined sparse PPR matrix to obtain target embedding. Figure 2 is the framework of PSNE.

4.1 Spectral Sparsification for PPR Matrix

Definition 4.1 (Random-Walk Matrix Polynomials). For an undirected graph G and a non-negative vector $\beta = (\beta_1, \dots, \beta_T)$ with $\sum_{i=1}^T \beta_i = 1$, the matrix

$$L_{\beta}(G) = D - \sum_{r=1}^T \beta_r D (D^{-1}A)^r \quad (2)$$

is a T -degree random-walk matrix polynomial of G .

THEOREM 4.2. [Sparsifiers of Random-Walk Matrix Polynomials] For any undirected graph G and $0 < \epsilon \leq 0.5$, there exists a matrix \tilde{L} with $O(n \log n / \epsilon^2)$ non-zeros entries such that for any $x \in \mathbb{R}^n$, we have

$$(1 - \epsilon)x^T \tilde{L}x \leq x^T L_{\beta}(G)x \leq (1 + \epsilon)x^T \tilde{L}x \quad (3)$$

The matrix \tilde{L} satisfying Equation 3 is called spectrally similar with approximation parameter ϵ to $L_{\beta}(G)$, which can be constructed by the two-stage computing framework [9]. In the first stage, an initial sparsifier with $O(Tm \log n / \epsilon^2)$ non-zero entries is found. In the second stage, a standard spectral sparsification algorithm [35] is applied in the initial sparsifier to further reduce the number of non-zero entries to $O(n \log n / \epsilon^2)$. Note that the second stage requires complex graph theory to understand and consumes most of the time of the two-stage computing framework. Thus, in this paper, we first non-trivially utilize the first stage to obtain the coarse-grained sparsifier quickly. Then, we propose an effective multiple-perspective strategy to enhance the representation power

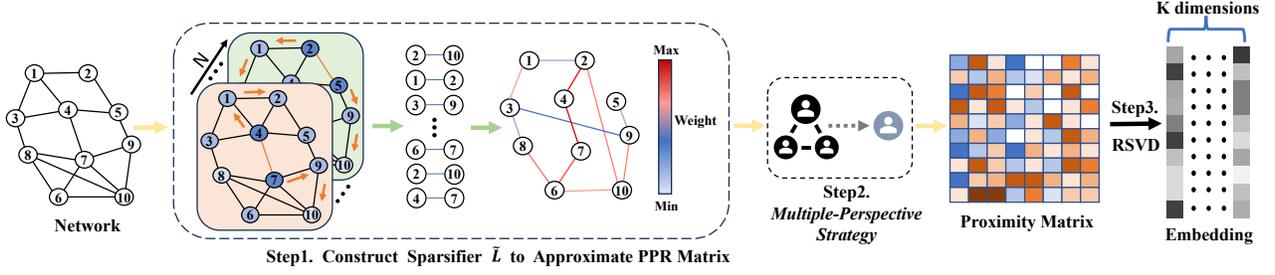


Figure 2: The design of PSNE framework. In Step 1 (i.e., Section 4.1), PSNE first constructs the sparsifier \tilde{L} by sampling N paths and assigning weights to the newly sampled edges. Then, Equations 4-9 and \tilde{L} are applied to *directly* approximate the PPR matrix, avoiding repeatedly computing each row or column of the PPR matrix in the traditional Local Push method. In Step 2 (i.e., Section 4.2), PSNE devises a multiple-perspective strategy to further enhance the representation of the coarse-grained and sparse PPR matrix obtained by Step 1. In Step 3 (i.e., Section 4.3), a randomized singular value decomposition (RSVD) algorithm is executed on the sparse and multiple-perspective PPR proximity matrix to obtain the target embedding matrix.

of the coarse-grained sparsifier in the next subsection. Specifically, the T -truncated PPR matrix is given as follows:

$$\Pi' = \Pi - \sum_{r=T+1}^{+\infty} \alpha(1-\alpha)^r \mathbf{P}^r = \sum_{r=0}^T \alpha(1-\alpha)^r \mathbf{P}^r \quad (4)$$

where T is the truncation order (T is also T in Definition 4.1). By combining Equation 2 and Equation 4, we have

$$\Pi' = \alpha \mathbf{I} + \sum_{r=1}^T \alpha(1-\alpha)^r \mathbf{P}^r \quad (5)$$

$$= \alpha \mathbf{I} + \mathbf{D}^{-1} \cdot \sum_{r=1}^T \alpha(1-\alpha)^r \mathbf{D} \mathbf{P}^r \quad (6)$$

$$= \alpha \mathbf{I} + \alpha_{sum} \mathbf{D}^{-1} \cdot \sum_{r=1}^T \alpha(1-\alpha)^r / \alpha_{sum} \mathbf{D} \mathbf{P}^r \quad (7)$$

$$= \alpha \mathbf{I} + \alpha_{sum} \mathbf{D}^{-1} \cdot (\mathbf{D} - \mathbf{L}_\beta(G)) \quad (8)$$

$$= \alpha \mathbf{I} + \alpha_{sum} \cdot (\mathbf{I} - \mathbf{D}^{-1} \mathbf{L}_\beta(G)) \quad (9)$$

Where $\alpha_{sum} = \sum_{i=1}^T \alpha(1-\alpha)^i$. Therefore, we establish a theoretical connection between the (truncated) PPR matrix and random-walk matrix polynomials. Based on this connection, we devise a novel sparsifier to obtain the approximate PPR matrix (Algorithm 1), reducing greatly the prohibitively high computational cost of existing local push-based embedding methods. Specifically, Algorithm 1 first initializes an undirected graph $\tilde{G} = (V, \emptyset)$, in which V is the vertex set of the input graph G (Line 1). Subsequently, Lines 2-7 of Algorithm 1 adds $O(N)$ edges to \tilde{G} by executing iteratively the Path_Sampling Function (Lines 11-18). Finally, Algorithm 1 applies Equations 4-9 to get an approximate PPR matrix $\tilde{\Pi}$ with $O(N)$ non-zeros entries (Lines 8-10).

Remark. The path length r in Line 4 is selected with the probability $\alpha(1-\alpha)^r / \sum_{i=1}^T \alpha(1-\alpha)^i$ for satisfying the condition of Definition 4.1, that is $\alpha_r = \alpha(1-\alpha)^r / \sum_{i=1}^T \alpha(1-\alpha)^i$ and $\sum_{r=1}^T \alpha_r = 1$. As a result, Algorithm 1 can obtain a sparse PPR matrix with a theoretical guarantee in terms of the Frobenius norm, which will be analyzed theoretically later. Besides, Algorithm 1 also leverages the intuition

Algorithm 1 Sparsifier of the PPR Matrix

Input: An undirected graph $G(V, E)$; the truncation order T ; the number of non-zeros N in the sparsifier; the decay factor α of PPR

Output: A sparse PPR matrix $\tilde{\Pi}$

- 1: Initializing an undirected graph $\tilde{G} = (V, \emptyset)$
 - 2: **for** $i = 1$ to N **do**
 - 3: Uniformly pick an edge $e = (u, v) \in E$
 - 4: Pick an integer $r \in [1, T]$ with a probability of $\alpha(1-\alpha)^r / \sum_{i=1}^T \alpha(1-\alpha)^i$
 - 5: $u', v', Z_p \leftarrow \text{Path_Sampling}(e, r)$
 - 6: Add weight $2rm / (NZ_p)$ to the edge (u', v') of \tilde{G}
 - 7: **end for**
 - 8: $\tilde{L} \leftarrow$ the Laplacian matrix of \tilde{G}
 - 9: $\tilde{\Pi} \leftarrow \alpha \mathbf{I} + \alpha_{sum} (\mathbf{I} - \mathbf{D}^{-1} \tilde{L})$
 - 10: **return** $\tilde{\Pi}$
 - 11: **Function** Path_Sampling($e = (u, v), r$) :
 - 12: Uniformly pick an integer $j \in [1, r]$
 - 13: Perform $(j-1)$ -step random walk from u to n_0
 - 14: Record anonymous trajectory from u to n_0 ($AnoTra_u$)
 - 15: Perform $(r-j)$ -step random walk from v to n_r
 - 16: Record anonymous trajectory from v to n_r ($AnoTra_v$)
 - 17: Calculate pattern similarity via $AnoTra_u$ and $AnoTra_v$ for the multiple-perspective strategy of Section 4.2
 - 18: **return** $n_0, n_r, \sum_{i=1}^r \frac{2}{A(n_{i-1}, n_i)}$
-

that closer nodes exhibit a higher propensity for information exchange. Therefore, the shorter the random walk, the greater the probability of being selected to promote local interactions, enabling more effective capture non-uniform high-order structural proximities among vertices for obtaining high-quality embedding vectors, which is verified in our experiments.

4.2 From PPR to Multiple-Perspective PPR

As depicted in Figure 1, the original PPR metric cannot effectively capture the structural similarity between vertices. Besides, if we

Table 1: Illustration of the original PPR and Multiple-Perspective PPR (MP-PPR) values on Figure 1.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
PPR	0.204	0.319	0.054	0.078	0.078	0.078	0.140	0.023	0.023
MP-PPR	0.142	0.180	0.142	0.145	0.145	0.145	0.095	0.062	0.062

directly take the approximate PPR matrix obtained by Section 4.1 as the input to matrix factorization, there will also be performance degradation. To overcome these issues, we propose a simple yet effective multiple-perspective strategy to achieve the following goals: (1) Alleviating the inherent defects of the original PPR metric; (2) Enhancing the representation power of the coarse-grained and sparse PPR matrix obtained in Section 4.1.

Through deep observation as shown in Figure 1, we found that the original PPR measures node pair proximity from a *single perspective* and ignores *pattern similarity*. For example, in a social network with users A , B , and C . C is the close friend of A , and A 's assessment of B is not only influenced by A 's subjective impression of B but also indirectly influenced by C 's impression of B . However, the original PPR metric only considers A 's impression of B . On top of that, the original PPR fails to illustrate that node v_1 exhibits a structural pattern more similar to nodes v_3 rather than v_7 .

Based on these intuitions, we propose a novel metric of Multiple-Perspective PPR (MP-PPR) to effectively compute the structural proximity between destination nodes and the source node with pattern similarity. Since considering the perspective of all nodes is computationally expensive with little performance improvement, we integrate only the perspectives of the one-hop neighbors of the source node. Consequently, the multiple-perspective proximity of destination node j w.r.t. the source node i is stated as follows.

$$M(i, j)_S = \sum_{h \in \mathcal{N}(i)} \lambda_{hi} S_{hj} + \lambda_{ii} S_{ij} \quad (10)$$

where S is any proximity matrix (e.g., the PPR matrix) and $\mathcal{N}(i)$ represents the neighbor set of i . λ_{hi} (resp., λ_{ii}) is the multiple-perspective coefficient of node h (resp., i) to node i . In this paper, we take $\lambda_{hi} = wp_{hi} / (\sqrt{(d_h + 1)} \sqrt{(d_i + 1)})$ and $\lambda_{ii} = 1 / (d_i + 1)$ where wp_{hi} is pattern similarity between (v_h, v_j) . To characterize the pattern similarity between nodes, we introduce the well-known anonymous walk [19] as follows.

Definition 4.3 (Anonymous Walk). If $A = (v_1, v_2, \dots, v_n)$ is a random walk trajectory, then its corresponding anonymous walk is the sequence of integers $AnoTra_A = (f(v_1), f(v_2), \dots, f(v_n))$, where $f(\cdot)$ is a mapping that maps nodes to positive integers.

Different nodes on the same trajectories are mapped to different positive integers, which may coincide on different paths. For example, trajectories $P_A = (v_1, v_2, v_3, v_2, v_3)$ and $P_B = (v_3, v_4, v_2, v_4, v_2)$ share the common anonymous trajectory $(1, 2, 3, 2, 3)$. We utilize the anonymous walk for pattern similarity calculation, which demands extensive trajectory sampling, limiting scalability. However, we have identified specific features in Equation 10 and Algorithm 1 as follows. (1) Equation 10 focuses on pattern similarity of neighboring nodes within one hop, reducing initial sampling needs. (2) Algorithm 1 already includes extensive path sampling, allowing for reduced sampling in anonymous random walks through strategic design (Lines 14 and 16 of Algorithm1).

Algorithm 2 PSNE

Input: An undirected graph $G(V, E)$; the truncation order T ; the number of non-zeros N used in the PPR matrix sparsifier; the decay factor α of PPR; the filter parameter μ ; the embedding dimension size k

Output: The network embedding matrix

- 1: Obtaining a sparse PPR matrix \tilde{M} by executing Algorithm 1
 - 2: Obtaining the multiple-perspective PPR $M_{\tilde{M}}$ by Equation 10
 - 3: $M_{\tilde{M}} \leftarrow \sigma_{\mu}(M_{\tilde{M}})$ // σ_{μ} is a non-linear activation function
 - 4: $U, \Sigma, V \leftarrow$ Randomized SVD ($M_{\tilde{M}}, k$)
 - 5: **return** $U\sqrt{\Sigma}$ as the network embedding matrix
-

After obtaining the anonymous trajectories, we utilize the Longest Common Subsequence [40] to ascertain the similarity between the two trajectories.

THEOREM 4.4. [Longest Common Subsequence(LCSS)] For two trajectories $P_A = (a_1, a_2, \dots, a_n)$ and $P_B = (b_1, b_2, \dots, b_m)$ with lengths n and m respectively, where the length of the longest common subsequence is:

$$LCSS(P_A, P_B) = \begin{cases} 0 & \text{if } P_A = \emptyset \text{ or } P_B = \emptyset \\ 1 + LCSS(a_{t-1}, b_{i-1}), & \text{if } a_t = b_i \\ \max(LCSS(a_{t-1}, b_i), LCSS(a_t, b_{i-1})), & \text{otherwise} \end{cases} \quad (11)$$

where $t = 1, 2, \dots, n$ and $i = 1, 2, \dots, m$ and \emptyset is empty trajectory.

Therefore, the pattern similarity $wp_{Pattern}$ in Equation 10 is defined via anonymous walk paths and LCSS as follows:

$$wp_{ij} = \frac{1}{s} \sum_1^s LCSS(AnoTra_i, AnoTra_j) / lenth(AnoTra_i) \quad (12)$$

where $AnoTra_i$ and $AnoTra_j$ are anonymous random walk trajectories starting from nodes i and j , respectively. $lenth(\cdot)$ is a function of trajectory length and s is the sampling numbers node pair (i, j) .

As shown in Table 1, v_3 exhibits a higher MP-PPR value than v_7 . Thus, MP-PPR captures more reasonable proximity for network embedding from multiple perspectives without compromising the proximity between different nodes as reflected in the original PPR.

4.3 Our PSNE and Theoretical Analysis

Based on the above theoretical backgrounds, we devise an efficient spectral sparsification algorithm for scaling network embedding (Algorithm 2). Firstly, Algorithm 2 obtains a sparse PPR matrix with a theoretical guarantee in terms of the Frobenius norm (Line 1). Subsequently, it obtains the multiple-perspective PPR matrix $M_{\tilde{M}}$ (Line 2). Finally, Lines 3-5 obtain the network embedding matrix by executing the randomized singular value decomposition (RSVD) algorithm [15]. Here, σ_{μ} is a non-linear activation function (e.g., $\sigma_{\mu}(x) = \max(0, \log(xn\mu))$ [44, 47, 51]) with the filter parameter μ . Next, we analyze the time&space complexities of the proposed PSNE and the corresponding approximation errors.

THEOREM 4.5. *The time complexity and space complexity of Algorithm 2 are $O(m \log n + mk + nk^2)$ and $O(m \log n + nk)$, respectively.*

PROOF. PSNE (i.e., Algorithm 2) has three main steps as follows:

- Step 1 (i.e., Algorithm 1): Random-Walk Molynomial Sparsifier for PPR matrix. Lines 1-7 of Algorithm 1 sample $O(N)$ paths to construct $O(N)$ edges for the sparse graph \tilde{G} . The expected value of r (r is the length of sample path), denoted as $\mathbb{E}(r)$, is given by $\sum_{i=1}^T i(\alpha(1-\alpha))^i / \sum_{i=1}^T (\alpha(1-\alpha))^i \leq 1/\alpha$. As a result, Lines 1-7 of Algorithm 1 consume $O(N\mathbb{E}(r)) = O(N/\alpha)$ time. In Lines 8-9, Algorithm 1 consumes $O(N)$ time to compute the sparse PPR matrix $\tilde{\Pi}$. Thus, the time complexity of Algorithm 1 is $O(N/\alpha)$. For space complexity, Algorithm 1 takes $O(N)$ extra space to store graph \tilde{G} and matrix $\tilde{\Pi}$. So, the space complexity of Algorithm 1 is $O(N + n + m)$.
- Step 2 (i.e., Lines 2-3 of Algorithm 2): Multiple-Perspective strategy. In particular, according to Equation 10, we can know that this step consumes $O(mx_{avg})$ time to obtain MP-PPR matrix $M_{\tilde{\Pi}}$, in which $x_{avg} = \frac{N}{n}$ is the average number of non-zero elements per row of $\tilde{\Pi}$. Thus, the time complexity of step 2 is $O(m + m \frac{N}{n})$. In most real-life graphs, $m = O(n \log n)$, thus, the time complexity of step 2 can be further reduced to $O(m + N \log n)$. The space complexity of step 2 is $O(m + N \log n + n)$.
- Step 3 (i.e., Lines 4-5 of Algorithm 2): Randomized Singular Value Decomposition. By [15], we know that this step needs $O(Nk + nk^2 + k^3)$ time and $O(N + nk)$ space to get the network embedding matrix.

In a nutshell, the time (resp., space) complexity of PSNE is $O(N/\alpha + m + N \log n + Nk + nk^2 + k^3)$ (resp., $O(m + N \log n + nk)$). Following the previous methods [35, 46], α is a constant and $N = O(m)$, thus, the time (resp., space) complexity of PSNE can be reduced to $O(m \log n + mk + nk^2)$ (resp., $O(m \log n + nk)$). Thus, we have completed the proof of Theorem 4.5. \square

Missing proofs are deferred to our Appendix Section.

THEOREM 4.6. *Let Π be the exact PPR matrix (i.e., Equation 1) and $\tilde{\Pi}$ be the approximate sparse matrix obtained by Algorithm 1, we have $\|\Pi - \tilde{\Pi}\|_F \leq \sqrt{n}((1-\alpha)^{T+1} + 4\epsilon \cdot \alpha_{sum})$.*

THEOREM 4.7. *Let M_{Π} (resp., $M_{\tilde{\Pi}}$) be the MP-PPR matrix by executing Equation 10 on Π (resp., $\tilde{\Pi}$), we have $\|\sigma(M_{\Pi}, \mu) - \sigma(M_{\tilde{\Pi}}, \mu)\|_F \leq n((1-\alpha)^{T+1} + 4\epsilon \cdot \alpha_{sum})$.*

5 Empirical Results

In this section, we answer the following Research Questions:

- **RQ1:** How much improvement in effectiveness and efficiency is our PSNE compared to other baselines?
- **RQ2:** Whether the *multiple-perspective* strategies can be integrated into other baselines to improve qualities?

5.1 Experimental Setup

Datasets. We evaluate our proposed solutions on several publicly-available datasets (Table 2), which are widely used benchmarks for network embedding [39, 44, 46, 51]. BlogCatalog, Flickr, and YouTube are undirected social networks where nodes represent users and edges represent relationships. The Protein-Protein Interaction (PPI) dataset is a subgraph of the Homo sapiens PPI network,

Table 2: Dataset statistics.

Dataset	V	E	#labels
PPI[50]	3,890	76,584	50
Wikipedia[50]	4,777	184,812	39
BlogCatalog[50]	10,312	333,983	39
Flickr[47]	80,513	5,899,882	195
Youtube[50]	1,138,499	2,990,443	47

with vertex labels from hallmark gene sets indicating biological states. The Wikipedia dataset is a word co-occurrence network from the first million bytes of a Wikipedia dump, with nodes labeled by Part-of-Speech tags.

Baselines and parameters. The following ten competitors are implemented for comparison: DeepWalk [31], Grarep [7], HOPE [28], NetSMF [32], ProNE [50], STRAP [47], NRP [46], Lemane [51], FREDE [39], and SketchNE [44]. Note that STRAP, NRP, and Lemane are PPR-based embedding methods. For the ten competitors, we take their corresponding default parameters. The detailed parameter settings of the proposed PSNE are summarized in Table 3. All experiments are conducted on a Ubuntu server with Intel (R) Xeon (R) Silver 4210 CPU (2.20GHz) and 1T RAM.

5.2 Effectiveness Testing

We apply node classification to evaluate the effectiveness of our solutions. Node classification aims to accurately predict the labels of nodes. Specifically, a node embedding matrix is first constructed from the input graph. Subsequently, a one-vs-all logistic regression classifier is trained using the embedding matrix and the labels of randomly selected vertices. Finally, the classifier is tested with the labels of the remaining vertices. The training ratio is adjusted from 10% to 90%. To be more reliable, we execute each method five times and report their Micro-F1 and Macro-F1 in Figure 4¹. As can be seen, we can obtain the following observations: (1) Under different training ratios, our PSNE consistently achieves the highest Micro-F1 scores on four of the five datasets, and the highest Macro-F1 scores on PPI, BlogCatalog, and YouTube. For example, on YouTube, PSNE is 1.5% and 0.9% better than the runner-up in Micro-F1 and Macro-F1 scores, respectively. (2) The Micro-F1 and Macro-F1 of all baselines vary significantly depending on the dataset and training ratio. For example, DeepWalk outperforms other methods on Flickr (PSNE is the runner-up and slightly worse than DeepWalk) but has inferior Micro-F1 scores on other datasets. HOPE achieves the highest Macro-F1 score on Wikipedia but performs poorly on the Micro-F1 metric. (3) PSNE outperforms other PPR-based methods, including STRAP, NRP, and Lemane, with a margin of at least 2% in most cases. Specifically, for the Micro-F1 metric, our PSNE achieves improvements of 6%, 3%, 8%, 7%, and 2% over NPR on PPI, Wikipedia, BlogCatalog, Flickr, and Youtube, respectively. For example, for Flickr with more than a few million edges, our PSNE achieves 41% while NPR is 34% in the micro-F1 metric. For the Macro-F1 metric, PSNE surpasses all other PPR-based algorithms, achieving a notable lead of 0.5%, 0.6%, 2%, 4%, and 2% over the

¹For Youtube, since FREDE, GraRep, and HOPE cannot obtain the result within 48 hours or out of memory, we ignore their results.

Table 3: Parameter settings of our proposed PSNE.

Datasets	Parameters
PPI	$\alpha=0.35, T=10, c=25, \mu=10$
Wikipedia	$\alpha=0.50, T=10, c=25, \mu=0.2$
BlogCatalog	$\alpha=0.35, T=10, c=35, \mu=25$
Flickr	$\alpha=0.35, T=5, c=25, \mu=1$
Youtube	$\alpha=0.35, T=5, c=30, \mu=1$

runner-up PPR-based algorithm on these five datasets, respectively. These results show that PSNE’s multi-perspective strategy indeed can enhance the representation power of the original PPR matrix. Besides, these results give clear evidence that our PSNE has high embedding quality compared with the baselines.

5.3 Efficiency Testing

For efficiency testing, we do not include non-PPR-based algorithms because all of them are outperformed by *Lemane* and *NRP*, as reported in their respective studies. On top of that, our proposed PSNE is a PPR-based method, so we test the runtime of other PPR-based methods (i.e., STRAP, NRP, and Lemane) for comparison. Table 4 presents the wall-clock time of each PPR-based method with 20 threads. As can be seen, NRP outperforms other methods (but it has poor node classification quality and improved by 2% ~ 25% by PSNE, as stated in Figure 4), and PSNE is runner-up and slightly worse than NRP. The reasons can be explained as follows: NRP integrates the calculation and factorization of the PPR matrix in an iterative framework to improve efficiency but lacks the non-linear representation powers for node embedding. However, our PSNE devises the sparsifiers of random-walk matrix polynomials for the truncated PPR matrix, avoiding repeatedly computing each row or column of the PPR matrix in the push-based methods (e.g., STRAP, Lemane). These results show that PSNE achieves significant speedup with high embedding quality compared with the baselines, which is consistent with our theoretical analysis (Section 4.3).

5.4 Scalability Testing on Synthetic Graphs

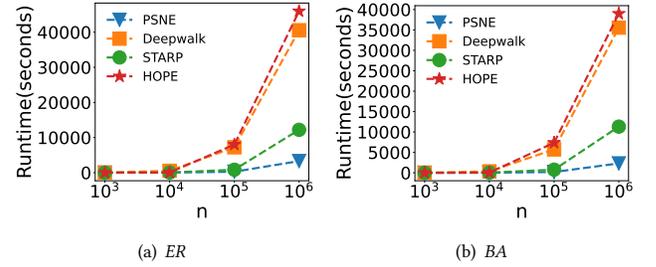
We use the well-known NetworkX Python package [4] to generate two types of synthetic graphs *ER* [13] and *BA* [5] to test the scalability of our PSNE. Figure 3 only presents the results of Deepwalk, STRAP, HOPE, and our PSNE, with comparable trends across other methods. By Figure 3, we can know that when the number of nodes is small, the DeepWalk and HOPE have a runtime comparable to PSNE and STRAP. However, as the number of nodes increases, the runtime of DeepWalk and HOPE significantly rises, surpassing that of PSNE by an order of magnitude. Additionally, the increase in STRAP’s runtime is greater than that of PSNE. These results indicate that our PSNE has excellent scalability over massive graphs while the baselines do not.

5.5 Ablation Studies

To illustrate the impact of the multiple-perspective strategy on PSNE and other baselines, we report the results of the ablation study in Table 5. As can be seen, on PPI, both GraRep and HOPE have a 2%-3% improvement, while NetSMF, STRAP, ProNE, and

Table 4: Runtime (seconds) of different PPR-based methods.

Dataset	STRAP	NRP	Lemane	PSNE
PPI	4.8e1	2e0	1.9e2	2.4e1
Wikipedia	1.1e2	3e0	5.3e2	4.4e1
BlogCatalog	3.7e2	1.1e1	7.4e3	2.6e2
Flickr	5.0e3	2.0e2	1.8e4	2.3e3
Youtube	2.1e4	1.8e2	>24h	7.0e3

**Figure 3: Scalability testing on synthetic graphs.****Table 5: Ablation studies. MP (resp., NMP) is the corresponding method with (resp., without) multiple-perspective strategy. The best result is marked in bold.**

Model	PPI		Wikipedia		BlogCatalog	
	NMP	MP	NMP	MP	NMP	MP
GraRep	20.57	22.93	50.51	53.60	33.67	37.21
HOPE	20.72	23.73	52.23	53.46	34.25	38.63
NetSMF	23.1	23.64	43.4	44.75	39.33	41.86
STRAP	23.51	24.31	51.87	52.78	40.33	41.41
ProNE	23.84	24.11	50.87	51.32	40.73	41.23
PSNE	24.07	24.52	52.19	53.99	41.02	43.14

PSNE exhibit more modest gains of 0.3%-0.8%. On the Wikipedia and BlogCatalog datasets, all methods benefit significantly from the multiple-perspective strategy, with HOPE achieving the highest improvement on BlogCatalog (approximately 4.38%) and other methods averaging around 1.2% improvements. All experimental results were presented with a 50% training rate.

5.6 Sampling Quality Analysis

We also observed that both PSNE and NetSMF use similar but completely different² path sampling strategies to derive the node proximity matrix. Therefore, we will closely examine their differences. In particular, Figure 4 has revealed that NetSMF exhibits lower accuracy than PSNE. In addition, NetSMF requires significantly more path sampling to achieve acceptable accuracy. To illustrate this point, we compare the impact of sampling size on the

²New sampling probabilities and anonymous random walks are used in our solutions.

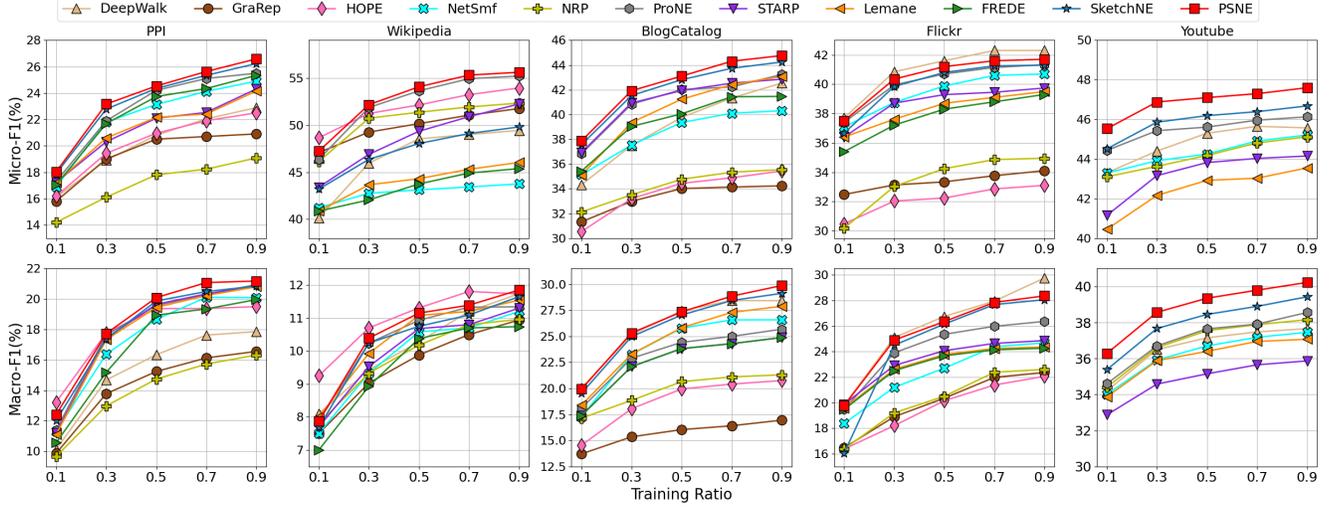


Figure 4: The performance of different network embedding methods

F1 scores of NetSMF and PSNE. Following the previous methods [9], we also set the number of samples to cTm (T is the path length) and vary c to adjust the number of samples. As shown in Figure 5, under the same sampling scale, PSNE outperforms NetSMF, with a maximum difference of approximately 8%. On top of that, when $c = 30$, PSNE essentially meets the sampling quantity requirement, whereas NetSMF requires nearly 10 times (i.e., $c = 300$) the sampling quantity of PSNE to achieve comparable performance. We believe there are two main reasons for this phenomenon: (1) NetSMF treats distant nodes and nearby nodes equally, missing the non-uniform higher-order structure information (Table 1). To this end, we use the α -decay random walk (i.e., the PPR matrix in Equation (1)) to measure node proximity such that nearby nodes receive more attention. (2) NetSMF obtains proximity information of node pair (v_i, v_j) only by sampling path $\{v_i \dots v_j\}$, which requires more than n^2 paths to calculate the proximity of all node pairs. On the contrary, by the proposed multi-perspective strategy, v_i 's one-hop neighbors propagate their PPR values (w.r.t v_j) to v_i to restore the PPR values (Equation 10), allowing us to greatly reduce the number of samples while obtaining high-quality when compared to NetSMF.

5.7 Parameter Analysis

Following the previous methods [9], we also set $N = cTm$. Figure 6 only shows the effect of different parameter settings on Micro-F1 of PSNE in BlogCatalog and PPI, with comparable trends across Macro-F1 and other datasets. We have the following results: (1) By Figure 6(a), Micro-F1 increases first and then decreases with increasing α . This is because the parameter α controls how much of the graph is explored by our PSNE. Thus, when α is close to 1, the proximity matrix focuses on one-hop neighbors and thus preserves the adjacency information. As α approaches 0, the proximity matrix incorporates more information coming from multi-hop higher-order neighbors, resulting in good performance. (2) By Figure 6(b) and 6(c), we observe that Micro-F1 increases first and then decreases

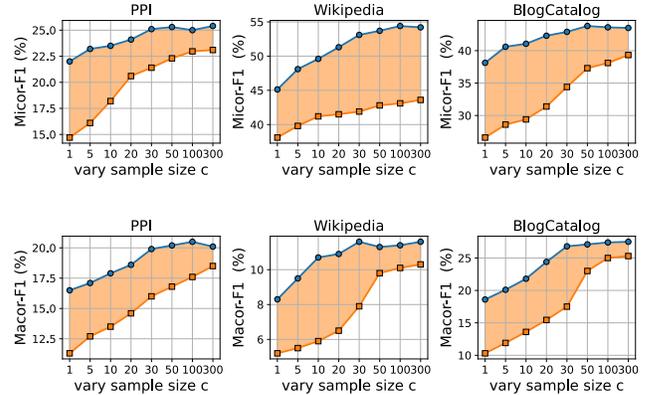


Figure 5: Sampling quantity analysis (orange line and blue line represent NetSMF and PSNE, respectively).

with increasing c or T , and the optimal result is taken when $c = 50$ or $T = 10$. The reasons can be explained as follows: Although the larger c or T , the closer the obtained $\tilde{\Pi}$ by Algorithm 1 is to the exact PPR matrix Π , the gain they bring leads to overfitting due to the small-world phenomenon [43] (e.g., numerous redundant and valueless neighbors lead to noise [22]). (3) By Figure 6(d), we know that both excessively large and small values of μ result in performance degradation. This observation suggests that small entries in the proximity matrix may introduce noise and impede the qualities of embedding vectors (Line 3 of Algorithm 2).

6 Conclusion

This paper presents an efficient spectral sparsification algorithm PSNE that first devises a matrix polynomial sparser to *directly* approximate the whole PPR matrix with theoretical guarantee, which avoids repeatedly computing each row or column of the PPR matrix.

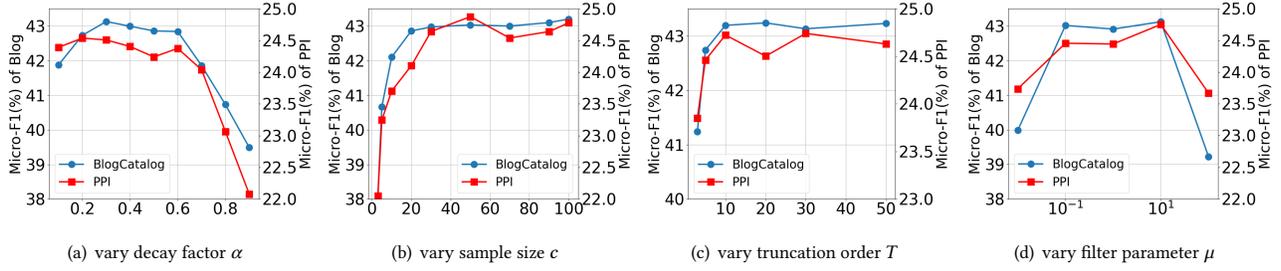


Figure 6: The performance of our proposed PSNE with varying parameters

Then, PSNE introduces a simple yet effective multiple-perspective strategy to enhance further the representation power of the obtained sparse and coarse-grained PPR matrix. Finally, extensive empirical results show that PSNE can quickly obtain high-quality embedding vectors compared with ten competitors.

7 ACKNOWLEDGMENTS

The work was supported by (1) Fundamental Research Funds for the Central Universities under Grant SWU-KQ22028, (2) University Innovation Research Group of Chongqing (No. CXQT21005) (3) the Fundamental Research Funds for the Central Universities (No. SWU-XDJH202303) (4) the Natural Science Foundation of China (No. 72374173) (5) the High Performance Computing clusters at Southwest University.

8 Appendix

8.1 Approximate Error Analysis

LEMMA 8.1. Let $\mathcal{L} = D^{-1/2}L_{\beta}(G)D^{-1/2}$ and the corresponding sparsifier $\tilde{\mathcal{L}} = D^{-1/2}\tilde{L}D^{-1/2}$, in which \tilde{L} is the Laplacian matrix of \tilde{G} (Line 8 of Algorithm 1). Then, all the singular values of $\tilde{\mathcal{L}} - \mathcal{L}$ are less than 4ϵ .

PROOF. According to Theorem 1, we have

$$\frac{1}{(1+\epsilon)} \cdot \mathbf{x}^T L_{\beta}(G) \mathbf{x} \leq \mathbf{x}^T \tilde{L} \mathbf{x} \leq \frac{1}{(1-\epsilon)} \cdot \mathbf{x}^T L_{\beta}(G) \mathbf{x}$$

Let $\mathbf{x} = D^{-1/2} \mathbf{y}$, we have

$$-\frac{\epsilon}{(1+\epsilon)} \cdot \mathbf{y}^T \mathcal{L} \mathbf{y} \leq \mathbf{y}^T (\tilde{\mathcal{L}} - \mathcal{L}) \mathbf{y} \leq \frac{\epsilon}{(1-\epsilon)} \cdot \mathbf{y}^T \mathcal{L} \mathbf{y}$$

Since $\epsilon \leq 0.5$, we have

$$|\mathbf{y}^T (\tilde{\mathcal{L}} - \mathcal{L}) \mathbf{y}| \leq \frac{\epsilon}{(1-\epsilon)} \cdot \mathbf{y}^T \mathcal{L} \mathbf{y} \leq 2\epsilon \mathbf{y}^T \mathcal{L} \mathbf{y}$$

By Courant-Fisher Theorem [34], we have

$$|\lambda_i(\tilde{\mathcal{L}} - \mathcal{L})| \leq 2\epsilon \lambda_i(\mathcal{L}) < 4\epsilon$$

where $i \in [n]$, $\lambda_i(A)$ is i -th largest eigenvalue of matrix A . This is because \mathcal{L} is a normalized graph Laplacian matrix with eigenvalues in the interval $[0, 2)$. \square

LEMMA 8.2. [18] If B, C be two $n \times n$ symmetric matrices, for the decreasingly-ordered singular values λ of B, C and BC ,

$$\lambda_{i+j-1}(BC) \leq \lambda_i(B) \times \lambda_j(C)$$

where $i \leq i, j \leq n$ and $i + j \leq n + 1$.

Based on these lemmas, we give the detailed proofs of Theorem 4 and Theorem 5 as follows.

The Proof of Theorem 4. According to Equation 4, we have

$$\|\Pi - \Pi'\|_F = \left\| \alpha \sum_{i=T+1}^{\infty} (1-\alpha)^i (D^{-1}A)^i \right\|_F \quad (13)$$

$$= \alpha \sum_{i=T+1}^{\infty} (1-\alpha)^i \sqrt{\sum_{j \in [n]} \lambda_j^2((D^{-1}A)^i)} \quad (14)$$

$$\leq \alpha \sum_{i=T+1}^{\infty} (1-\alpha)^i \sqrt{\sum_{j \in [n]} \lambda_j^2(D^{-1}A)} \quad (15)$$

$$\leq \sqrt{n} (1-\alpha)^{T+1} \quad (16)$$

This is because $\lambda_j((D^{-1}A)^i) \leq \lambda_j(D^{-1}A) * \lambda_1(D^{-1}A) * \dots * \lambda_1(D^{-1}A) \leq \lambda_j(D^{-1}A)$ by Lemma 8.2 and $\lambda_s(D^{-1}A) \in [0, 1]$ for any $s \in [n]$. Furthermore, by Lemma 8.1 and Lemma 8.2, we can know that $\Pi' - \tilde{\Pi} = \alpha_{sum} D^{-1/2} (\tilde{L} - L_{\alpha}(G)) D^{-1/2} D^{1/2}$. Thus, $\lambda_i(\Pi' - \tilde{\Pi}) \leq \alpha_{sum} \lambda_1(D^{-1/2}) \lambda_i(\tilde{L} - L_{\alpha}(G)) \lambda_1(D^{1/2}) \leq 4\epsilon \cdot \alpha_{sum}$.

As a result, $\|\Pi' - \tilde{\Pi}\|_F = \sqrt{\sum_{i \in [n]} \lambda_i^2(\Pi' - \tilde{\Pi})} \leq 4\epsilon \cdot \alpha_{sum} \sqrt{n}$. Based on these analyses, we have

$$\|\Pi - \tilde{\Pi}\|_F \leq \|\Pi - \Pi'\|_F + \|\Pi' - \tilde{\Pi}\|_F \quad (17)$$

$$\leq \sqrt{n} (1-\alpha)^{T+1} + 4\epsilon \cdot \alpha_{sum} \sqrt{n} \quad (18)$$

Therefore, we have completed the proof of Theorem 4.

The Proof of Theorem 5. To simplify the proof, let's assume that $\lambda_{hi} = 1/(\sqrt{(d_h+1)}\sqrt{(d_i+1)})$ and $\lambda_{ii} = 1/(d_i+1)$. Therefore, by Definition 10, we can know that $M_{\Pi} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \Pi$ and $M_{\tilde{\Pi}} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \tilde{\Pi}$, in which $\tilde{A} = A + I$ (I is the identify matrix) and \tilde{D} is degree matrix of \tilde{A} . Since $\max(0, \log(xn\mu))$ is 1-Lipchitz w.r.t Frobenius norm, we have

$$\|\sigma_{\mu}(M_{\Pi}) - \sigma_{\mu}(M_{\tilde{\Pi}})\|_F \quad (19)$$

$$= \|\sigma_{\mu}(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \Pi) - \sigma_{\mu}(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \tilde{\Pi})\|_F \quad (20)$$

$$\leq \|\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} (\Pi - \tilde{\Pi})\|_F \quad (21)$$

$$\leq n((1-\alpha)^{T+1} + 4\epsilon \cdot \alpha_{sum}) \quad (22)$$

Therefore, we have completed the proof of Theorem 5.

Since $(1-\alpha)^{T+1} + 4\epsilon \cdot \alpha_{sum}$ is a constant due to $0 < \epsilon \leq 0.5$, $\alpha_{sum} \leq 1-\alpha$, and $(1-\alpha)^{T+1} \leq 1$, we have $\|\Pi - \tilde{\Pi}\|_F = O(\sqrt{n})$, $\|\sigma(M_{\Pi}, \mu) - \sigma(M_{\tilde{\Pi}}, \mu)\|_F = O(n)$.

References

- [1] Sami Abu-El-Hajia, Bryan Perozzi, Rami Al-Rfou, and Alexander A. Alemi. 2018. Watch Your Step: Learning Node Embeddings via Graph Attention. In *NIPS*. 9198–9208.
- [2] Charu C. Aggarwal and Haixun Wang. 2010. An Introduction to Graph Data. In *Managing and Mining Graph Data*. Vol. 40. 1–11.
- [3] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local Graph Partitioning using PageRank Vectors. In *FOCS*. 475–486.
- [4] Daniel A. Schult, Aric A. Hagberg, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference 2*, 1 (2008), 11–15.
- [5] Albert-Laszlo Barabasi and Reka Albert. 1999. Emergence of Scaling in Random Networks. *science* (1999).
- [6] Yuchen Bian, Yaowei Yan, Wei Cheng, Wei Wang, Dongsheng Luo, and Xiang Zhang. 2018. On Multi-query Local Community Detection. In *2018 IEEE International Conference on Data Mining (ICDM)*. 9–18.
- [7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *CIKM*. ACM, 891–900.
- [8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *AAAI*.
- [9] Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. 2015. Spectral Sparsification of Random-Walk Matrix Polynomials. *CoRR* abs/1502.03496 (2015).
- [10] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine* 35, 1 (2018), 53–65.
- [11] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. 2019. Adversarial Training Methods for Network Embedding. In *WWW*. 329–339.
- [12] Xinyu Du, Xingyi Zhang, Sibao Wang, and Zengfeng Huang. 2023. Efficient Tree-SVD for Subset Node Embedding over Large Dynamic Graphs. *Proc. ACM Manag. Data* 1, 1 (2023), 96:1–96:26.
- [13] Paul Erdos, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.
- [14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.
- [15] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. , 217–288 pages.
- [16] William L. Hamilton. 2020. *Graph Representation Learning*. Morgan & Claypool Publishers.
- [17] Yue He, Longlong Lin, Pingpeng Yuan, Ronghua Li, Tao Jia, and Zeli Wang. 2024. CCSS: Towards conductance-based community search with size constraints. *Expert Syst. Appl.* 250 (2024), 123915.
- [18] Roger A. Horn and Charles R. Johnson. 1991. *Topics in Matrix Analysis*.
- [19] Sergey Ivanov and Evgeny Burnaev. 2018. Anonymous Walk Embeddings. arXiv:1805.11921 [cs.LG]
- [20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
- [21] Yi-An Lai, Chin-Chi Hsu, Wen-Hao Chen, Mi-Yen Yeh, and Shou-De Lin. 2017. PRUNE: Preserving Proximity and Global Ranking for Network Embedding. In *NIPS*. 5257–5266.
- [22] Haoyang Li and Lei Chen. 2023. EARLY: Efficient and Reliable Graph Neural Network for Dynamic Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 163:1–163:28.
- [23] Meihao Liao, Rong-Hua Li, Qiangqiang Dai, Hongyang Chen, Hongchao Qin, and Guoren Wang. 2023. Efficient Personalized PageRank Computation: The Power of Variance-Reduced Monte Carlo Approaches. *Proc. ACM Manag. Data* 1, 2 (2023), 160:1–160:26.
- [24] Longlong Lin, Tao Jia, Zeli Wang, Jin Zhao, and Rong-Hua Li. 2024. PSMC: Provable and Scalable Algorithms for Motif Conductance Based Graph Clustering. *CoRR* abs/2406.07357 (2024).
- [25] Longlong Lin, Ronghua Li, and Tao Jia. 2023. Scalable and Effective Conductance-Based Graph Clustering. In *AAAI*. 4471–4478.
- [26] Longlong Lin, Pingpeng Yuan, Rong-Hua Li, Chun-Xue Zhu, Hongchao Qin, Hai Jin, and Tao Jia. 2024. QTCS: Efficient Query-Centered Temporal Community Search. *Proc. VLDB Endow.* 17, 6 (2024), 1187–1199.
- [27] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.
- [28] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *KDD*. 1105–1114.
- [29] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*. 1105–1114.
- [30] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking : Bringing Order to the Web. In *The Web Conference*.
- [31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
- [32] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WWW*. 1509–1520.
- [33] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and Node2vec. In *WSDM*. 459–467.
- [34] Daniel A. Spielman. 2007. Spectral Graph Theory and its Applications. In *FOCS*. 29–38.
- [35] Daniel A. Spielman and Nikhil Srivastava. 2011. Graph Sparsification by Effective Resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.
- [36] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
- [37] Mingyue Tang, Pan Li, and Carl Yang. 2022. Graph Auto-Encoder via Neighborhood Wasserstein Reconstruction. In *ICLR*.
- [38] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *WWW*. 539–548.
- [39] Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan V. Oseledets, and Emmanuel Müller. 2021. FREDE: Anytime Graph Embeddings. *Proc. VLDB Endow.* 14, 6 (2021), 1102–1110.
- [40] M. Vlachos, G. Kollios, and D. Gunopoulos. 2002. Discovering similar multi-dimensional trajectories. In *Proceedings 18th International Conference on Data Engineering*. 673–684. <https://doi.org/10.1109/ICDE.2002.994784>
- [41] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *KDD*. 1225–1234.
- [42] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2017. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. *CoRR* abs/1711.08267 (2017).
- [43] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of ‘small-world’ networks. *Nature* 393 (1998), 440–442.
- [44] Yuyang Xie, Yuxiao Dong, Jiezhong Qiu, Wenjian Yu, Xu Feng, and Jie Tang. 2023. SketchNE: Embedding Billion-Scale Networks Accurately in One Hour. *IEEE Transactions on Knowledge and Data Engineering* (2023), 1–14. <https://doi.org/10.1109/TKDE.2023.3250703>
- [45] Renchi Yang and Jieming Shi. 2024. Efficient High-Quality Clustering for Large Bipartite Graphs. *Proc. ACM Manag. Data* 2, 1, Article 23 (2024), 27 pages.
- [46] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, and Sourav S. Bhowmick. 2020. Homogeneous Network Embedding for Massive Graphs via Reweighted Personalized PageRank. *Proc. VLDB Endow.* 13, 5 (2020), 670–683.
- [47] Yuan Yin and Zhewei Wei. 2019. Scalable graph embeddings via sparse transpose proximities. In *KDD*. 1429–1437.
- [48] Yunfeng Yu, Longlong Lin, Qiyu Liu, Zeli Wang, Xi Ou, and Tao Jia. 2024. GSD-GNN: Generalizable and Scalable Algorithms for Decoupled Graph Neural Networks. In *ICMR*. 64–72.
- [49] Pingpeng Yuan, Longlong Lin, Zhijuan Kou, Ling Liu, and Hai Jin. 2019. Big RDF Data Storage, Computation, and Analysis: A Strawman’s Arguments. In *ICDCS*. 1693–1703.
- [50] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI*. 4278–4284.
- [51] Xingyi Zhang, Kun Xie, Sibao Wang, and Zengfeng Huang. 2021. Learning Based Proximity Matrix Factorization for Node Embedding. In *KDD*. 2243–2253.
- [52] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In *KDD*. ACM, 2778–2786.
- [53] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *AAAI*.