

Adaptive-similarity node embedding for scalable learning over graphs

Dimitris Berberidis, *Student Member, IEEE*, and Georgios B. Giannakis, *Fellow, IEEE*,

Abstract—Node embedding is the task of extracting informative and descriptive features over the nodes of a graph. The importance of node embeddings for graph analytics, as well as learning tasks such as node classification, link prediction and community detection, has led to increased interest on the problem leading to a number of recent advances. Still, node embedding is faced with a 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. The proposed model is carefully studied, interpreted, and numerically evaluated using stochastic block models. Moreover, an algorithmic scheme is proposed for training the model parameters efficiently and in an unsupervised manner. We perform extensive node classification, link prediction, 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.

Index Terms—SVD, SVM, unsupervised, multiscale, random walks, spectral

1 INTRODUCTION

Supervised node embedding is an exciting field, in which a significant amount of progress has been made in recent years [15]. The task consists of mapping each node of a graph to a vector in a low-dimensional Euclidean 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 [16]. 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 Euclidean 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 [22], [23], [24], [25], [26]. 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 [27], [28], [29], [30], [31], [32], [33], [34]. Efforts in this direction mostly focus on designing meaningful similarity metrics to factorize. While some methods (e.g. [27], [29]) maintain scalability by factorizing similarity matrices in an implicit manner (i.e., without explicitly forming them), others such as [30], [31] 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 [35], [37]. Such approaches are naturally scalable and entail a high degree of locality. Recently, stochastic edge-fitting has been generalized to implicitly accommodate long-range node similarities [36]. Meanwhile, other works have approached node embeddings using random-walk-based tools and concepts that originate in natural language processing [38], [39], [40]; see also related works on embedding of knowledge graphs [41], [42]. Methods that rely on graph convolutional neural networks and autoencoders have also been proposed for node embedding [45], [46], [47]. Moreover, a gamut of related embedding tasks are gaining traction, such as embedding based on structural roles of nodes [43], [44], supervised embeddings for classification [11], 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:

- The authors are with the Department of Electrical and Computer Engineering, and Digital Technology Center, University of Minnesota MN, 55455. E-mails: {bermp001,anikolakis,georgios}@umn.edu
- Work was supported by NSF 171141, 1514056 and 1500713.

Manuscript received XXX.

- **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. Specifically, graphs such as social networks, transportation networks, knowledge graphs and others, typically scale to millions of nodes and tens of millions of edges. Thus, strict computational constraints need to be incorporated into the design of node embedding methods.

Targeting at the three aforementioned challenges, we propose a scalable node embedding framework that is based on factorizing an adaptive node similarity matrix. The first challenge is addressed by utilizing a large family of node similarity metrics, parametrized by placing different weights on node proximities of different orders; see also our related work [20]. Experiments indicate that the proposed model for similarity metrics is expressive enough to describe real-world graph from diverse domains and with different structures. Targeting at the second challenge (lack of supervision), we put forth a self-supervised parameter learning scheme based on predicting randomly removed edges. Finally, scalability is achieved by constraining the parametrization of similarity matrices such that the proximity order parameters carry over to the embedded vectors in a smooth manner. This allows for learning proximity order parameters directly on the feature vectors. Consequently, dense similarity matrices do not need to be explicitly formed and factorized, ensuring the scalability of the proposed method.

The rest of the paper is organized as follows. Section 1 introduces the problem and the proposed similarity model. In Section 2, we present a numerical study on the properties of the model. In Section 3, we develop a procedure to learn the parameters of the model in an unsupervised manner. Finally, Sections 4 presents experiments on real graphs, comparisons with competing alternatives, and interpretation of the results.

Notation. Bold lower-case letters denote column vectors (e.g., \mathbf{v}); bold upper-case letters denote matrices (e.g., \mathbf{Q}). Vectors \mathbf{q}_j and \mathbf{q}_i^T denote the j^{th} column and the i^{th} row of \mathbf{Q} , respectively; whereas Q_{ij} (or sometimes for clarity $[\mathbf{Q}]_{ij}$) denotes the ij^{th} entry of \mathbf{Q} . Vector \mathbf{e}_K denotes the K^{th} canonical column vector; and $\|\cdot\|$ denotes the Euclidean norm, unless stated otherwise. Calligraphic upper-case letters denote sets (e.g., \mathcal{U}, \mathcal{V}); and finally, symbol $:=$ is used in definition statements.

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 Euclidian space; typically, the embedding is low-dimensional with d being much smaller than the number of nodes. Given $f(\cdot)$

we can obtain the low-dimensional vector representation of each node v_i as

$$\mathbf{e}_i = f(v_i).$$

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 a the following minimization problem wrt to the embedded vectors

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

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 Euclidian space. Thus, according to (1), node embedding can be viewed as the design of vectors $\{\mathbf{e}_i\}_{i \in \mathcal{V}}$ 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. $s_{\mathcal{G}}(v_i, v_j) = s_{\mathcal{G}}(v_j, v_i) \forall v_i, v_j \in \mathcal{V}$. 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 $[\mathbf{S}_{\mathcal{G}}]_{i,j} = [\mathbf{S}_{\mathcal{G}}]_{j,i} = s_{\mathcal{G}}(v_i, v_j)$, and matrix $\mathbf{E} = [\mathbf{e}_1 \dots \mathbf{e}_N]^T$ 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 $\mathbf{S}_{\mathcal{G}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where \mathbf{U} and \mathbf{V} are the $N \times N$ unitary matrices containing the left and right singular vectors, and $\mathbf{\Sigma}$ 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 [19] that $\mathbf{E}^* = \mathbf{U}_d \sqrt{\mathbf{\Sigma}_d}$, where $\mathbf{\Sigma}_d$ contains the d largest singular values and \mathbf{U}_d the corresponding singular vectors. Fortunately, \mathbf{U}_d and $\mathbf{\Sigma}_d$ 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}_G . Towards this directions, and to maintain expressibility, we will aim at designing a parametrized model for \mathbf{S}_G , where each pairwise node similarity is given as

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

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, $s_G(\cdot, \cdot; \boldsymbol{\theta})$ 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, $[\mathbf{S}^k]_{i,j} = s(v_i, v_j, k)$. For instance, selecting $\mathbf{S} = \mathbf{A}$ is equivalent to using the k -step similarity $s(v_i, v_j, k) = |\{k\text{-length paths connecting } v_i \text{ to } v_j\}|$ [12]. 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., [31]. 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

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

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 $\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$, and given that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, it readily follows that

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

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

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

Furthermore, the truncated singular pairs of $\mathbf{S}_G(\boldsymbol{\theta})$ conveniently follows from that of \mathbf{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 $\boldsymbol{\Sigma}_d(\boldsymbol{\theta}) = \sum_{k=1}^K \theta_k \boldsymbol{\Sigma}_d^k$ respectively. Thus, if $\mathbf{S} \in \text{Sym}_N$ the solution to (2) with \mathbf{S}_G parametrized by $\boldsymbol{\theta}$ is simply given as

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

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 $\boldsymbol{\Sigma}_d(\boldsymbol{\theta})$ cannot be guaranteed to be non-negative and sorted in decreasing order, which would cause $(\mathbf{U}_d(\boldsymbol{\theta}), \boldsymbol{\Sigma}_d(\boldsymbol{\theta}))$ to *not* be a valid SVD pair. Finally, having narrowed down \mathbf{S}_G to belong to the parametrized family in (5), we arrive at selecting an appropriate sparsity-preserving \mathbf{S} in order to obtain a solid model.

2.3 Spectral multi-length embeddings

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

$$\mathbf{S} \in \mathcal{P}_N^+ \iff \mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T, \quad (9)$$

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

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

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

Consider now that we specify \mathbf{S} to be

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

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 $\lambda_i(\mathbf{S}) \in [0, 1] \forall i$. More importantly, it can easily be verified that the first d eigenvectors of \mathbf{S} are the same as the eigenvectors that correspond to the d smallest eigenvalues of the symmetric normalized Laplacian matrix

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

The latter are known to contain useful information on cluster structures of different resolution levels, a key property that has been successfully used by spectral clustering [17]. 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 $\boldsymbol{\Lambda}_d(\boldsymbol{\theta})$. 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.

2.4 Relation to random walks

Apart from the spectral embedding interpretation discussed in the previous section, using powers of (11) to capture multi-length similarities also admits an interesting random walk

interpretation. We begin by expressing the k -th power of \mathbf{S} as

$$\begin{aligned}\mathbf{S}^k &= \frac{1}{2^k} \left(\mathbf{I} + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right)^k \\ &= \sum_{\tau=0}^k \alpha_\tau(k) \left(\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right)^\tau\end{aligned}\quad (13)$$

where the sequence

$$\alpha_\tau(k) = \begin{cases} \frac{1}{2^k} \binom{k}{\tau}, & 0 \leq \tau \leq k \\ 0, & \text{else} \end{cases}\quad (14)$$

can be interpreted as the non-zero weights that \mathbf{S}^k assigns to all paths of up to length k (see Fig. ??).

Using (13) and (14), the multi-length similarity in (5) becomes

$$\begin{aligned}\mathbf{S}_G(\boldsymbol{\theta}) &= \sum_{\tau=0}^K c_\tau(\boldsymbol{\theta}) \left(\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right)^\tau \\ &= \mathbf{D}^{-1/2} \left(\sum_{\tau=0}^K c_\tau(\boldsymbol{\theta}) \mathbf{P}^\tau \right) \mathbf{D}^{1/2},\end{aligned}\quad (15)$$

where

$$c_\tau(\boldsymbol{\theta}) = \sum_{k=1}^K \theta_k \alpha_\tau(k),\quad (16)$$

and $\mathbf{P} = \mathbf{A} \mathbf{D}^{-1}$ is the probability transition matrix of a simple random walk defined over \mathcal{G} ; that is, $P_{i,j}$ is the probability that a random walker positioned on node (state) j transitions to node i in one step. Thus, the k -length similarity function defined in (3) is expressed as

$$s_G(v_i, v_j, \boldsymbol{\theta}) = \sqrt{\frac{d_j}{d_i}} \sum_{\tau=0}^K c_\tau(\boldsymbol{\theta}) \Pr\{X_\tau = v_i | X_0 = v_j\},\quad (17)$$

where $\Pr\{X_\tau = v_i | X_0 = v_j\} := [\mathbf{P}^\tau]_{ij}$ is the probability that a random walk starting from v_j lands on v_i after τ steps.

Interestingly, $\mathbf{S}_G(\boldsymbol{\theta})$ does not weigh landing probabilities of different lengths independently. Instead, it accumulates the latter as weighted combinations (cf. (16)) in a basis of “wavelet”-type functions of different resolution (see Fig. 1).

With established links to spectral clustering and random walks, $\mathbf{S}_G(\boldsymbol{\theta})$ appears promising as a family of node similarity matrixes. Nevertheless, before devising an algorithm for learning $\boldsymbol{\theta}$ and testing it on real graphs, we evaluate how well the basis on which $\mathbf{S}_G(\boldsymbol{\theta})$ is built (i.e., $\{\mathbf{S}^k\}_{k=1}^K$) can capture underlying node similarities.

3 MODEL EXPRESSIVENESS

In the present section, a performance metric that quantifies how well a node similarity matrix that is derived from the graph itself matches the “true” underlying similarity structure between nodes is proposed. The discussion is followed by numerical evaluation of the performance of different similarity matrices (including the proposed in (13)) on graphs that are generated according to the stochastic block model [2].

Let us begin by assuming that for a given set of nodes, an adjacency matrix \mathbf{A} is generated as

$$\mathbf{A} \sim f_A(\mathbf{A})$$

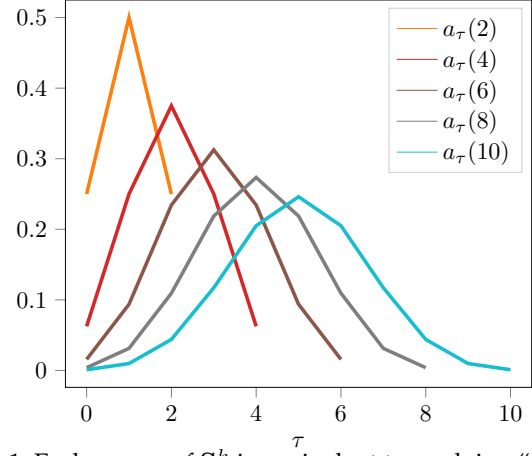


Fig. 1: Each power of \mathbf{S}^k is equivalent to applying “wavelet”-type weights $\alpha_\tau(k)$ over walks of length $\leq k$.

where $f_A(\mathbf{A})$ is a distribution defined over the space of all possible adjacency matrices. We define the “true” underlying similarity between nodes v_i and v_j to be

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

which is the probability that the two nodes are connected. The “true” similarity matrix is thus given as the expected adjacency matrix

$$\mathbf{S}^* := \mathbb{E}_{f_A} [\mathbf{A}].$$

We define the quality-of-match (QoM) between the underlying \mathbf{S}^* and any similarity $\hat{\mathbf{S}} = F(\mathbf{A})$ estimated from the adjacency matrix as

$$\text{QoM} := \mathbb{E}_{f_A} [\text{PC}(\mathbf{S}^*, F(\mathbf{A}))]\quad (18)$$

where

$$\text{PC}(\mathbf{X}_1, \mathbf{X}_2) := \frac{(\text{vec}(\mathbf{X}_1))^T \text{vec}(\mathbf{X}_2)}{\|\mathbf{X}_1\|_F \|\mathbf{X}_2\|_F},\quad (19)$$

is the Pearson correlation between two matrices \mathbf{X}_1 and \mathbf{X}_2 , with $\text{vec}(\mathbf{X})$ denoting matrix vectorization. The latter is used for appropriate rescaling of the “true” similarity matrix in order for the comparison with \mathbf{S}_G to be meaningful. Intuitively, (18) measures how well the estimated node similarities in $\hat{\mathbf{S}}$ are expected to match the pattern of true underlying similarities in \mathbf{S}^* , when edges are generated according to known $f_A(\cdot)$.

3.1 Numerical experiments and observations

We numerically evaluate the QoM achieved by different similarity matrices, on a set of N nodes whose interconnections are generated according to a stochastic block model (SBM). For this set of experiments, we divided the nodes into three clusters of equal size

$$\mathcal{C}_l = \{i : (l-1)N/3 \leq i \leq lN/3\}, \quad l \in \{1, 2, 3\}$$

with inter- and intra-connection probabilities given as

$$\Pr\{(i, j) \in \mathcal{E}\} = \begin{cases} p & , (i, j) \text{ in the same } \mathcal{C}_l \\ cq & , i \in \mathcal{C}_1 \text{ and } j \in \mathcal{C}_3 \\ q & \text{else} \end{cases}\quad (20)$$

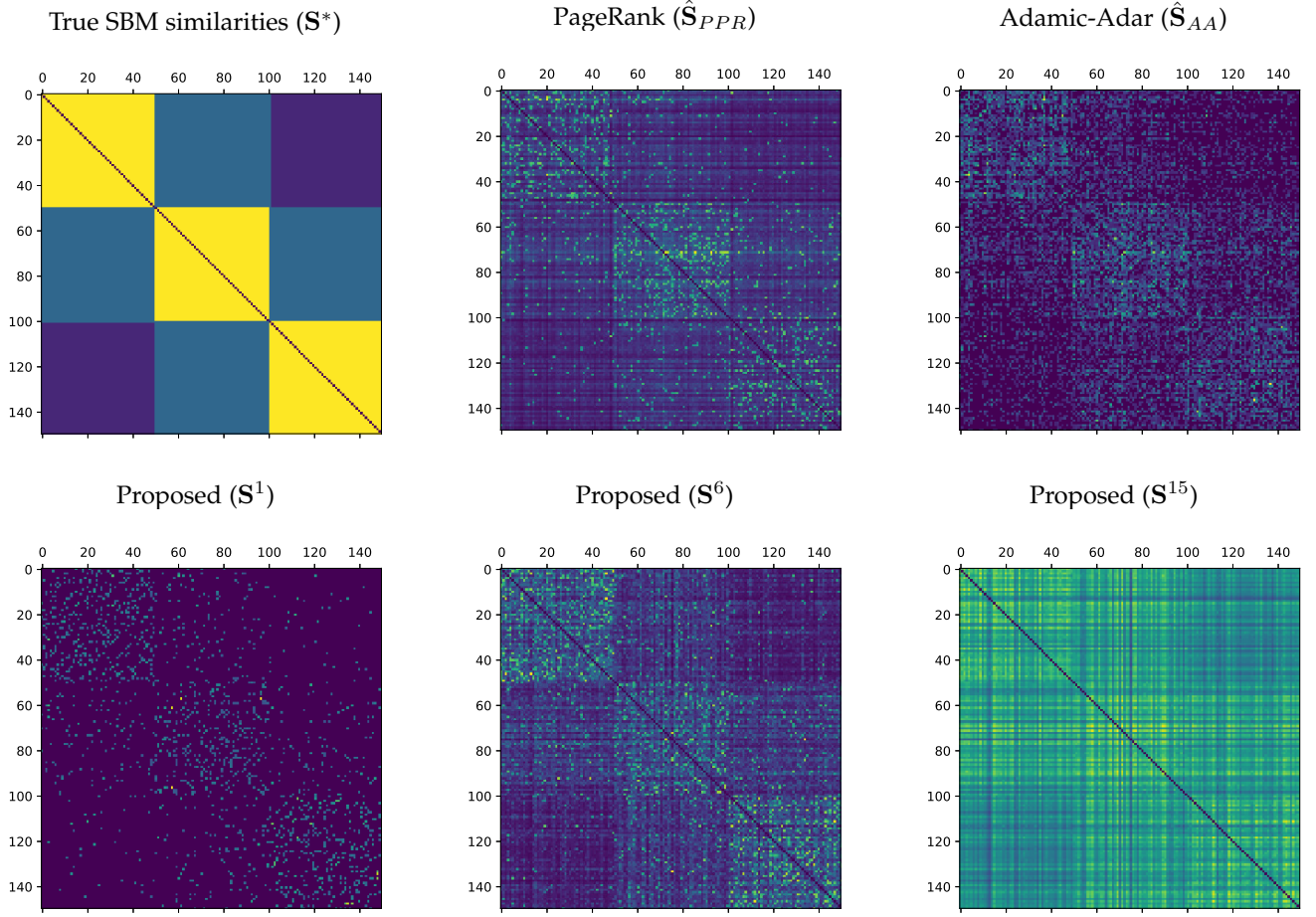


Fig. 2: Depiction of groundtruth and estimated similarity matrices, as yielded from an instance of the numerical experiments described in Section 3.1.

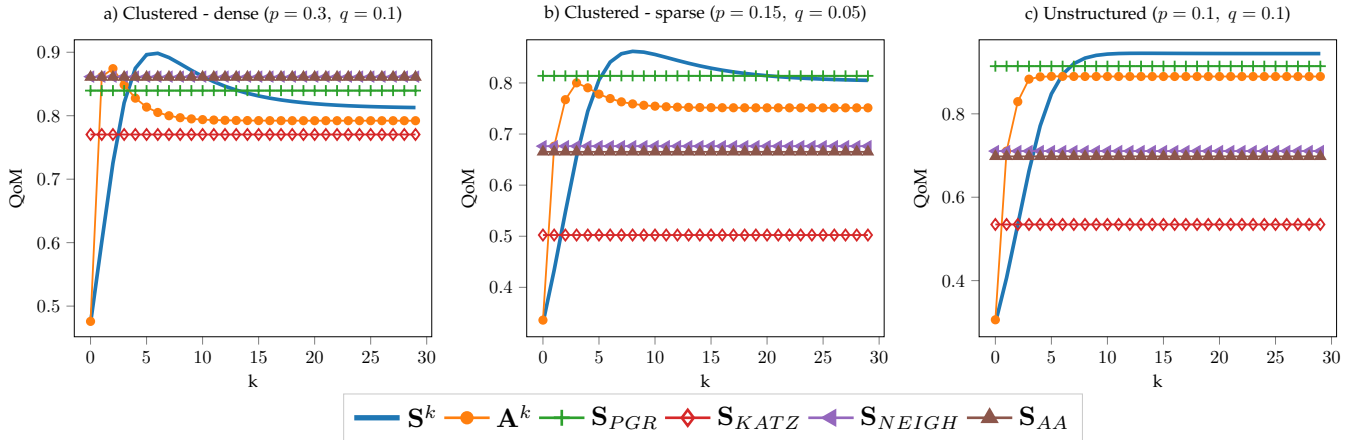


Fig. 3: Quality of match between true SBM similarity and various estimates, as yielded from experiments of Section 3.1.

where p is the probability of connection when two nodes belong to the same cluster, and $c < 1$ introduces asymmetry and a hierarchical clustering organization (see Fig. 2-top left), by making two of the clusters less likely to connect; we have related Python scripts available¹. The SBM probability matrix

[2] is given as

$$\mathbf{W}_{\text{sbm}} = \begin{bmatrix} p & q & cq \\ q & p & q \\ cq & q & p \end{bmatrix} \quad (21)$$

and the underlying similarity can be expressed as

$$\mathbf{S}^* = \mathbb{E}[\mathbf{A}] = \mathbf{W}_{\text{sbm}} \otimes (\mathbf{1}_{N/3} \mathbf{1}_{N/3}^T) - \text{diag}(p \mathbf{1}_N) \quad (22)$$

1. https://github.com/DimBer/ASE-project/tree/master/sim_tests

where \otimes denotes the Kronecker product.

For each experiment, se set $N = 150$ and generated a graph according to (20). We then compared the QoM between (22) and the k -th power of the proposed (11), the k -th power of the adjacency (\mathbf{A}^k), as well as each of the following well known similarity metrics

- $\hat{\mathbf{S}}_{PPR} := (\mathbf{I} - \alpha)(\mathbf{I} - \alpha\mathbf{A}\mathbf{D}^{-1})^{-1}$: the steady state probability that a random walk restarting at v_j with probability $1 - \alpha$ at every step is located at v_i . Essentially a personalized PageRank (PPR) computed for every node of the graph, inheriting the properties of the celebrated centrality measure [7], [8], [9].
- $\hat{\mathbf{S}}_{KATZ} := (\mathbf{I} - \beta)(\mathbf{I} - \beta\mathbf{A})^{-1}\mathbf{A}$: the Katz index [12], an exponentially weighted summation over paths of all possible lengths between two nodes.
- $\hat{\mathbf{S}}_{NEIGH} := \mathbf{A}^2$: the number of common neighbors that every pair of nodes shares.
- $\hat{\mathbf{S}}_{AA} := \mathbf{A}\mathbf{D}^{-1}\mathbf{A}$: Adamic-Adar [4] is a variant of common neighbors where each set of neighbors is weighed reciprocally its cardinality.

The resulting QoM was averaged over 200 experiments. Parameters α in $\hat{\mathbf{S}}_{PPR}$ and β in $\hat{\mathbf{S}}_{KATZ}$ were tuned to maximize the performance of the metrics. Figure 3 depicts QoM as a function of k , for three different scenarios.

In the first scenario (Fig. 3-a), with graphs being dense and clustered ($p = 0.3, q = 0.1$), the proposed \mathbf{S}^k improves sharply in the first few steps, reaching maximum QoM after 4 or 5 steps, and gradually decreases as k continues to increase. The k -th order proximities that are given as entries of \mathbf{A}^k follow a similar trend, however their QoM peaks shortly after 2 or 3 steps and declines fast for larger k . The matrix plots of a randomly selected experiment that are presented in Fig. 2 can aid in understanding the underlying mechanism that gives rise this highly step-dependent behavior. Specifically, \mathbf{S}^1 (bottom left) that has the same sparsity pattern as the adjacency is a poor match to the dense block-structure of \mathbf{S}^* . On the other side of the spectrum, \mathbf{S}^{15} (bottom right) is too “flat” and also a poor similarity metric. Meanwhile, taking $k = 6$ promotes enough mixing without “dissipating”, thus \mathbf{S}^6 (bottom center) visibly matches the structure of \mathbf{S}^* . Interestingly, for $k \in [4, 10]$ the proposed \mathbf{S}^k surpasses in QoM all other similarity metrics that were tested. Nevertheless, the simple 2-hop Adamic-adar, common-neighbors similarities perform reasonably well by exploiting the relatively dense structure of the graphs.

Results were markedly different in the second scenario; see, Fig. 3-b. Here, graphs were generated with the same clustering structure but significantly sparser, with edge probability parameters $p = 0.15$ and $q = 0.05$. For sparser graphs, \mathbf{A}^k and \mathbf{S}^k require more steps to reach peak QoM (4 and 9 respectively). Similarly, PPR which relies on long paths performs much better than the short-reaching Adamic-Adar. This behavior is intuitively pleasing; the sparser a graph is, the longer become the paths that need to be explored around each node, in order for the latter to “gauge” its position on the graph.

Finally, a third scenario (Fig. 3-c) was examined, where each graph was generated without a clustering structure ($p = q = 0.1$ and $c = 1$); essentially an Erdos-Renyi graph. For this degenerate case that is of no real practical interest,

all pairs of nodes are equally similar; this type of similarity requires infinite long paths to be described.

To conclude, the presented numerical study hints at the two following facts. First, \mathbf{S}^k can successfully model similarities that are based on the grouping of nodes in arbitrary and multilevel sets with varying degrees of homophily and heterophily. The second fact, is that the performance of \mathbf{S}^k varies significantly with k . Moreover, the way that k affects performance may also vary between graphs, depending on their properties; a type of graph “signature” that is also validated in real graphs in Section 6. Thus, a principled way to build $\mathbf{S}_G(\theta)$ by learning the parameters that match the this graph “signature” in an unsupervised way, is highly motivated.

4 UNSUPERVISED SIMILARITY LEARNING

We have arrived at the point where, for a given graph, we must select a specific $\theta \in \mathcal{S}^K$ without supervision. Following the discussion in Section 3, it would be ideal to fit $\mathbf{S}_G(\theta)$ to a true \mathbf{S}^* by minimizing an expected cost

$$\theta^* = \arg \min_{\theta \in \mathcal{S}^K} \mathbb{E}_{f_A} [\ell(\mathbf{S}^*, \mathbf{S}_G(\mathbf{A}; \theta))] \quad (23)$$

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

$$\min_{\theta \in \mathcal{S}^K} \ell(\mathbf{A}, \mathbf{S}_G(\mathbf{A}; \theta)). \quad (24)$$

While straightforward, (24) yields embeddings with limited generalization capability. Simply put, regardless of the choice of $\ell(\cdot)$, solving (24) 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 following 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 $\mathbf{S}^* \approx \mathbf{A}_1$, and the other to form the multilength similarity matrix $\mathbf{S}_G(\mathbf{A}_2; \theta)$; parameters θ can then be learned by solving

$$\min_{\theta \in \mathcal{S}^K} \ell(\mathbf{A}_1, \mathbf{S}_G(\mathbf{A}_2; \theta)). \quad (25)$$

Since separate samples are not available, we approximate the above process by randomly extracting part of \mathbf{A} and approaching (25) as

$$\min_{\theta \in \mathcal{S}^K} \ell_S(\mathbf{A}, \mathbf{S}_G(\mathbf{A} * \mathbf{S}^c; \theta)), \quad (26)$$

where $\mathcal{S} \in \{1, \dots, N\}^2$ is a subset of all possible pairs of nodes with $|\mathcal{S}| = N_s$, and \mathbf{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, $\ell_S(\cdot, \cdot)$ in (26) 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 $\mathcal{S}^- \in \{1, \dots, N\}^2 \setminus \mathcal{E}$ 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 (26) to an applicable method, it remains to specify two modular sub-systems: one responsible for sampling edges, and one that specifies $\ell(\cdot, \cdot)$ and finds θ^* by solving (26).

4.1 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

$$\begin{aligned} \Pr\{e_0 = (i, j)\} &= \Pr\{v_1 = i, v_2 = j\} + \Pr\{v_1 = j, v_2 = i\} \\ &= \Pr\{v_2 = i | v_1 = j\} \Pr\{v_1 = j\} \\ &\quad + \Pr\{v_2 = j | v_1 = i\} \Pr\{v_1 = i\} \\ &= \frac{1}{d_j} \frac{1}{N} + \frac{1}{d_i} \frac{1}{N} \propto \frac{d_i + d_j}{d_i d_j}, \end{aligned} \quad (27)$$

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 [14], 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 tends to lead 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 (27) 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.

4.2 Parameter training

Subsequently, for a given sample \mathcal{S} , we can obtain the corresponding optimal parameters as (cf. (26))

$$\theta_S^* = \arg \min_{\theta \in \mathcal{S}^K} \sum_{i,j \in \mathcal{S}} \ell(A_{i,j}, s_{\mathcal{G}^-}(u_i, v_j; \theta)) \quad (28)$$

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

Interestingly, one way that (28) could be solved is by explicitly computing the entries of $\mathbf{S}_{\mathcal{G}}(\theta)$ that are in \mathcal{S} . This would require performing K sparse matrix-vector products to obtain every column of \mathbf{S}^k for $k \in [1, K]$, for all the columns that contain sampled entries. At the worst case, if all nodes in the tuples in \mathcal{S} correspond to different columns of $\mathbf{S}_{\mathcal{G}}(\theta)$, two random walk are required for every tuple, a total of $2 * N_s$ random walks. This requires $\mathcal{O}(N_s K |\mathcal{E}|)$ computations, and $\mathcal{O}(N_s N)$ memory if they are to be performed concurrently or in matrix form. Since K will typically be in the order of tens, these requirements may be affordable if N_s is relatively small. Nevertheless, they quickly become cumbersome for $N_s \gg K$ which may be necessary to estimate the K -dimensional θ .

Algorithm 1 ADAPTIVE SIMILARITY EMBEDDING

Input: \mathcal{G} **Output:** \mathbf{E}
 // Training phase
 $\Theta = \emptyset$
while $|\Theta| < T_s$ **do**
 $\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^- = \text{SAMPLE EDGES}(\mathcal{G})$
 $\theta_S^* = \text{TRAIN PARAMETERS}(\mathcal{G}^-, \mathcal{S}^+, \mathcal{S}^-)$
 $\Theta = \Theta \cup \theta_S^*$
end while
 $\theta^* = T_s^{-1} \sum_{\theta \in \Theta} \theta$
 // Embedding phase
 $\mathbf{S} = \frac{1}{2} (\mathbf{I} + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2})$
 $\mathbf{S} = \mathbf{U}_d \Sigma_d \mathbf{U}_d^T$
 $\Sigma_d(\theta^*) = \sum_{k=1}^K \theta_k^* \Sigma_d^k$
return $\mathbf{E} = \mathbf{U}_d \sqrt{\Sigma_d(\theta^*)}$

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 $v_1 \sim \text{Unif}(\mathcal{V})$
 if $|\mathcal{N}_{\mathcal{G}^-}(v_1)| > 1$ **then**
 Sample $v_2 \sim \text{Unif}(\mathcal{N}_{\mathcal{G}^-}(v_1))$
 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 $(v_1, v_2) \sim \text{Unif}(\mathcal{V} \times \mathcal{V})$
 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:** θ_S^*
 $\mathbf{S} = \frac{1}{2} (\mathbf{I} + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2})$
 $\mathbf{S} = \mathbf{U}_d \Sigma_d \mathbf{U}_d^T$
 $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$
Form $\mathcal{X}_S = \{\mathbf{x}_{(i,j)}\}_{(i,j) \in \mathcal{S}}$ as in (30)
return $\theta_S^* = \text{SIMPLEXSVM}(\mathcal{X}_S, \mathcal{S}^+, \mathcal{S}^-)$

Algorithm 4 SIMPLEXSVM

Input: $\mathcal{X}, \mathcal{S}^+, \mathcal{S}^-$ **Output:** θ^*
 $\theta_0 = \frac{1}{K} \mathbf{1}, t = 1$
while $\|\theta_t - \theta_{t-1}\|_\infty \geq \text{tol}$ **do**
 $t = t + 1, \eta_t = a/\sqrt{t}$
 $\mathcal{S}_a^+ = \{e \in \mathcal{S}^+ \mid \mathbf{x}_e^T \theta_{t-1} \leq \epsilon\}$
 $\mathcal{S}_a^- = \{e \in \mathcal{S}^- \mid \mathbf{x}_e^T \theta_{t-1} \geq -\epsilon\}$
 $\mathbf{g}_t = \sum_{e \in \mathcal{S}_a^+} \mathbf{x}_e - \sum_{e \in \mathcal{S}_a^-} \mathbf{x}_e$
 $\mathbf{z}_t = (1 - 2\eta_t \lambda) \theta_{t-1} - \frac{\eta_t}{N_s} \mathbf{g}_t$
 $\theta_t = \text{SIMPLEXPROJ}(\mathbf{z}_t)$
end while
return θ_t

Instead, we will rely on the fact that the proposed embeddings are smooth and differentiable wrt to θ (cf. (10)), to develop a solution that allows for selecting arbitrarily large N_s , using the approximation

$$\begin{aligned}
s_{\mathcal{G}^-}(u_i, v_j; \theta) &\approx s_{\mathcal{E}^-}(\mathbf{e}_i^-(\theta), \mathbf{e}_j^-(\theta)) \\
&= (\mathbf{e}_i^-(\theta))^T \mathbf{e}_j^-(\theta) \\
&= \left(\sqrt{\Sigma_d^-(\theta)} \mathbf{u}_i^- \right)^T \sqrt{\Sigma_d^-(\theta)} \mathbf{u}_j^- \\
&= (\mathbf{u}_i^-)^T \Sigma_d^-(\theta) \mathbf{u}_j^- \\
&= \mathbf{x}_{i,j}^T \theta
\end{aligned} \tag{29}$$

where

$$\mathbf{x}_{i,j} = (\mathbf{u}_i^- * \mathbf{u}_j^-)^T \Sigma_d^K, \tag{30}$$

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 θ to produce similarities; this allows us to approach (28) using well-understood learning and optimization tools. For instance, let us define $\ell(\cdot)$ to be the Hinge loss

$$\ell(y, f) := \max(0, \epsilon - yf), \tag{31}$$

and, upon defining targets $y_{i,j} = 2 * A_{i,j} - 1$ such that $y_{i,j} \in \{-1, 1\}$, (28) can be equivalently expressed as

$$\theta_S^* = \arg \min_{\theta \in \mathcal{S}^K} \sum_{i,j \in \mathcal{S}} \max(0, \epsilon - y_{i,j} \mathbf{x}_{i,j}^T \theta) + \lambda \|\theta\|_2^2 \tag{32}$$

where $\lambda \geq 0$ is the regularization parameter of the ℓ_2 regularization typically used to improve the generalization of SVMs [13]. To solve our variant of simplex-constrained SVM's (cf. (32)), we employ the projected-gradient descent [3] approach that we describe in Algorithm 4, where $\text{SIMPLEXPROJ}(\cdot)$ is a subroutine that implements projections onto \mathcal{S}^K ; the latter can be performed with $\mathcal{O}(K \log K)$ complexity as described in [21]. The overall parameter learning procedure for a given sample is summarized in Algorithm 3.

In general, if the runtime or computational allows, the sampling and training process described in the last two sections can be repeated for T_s times to obtain different θ_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 θ . The overall proposed adaptive-similarity embedding (ASE) framework is summarized in Algorithm 1.

4.3 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 phase of Algorithm 1. Relying on the sparsity ($|\mathcal{E}| \ll N^2$) and symmetricity of \mathbf{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 $\mathcal{O}(|\mathcal{E}|d)$ time and using $\mathcal{O}(Nd)$ memory. Therefore, for the $T_s \geq 1$ training rounds and single embedding round in Algorithm 1, the total complexity is $\mathcal{O}((T_s + 1)|\mathcal{E}|d)$.

5 RELATED WORK

Two recent embedding methods also entertain the notion of similarity matrices that combine walks of different lengths [12], [38]. Most relevant to the proposed ASE is the ‘‘Arbitrary-Order Proximity Preserved Network Embedding’’ [12] framework, where a method is proposed for obtaining the SVD of a polynomial of the adjacency matrix without having to recompute the singular vectors. Compared to [12], we put forth the following contributions. First, we introduce a family of multi-length similarities whose decomposition leads to embeddings that inherit the rich information contained in the spectral embeddings (cf. Section 2.3). An equally important contribution in terms of modeling is the fact that the embeddings produced by ASE can be differentiated with respect to weights θ (cf. (29)-(32)), whereas the resulting embeddings of [12] are non-differentiable wrt to the weights. Hence, [12] can only produce in a ‘‘forward’’ fashion given some order proximity weights θ ; whereas, our approach allows for ‘‘navigating’’ the space of possible similarity functions $s(v_i, v_j; \theta)$ in a smooth fashion, meaning that θ can be learned with simple optimization on well defined fitting models such as logistic regression or SVMs (cf. (32)). This leads to the third main contribution which is a way to learn ‘‘personalize’’ θ (cf. Section 4) in an unsupervised fashion, i.e., without downstream information such as node or edge labels/attributes guiding crossvalidation in high-dimensional discretized parameter grids. The second related embedding method presented in [38] builds on the concept of graph attention mechanisms to place weights on lengths

of truncated random walks that are used to build a similarity matrix containing co-occurrence probabilities; the latter is jointly decomposed by maximizing a graph-likelihood function. The model in [38] is a generalization of the ones implicitly used in [39] and [40], building on similar tools and concepts that emerge from natural language processing. Nevertheless, unlike [39], [40] and the proposed ASE, [38] explicitly constructs and factorizes a dense $N \times N$ similarity matrix; the exact process entails complexity that is *cubic* wrt to N , and becomes at best *quadratic* with model approximations, meaning that [38] scales poorly beyond small graphs.

6 EXPERIMENTAL EVALUATION

The present section reports extensive experimental results on a variety of real-world networks. 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.

Datasets. For our experiments, we used the following real-world networks (see also Table 1).

- **ca-AstroPh.** The Astro Physics [51] collaboration network is from the e-print arXiv and covers scientific collaborations between authors papers submitted to Astro Physics category. If an author i co-authored a paper with author j , the graph contains a undirected edge from i to j . If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes.
- **ca-CondMat.** Condense Matter Physics collaboration network from ArXiv [51].
- **CoCit.** A co-citation network of papers citing other papers extracted by [36]; labels represent conferences in which papers were published.
- **com-DBLP.** Computer science research bibliography collaboration network [51].
- **com-Amazon.** Network collected by crawling Amazon website [51]. It is based on “Customers Who Bought This Item Also Bought” feature of the Amazon website. If a product i is frequently co-purchased with product j , the graph contains an undirected edge from i to j .
- **vk2016-17.** VK is a Russian all-encompassing social network. In [36], two snapshots of the network were extracted in November 2016 and May 2017, to obtain information about link appearance.
- **email-Enron.** Enron email communication network covering all the email communication within a dataset of around half million emails [51].
- **PPI (H. Sapiens).** Subgraph of the protein-protein interaction network for Homo Sapiens. The subgraph corresponds to the graph induced by nodes for which labels (represent biological states) were obtained from the hallmark gene sets [40].
- **Wikipedia.** This is a cooccurrence network of words appearing in the first million bytes of the Wikipedia dump. The labels represent the Part-of-Speech (POS) tags inferred using the Stanford POS-Tagger [40].

TABLE 1: Network Characteristics

Graph	$ \mathcal{V} $	$ \mathcal{E} $	$ \mathcal{Y} $	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}
ca-AstroPh	18,772	198,110	-	1.1×10^{-3}
email-Enron	36,692	183,831	-	2.7×10^{-4}
CoCit	44,312	195,362	15	2×10^{-4}
vk2016-17	78,593	2,680,542	-	8.7×10^{-4}
com-Amazon	334,863	925,872	-	1.7×10^{-5}
com-DBLP	317,080	1,049,866	-	2.1×10^{-5}

- **BlogCatalog.** A network of social relationships of the bloggers listed on the BlogCatalog website. The labels represent blogger interests inferred through the meta- data provided by the bloggers.

Methods. Experiments were run using the following *unsupervised* and *scalable* embedding methods.

- **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 ².
- **VERSE** [36]. 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 ³ provided by the authors.
- **Deepwalk** [39]. 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 ⁴ provided in [36].
- **HOPE** [29]. 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.
- **LINE** [35]. 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 ⁵ 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.

2. <https://github.com/DimBer/ASE-project>

3. <https://github.com/xgfs/verse>

4. <https://github.com/xgfs/deepwalk-c>

5. <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
ca-AstroPh	0.76	0.24	0.00	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
vk2016-17	0.71	0.29	0.00	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
com-DBLP	0.11	0.10	0.10	0.09	0.09	0.09	0.09	0.09	0.09	0.08	short	very weak

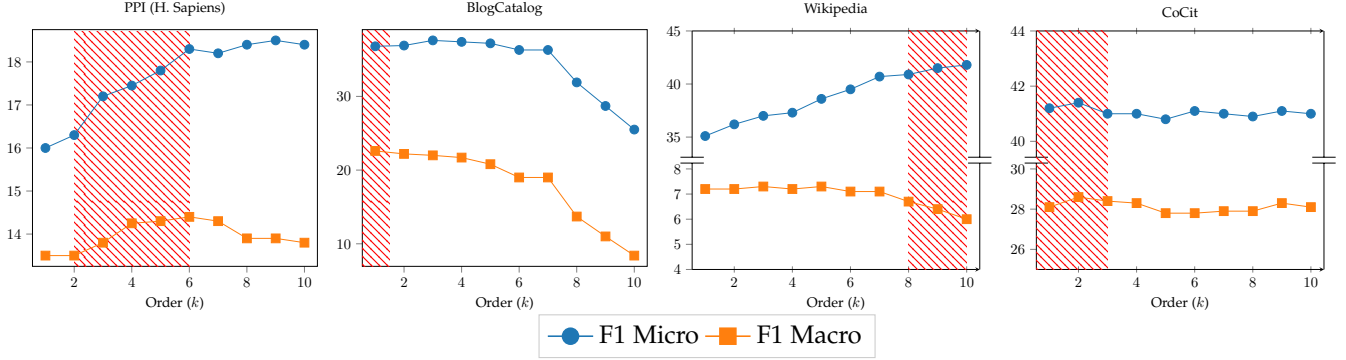


Fig. 4: Micro and Macro F_1 scores for the four labeled graphs, when the “pure” k -order \mathbf{S}^k is used for embedding, given as a function of k . Red shade denotes the corresponding k ’s where ASE assigned non-zero θ_k ’s; see also Table 2.

- **Spectral.** This the first d eigenvectors of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. This baseline was developed for clustering [17], and has also been run as a benchmark for node embeddings [40]. 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 [40] and AROPE [12] because they use cross-validation for hyperparameter 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 [31] and M-NMF [30] due to their limited scalability ($\mathcal{O}(N^2d)$ computational and $\mathcal{O}(N^2)$ memory complexity).

Evaluation methodology. Our experiment setting follows the one in [36]. 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 θ^* 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 θ^* are and depend on the value of λ . Since the default value was $\lambda = 1$ for all graphs, the results can be interpreted as relative interaction strengths between them. The resulting θ^* ’s for all graphs are collected in Table 2.

It can immediately 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*, *ca-AstroPh*, 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; while 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, where k_i is the true number of labels of node v_i . We then computed the Micro- and Macro-averaged F_1

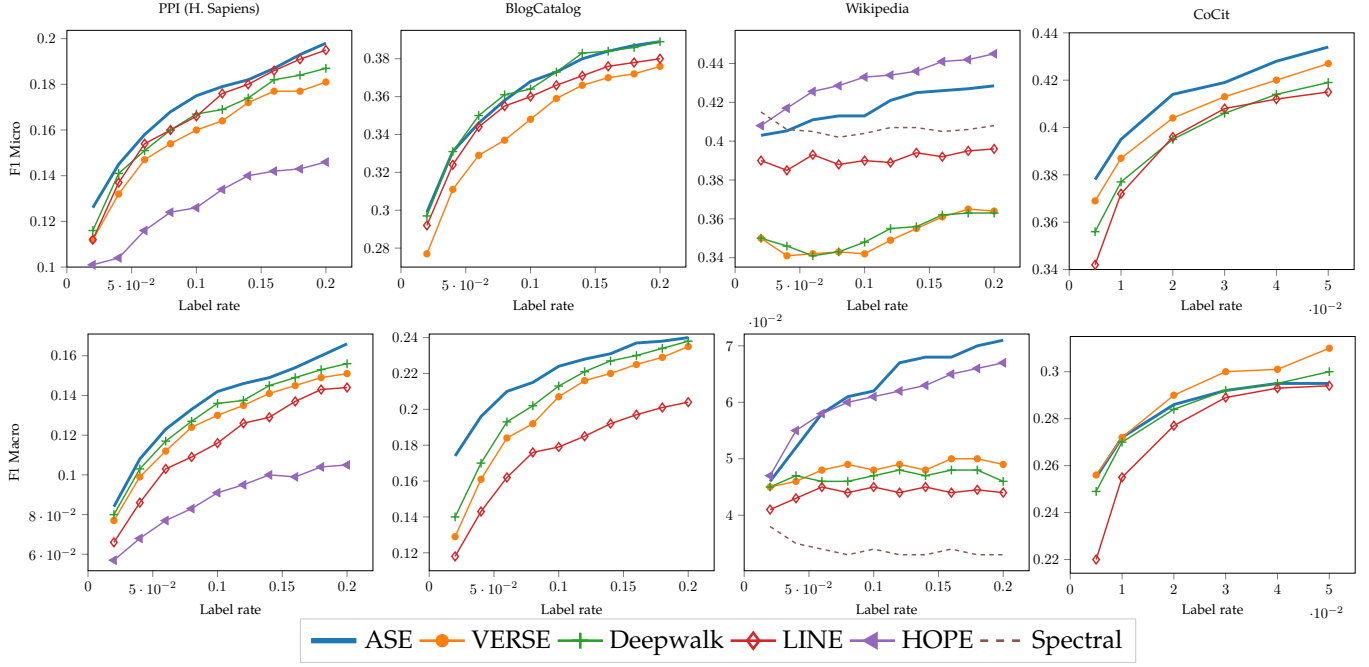


Fig. 5: Micro (upper row) and Macro (lower row) F_1 scores that different embeddings + logistic regression yield on labeled graphs, as a function of the labeling rate (percentage of training data)

scores [10] of the predicted labels.

Apart with 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” $\theta^* = e_k$ in Algorithm 1, and then using each embedding for classification with 10% labeling rate. Figure 4 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 θ^* are non-zero (cf. Table 1). As seen in Fig. 4, 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 θ^* depends only on the graph, since ASE does *not* use labels for training or validation.

We also compared the classification accuracy of ASE embeddings with those of the alternative embedding approaches, with results plotted in Fig. 5. The plots for some method-graph pairs are not visible due to values being too low. While the relative performance of any given method varies among different 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 respectively. Interestingly, HOPE achieving high Micro F_1 and low Macro F_1 in Wikipedia is in agreement with the findings in Fig. 4, combined with the fact that HOPE focuses on longer paths.

Link prediction. Link prediction is the task of estimating the probability that a link between two unconnected nodes will appear in the future. We repeat the experiment performed in [36] on the vk2016-17 social network. For every possible edge, we build a feature vector as the Hadamard product between the embedded vectors of its two adjacent nodes. Using the two time instances of vk2016-17, we predict whether a new friendship link appears between November 2016 and May 2017, using 50% of the new links for training and 50% for testing. To train the binary logistic regression classifier, we also randomly sample non-existing edges as negative examples. The link prediction accuracy for different embeddings is reported in Table 3. While for this experiment ASE does not reach the accuracy of VERSE, it provides the second most accurate link prediction, far surpassing the also SVD-based HOPE and Spectral embeddings.

TABLE 3: Link Prediction Accuracy on vk2016-17

	VERSE	ASE	LINE	Deepwalk	HOPE	Spectral
Acc.	0.79	0.75	0.74	0.69	0.62	0.60

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 [18]. 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. 6 gives the average conductance across communities, as a

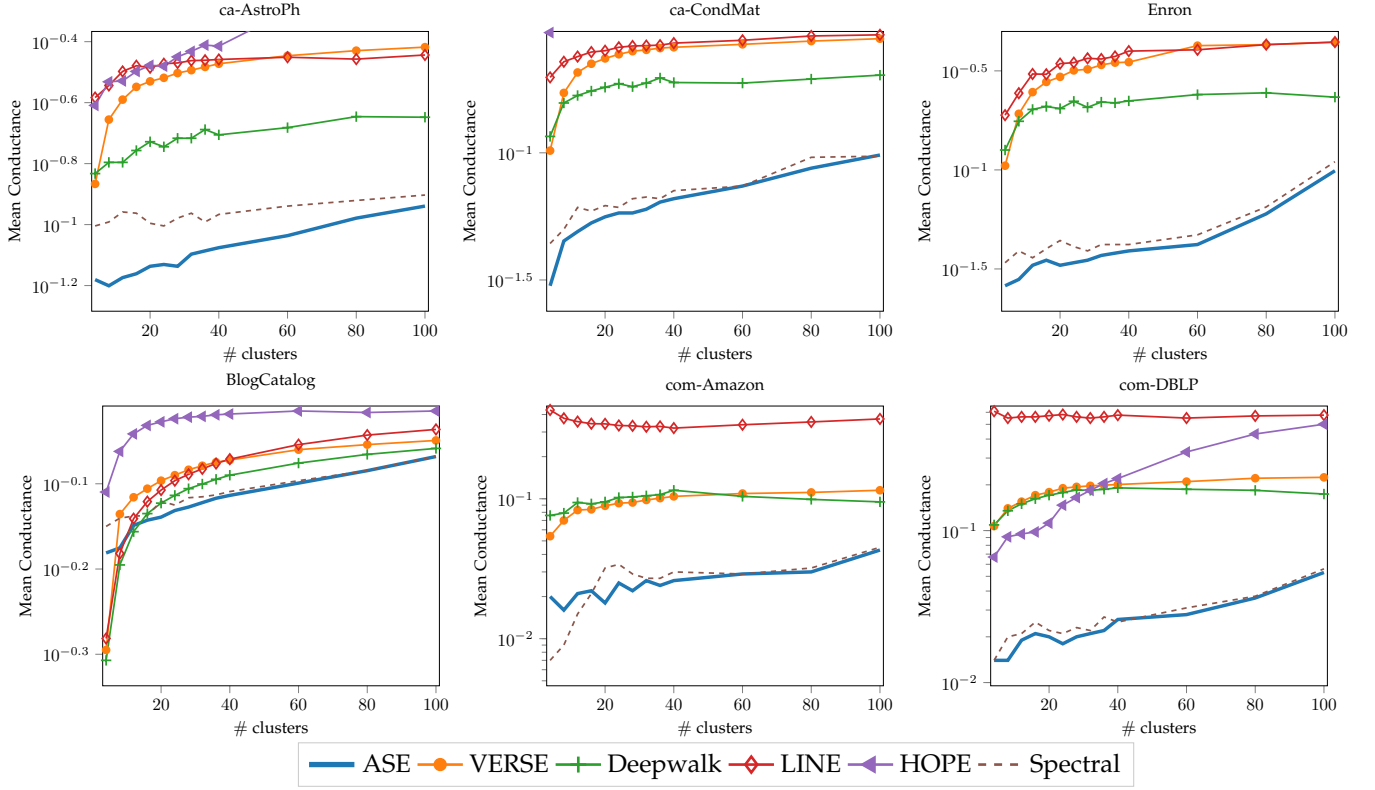


Fig. 6: Average conductance of different embeddings used by kmeans for clustering, as a function of number of clusters.

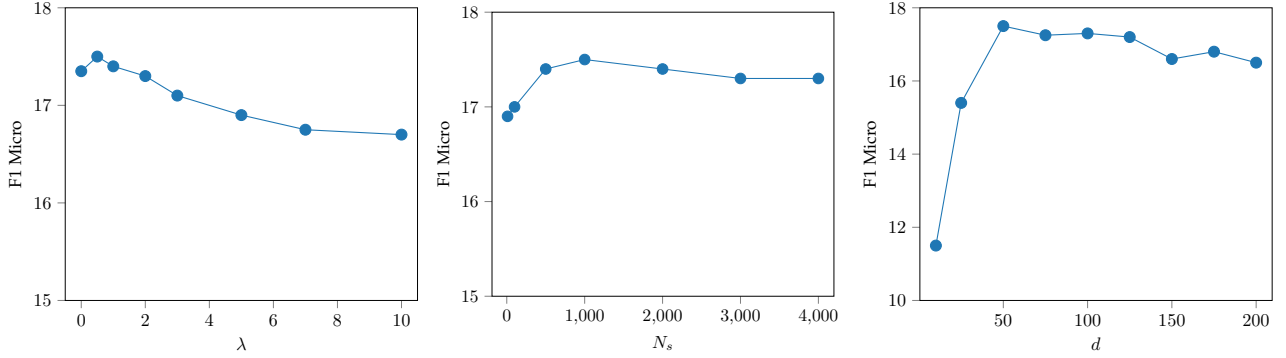


Fig. 7: Sensitivity of ASE on PPI graphs wrt various parameters.

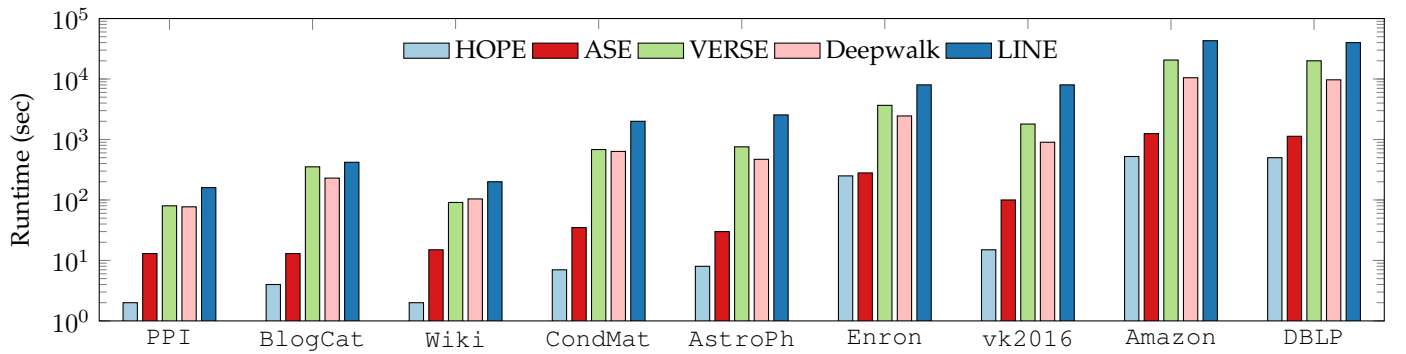


Fig. 8: Runtime of various embedding methods across different graphs

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 [17], which is very similar to conductance. A closer look at the resulting clusters, reveals that clustering based 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 below 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.

Parameter sensitivity. We also present results where we varied ASE parameters and measured classification Micro F_1 accuracy for PPI with 10% labeling rate. The aim is to demonstrate the sensitivity of ASE with respect to its basic parameters. The plot on the left shows how increasing λ (cf. (32)) may decrease accuracy by forcing the entries of θ^* to be close to uniform, thus losing the benefits of graph-specific adaptation. Regarding the number of sampled edges N_s , results (middle plot) indicate relative robustness of ASE embeddings, given a minimum number of samples. As expected, sampling a large number of edges may cause noticeable perturbation on the graph (even using the minimally-perturbing Algorithm 2); this may be causing the slight decrease in accuracy. Finally, the plot on the left presents accuracy across a range of embedding dimensions d .

Runtime. Finally, we compared different embedding methods in terms of runtime. Results for all graphs are reported in Fig. 8. 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 implementation⁶ that uses the SVDLIBC [50] library for sparse SVD. We also developed a more scalable implementation⁷ that relies on (and requires installation of) the SLEPc package [49]; this scalable version can perform large-scale sparse SVD on multiple processes and distributed memory environments using the message-passing interface (MPI) [48]. 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 greatly benefit from decades of research on

how to efficiently perform these decomposition, and a suite of stable and highly optimized software tools.

7 CONCLUSIONS AND FUTURE WORK

We present a scalable node embedding framework that is based on factorizing an adaptive node similarity matrix. The model is carefully studied, interpreted, and numerically evaluated using stochastic block models, with an algorithmic scheme proposed for training the model parameters efficiently and without supervision.

The proposed framework opens up several interesting future directions. For instance, one can explore larger families of node similarity metrics that can be learned using the graph. Furthermore, it would be interesting to assess the performance of different randomized edge sampling methods, and generalize the notion of adaptive-similarity to heterogeneous and multilayered graph embedding, as well as to edge embedding.

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] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Comput. Netw.*, vol. 56, no. 18, pp. 3825–3833, 2012.
- [8] D. F. Gleich, "Pagerank beyond the web," *SIAM Rev.*, vol. 57, no. 3, pp. 321–363, 2015.
- [9] I. M. Kloumann, J. Ugander, and J. Kleinberg, "Block models and personalized pagerank," *Proc. Natl. Acad. Sci.*, vol. 114, no. 1, pp. 33–38, 2017.
- [10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, MA: Cambridge University Press, 2008.
- [11] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.
- [12] 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.
- [13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," *Proc. of the workshop on Comp. Learning Theory*, pp. 144–152, 1992.
- [14] 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.
- [15] 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.
- [16] P. Goyal, and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

6. <https://github.com/DimBer/ASE-project/tree/master/portable>

7. https://github.com/DimBer/ASE-project/tree/master/slep_based

- [17] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
- [18] D. Arthur, and S. Vassilvitskii, "k-means++: The advantages of careful seeding," *SIAM*, pp. 1027–1035, 2007.
- [19] G. H. Golub, and C. Reinsch, "Singular value decomposition and least squares solutions," *Numer. Math.*, vol. 14, no. 5, pp. 403–420, 1970.
- [20] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive Diffusions for Scalable Learning over Graphs," *arXiv preprint arXiv:1804.02081*, 2018.
- [21] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *In Mathematical Programming*, vol. 158, no. 1-2, pp. 575–585, 2016.
- [22] Y. Han, and Y. Shen, "Partially supervised graph embedding for positive unlabeled feature selection," in *Proc. Intl Joint Conf. on Artificial Intelligence*, pp. 1548–1554, New York, NY, 2016.
- [23] T. Hofmann, and J. M. Buhmann, "Multidimensional scaling and data clustering," *Advances in Neural Inf. Proc. Systems*, 1994, pp. 459–466.
- [24] M. Balasubramanian, and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.
- [25] X. He, and P. Niyogi, "Locality preserving projections," in *Advances in Neural Inf. Proc. Systems*, pp. 1531–1600, 2003.
- [26] S. T. Roweis, and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [27] 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. 3748, Rio de Janeiro, Brazil, 2013.
- [28] 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. 2111–2117, 2015.
- [29] 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. 1105–1114, San Francisco, CA, 2016.
- [30] 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.
- [31] 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.
- [32] B. Shaw, and T. Jebara, "Structure preserving embedding," in *Proc. of Intl Conf. on Machine Learning*, pp. 937–944, Montreal, Canada, 2009.
- [33] 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. 1412–1418, Buenos aires, Argentina, 2015.
- [34] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *In IEEE Computer*, np. 8, pp. 30–37, 2009.
- [35] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. of the World Wide Web Conf.*, pp. 1067–1077, Florence, Italy, 2015.
- [36] 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.
- [37] 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. 1165–1174, Sydney, Australia, 2015.
- [38] 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.
- [39] 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.
- [40] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, 2016, pp. 855–864.
- [41] 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. 2787–2795, Lake Tahoe, CA, 2013.
- [42] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types," in *Proc. Intl Joint Conf. on Artificial Intelligence*, pp. 2965–2971, New York, NY, 2016.
- [43] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Spectral Graph Wavelets for Structural Role Similarity in Networks," *arXiv preprint arXiv:1710.10321*, 2017.
- [44] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, Halifax, Canada, 2017, pp. 385–394.
- [45] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 1225–1234, San Francisco, CA, 2016.
- [46] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering," in *Assoc. for the Advanc. of Artif. Intel.*, pp. 1293–1299, Quebec, Canada, 2014.
- [47] 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.
- [48] B. Barker, "Message passing interface (mpi)," in *Workshop: High Performance Computing on Stampede*, vol. 256, 2015.
- [49] 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.
- [50] <https://tedlab.mit.edu/~dr/SVDLIBC/>
- [51] <https://snap.stanford.edu/data/index.html>



graphs, including semi-supervised classification, and node embedding.



Minnesota McKnight Presidential Chair in ECE, and serves as director of the Digital Technology Center.

His general interests span the areas of communications, networking and statistical learning - subjects on which he has published more than 430 journal papers, 720 conference papers, 25 book chapters, two edited books and two research monographs (h-index 134). Current research focuses on learning from Big Data, wireless cognitive radios, and network science with applications to social, brain, and power networks with renewables. He is the (co-) inventor of 32 patents issued, and the (co-) recipient of 9 best journal paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (inaugural recipient in 2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SP Society.