Node Embedding via Adaptive Similarities

Dimitris Berberidis and Georgios B. Giannakis
Dept. of Electrical & Computer Engineering, University of Minnesota
{bermp001,georgios}@umn.edu

ABSTRACT

Node embedding is faced with several important challenges. Practical node embedding methods are required to cope with real-world graphs that arise from a variety of different domains, with inherently diverse underlying processes and similarity structures. On the other hand, much like PCA in the feature domain, node embedding is an inherently *unsupervised* task; in lack of metadata used for validation, practical methods may require standardization and limiting the use of tunable hyperparameters. Finally, node embedding methods are faced with maintaining scalability in the face of large-scale real-world graphs of ever-increasing sizes. In the present work, we propose an adaptive node embedding framework that adjusts the embedding process to a given underlying graph, in a fully unsupervised manner. To achieve this, we adopt the notion of a tunable node similarity matrix that assigns weights on paths of different length. The design of the multilength similarities ensures that the resulting embeddings also inherit interpretable spectral properties. An algorithmic scheme is proposed for training the model parameters effieciently and in an unsupervised manner. We perform extensive node classification, and clustering experiments on many real world graphs from various domains, and compare with state-of-the-art scalable and unsupervised node embedding alternatives. The proposed method enjoys superior performance in many cases, while also yielding interpretable information on the underlying structure of the graph.[1]

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; • **Information systems** → *Data mining*;

KEYWORDS

SVD, SVM, unsupervised, multiscale, random walks, spectral

ACM Reference Format:
Dimitris Berberidis and Georgios B. Giannakis. 2019. Node Embedding via Adaptive Similarities. In *Proceedings of 2019 ACM SIGKDD conference (KDD19 (MLG Workshop)).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Unsupervised node embedding is an exciting field, in which a significant ammount of progress has been made in recent years [10]. The task consists of mapping each node of a graph to a vector in a low-dimensional Eucledian space. The main goal is to *extract features* that can be utilized downstream in order to perform a variety of unsupervised or semi-supervised learning tasks, i.e. node classification, link prediction, or clustering. In theory, the original graph will contain at least as much information as the resulting embedded vectors. Nevertheless, an appropriate embedding can boost the performance of certain learning tasks by allowing us to work with the more "friendly" and intuitive Eucledian representation, and deploy mature and widely implemented feature-based algorithms such as SVMs, logistic regression, and K-means.

Early embedding work mostly focused on a structure-preserving dimensionality reduction of feature vectors (instead of nodes); see for instance [15–19]. In this context, graphs are constructed from pairwise feature-vector relations and are treated as representations of the manifold that data lie on; embedded vectors are then generated such that they preserve the corresponding pair-wise proximities on the manifold. More recently, the task of embedding the nodes of a graph has attracted considerable attention in different fields, and is often posed as the factorization of a properly defined node similarity matrix [20–27]. Efforts in this direction mostly focus on designing meaningful similarity metrics to factorize. While some methods (e.g. [20, 22]) maintain scalability by factorizing similarity matrices in an implicit manner (i.e., without explicitly forming them) , others such as [23, 24] form and/or factorize dense similarity matrices that scale poorly to large graphs. Another line of work opts to gradually fit pairs of embedded vectors to existing edges using stochastic optimization tools [28, 30]. Recently, stochastic edge-fitting has been generalized to implicitly accommodate long-range node similarities [29]. Meanwhile, other works have approached node embeddings using random-walk-based tools and concepts that originate in natural language processing [31–33]; see also related works on embedding of knowledge graphs [34, 35]. Methods that rely on graph convolutional neural networks and autoencoders have also been proposed for node embedding [38, 39]. Moreover, a gamut of related embedding tasks are gaining traction, such as embedding based on structural roles of nodes [36, 37], supervised embeddings for classification [7], and inductive embedding methods that utilize multiple graphs [6].

We identify the following *challenges* that need to be addressed in order to design embedding methods that are applicable in practice:

- **Diversity**. Since graphs that arise from different domains are generally characterized by a diverse set of properties, there may not be a "one-size-fits-all" node embedding approach.
- **No supervision**. At the same time, node embedding may need to be performed in a *fully unsupervised* manner, that is, without extra information (node attributes, labels, or groundtruth communities) to guide the parameter tuning process with cross-validation.
- **Scalability**. While some real world networks are of moderate size, others may contain massive numbers of nodes and edges. Thus, strict computational constrains need to be incorporated into the design of node embedding methods.

[1] 1This work was supported by NSF 171141, NSF 1514056 and NSF 1500713.

KDD19 (MLG Workshop), August 2019, Alaska, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Targeting at the three aforementioned challenges, we propose a scalable node embedding framework that is based on *jointly* factorizing and learning an adaptive node similarity matrix that places weights on node proximities of different orders. The learning approach is unsupervised, and uses only the graph structure. Experiments indicate that the proposed similarity model is expressive enough to effectively embed real-world graphs from diverse domains and with different structures and properties. [2]

2 PROBLEM STATEMENT AND MODELING

Given an undirected graph $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of N nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, the task of node embedding boils down to determining $f(\cdot) : \mathcal{V} \rightarrow \mathbb{R}^d$, where $d \ll N$. Thus, we seek for a functions that maps every node of the graph to a vector in the $d-$dimensional Eucledian space; typically, the embedding is low-dimensional with d being much smaller than the number of nodes. Since the number of node on a graph is finite, instead of finding a general $f(\cdot)$ (induction), one may pose the embedding task in its most general form as the following minimization problem wrt to the embedded vectors

$$\{\mathbf{e}_i^*\}_{i \in \mathcal{V}} = \arg \min_{\{\mathbf{e}_i\}_{i \in \mathcal{V}}} \sum_{i,j \in \mathcal{V}} \ell \left(s_{\mathcal{G}}(v_i, v_j), s_{\mathcal{E}}(\mathbf{e}_i, \mathbf{e}_j) \right), \quad (1)$$

where $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a loss function; $s_{\mathcal{G}}(\cdot,\cdot) : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ is a similarity metric defined over every pair of *nodes* of the graph; and, $s_{\mathcal{E}}(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a similarity metric defined over every pair of *vectors* in the $d-$dimensional Eucledian space. Thus, according to (1), node embedding can be viewed as the design of vectors $\{\mathbf{e}_{i \in \mathcal{V}}\}$ that successfully "encode" a certain notion of pairwise similarities between nodes.

2.1 Embedding as matrix factorization

Starting from the generalized framework in (1), one may arrive at more concrete approaches by imposing specifications to $s_{\mathcal{G}}(\cdot,\cdot)$, $s_{\mathcal{E}}(\cdot,\cdot)$, and $\ell(\cdot,\cdot)$. Thus, let us specify the node similarity metric to be symmetric, i.e. $s_{\mathcal{G}}(v_i, v_j) = s_{\mathcal{G}}(v_i, v_j) \, \forall v_i, v_j \in \mathcal{V}$. Furthermore, let the loss function be quadratic $\ell(x, x') = (x - x')^2$, and the vector similarity be the inner product $s_{\mathcal{E}}(\mathbf{e}_i, \mathbf{e}_j) = \mathbf{e}_i^T \mathbf{e}_j$. Using the above specifications, (1) becomes equivalent to the following symmetric matrix factorization problem

$$\mathbf{E}^* = \arg \min_{\mathbf{E} \in \mathbb{R}^{N \times d}} \|\mathbf{S}_{\mathcal{G}} - \mathbf{E}\mathbf{E}^T\|_F^2 \quad (2)$$

where $\mathbf{S}_{\mathcal{G}} \in \mathbb{R}^{N \times N}$ is the symmetric similarity matrix with $\left[\mathbf{S}_{\mathcal{G}}\right]_{i,j} = \left[\mathbf{S}_{\mathcal{G}}\right]_{j,i} = s_{\mathcal{G}}(v_i, v_j)$, and matrix $\mathbf{E} = [\mathbf{e}_1 \ldots \mathbf{e}_N]^T$ concatenates all the node embeddings as rows. A well known way to obtain an analytical solution to (2) is via the singular value decomposition (SVD) of the similarity matrix, that is $\mathbf{S}_{\mathcal{G}} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the $N \times N$ unitary matrices containing the left and right singular vectors, and Σ is diagonal with non-negative singular values sorted in decreasing order; in our case, $\mathbf{U} = \mathbf{V}$ since $\mathbf{S}_{\mathcal{G}}$ is symmetric. Given the SVD of $\mathbf{S}_{\mathcal{G}}$, it can be shown [13] that $\mathbf{E}^* = \mathbf{U}_d\sqrt{\Sigma_d}$, where Σ_d contains the d largest singular values and $\mathbf{U}_d$ the corresponding singular vectors. Fortunately, $\mathbf{U}_d$ and Σ_d can be obtained directly, without computing the full SVD, via a process known as the *truncated* SVD that has reduced complexity. Moreover, if $\mathbf{S}_{\mathcal{G}}$ is *sparse*, (2) can be solved even more efficiently, with complexity that scales with the number of edges. One example of such sparse similarities is the adjacency matrix itself $\mathbf{A}$, i.e., using $\mathbf{S}_{\mathcal{G}} = \mathbf{A}$. In general, embeddings can achieve computational scalability by avoiding the explicit construction of a *dense* $\mathbf{S}_{\mathcal{G}}$. In fact, simply storing $\mathbf{S}_{\mathcal{G}}$ into working memory becomes prohibitive even for graphs of moderate sizes ($N > 10^5$). In the following section, we design a family of dense similarity matrices that (among other properties) can be decomposed implicitly, at the cost of input sparsity.

2.2 Multi-length node similarities

Having reduced the node embedding problem to the one in (2), it remains to specify the node similarity metric that gives rise to $\mathbf{S}_{\mathcal{G}}$. Towards this directions, and to maintain expressibility, we will aim at designing a parametrized model for $\mathbf{S}_{\mathcal{G}}$, where each pairwise node similarity is given as

$$s_{\mathcal{G}}(u_i, v_j; \boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k s(v_i, v_j, k), \text{ s.t. } \boldsymbol{\theta} \in \mathcal{S}^K, \quad (3)$$

where $\mathcal{S}^K := \{\boldsymbol{\theta} \in \mathbb{R}^K : \boldsymbol{\theta} \geq \mathbf{0}, \boldsymbol{\theta}^T\mathbf{1} = 1\}$ is the $K-$th dimensional probability simplex, and $s(v_i, v_j, k)$ is a similarity function that depends on all $k-$length paths (of possibly repeated nodes) that start from v_i and end in v_j (or vice-versa). Thus, $s_{\mathcal{G}}(\cdot,\cdot;\boldsymbol{\theta})$ contains all $k-$length (for $k \leq K$) interactions between two nodes, each weighted with a non-negative importance score θ_k.

Let $\mathbf{S}$ be any similarity matrix that is characterized by the same sparsity pattern as the adjacency matrix, that is

$$S_{i,j} = \begin{cases} s_{i,j}, & (i,j) \in \mathcal{E} \\ 0, & (i,j) \notin \mathcal{E} \end{cases}, \quad (4)$$

where $s_{i,j}$'s denote the generic non-negative values of entries that correspond to edges of $\mathcal{G}$. Maintaining the same sparsity pattern as $\mathbf{A}$ allows for the (i,j) the $k-$th power $\mathbf{S}^k$ to be interpreted as a measure of influence between v_i and v_j that depends on all $k-$length paths that connect them; that is, $\left[\mathbf{S}^k\right]_{i,j} = s(v_1, v_2, k)$. For instance, selecting $\mathbf{S} = \mathbf{A}$ is equivalent to using the $k-$step similarity $s(v_i, v_j, k) = |\{k-\text{length paths connecting } v_i \text{ to } v_j\}|$ [8]. Similarly, if $\mathbf{S} = \mathbf{A}\mathbf{D}^{-1}$ where $\mathbf{D} = \text{diag}(\mathbf{1}^T\mathbf{A})$, then $s(v_i, v_j, k)$ can be interpreted as the probability that a random walk starting from v_j lands on v_i after exactly k steps, e.g., [24]. Thus, for an appropriately selected $\mathbf{S}$ that follows (4), tunable multi-length similarity metrics in (3) can be collected as matrix entries in the form of a power series, that is

$$\mathbf{S}_{\mathcal{G}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \mathbf{S}^k, \text{ s.t. } \boldsymbol{\theta} \in \mathcal{S}^K \quad (5)$$

Upon, substituting (5) into (2) yields the tunable embeddings $\mathbf{E}^*(\boldsymbol{\theta})$ that depend on the choice of parameters $\boldsymbol{\theta}$. Moreover, from the SVD $\mathbf{S} = \mathbf{U}\Sigma\mathbf{U}^T$, and given that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, it readily follows that

$$\mathbf{S}^k = \mathbf{U}\Sigma^k\mathbf{U}^T, \quad (6)$$

[2]Full version of paper available here: https://arxiv.org/pdf/1811.10797.pdf

and by using (6) in (5) we obtain

$$\mathrm{S}_{\mathcal{G}}(\boldsymbol{\theta}) = \mathbf{U}\left(\sum_{k=1}^{K} \theta_k \Sigma^k\right)\mathbf{U}^T, \text{ s.t. } \boldsymbol{\theta} \in \mathcal{S}^K. \tag{7}$$

Furthermore, the truncated singular pairs of $\mathrm{S}_{\mathcal{G}}(\boldsymbol{\theta})$ conveniently follows from that of S and thus only needs to be computed once. Specifically, the truncated singular vectors and singular values are given as $\mathbf{U}_d(\boldsymbol{\theta}) = \mathbf{U}_d$ and $\Sigma_d(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \Sigma_d^k$ respectively. Thus, if $\mathrm{S} \in \mathrm{Sym}_N$ the solution to (2) with $\mathrm{S}_{\mathcal{G}}$ parametrized by $\boldsymbol{\theta}$ is simply given as

$$\mathbf{E}^*(\boldsymbol{\theta}) = \mathbf{U}_d\sqrt{\Sigma_d(\boldsymbol{\theta})} \tag{8}$$

Note that this holds only for non-negative parameters, i.e. $\theta_k \geq 0 \; \forall \; k$. If $\theta_k < 0$ for at least one $k \in [1, K]$, then the diagonal elements of $\Sigma_d(\boldsymbol{\theta})$ cannot be guaranteed to be non-negative and sorted in decreasing order, which would cause $(\mathbf{U}_d(\boldsymbol{\theta}), \Sigma_d(\boldsymbol{\theta}))$ to *not* be a valid SVD pair. Finally, having narrowed down $\mathrm{S}_{\mathcal{G}}$ to belong to the parametrized family in (5), we arrive at selecting an appropriate sparsity-preserving S in order to obtain a solid model.

2.3 Spectral multi-length embeddings

While any symmetric S that obeys (4) can be used for constructing multi-length similarities (cf. (5)), certain desirable properties may materialize by properly designing S. We begin by recalling the following identity

$$\mathrm{S} \in \mathcal{P}_N^+ \iff \mathrm{S} = \mathbf{U}\Sigma\mathbf{U}^T = \mathbf{U}\Lambda\mathbf{U}^T, \tag{9}$$

where $\mathcal{P}_N^+$ denotes the space of $N \times N$ symmetric positive definite (SPD) matrices, and Λ is the diagonal matrix that contains the eigenvalues of S sorted in decreasing order. According to (9), for SPD matrices, the SVD is identical to the eigenvalue decomposition (EVD). Thus, if $\mathrm{S} \in \mathcal{P}_N^+$, the solution to (2) is also given as (cf.(8))

$$\mathbf{E}^*(\boldsymbol{\theta}) = \mathbf{U}_d\sqrt{\Lambda_d(\boldsymbol{\theta})}, \tag{10}$$

where $\mathbf{U}_d$ are also the first d *eigenvectors* of S, and $\Lambda_d(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \Lambda_d^k$ is the $K-$order polynomial of its eigenvalues defined by $\boldsymbol{\theta}$.

Consider now that we specify S to be

$$\mathrm{S} = \frac{1}{2}\left(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right). \tag{11}$$

Clearly, (11) is SPD; this follows upon recalling that $\lambda_i\left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right) \in [-1, 1] \; \forall \; i$, and from identity shifting and scaling, it readily follows that $\lambda_i(\mathrm{S}) \in [0, 1] \; \forall \; i$. More importantly, it can easily be verified that the first d eigenvectors of S are the same as the eigenvectors that correspond to the d smallest eigenvalues of the symmetric normalized Laplacian matrix

$$\mathbf{L}_{\text{sym}} := \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{12}$$

The latter are known to contain useful information on cluster structures of different resolution levels, a key property that has been succesfully used by spectral clustering [11]. Intuitively, assigning weight θ_k to $k-$length paths in the node similarity in (5), is equivalent (10) to shrinking the $d-$dimensional spectral node embeddings (rows of $\mathbf{U}_d$) coordinates according to $\Lambda_d(\boldsymbol{\theta})$. Interestingly, assigning large weights to longer paths ($K \gg 1$) is equivalent to fast shrinking of the coordinates that correspond to small eigenvalues and capture the fine-grained structures and local relations, and leads to a coarse, high-level cluster description of the graph.

3 UNSUPERVISED SIMILARITY LEARNING

For a given graph, we must select a specific $\boldsymbol{\theta} \in \mathcal{S}^K$ without supervision. Let us begin by assuming that for a given set of nodes, an adjacency matrix A is generated according to a distribution $f_A(\mathbf{A})$ defined over the space of all possible adjacency matrices. We define the "true" underlying similarity between nodes v_i and v_j to be

$$s^*(v_i, v_j) := \Pr\{(i, j) \in \mathcal{E}\} = \mathbb{E}_{f_A}\left[A_{i,j}\right],$$

which is the probability that the two nodes are connected. The "true" similarity matrix is thus given as the expected adjacency matrix $\mathrm{S}^* := \mathbb{E}_{f_A}[\mathbf{A}]$. Since $\mathrm{S}_{\mathcal{G}}(\boldsymbol{\theta}) = \mathrm{S}_{\mathcal{G}}(\mathbf{A}; \boldsymbol{\theta})$ essentially acts as an estimate of S^* from one graph realization (A), one is motivated to fit the parameters $\boldsymbol{\theta}$, ideally by minimizing an expected cost as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \mathbb{E}_{f_A}\left[\ell\left(\mathrm{S}^*, \mathrm{S}_{\mathcal{G}}(\mathbf{A}; \boldsymbol{\theta})\right)\right] \tag{13}$$

Unfortunately, we only have one realization A of $f_A(\cdot)$ which means that, in the absence of some prior knowledge, the best approximation of S^* that we can obtain is the adjacency matrix itself, that is $\mathrm{S}^* \approx \mathbf{A}$. Using this approximation yields

$$\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \ell\left(\mathbf{A}, \mathrm{S}_{\mathcal{G}}(\mathbf{A}; \boldsymbol{\theta})\right). \tag{14}$$

While straightforward, (14) yields embeddings with limited generalization capability. Simply put, regardless of the choice of $\ell(\cdot)$, solving (14) amounts to predicting a set of edges by tuning a similarity metric that is generated by the *same* set of edges.

To mitigate overfitting and promote generalization of the similarity metric, and of the resulting embeddings, we explore the folowing idea. Assume that we are given a pair $\mathbf{A}_1, \mathbf{A}_2$ of adjacency matrices both drawn independently from $f_A(\cdot)$. In that case, we would be able to use one as approximation of $\mathrm{S}^* \approx \mathbf{A}_1$, and the other to form the multilength similarity matrix $\mathrm{S}_{\mathcal{G}}(\mathbf{A}_2; \boldsymbol{\theta})$; parameters $\boldsymbol{\theta}$ can then be learned by solving

$$\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \ell\left(\mathbf{A}_1, \mathrm{S}_{\mathcal{G}}(\mathbf{A}_2; \boldsymbol{\theta})\right). \tag{15}$$

Since separate samples are not available, we approximate the above process by randomly extracting part of A and approaching (15) as

$$\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \ell_{\mathcal{S}}\left(\mathbf{A}, \mathrm{S}_{\mathcal{G}}(\mathbf{A} * \mathrm{S}^c; \boldsymbol{\theta})\right), \tag{16}$$

where $\mathcal{S} \in \{1, \ldots, N\}^2$ is a subset of all possible pairs of nodes with $|\mathcal{S}| = N_s$, and S^c is an $N \times N$ binary section matrix with $S_{i,j}^c = 0$ if $\{i, j\} \in \mathcal{S}$ and $S_{i,j}^c = 1$ otherwise; furthermore, $\ell_{\mathcal{S}}(\cdot, \cdot)$ in (16) denotes cost $\ell(\cdot, \cdot)$ applied selectively only for the entries of the matrix variables that belong to $\mathcal{S}$. Here, such that $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, with $\mathcal{S}^+ \in \mathcal{E}$ being as subset of the edges and $\mathcal{S}^- \in \{1, \ldots, N\}^2 \setminus \mathcal{E}$ a subset of node index tuples that are not connected (non-edges). To balance the influence of existing and non-existing edges, we use subsets of equal cardinality, that is $|\mathcal{S}^+| = |\mathcal{S}^-| = N_s/2$.

To arrive from the unsupervised similarity learning framework (16) to an applicable method, it remains to specify two modular subsystems: one responsible for sampling edges, and one that specifies $\ell(\cdot, \cdot)$ and finds $\boldsymbol{\theta}^*$ by solving (16).

3.1 Edge sampling

The choice of sampling scheme for $\mathcal{S}$ plays an important role in the overall performance of the proposed adaptive embedding framework. Ideally, edge sampling should satisfy the following criteria

- Sample $\mathcal{S}^+$ should be representative of the graph.
- Edge removal should inflict minimal perturbation.
- Edge removal should avoid isolating nodes.
- Simplicity and scalability.

To strike a good balance between the above objectives, we populate $\mathcal{S}^+$ by sampling edges according to the following procedure: first, a node v_1 is sampled uniformly at random from $\mathcal{V}$; then, a second node v_2 is sampled uniformly from the neighborhood set $\mathcal{N}_{\mathcal{G}}(v_1)$ of v_1. The selected edge is removed only if both adjacent nodes have degree larger than one. Non-edges $\mathcal{S}^-$ are obtained by uniform sampling without replacement over $\{1,\dots,N\}^2 \setminus \mathcal{E}$. The overall procedure is summarized in Algorithm 2. For $N_s \ll N$, sampling probabilities remain approximately unchanged despite the removals, since the probability of selecting the same node is relatively small. Thus, one may approximate $\Pr\{e_t = (i,j)\} \approx \Pr\{e_0 = (i,j)\}$, and assuming for simplicity that $d_i > 1 \forall i$, it follows that

$$\begin{aligned}\Pr\{e_0 = (i,j)\} &= \Pr\{v_1 = i, v_2 = j\} + \Pr\{v_1 = j, v_2 = i\} \\ &= \Pr\{v_2 = i | v_1 = j\}\Pr\{v_1 = j\} \\ &\quad + \Pr\{v_2 = j | v_1 = i\}\Pr\{v_1 = i\} \\ &= \frac{1}{d_j}\frac{1}{N} + \frac{1}{d_i}\frac{1}{N} \propto \frac{d_i + d_j}{d_i d_j}, \end{aligned} \tag{17}$$

meaning that edge $e = (i,j)$ is removed with probability that is proportional to the harmonic mean of the degrees of the nodes that it connects. As shown in [9], the perturbation that the removal of edge $e = (i,j)$ inflicts on the spectrum of an undirected graph is proportional to $d_i d_j$; that is, removing edges that connect high-degree nodes leads to higher perturbation. Thus, Algorithm 2 tends to inflict minimal perturbation by sampling with probability that is inversely proportional to $d_i d_j$ for $d_i,\ d_j \gg 1$; this follows the fact that the denominator of (17) dominates its numerator for large degrees. On the other hand, for smaller d_i and d_j, the numerator ensures relatively high probabilities for moderate-degree nodes. The combination of the two effects produces edge samples that are fairly representative of the graph, while inflicting low perturbation when removed.

3.2 Parameter training

Subsequently, for a given sample $\mathcal{S}$, we can obtain the corresponding optimal parameters as (cf. (16))

$$\boldsymbol{\theta}^*_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta} \in \mathcal{S}^K} \sum_{i,j \in \mathcal{S}} \ell\left(A_{i,j}, s_{\mathcal{G}^-}(u_i, v_j; \boldsymbol{\theta})\right) \tag{18}$$

where $\mathcal{G}^- = \left(\mathcal{V}, \mathcal{E} \setminus \mathcal{S}^+\right)$ is the original graph with the randomly sampled subset $\mathcal{S}^+$ of edges removed.

Instead, we will rely on the fact that the proposed embeddings are smooth and differentiable wrt to $\boldsymbol{\theta}$ (cf. (10)), to develop a solution

Algorithm 1 Adaptive Similarity Embedding

```
Input: $\mathcal{G}$ Output: E
// Training phase
$\Theta = \emptyset$
while $|\Theta| < T_s$ do
    $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$ = Sample Edges( $\mathcal{G}$ )
    $\boldsymbol{\theta}^*_{\mathcal{S}}$ = Train Parameters( $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$ )
    $\Theta = \Theta \cup \boldsymbol{\theta}^*_{\mathcal{S}}$
end while
$\boldsymbol{\theta}^* = T_s^{-1} \sum_{\boldsymbol{\theta} \in \Theta} \boldsymbol{\theta}$
// Embedding phase
$\mathrm{S} = \frac{1}{2}\left(\mathrm{I} + \mathrm{D}^{-1/2}\mathrm{A}\mathrm{D}^{-1/2}\right)$
$\mathrm{S} = \mathrm{U}_d \Sigma_d \mathrm{U}_d^T$
$\Sigma_d(\boldsymbol{\theta}^*) = \sum_{k=1}^K \theta^*_k \Sigma_d^k$
return $\mathrm{E} = \mathrm{U}_d\sqrt{\Sigma_d(\boldsymbol{\theta}^*)}$
```

Algorithm 2 Sample Edges

```
Input: $\mathcal{G}$ Output: $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$
// Sample edges
$\mathcal{S}^+ = \emptyset, \mathcal{G}^- = \mathcal{G}$
while $|\mathcal{S}^+| < N_s/2$ do
    Sample $v_1 \sim$ Unif $(\mathcal{V})$
    if $|\mathcal{N}_{\mathcal{G}^-}(v_1)| > 1$ then
        Sample $v_2 \sim$ Unif $\left(\mathcal{N}_{\mathcal{G}^-}(v_1)\right)$
        if $|\mathcal{N}_{\mathcal{G}^-}(v_2)| > 1$ then
            $\mathcal{S}^+ = \mathcal{S}^+ \cup (v_1, v_2)$
            $\mathcal{G}^- = \mathcal{G}^- \setminus (v_1, v_2)$
        end if
    end if
end while
// Sample non-edges
$\mathcal{S}^- = \emptyset$
while $|\mathcal{S}^-| < N_s/2$ do
    Sample $(v_1, v_2) \sim$ Unif $(\mathcal{V} \times \mathcal{V})$
    if $(v_1, v_2) \notin \mathcal{E}$ then
        $\mathcal{S}^- = \mathcal{S}^- \cup (v_1, v_2)$
    end if
end while
return $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-$
```

Algorithm 3 Train Parameters

```
Input: $\mathcal{G}, \mathcal{S}^+, \mathcal{S}^-$ Output: $\boldsymbol{\theta}^*_{\mathcal{S}}$
$\mathrm{S} = \frac{1}{2}\left(\mathrm{I} + \mathrm{D}^{-1/2}\mathrm{A}\mathrm{D}^{-1/2}\right)$
$\mathrm{S} = \mathrm{U}_d \Sigma_d \mathrm{U}_d^T$
$\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$
Form $\mathcal{X}_{\mathcal{S}} = \{\mathbf{x}_{(i,j)}\}_{(i,j) \in \mathcal{S}}$ as in (20)
return $\boldsymbol{\theta}^*_{\mathcal{S}}$ = SimplexSVM( $\mathcal{X}_{\mathcal{S}}, \mathcal{S}^+, \mathcal{S}^-$ )
```

that allows for selecting arbitrarily large N_s, using the approximation

$$\begin{aligned} s_{\mathcal{G}^-}(u_i, v_j; \boldsymbol{\theta}) &\approx s_{\mathcal{E}}(\mathbf{e}_i^-(\boldsymbol{\theta}, \mathbf{e}_j^-(\boldsymbol{\theta})) = \left(\mathbf{e}_i^-(\boldsymbol{\theta})\right)^T \mathbf{e}_j^-(\boldsymbol{\theta}) \\ &= \left(\sqrt{\Sigma_d^-(\boldsymbol{\theta})}\, \mathbf{u}_i^-\right)^T \sqrt{\Sigma_d^-(\boldsymbol{\theta})}\, \mathbf{u}_j^- \\ &= \left(\mathbf{u}_i^-\right)^T \Sigma_d^-(\boldsymbol{\theta}) \mathbf{u}_j^- = \mathbf{x}_{i,j}^T \boldsymbol{\theta} \end{aligned} \tag{19}$$

where

$$\mathbf{x}_{i,j} = \left(\mathbf{u}_i^- * \mathbf{u}_j^-\right)^T \Sigma_d^K, \tag{20}$$

and

$$\Sigma_d^K = \begin{bmatrix} \sigma_1 & \sigma_1^2 & \cdots & \sigma_1^K \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d-1} & \sigma_{d-1}^2 & \cdots & \sigma_{d-1}^K \\ \sigma_d & \sigma_d^2 & \cdots & \sigma_d^K \end{bmatrix}.$$

Conveniently, $\mathbf{x}_{i,j}$'s act as features over every possible pair of nodes, which when linearly combined with weights $\boldsymbol{\theta}$ to produce similarities; this allows us to approach (18) using well-understood learning and optimization tools. For instance, let us define $\ell(\cdot)$ to be the Hinge loss $\ell(y, f) := \max(0, \epsilon - yf)$, and, upon defining targets $y_{i,j} = 2 * A_{i,j} - 1$ such that $y_{i,j} \in \{-1, 1\}$, (18) can be equivalently expressed as

$$\boldsymbol{\theta}_{\mathcal{S}}^* = \arg \min_{\boldsymbol{\theta} \in \mathcal{S}^K} \sum_{i,j \in \mathcal{S}} \max(0, \epsilon - y_{i,j} \mathbf{x}_{i,j}^T \boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \tag{21}$$

where $\lambda \geq 0$ is the regularization parameter of the ℓ_2 regularization typically used to improve the generalization of SVMs. To solve our variant of simplex-constrained SVM's (cf. (21)), we employ the projected-gradient descent [3] approach where projection onto $\mathcal{S}^K$ is performed with $O(K \log K)$ complexity as described in [14]. The overall parameter learning procedure for a given sample is summarized in Algorithm 3.

In general, if the runtime or computational budget allows, the sampling and training process described in the last two sections can be repeated for T_s times to obtain different $\boldsymbol{\theta}_{\mathcal{S}}^*$'s, which can then be averaged in order to reduce their variance. In practice, this may not be necessary if N_s is large enough, which will yield a near-deterministic $\boldsymbol{\theta}$. The overall proposed adaptive-similarity embedding (ASE) framework is summarized in Algorithm 1.

Complexity. The computational complexity of ASE is dominated by the cost of performing the truncated SVD of $\mathbf{S}$ in the training as well as testing phaze of Algorithm 1. Relying on the sparsity ($|\mathcal{E}| \ll N^2$) and symmetricity of S, the Lanczos algorithm followed by EVD of a tridiagonal matrix yield the truncated SVD in a very efficient manner. Provided that $d \ll N$, the decomposition can be achieved in $O(|\mathcal{E}|d)$ time and using $O(Nd)$ memory. Therefore, for the $T_s \geq 1$ training rounds and single embedding round in Algorithm 1, the total complexity is $O((T_s + 1)|\mathcal{E}|d)$.

4 EXPERIMENTAL EVALUATION

The present section reports extensive experimental results on a variety of real-world networks [3]. The aim of the experimentation was twofold. First, to determine and quantify the quality of the proposed ASE embeddings for different downstream learning tasks. Second, to analyze and interpret the resulting embedding parameters for different networks.

[3]https://snap.stanford.edu/data/index.html

Table 1: Network Characteristics

Graph	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{Y}\rvert$	Density
PPI (H. Sapiens)	3,890	76,584	50	10^{-2}
Wikipedia	4,733	184,182	40	1.6×10^{-2}
BlogCatalog	10,312	333,983	39	6.2×10^{-3}
ca-CondMat	23,133	93,497	-	3.5×10^{-4}
email-Enron	36,692	183,831	-	2.7×10^{-4}
CoCit	44,312	195,362	15	2×10^{-4}
com-Amazon	334,863	925,872	-	1.7×10^{-5}

Methods. Experiments were run using the following *unsupervised* and *scalable* embedding methods: **a) ASE**. Our proposed adaptive similarity embedding. Based on observations made in Sections 3, and to retain optimization stability, we set the maximum number of steps to $K = 10$. We also use the default SVM regularizer $\lambda = 1$, and sampling $N_s/2 = 1000$ allowed for a single learning round $T_s = 1$ since parameters are learned with small enough variance. We made our implementation of ASE freely available [4]. **b) VERSE** [29]. This is a scalable framework for generating node embeddings according to a similarity function by minimizing s KL-divergence-objective via stochastic optimization. We used the default version with similarity (PPR with $\alpha = 0.85$), as implemented by the code [5] provided by the authors. **c) Deepwalk** [32]. This approach learns an embedding by sampling random walks from each node, applying word2vec-based learning on those walks. We use the default parameters described in the paper, i.e., walk length $t = 80$, number of walks per node $\gamma = 80$, and window size $w = 10$, and the scalable C++ implementation [6] provided in [29]. **d) HOPE** [22]. This SVD-based approach approximates high-order proximities and leverages directed edges. We report the results obtained with the default parameters, i.e, Katz similarity as the similarity measure with β inversely proportional to the spectral radius. **e) LINE** [28]. This approach learns a $d-$dimensional embedding in two steps, both using adjacency similarity. First, it learns $d/2$ dimensions using first-order proximity; then, it learns another $d/2$ features using second-order proximity. Last, the two halves are normalized and concatenated. We obtained a copy of the code [7] and run experiments with total $T = 10^{10}$ (although $T = 10^9$ yielded the same accuracy for smaller graphs) samples and $s = 5$ negative samples, as described in the paper. **f) Spectral**. This the first d eigenvectors of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. This baseline was developed for clustering [11], and has also been run as a benchmark for node embeddings [33]. In our case, spectral embedding is of particular interest since it can be obtained by column-wise normalization of the embeddings generated by the proposed method. We excluded comparisons with Node2vec [33] and AROPE [8] because they use cross-validation for hyper-parameter

[4]https://github.com/DimBer/ASE-project
[5]https://github.com/xgfs/verse
[6]https://github.com/xgfs/deepwalk-c
[7]https://github.com/tangjianpku/LINE

Table 2: Inferred parameters and interpretation

Graph	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	range	strength
PPI (H. Sapiens)	0.00	**0.14**	**0.31**	**0.29**	**0.21**	**0.04**	0.00	0.00	0.00	0.00	medium	medium
Wikipedia	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.01**	**0.37**	**0.62**	long	strong
BlogCatalog	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	very strong
ca-CondMat	**0.55**	**0.33**	**0.12**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
email-Enron	**0.24**	**0.25**	**0.18**	**0.14**	**0.1**	**0.06**	**0.02**	0.00	0.00	0.00	medium	weak
CoCit	**0.61**	**0.33**	**0.06**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	short	strong
com-Amazon	**0.10**	**0.10**	**0.10**	**0.10**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	short	very weak

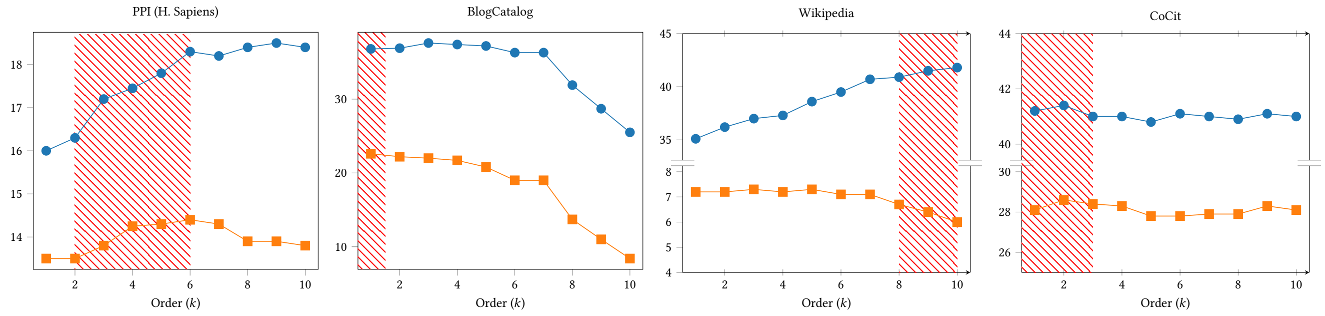

Figure 1: Micro (blue) and Macro (orange) F_1 scores for the four labeled graphs, when the "pure" k−order S^k is used for embedding, given as a function of k. Red shade denotes the k's where ASE assigned non-zero θ_k's; see also Table 2.

selection. Thus comparing Node2vec and AROPE to methods such as LINE, Deepwalk, HOPE, VERSE, and EMB that all operate with *fixed* hyperparameters in a fully *unsupervised* manner would be unfair. We also excluded comparisons with GraRep [24] and M-NMF [23] due to their limited scalability ($O(N^2d)$ computational and $O(N^2)$ memory complexity). Our experiment setting follows the one in [29]. All methods are set to embed nodes to dimension $d = 100$. Using the resulting embeddings as feature vectors, we evaluated their performance in terms of node classification and link prediction accuracy, and clustering quality. All experiments were repeated 10 times and reported are the averaged results.

Interpretation of results. One interesting aspect of the proposed ASE method, is that the inferred parameters $\boldsymbol{\theta}^*$ from the first phase of Algorithm 1 can be used to characterise the underlying similarity structure of the graph, and the way that nodes "interact" over different path lengths (short, medium and long range). The "strength" of interactions is inferred by how uniform the coefficients of $\boldsymbol{\theta}^*$ are and depend on the value of λ. Since the default value was $\lambda = 1$ for all graphs, the results can be interepreted as relative interaction strengths between them. The resulting $\boldsymbol{\theta}^*$'s for all graphs are collected in Table 2. It can immediatly be observed that the type of node interactions varies significantly among different graphs, with similar behavior for graphs that belong to the same domain. Specifically, ca-CondMat, and CoCit that belong to the citation/co-authorship domain all show relatively strong interactions of short range. BlogCatalog shows very strong short-range similarities of only one-hop neighborhood interactions among bloggers. On the other hand,the Wikipedia word cooccurrence network shows a strong tendency for long-range interactions; other graphs, such as the PPI protein interaction network stay on the medium range.

Node classification. Graphs with labeled nodes are frequently used to measure the ability of embedding methods to produce features suitable for classification. For each experiment, nodes were randomly split to a training set and a test set. Similar to other works, and to cope with multi-label targets, we fed the training features and labels into the one-vs-the-rest configuration of logistic regression classifier provided by the sklearn Python library. In the testing phase, we sorted the predicted class probabilities for each node in decreasing order, and extracted the top-k_i ranking labels, were k_i is the true number of labels of node v_i. We then computed the Micro- and Macro-averaged F_1 scores of the predicted labels. Apart from comparisons to alternative embedding methods, node classification can reveal whether available node labels (metadata) are distributed in a manner that matches the node relations – interactions that are inferred by ASE. To reveal this information, we obtain embeddings for every length $k \in [1, 10]$ by ignoring the training phase and "forcing" $\boldsymbol{\theta}^* = \mathbf{e}_k$ in Algorithm 1, and then using each embedding for classification with 10% labeling rate. Figure 1 plots Micro and Macro F_1 for all labeled graphs as a function of k, while red shade is placed on the lengths where the *unsupervised* ASE parameters $\boldsymbol{\theta}^*$ are non-zero (cf. Table 1). As seen in Fig. 1, the accuracy on the four labeled graphs evolves with k in a markedly different manner. Nevertheless, ASE identifies the trends and tends to assign non-zero weights to lengths that yield a good trade-off between Micro and Macro F_1. This is rather remarkable considering the fact that $\boldsymbol{\theta}^*$ depends only on the graph, since ASE does *not* use labels for training or validation.

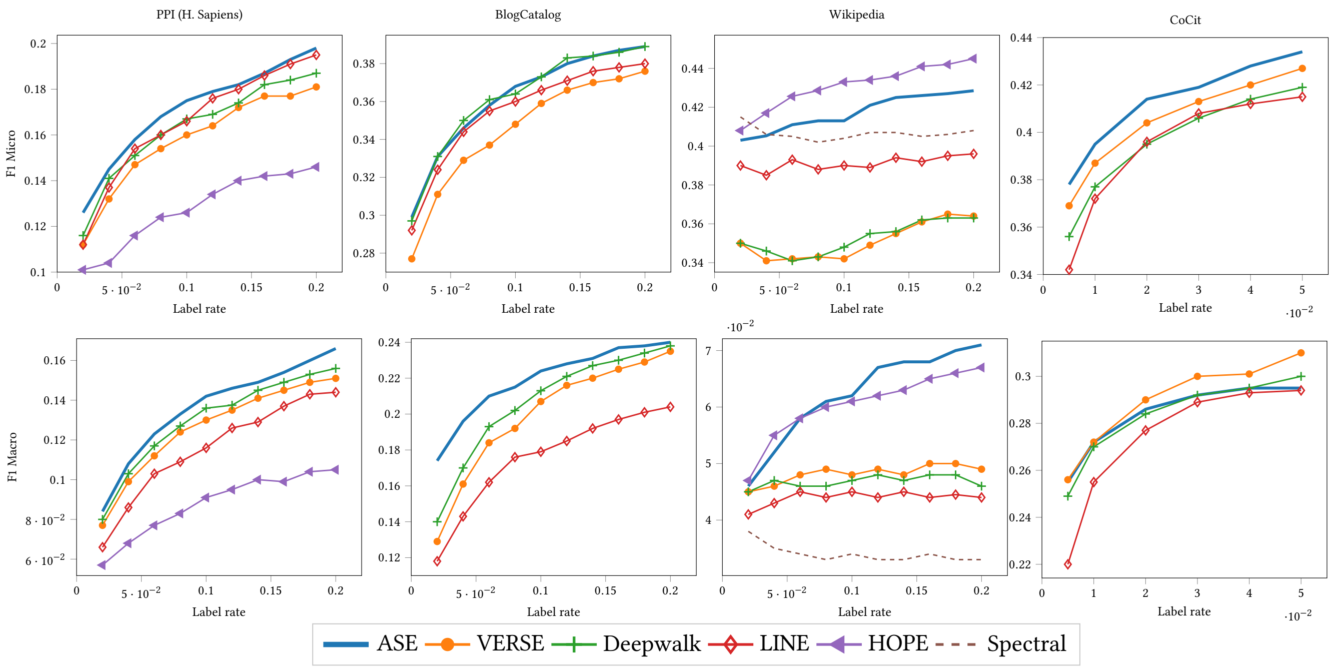

Figure 2: Micro (upper row) and Macro (lower row) F_1 scores that different embeddings + logistic regression yield on labeled graphs, as a function of the labeling rated (percentage of training data)

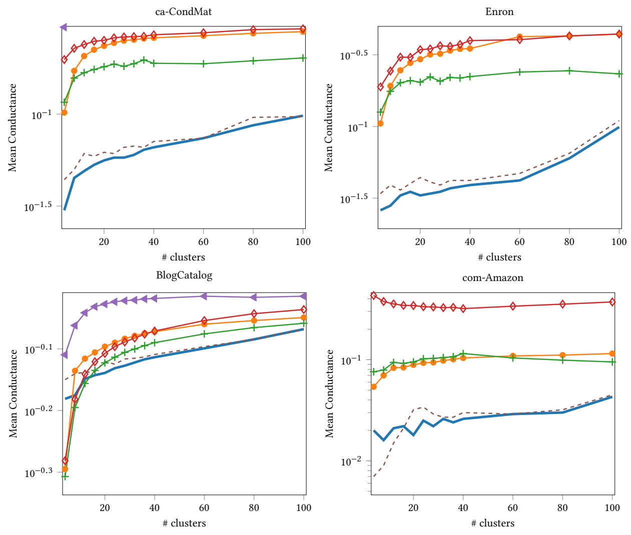

Figure 3: Average conductance of different embeddings used by kmeans for clustering, w.r.t number of clusters.

We also compared the classification accuracy of ASE embeddings with those of the alternative embedding approaches, with results plotted in Fig. 2. The plots for some method-graph pairs are not visible due to values being too low. While performance varies among graphs, ASE adapts to each graph and yields consistently reliable embeddings, with accuracy that in most cases reaches or surpasses that of state-of-the-art methods, especially in terms of Macro F_1. The two exceptions are the Macro F_1 in `CoCit`, and Micro F_1 in `Wikipedia`, where VERSE and HOPE being more accurate

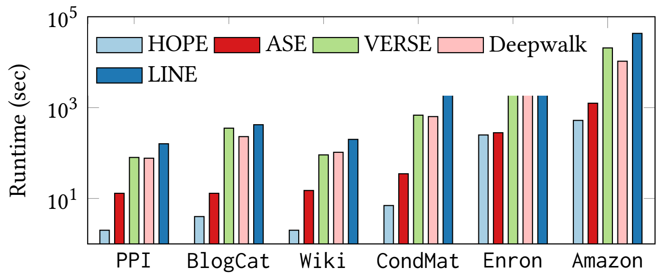

Figure 4: Runtime of various embedding methods across different graphs

respectively. Interestingly, HOPE achieving high Micro F_1 and low Macro F_1 in `Wikipedia` is in agreement with the findings in Fig. 1, combined with the fact that HOPE focuses on longer paths.

Node clustering. Finally, the embedded vectors were used to cluster the nodes into different communities, using the `sklearn` library K-means with the default K-means++ initialization [12]. We evaluate the quality of node clustering with conductance, a well-known metric for measuring the goodness of a community [5]; conductance is minimized for large, well connected communities that are also well separated from the rest of the graph. Each plot in Fig. 3 gives the average conductance across communities, as a function of the total number of clusters. Results indicate that the proposed ASE as well as the spectral clustering benchmark yield much lower conductance compared to other embeddings. Apparently, since ASE builds on the same basis of eigenvectors used by normalized spectral clustering, it inherits the property of the latter to approximately minimize the normalized-cut metric [11], which is very similar to conductance. A closer look at the resulting clusters, reveals that clustering beased on VERSE, Deepwalk, LINE, and HOPE splits graphs into very large communities of roughly equal size, cutting a large number of edges in the process. This is an indication that these methods are subject to a *resolution limit*, which is the inability to detect well-separated communities that are bellow a certain size [1]. On the other hand, Spectral (and the proposed ASE) separate the graph into a large-core component, and many smaller well-separated communities, a structure that many large-scale information networks have been observed to have [5]. Indeed, the conductance gap is smaller for `BlogCatalog` which is relatively small and with less pronounced communities.

Runtime. Finally, we compared different embedding methods in terms of runtime. Results for all graphs are reported in Fig. 4. All experiments were run on a personal workstation with a quad-core i5 processor, and 16 GB of RAM. For our proposed ASE, we provide a light-weight yet highly portable implementation that uses the SVDLIBC [41] library for sparse SVD. We also developed a more scalable implementation that relies on (and requires installation of) the SLEPc package [40]; this scalable version can perform large-scale sparse SVD on multiple processes and distributed memory environments using the message-passing interface (MPI). We used the high-performance implementation for the five larger graphs, and the portable-one for the five smallest ones. Evidently, ASE and HOPE that are SVD-based are orders of magnitudes faster than VERSE, Deepwalk, and LINE. The main factor that seems to slow the latter down seems to be the large number of stochastic-optimization iterations that these methods need to perform in order to reach accurate embeddings. Nevertheless, it should be noted that sampling based methods enjoy nearly-full parallelization and could thus benefit more from highly multi-threaded environments. On the other hand, methods that rely on SVD (and EVD) can benefit from decades of research on efficiently performing the decomposition, and a suite of stable and highly optimized software tools.

REFERENCES

[1] S. Fortunato, and M. Barthelemy, “Resolution limit in community detection,” *Proc. of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[2] Y. Zhao, E. Levina, and J. Zhu, “Consistency of community detection in networks under degree-corrected stochastic block models,” *The Annals of Statistics*, vol. 40, no. 4, pp. 2266–2292, 2012.

[3] D. P. Bertsekas, *Nonlinear programming*, Athena scientific Belmont, 1999

[4] L. A. Adamic, and E. Adar, “Friends and neighbors on the web,” In *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[5] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters,” *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[6] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in Neural Inf. Proc. Systems*, pp. 1024–1034, Long Beach, CA, 2017.

[7] Z. Yang, W. W. Cohen, and R. Salakhutdinov, “Revisiting semi-supervised learning with graph embeddings,” *arXiv preprint arXiv:1603.08861*, 2016.

[8] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, “Arbitrary-order proximity preserved network embedding,” in *Proc. of Intl Conf. on Knowledge Discovery and Data Mining*, pp. 2778–2786, London, UK, 2018.

[9] A. Milanese, J. Sun, and T. Nishikawa, “Approximating spectral impact of structural perturbations in large networks,” *Physical Review E*, vol. 81, no. 4, pp. 046–112, 2010.

[10] H. Cai, V. W. Zheng, and K. Chang, “A comprehensive survey of graph embedding: problems, techniques and applications,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[11] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[12] D. Arthur, and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” *SIAM*, pp. 1027–1035, 2007

[13] G. H. Golub, and C. Reinsch, “Singular value decomposition and least squares solutions,” *Numer. Math.*, vol. 14, no. 5, pp. 403–420, 1970.

[14] L. Condat, “Fast projection onto the simplex and the ℓ_1 ball,” In *Mathematical Programming*, vol. 158, no. 1-2, pp. 575–585, 2016.

[15] Y. Han, and Y. Shen, “Partially supervised graph embedding for positive unlabeled feature selection,” in *Proc. Intl Joint Conf. on Artificial Intelligence* , pp. 1548âĂŞ 1554, New York, NY, 2016.

[16] T. Hofmann, and J. M. Buhmann, “Multidimensional scaling and data clustering,” *Advances in Neural Inf. Proc. Systems*, 1994, pp. 459âĂŞ466.

[17] M. Balasubramanian, and E. L. Schwartz, “The isomap algorithm and topological stability,” *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[18] X. He, and P. Niyogi, “Locality preserving projections,” in *Advances in Neural Inf. Proc. Systems*, pp. 153âĂŞ160, 2003.

[19] S. T. Roweis, and L. K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[20] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, “Distributed large-scale natural graph factorization,” in *Proc. of the World Wide Web Conf.*, pp. 37âĂŞ48, Rio de Janeiro, Brazil, 2013.

[21] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, “Network representation learning with rich text information,” in *Proc. Intl Joint Conf. on Artificial Intelligence* , pp. 2111âĂŞ2117, 2015.

[22] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *Proc. of Intl Conf. on Knowledge Discovery and Data Mining*, pp. 1105âĂŞ1114, San Fransisco, CA, 2016.

[23] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, “Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec,” *Proc. of Intl Conf. on Web Search and Data Mining*, pp. 459–467, Los Angeles, CA, 2018.

[24] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” *Proc. of Intl on Conf. on Information and Knowledge Management*, pp. 891–900, 2015.

[25] B. Shaw, and T. Jebara, “Structure preserving embedding,” in *Proc. of Intl Conf. on Machine Learning*, pp. 937âĂŞ944, Montreal, Canada, 2009

[26] Y. Zhao, Z. Liu, and M. Sun, “Representation learning for measuring entity relatedness with rich information,” in *Proc. Intl Joint Conf. on Artificial Intelligence* , pp. 1412âĂŞ1418, Buenos aires, Argentina, 2015

[27] Y. Koren, R. M. Bell, and C. Volinsky. “Matrix factorization techniques for recommender systems,” In *IEEE Computer*, np. 8, pp. 30–37, 2009.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, âĂIJLine: Large-scale information network embedding,âĂİ in *Proc. of the World Wide Web Conf.*, pp. 1067âĂŞ1077, Florence, Italy, 2015.

[29] A. Tsitsulin, D. Mottin, P. Karras, and E. Muller, “VERSE: Versatile Graph Embeddings from Similarity Measures,” *Proc. of the World Wide Web Conf.*, pp. 539–548, Lyon, France, 2018.

[30] J. Tang, M. Qu, and Q. Mei, “Pte: Predictive text embedding through large-scale heterogeneous text networks,” in *Proc. of Intl Conf. on Knowledge Discovery and Data Mining*, pp. 1165âĂŞ1174, Sidney, Australia, 2015.

[31] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. Alemi, “Watch Your Step: Learning Graph Embeddings Through Attention,” *arXiv preprint arXiv:1710.09599*, 2017.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” *Proc. ACM SIGKDD Intl. Conf. on Knowl. Disc. and Data Mining*, New York, NY, 2014, pp. 701–710.

[33] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, 2016, pp. 855–864.

[34] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multirelational data,” in *Advances in Neural Inf. Proc. Systems*, pp. 2787âĂŞ2795, Lake Tahoe, CA, 2013.

[35] R. Xie, Z. Liu, and M. Sun, “Representation learning of knowledge graphs with hierarchical types,” in *Proc. Intl Joint Conf. on Artificial Intelligence* , pp. 2965âĂŞ 2971, New York, NY, 2016.

[36] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, “Spectral Graph Wavelets for Structural Role Similarity in Networks,” *arXiv preprint arXiv:1710.10321*, 2017.

[37] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, “struc2vec: Learning node representations from structural identity,” in *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Halifax, Canada, 2017, pp. 385–394.

[38] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 1225–1234, San Fransisco, CA, 2016.

[39] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in *Assoc. for the Advanc. of Artif. Intel.*, pp. 1145–1152, Phoenix, AZ, 2016.

[40] V. Hernandez, J. E. Roman, and V. Vidal, “SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems,”, in *ACM Trans. Math. Software*, vol. 31, no. 3, pp. 351–362, 2005.

[41] https://tedlab.mit.edu/~dr/SVDLIBC/