ProNE: Fast and Scalable Network Representation Learning

Jie Zhang[1] , **Yuxiao Dong**[2] , **Yan Wang**[1] , **Jie Tang**[1] and **Ming Ding**[1]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Microsoft Research, Redmond
{j-z16, wang-y17, dm18}@mails.tsinghua.edu.cn, yuxdong@microsoft.com, jietang@tsinghua.edu.cn

Abstract

Recent advances in network embedding have revolutionized the field of graph and network mining. However, (pre-)training embeddings for very large-scale networks is computationally challenging for most existing methods. In this work, we present ProNE[1]—a fast, scalable, and effective model, whose single-thread version is 10–400× faster than efficient network embedding benchmarks with 20 threads, including LINE, DeepWalk, node2vec, GraRep, and HOPE. As a concrete example, the single-thread ProNE requires only 29 hours to embed a network of hundreds of millions of nodes while it takes LINE weeks and DeepWalk months by using 20 threads. To achieve this, ProNE first initializes network embeddings efficiently by formulating the task as sparse matrix factorization. The second step of ProNE is to enhance the embeddings by propagating them in the spectrally modulated space. Extensive experiments on networks of various scales and types demonstrate that ProNE achieves both effectiveness and significant efficiency superiority when compared to the aforementioned baselines. In addition, ProNE's embedding enhancement step can be also generalized for improving other models at speed, e.g., offering $>10\%$ relative gains for the used baselines.

1 Introduction

Over the past years, representation learning has offered a new paradigm for network mining and analysis [Hamilton *et al.*, 2017b]. Its goal is to project a network's structures into a continuous space—embeddings—while preserving its certain properties. Extensive studies have shown that the learned embeddings can benefit a wide range of network mining tasks [Perozzi *et al.*, 2014; Hamilton *et al.*, 2017b].

The recent advances in network embedding can roughly fall into three categories: matrix factorization based methods such as SocDim [Tang and Liu, 2009], GraRep [Cao *et al.*, 2015], HOPE [Ou *et al.*, 2016], and NetMF [Qiu *et al.*, 2018]; skip-gram based models, such as DeepWalk [Perozzi *et al.*, 2014], LINE [Tang *et al.*, 2015], and node2vec [Grover and Leskovec, 2016]; and graph neural networks (GNNs), such as Graph Convolution [Kipf and Welling, 2017], GraphSage [Hamilton *et al.*, 2017a], and Graph Attention [Veličković *et al.*, 2018]. Commonly, the embeddings pre-trained by the first two types of methods are fed into downstream tasks' learning models, such as GNNs.

Therefore, it is critical to generate effective network embeddings efficiently in order to serve large-scale real network applications. However, most of existing models focus on improving the effectiveness of embeddings, leaving their efficiency and scalability limited. For example, in factorization based models, the time complexity of GraRep is $O(n^3)$ with n being the number of nodes in a network, making it prohibitively expensive to compute for large networks; In skip-gram based models, with the default parameter settings, it would cost LINE weeks and DeepWalk/node2vec months to learn embeddings for a network of 100,000,000 nodes and 500,000,000 edges by using 20 threads on a modern server.

To address the efficiency and scalability limitations of current work, we present a fast and scalable network embedding algorithm—ProNE. The general idea of ProNE is to first initialize network embeddings in an efficient manner and then to enhance the representation power of these embeddings. Inspired by the long-tailed distribution of most real networks and its resultant network sparsity, the first step is achieved by formulating network embedding as sparse matrix factorization; The second step is to leverage the higher-order Cheeger's inequality to spectrally propagate the initial embeddings with the goal of capturing the network's localized smoothing and global clustering information.

This design makes ProNE an extremely fast embedding model with the effectiveness superiority. We conduct experiments in five real networks and a set of random graphs. Extensive demonstrations show that the one-thread ProNE model is about 10–400× faster than popular network embedding benchmarks with 20 threads, including DeepWalk, LINE, node2vec (See Figure 1). Our scalability analysis suggests that the time cost of ProNE is linearly correlated with network volume and density, making it scalable for billion-scale networks. In fact, by using one thread, ProNE requires only 29 hours to learn embeddings for the aforementioned network of 100,000,000 nodes.

[1]Code is available at https://github.com/THUDM/ProNE

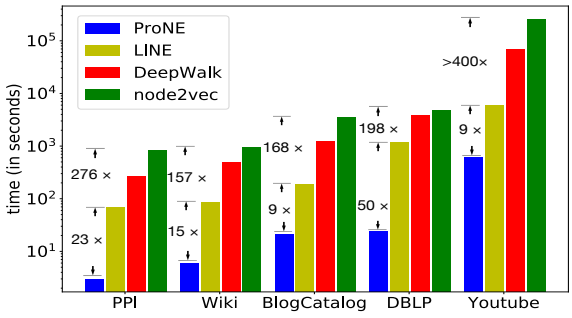


Figure 1: The efficiency comparison between ProNE and baselines.

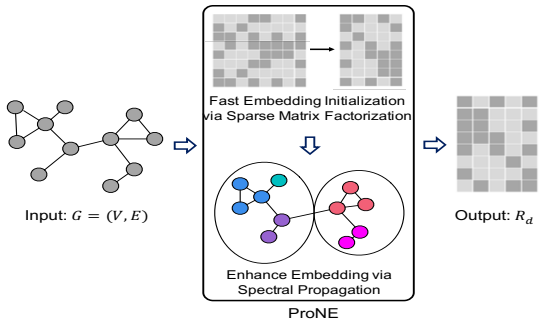


Figure 2: The ProNE model: 1) Sparse matrix factorization for fast embedding initialization and 2) Spectral propagation in the modulated networks for embedding enhancement.

In addition to its efficiency and scalability advantage, ProNE also consistently outperforms all baselines across all datasets for the multi-label node classification task. More importantly, the second step—spectral propagation—in ProNE is a general framework for enhancing network embeddings. By taking the embeddings generated by DeepWalk, LINE, node2vec, GraRep, and HOPE as the input, our spectral propagation strategy offers on average $+10\%$ relative improvements for all of them.

2 The Network Embedding Problem

We use $G = (V, E)$ to denote an undirected network with V as the node set of n nodes and E as the edge set. In addition, we denote G's adjacency matrix (binary or weighted) as A and its diagonal degree matrix as D with $D_{ii} = \sum_j A_{ij}$.

Given a network $G = (V, E)$, the problem of network embedding aims to learn a mapping function $f : V \mapsto R^d$ that projects each node to a d-dimensional space ($d \ll |V|$) to capture the structural properties of the network.

Extensive studies have shown that the learned node representations can benefit various graph mining tasks. , such as node classification and link prediction. However, one major challenge is that it is computationally infeasible for most network embedding models to handle large-scale networks. For example, it takes the popular DeepWalk model months to learn embeddings for a sparse random graph of 100,000,000 nodes by using 20 threads [Mikolov *et al.*, 2013a].

3 ProNE: Fast Network Embedding

In this section, we present ProNE—a very fast and scalable model for large-scale network embedding (NE). ProNE composes of two steps as illustrated in Figure 2. First, it formulates network embedding as sparse matrix factorization to efficiently achieve initial node representations. Second, it utilizes the higher-order Cheeger's inequality to modulate the network's spectral space and propagate the learned embeddings in the modulated network, which incorporates both the localized smoothing and global clustering information.

3.1 Fast NE as Sparse Matrix Factorization

The distributional hypothesis [Harris, 1954] has inspired the recent emergence of word and network embedding. Here we show how distributional similarity-based network embedding can be formulated as matrix factorization and more importantly, how it can enable efficient network embedding.

Network Embedding as Sparse Matrix Factorization

Node similarities are usually modeled by structural contexts. We propose to leverage the simplest structure—edge—to represent a node-context pair. The edge set then forms a node-context pair set $\mathcal{D} = E$. Formally, we define the occurrence probability of context v_j given node v_i as

$$\hat{p}_{i,j} = \sigma(r_i^T c_j) \tag{1}$$

where $\sigma(.)$ is the sigmoid function and $r_i, c_i \in R^d$ represent the embedding and context vectors of node v_i, respectively. Accordingly, the objective can be expressed as the weighted sum of log loss over all edges $l = -\sum_{(i,j)\in\mathcal{D}} p_{i,j} \ln \hat{p}_{i,j}$, where $p_{ij} = A_{ij}/D_{ii}$ indicates the weight of (v_i, v_j) in $\mathcal{D}$.

To avoid the trivial solution (r_i=c_j & $\hat{p}_{i,j}$=1), for each observed pair (v_i, v_j), the appearance of a context v_j is also accompanied by negative samples $P_{\mathcal{D},j}$, updating the loss as:

$$l = -\sum_{(i,j)\in\mathcal{D}} [p_{i,j} \ln \sigma(r_i^T c_j) + \tau P_{\mathcal{D},j} \ln \sigma(-r_i^T c_j)] \tag{2}$$

where τ is the negative sample ratio and $P_{\mathcal{D},j}$—the negative samples associated with context node v_j—can be defined as $P_{\mathcal{D},j} \propto (\sum_{i:(i,j)\in\mathcal{D}} p_{i,j})^\alpha$ with $\alpha = 1$ or 0.75 [Mikolov *et al.*, 2013b]. A sufficient condition for minimizing the objective in Eq. 2 is to let its partial derivative with respect to $r_i^T c_j$ be zero. Hence,

$$r_i^T c_j = \ln p_{i,j} - \ln(\tau P_{D,j}), \quad (v_i, v_j) \in \mathcal{D} \tag{3}$$

In observation of $r_i^T c_j$ representing the similarity between v_i's embedding and v_j's context embedding, we propose to define a proximity matrix M with each entry as $r_i^T c_j$, i.e.,

$$M_{i,j} = \begin{cases} \ln p_{i,j} - \ln(\tau P_{D,j}) & , (v_i, v_j) \in \mathcal{D} \\ 0 & , (v_i, v_j) \notin \mathcal{D} \end{cases} \tag{4}$$

Naturally, the objective of distributional similarity-based network embedding is transformed to matrix factorization. We use a spectral method—truncated Singular Value Decomposition (tSVD), i.e., $M \approx U_d \Sigma_d V_d^T$, where Σ_d is the diagonal matrix formed from the top-d singular values, and U_d and V_d are $n \times d$ orthonormal matrices corresponding to the selected singular values. Finally, $R_d \leftarrow U_d \Sigma_d^{1/2}$ is the embedding matrix with each row representing one node's embedding.

Sparse Randomized tSVD for Fast Embedding

By far, we show how general distributional similarity-based network embedding can be understood as matrix factorization. However, tSVD for large-scale matrices (networks) is still time and space expensive. To achieve fast network embedding, we propose to use the randomized tSVD, which offers a significant speedup over tSVD with a strong approximation guarantee [Halko *et al.*, 2011].

Here we show the basic idea of randomized tSVD. First, we seek to find Q with d orthonormal columns, i.e., $M \approx QQ^TM$. Assuming such Q has been found, we define $H=Q^TM$, which is a small matrix ($d \times |V|$) and can be decomposed efficiently by the standard SVD. Thus we can have $H=S_d\Sigma_d V_d^T$ for S_d, V_d orthogonal and Σ_d diagonal. Finally M can be decomposed as $M \approx QQ^TM = (QS_d)\Sigma_d V_d^T$ and the final output embedding matrix of this step is

$$R_d = QS_d\Sigma_d^{1/2} \tag{5}$$

The random matrix theory empowers us to find Q efficiently. The first step is to generate a $|V| \times d$ Gaussian random matrix Ω where $\Omega_{ij} \sim \mathcal{N}(0, \frac{1}{d})$. Given this, we can get $Y=M\Omega$ and take the QR decomposition , i.e., $Y=QR$, where Q is a $|V| \times d$ matrix whose columns are orthonormal.

Note that inspired by [Levy and Goldberg, 2014], a recent study shows that skip-gram based network embedding models can be viewed as implicit matrix factorization [Qiu *et al.*, 2018]. However, the matrix to be implicitly factorized is a dense one, resulting in the $O(|V|^3)$ time complexity for its associated matrix factorization model, while our case involves sparse matrix factorization in $O(|E|)$ (Cf. Eq. 4).

3.2 NE Enhancement via Spectral Propagation

Similar to DeepWalk and LINE, the embeddings learned above can only capture local structural information. To further incorporate global network properties, i.e., community structures, we propose to propagate the initial embeddings in the spectrally modulated network.

Formally, given the input/initial embeddings R_d, we conduct the following propagation rule:

$$R_d \leftarrow D^{-1}A(I_n - \widetilde{L})R_d \tag{6}$$

where I_n is the identity matrix, $\widetilde{L}$ is the Laplacian filter, and all together, $D^{-1}A(I_n - \widetilde{L})$ is the modulated network of the input G. This is inspired by the higher-order Cheeger's inequality [Lee *et al.*, 2014; Bandeira *et al.*, 2013], which will be shown below. Note that *the propagation strategy is general and can be also used to enhance existing embedding models, such as DeepWalk, LINE, etc. (Cf. Figure 4).*

Bridge NE, Graph Spectrum, and Graph Partition

Higher-order Cheeger's inequality suggests that eigenvalues in graph spectral are closely associated with a network' spatial locality smoothing and global clustering. First, in graph spectral theory, the random walk normalized graph Laplacian is defined as $L=I_n-D^{-1}A$. The normalized Laplacian can be decomposed as $L=U\Lambda U^{-1}$, where $\Lambda=diag([\lambda_1, ..., \lambda_n])$ with $0=\lambda_1 \leq \cdots \leq \lambda_n$ as its eigenvalues and U is the $n \times n$ square matrix whose i^{th} column is the eigenvector u_i. The graph Fourier transform of a signal x is defined as $\hat{x} = U^{-1}x$ while the inverse transform is $x = U\hat{x}$. Then a network propagation $D^{-1}Ax$ can be interpreted that x is first transformed into the spectral space and scaled by the eigenvalues, and then transformed back.

Second, the graph partition effect can be measured by the Cheeger constant (a.k.a., graph conductance). For a partition $S \subseteq V$, the constant is defined as $\phi(S) = \frac{|E(S)|}{min\{vol(S), vol(V-S)\}}$ where E(S) is the set of edges with one endpoint in S and vol(S) is the sum of nodes' degree in node set S. The k-way Cheeger constant is defined as $\rho_G(k) = min\{max\{\phi(S_i) : S_1, S_2, ..., S_k \subseteq V \ disjoint\}\}$ which reflects the effect of the graph partitioned into k parts. A smaller value means a better partitioning effect.

Higher-order Cheeger's inequality bridges the gap between graph spectrum and graph partition via controlling the bounds of k-way Cheeger constant as follows:

$$\frac{\lambda_k}{2} \leq \rho_G(k) \leq O(k^2)\sqrt{\lambda_k} \tag{7}$$

In spectral graph theory, the number of connected components in an undirected graph is equal to the multiplicity of the eigenvalue zero in graph Laplacian [Von Luxburg, 2007], which can be concluded from $\rho_G(k)$=0 when setting λ_k=0.

Eq. 7 indicates that small (large) eigenvalues control the network's global clustering (local smoothing) effect by partitioning it into a few large (many small) parts. This inspires us to incorporate the global and local network information into network embeddings by propagating the embeddings R_d in the partitioned/modulated network $D^{-1}A(I_n - \widetilde{L})$, where the Laplacian filter $\widetilde{L} = Ug(\Lambda)U^{-1}$ with g as the spectral modulator. To take both global and local structures into consideration, we design the spectral modulator as $g(\lambda) = e^{-\frac{1}{2}[(\lambda-\mu)^2-1]\theta}$. Therefore, we have the Laplacian filter

$$\widetilde{L} = Udiag([g(\lambda_1), ..., g(\lambda_n)])U^T \tag{8}$$

$g(\lambda)$ can be considered as a band-pass filter kernel [Shuman *et al.*, 2016; Hammond *et al.*, 2011] that passes eigenvalues within a certain range and attenuates eigenvalues outside that range. Hence $\rho_G(k)$ is attenuated for corresponding top largest and smallest eigenvalues, leading to the amplified local and global network information, respectively. Note that the band-pass filter is a general spectral network modulator and the other kinds of filters can also be used.

Chebyshev Expansion for Efficiency

To avoid the explicit eigendecomposition and Fourier transformation in Eq. 8, we utilize the truncated Chebyshev expansion. The Chebyshev polynomials of the first kind are defined recurrently as $T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x)$ with $T_0(x) = 1, T_1(x) = x$. Then

$$\widetilde{L} \approx U\sum_{i=0}^{k-1} c_i(\theta)T_i(\bar{\Lambda})U^{-1} = \sum_{i=0}^{k-1} c_i(\theta)T_i(\bar{L}) \tag{9}$$

where $\bar{\Lambda} = -\frac{1}{2}[(\Lambda - \mu I_n)^2 - I_n]$, $\bar{L} = -\frac{1}{2}[(L - \mu I_n)^2 - I_n]$. $\bar{\lambda} = \frac{1}{2}[(\lambda - \mu)^2 - 1]$, and the new kernel is $f(\bar{\lambda}) = e^{-\bar{\lambda}\theta}$.

As T_i is orthogonal with the weight $1/\sqrt{1-x^2}$ on the interval $[-1, 1]$, the coefficient of Chebyshev expansion for $e^{-x\theta}$ can be obtained by:

$$c_i(\theta) = \frac{\beta}{\pi} \int_{-1}^{1} \frac{T_i(x)e^{-x\theta}}{\sqrt{1-x^2}} dx = \beta(-)^i B_i(\theta) \tag{10}$$

where β=1 if i=0 otherwsie β=2 and $B_i(\theta)$ is the modified Bessel function of the first kind [Andrews and Andrews, 1992]. Then the series expansion of the Laplacian filter:

$$\widetilde{L} \approx B_0(\theta)T_0(\bar{L}) + 2\sum_{i=1}^{k-1}(-)^i B_i(\theta)T_i(\bar{L}) \tag{11}$$

Truncated Chebyshev expansion provides an approximation for $e^{-x\theta}$ with a very fast convergence rate. By combining Eqs. 11 and 6, the embeddings R_d can be enhanced by the propagation in the spectrally modulated network in a very efficient manner. In addition, to maintain the orthogonality of the original embedding space achieved by the sparse tSVD, finally we apply SVD on R_d again.

3.3 Complexity Analysis

The time complexity of SVD of H and QR decomposition of Y is $O(|V|d^2)$. Since $|\mathcal{D}| \ll |V| \times |V|$, M is a sparse matrix and the time complexity of the multiplication involved in the process above is $O(|E|)$. Therefore, the overall complexity for the first step is $O(|V|d^2 + |E|)$, which is very efficient.

The computation of Eqs. 6 and 11 can be efficiently executed in a recurrent manner. Denote $\bar{R}_d^{(i)} = T_i(\bar{L})R_d$, then $\bar{R}_d^{(i)} = 2\bar{L}\bar{R}_d^{(i-1)} - \bar{R}_d^{(i-2)}$ with $\bar{R}_d^{(0)} = R_d$ and $\bar{R}_d^{(1)} = \bar{L}R_d$. Note that $\bar{L} = -\frac{1}{2}[(L-\mu I_n)^2 - I_n]$ and L is sparse. SVD on a small matrix is $O(|V|d^2)$. Therefore, the overall complexity for the second step is $O(k|E| + |V|d^2)$.

All together, the time complexity of ProNE is $O(|V|d^2 + k|E|)$. Due to space constraint, we cannot include details about its space complexity, which is $O(|V|d + |E|)$.

3.4 Parallelizability

The computing time of ProNE is mainly spent in the sparse matrix multiplication, which is efficient enough for handling very large-scale graphs on a single thread. Nevertheless, there have been many progresses on sparse matrix multiplication parallelizability [Buluç and Gilbert, 2012; Smith *et al.*, 2015], which can offer a further speedup for our current implementation.

4 Experiments

We evaluate the efficiency and effectiveness of the ProNE method on multi-label node classification—a commonly used task for network embedding evaluation [Perozzi *et al.*, 2014; Tang *et al.*, 2015; Grover and Leskovec, 2016].

4.1 Experimental Setup

Datasets. We employ five widely-used datasets for demonstrating the effectiveness of ProNE. The dataset statistics are listed in Table 1. In addition, we also use a set of synthetic networks for evaluating its efficiency and scalability.

Dataset	*BlogCatalog*	*Wiki*	*PPI*	*DBLP*	*Youtube*
#nodes	10,312	4,777	3,890	51,264	1,138,499
#edges	333,983	184,812	76,584	127,968	2,990,443
#labels	39	40	50	60	47

Table 1: The statistics of datasets.

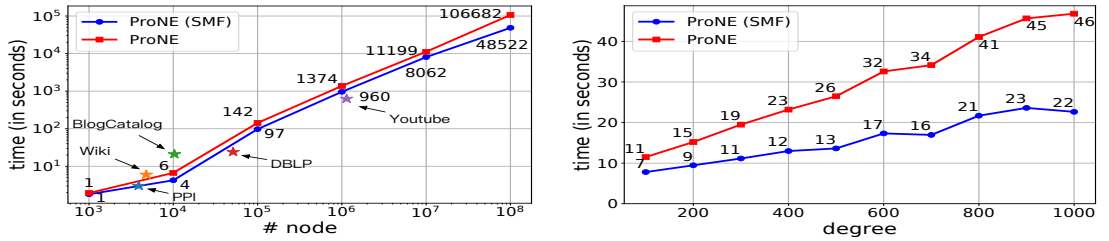
- BlogCatalog [Zafarani and Liu, 2009] is a social blogger network, in which Bloggers' interests are used as labels.
- Wiki[2] is a co-occurrence network of words in the first million bytes of the Wikipedia dump. Node labels are the Part-of-Speech tags.
- PPI [Breitkreutz *et al.*, 2008] is a subgraph of the PPI network for Homo Sapiens. Node labels are extracted from hallmark gene sets and represent biological states.
- DBLP [Tang *et al.*, 2008] is an academic citation network where authors are treated as nodes and their dominant conferences as labels.
- Youtube [Zafarani and Liu, 2009] is a social network between Youtube users. The labels represent groups of viewers that enjoy common video genres.

Baselines. We compare ProNE with popular benchmarks, including both skip-gram (DeepWalk, LINE, and node2vec) and matrix factorization (GraRep and HOPE) based methods. For a fair comparison, we set the embedding dimension $d = 128$ for all methods. For the other parameters, we follow the original authors' preferred choices. For DeepWalk and node2vec, windows size m=10, #walks per node r=80, walk length t=40. p, q in node2vec are searched over $\{0.25, 0.50, 1, 2, 4\}$. For LINE, #negative-samples $k = 5$ and total sampling budget T=$r\times t\times|V|$. For GraRep, the dimension of the concatenated embedding is d=128 for fairness. For HOPE, β is calculated in authors' code and searched over $(0, 1)$ for the best performance. For ProNE, the term number of the Chebyshev expansion k is set to 10, μ=0.2, and θ=0.5, which are the default settings. Note that convolution-based methods are excluded, as most of them are in (semi-)supervised learning settings and require side information features (such as embeddings) for training.

Running Environment. The experiments were conducted on a Red Hat server with Intel Xeon(R) CPU E5-4650 (2.70GHz) and 1T RAM. ProNE is implemented by Python 3.6.1.

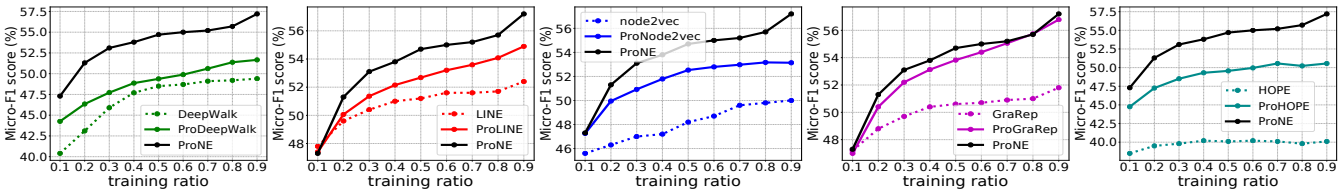
Evaluation. We follow the same experimental settings used in baseline works [Perozzi *et al.*, 2014; Grover and Leskovec, 2016; Tang *et al.*, 2015; Cao *et al.*, 2015]. We randomly sample different percentages of labeled nodes for training a liblinear classifier and use the remaining for testing. We repeat the training and predicting for *ten* times and report the average Micro-F1 for all methods. Analogous results also hold for Macro-F1, which thus are not shown due to space constraints. We follow the common practice for efficiency evaluation by the wall-clock time and ProNE's scalability is analyzed by the time cost in multiple-scale networks [Tang *et al.*, 2015].

[2] http://www.mattmahoney.net/dc/text.html



(a) The node degree is fixed to 10 and #nodes grows (b) #nodes is fixed to 10, 000 and the node degree grows

Figure 3: ProNE's scalability w.r.t. network volume and density. Blue: running time of ProNE's first step—sparse matrix factorization (SMF).



(a) ProDeepWalk (b) ProLINE (c) ProNode2vec (d) ProGraRep (e) ProHOPE

Figure 4: Spectral Propagation for enhancing baselines—ProDeepWalk, ProLINE, ProNode2vec, ProGrarep, and ProHOPE on Wiki.

Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

Table 2: Efficiency comparison based on running time (second).

4.2 Efficiency and Scalability

We compare the efficiency of different methods. The efficiency of all baselines is accelerated by using 20 threads/processes, while ProNE uses one single thread (Note that though in one minor step the number of threads used in the SciPy package is not controlled by users, its effect on efficiency is limited and all conclusions hold).

Table 2 reports the running time (both IO and computation time) of ProNE and the three fastest baselines—DeepWalk, LINE, and node2vec. The spectral matrix factorization baselines are much slower than them. For example, the time complexity of GraRep is $O(|V|^3)$, making it infeasible for relatively big networks, such as Youtube of 1.1 million nodes.

The running time results suggest that for PPI and Wiki—small networks (1,000+ nodes), ProNE requires less than 10 seconds to complete while the fastest baseline LINE is at least $14\times$ slower and DeepWalk/node2vec is about $100\times$ slower. Similar speedups can be consistently observed from BlogCatalog and DBLP—moderate-size networks (10,000+ nodes)—and YouTube—a relatively big network (1,000,000+ nodes). Remarkably, ProNE can embed the YouTube network within 11 minutes by using one thread while by using 20 threads LINE costs 100 minutes, DeepWalk requires 19 hours, and node2vec takes more than fives days. *To sum up, the one-thread ProNE model is about 10–400$\times$ faster than the 20-thread LINE, DeepWalk, and node2vec models, and its efficiency advantage over other spectral matrix factorization methods is even more significant.*

We use synthetic networks to demonstrate the scalability of ProNE and its potential for handling billion-scale networks in Figure 3. First, we generate random regular graphs with fixed node degree as 10 and the number of nodes ranging between 1,000 and 100,000,000. Figure 3(a) shows ProNE's running time for random graphs of different sizes, suggesting that the time cost of ProNE increases linearly as the network size grows. In addition, the running time for the five real datasets is also inserted into the plot, which is in line with the scalability trend on synthetic data. Therefore, we can project that it costs ProNE only $\sim$29 hours to embed a network of 0.1 billion nodes and 0.5 billion edges by using one thread, while it takes LINE over one week and may take DeepWalk/node2vec several months by using 20 threads.

Second, Figure 3(b) shows the running time of ProNE for random regular networks of a fixed size (10,000 nodes) and varied degree between 100 and 1,000. It can be clearly observed that the efficiency of ProNE is linearly correlated with network density. All together, we conclude that ProNE is a scalable network embedding approach for handling large-scale and even dense networks.

4.3 Effectiveness

We summarize the prediction performance in Table 3. Due to space limitation, we only report the results in terms of Micro-F1 and the standard deviation (σ) of the proposed model's results. Our conclusions below also hold for Macro-F1. In addition to ProNE, we also report the interim embedding results generated by the sparse matrix factorization (SMF) step in ProNE. For Youtube, we only use the two fastest and representative baselines, LINE and Deepwalk, to save time.

Dataset	training ratio	0.1	0.3	0.5	0.7	0.9
PPI	DeepWalk	16.4	19.4	21.1	22.3	22.7
	LINE	16.3	20.1	21.5	22.7	23.1
	node2vec	16.2	19.7	21.6	23.1	24.1
	GraRep	15.4	18.9	20.2	20.4	20.9
	HOPE	16.4	19.8	21.0	21.7	22.5
	ProNE (SMF)	15.8	20.6	22.7	23.7	24.2
	ProNE	**18.2**	**22.7**	**24.6**	**25.4**	**25.9**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.3)	($\pm$0.7)	($\pm$1.0)	($\pm$1.1)
Wiki	DeepWalk	40.4	45.9	48.5	49.1	49.4
	LINE	**47.8**	50.4	51.2	51.6	52.4
	node2vec	45.6	47.0	48.2	49.6	50.0
	GraRep	47.2	49.7	50.6	50.9	51.8
	HOPE	38.5	39.8	40.1	40.1	40.1
	ProNE (SMF)	47.6	51.6	53.2	53.5	53.9
	ProNE	47.3	**53.1**	**54.7**	**55.2**	**57.2**
	($\pm\sigma$)	($\pm$0.7)	($\pm$0.4)	($\pm$0.8)	($\pm$0.8)	($\pm$1.3)
BlogCatalog	DeepWalk	36.2	39.6	40.9	41.4	42.2
	LINE	28.2	30.6	33.2	35.5	36.8
	node2vec	**36.3**	39.7	41.1	42.0	42.1
	GraRep	34.0	32.5	33.3	33.7	34.1
	HOPE	30.7	33.4	34.3	35.0	35.3
	ProNE (SMF)	34.6	37.6	38.6	39.3	39.0
	ProNE	36.2	**40.0**	**41.2**	**42.1**	**42.7**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.3)	($\pm$0.6)	($\pm$0.7)	($\pm$1.2)
Dataset	training ratio	0.01	0.03	0.05	0.07	0.09
DBLP	DeepWalk	49.3	55.0	57.1	57.9	58.4
	LINE	48.7	52.6	53.5	54.1	54.5
	node2vec	48.9	55.1	57.0	58.0	58.4
	GraRep	50.5	52.6	53.2	53.5	53.8
	HOPE	**52.2**	55.0	55.9	56.3	56.6
	ProNE (SMF)	50.8	54.9	56.1	56.7	57.0
	ProNE	48.8	**56.2**	**58.0**	**58.8**	**59.2**
	($\pm\sigma$)	($\pm$1.0)	($\pm$0.5)	($\pm$0.2)	($\pm$0.2)	($\pm$0.1)
Youtube	DeepWalk	38.0	40.1	41.3	42.1	42.8
	LINE	33.2	35.5	37.0	38.2	39.3
	ProNE (SMF)	36.5	40.2	41.2	41.7	42.1
	ProNE	**38.2**	**41.4**	**42.3**	**42.9**	**43.3**
	($\pm\sigma$)	($\pm$0.8)	($\pm$0.3)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)

Table 3: The classification performance in terms of Micro-F1 (%).

We observe that ProNE consistently generates better results than baselines across five datasets, demonstrating its strong effectiveness. Interestingly, it turns out that the simple sparse matrix factorization (SMF) step for fast embedding initialization in ProNE is comparable to or sometimes even better than existing popular network embedding benchmarks. With the spectral propagation technique further incorporated, ProNE generates the best performance among all baselines due to its effective modeling of local structure smoothing and global clustering information.

Spectral Propagation for Embedding Enhancement

Recall that ProNE composes of two steps: 1) sparse matrix factorization for fast embedding initialization and 2) spectral propagation for enhancement. Can spectral propagation also help improve the baseline methods?

We input the embeddings learned by DeepWalk, LINE, node2vec, GraRep, and HOPE into ProNE's spectral propagation step. Figure 4 shows both the original and enhanced results (denoted as "ProBaseline') on Wiki, illustrating significant improvements achieved by the Pro versions for all five baselines. On average, our spectral propagation strategy offers $+10\%$ relative gains for all methods, such as the 25% improvements for HOPE. Moreover, all enhancement experiments are completed in one second. *The results demonstrate that the spectral propagation in ProNE is effective and a general and fast strategy for improving network embeddings.*

5 Related Work

The recent emergence of network embedding is largely triggered by representation learning natural language processing [Mikolov *et al.*, 2013b]. Its history can date back to spectral clustering [Chung, 1997] and social dimension learning [Tang and Liu, 2009]. Over the course of its development, most network embedding methods aim to model distributional similarities of nodes either implicitly or explicitly.

Inspired by the word2vec model, a line of skip-gram based embedding models have been presented to encode network structures into continuous spaces, such as DeepWalk [Perozzi *et al.*, 2014], LINE [Tang *et al.*, 2015], node2vec [Grover and Leskovec, 2016], and metapath2vec [Dong *et al.*, 2017]. Recently, learned from [Levy and Goldberg, 2014], a study shows that skip-gram based network embedding can be understood as implicit matrix factorization and it also presents the NetMF model to perform explicit matrix factorization for learning network embeddings [Qiu *et al.*, 2018]. The difference between NetMF and our model lies in that the matrix to be factorized by NetMF is a dense one, whose construction and factorization involve computation in $O(|V|^3)$ time complexity, while our ProNE model formalizes network embedding as sparse matrix factorization in $O(|E|)$.

The other recent matrix factorization based network embedding models include GraRep [Cao *et al.*, 2015] and HOPE [Ou *et al.*, 2016]. Spectral network embedding is related to spectral dimension reduction methods, such as Laplacian Eigenmaps [Belkin and Niyogi, 2001] and spectral clustering [Yan *et al.*, 2009]. These matrix decomposition based methods usually require expensive computation and excessive memory consumption due to their high time and space complexity.

Another significant line of work focuses on generalizing graph spectral into (semi-)supervised graph learning, such as graph convolution networks (GCNs) [Henaff *et al.*, 2015; Defferrard *et al.*, 2016; Kipf and Welling, 2017]. In GCNs, the convolution operation is defined in the spectral space and parametric filters are learned via back-propagation. Different from them, our ProNE model features a band-pass filter incorporating both spatial locality smoothing and global clustering properties. Furthermore, ProNE is an unsupervised and task-independent model that aims to pre-train general embeddings, while most GCNs are (semi-)supervised with side features as input, such as the network embeddings learned by ProNE.

6 Conclusions

In this work, we propose ProNE—a fast and scalable network embedding approach. It achieves both efficiency and effectiveness superiority over recent powerful network embedding benchmarks, such as DeepWalk, LINE, node2vec, GraRep, and HOPE. Remarkably, the single-thread ProNE model is $\sim$10–400$\times$ faster than the aforementioned baselines that are accelerated by using 20 threads. For future work, we would like to apply the sparse matrix multiplication parallelizability technique to speed up ProNE as discussed in Section 3.4. In addition, we are also interested in exploring the connection between graph spectral based factorization models and graph convolution and graph attention networks.

References

[Andrews and Andrews, 1992] Larry C Andrews and Larry C Andrews. *Special functions of mathematics for engineers*. McGraw-Hill New York, 1992.

[Bandeira *et al.*, 2013] Afonso S Bandeira, Amit Singer, and Daniel A Spielman. A cheeger inequality for the graph connection laplacian. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1611–1630, 2013.

[Belkin and Niyogi, 2001] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, pages 585–591, 2001.

[Breitkreutz *et al.*, 2008] Bobby-Joe Breitkreutz, Chris Stark, et al. The biogrid interaction database. *Nucleic acids research*, 36(suppl 1):D637–D640, 2008.

[Buluç and Gilbert, 2012] Aydin Buluç and John R Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal on Scientific Computing*, 34(4):C170–C191, 2012.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, pages 891–900, 2015.

[Chung, 1997] Fan RK Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[Dong *et al.*, 2017] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144. ACM, 2017.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[Halko *et al.*, 2011] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[Hamilton *et al.*, 2017a] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017.

[Hamilton *et al.*, 2017b] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data(base) Engineering Bulletin*, 40:52–74, 2017.

[Hammond *et al.*, 2011] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[Harris, 1954] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Lee *et al.*, 2014] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *JACM*, 61(6):37, 2014.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Ou *et al.*, 2016] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[Qiu *et al.*, 2018] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467, 2018.

[Shuman *et al.*, 2016] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[Smith *et al.*, 2015] Shaden Smith, Niranjay Ravindran, Nicholas D Sidiropoulos, and George Karypis. Splatt: Efficient and parallel sparse tensor-matrix multiplication. In *IPDPS*, pages 61–70. IEEE, 2015.

[Tang and Liu, 2009] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *KDD*, 2009.

[Tang *et al.*, 2008] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, pages 990–998, 2008.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Von Luxburg, 2007] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[Yan *et al.*, 2009] Donghui Yan, Ling Huang, and Michael I Jordan. Fast approximate spectral clustering. In *KDD*, pages 907–916, 2009.

[Zafarani and Liu, 2009] Reza Zafarani and Huan Liu. Social computing data repository at asu, 2009.