Clustered Embedding of Massive Social Networks

Han Hee Song[§], Berkant Savas[§*], Tae Won Cho[†], Vacha Dave[§],
Zhengdong Lu[‡], Inderjit S. Dhillon[§], Yin Zhang[§], Lili Qiu[§]
The University of Texas at Austin[§] AT&T Labs[†] Microsoft Research Asia[‡] Linköping University[*]
Austin, TX, USA Florham Park, NJ, USA Beijing, P. R. China Linköping, Sweden
{hhsong,berkant,vacha,inderjit,yzhang,lili}@cs.utexas.edu, twcho@research.att.com, zhengdol@microsoft.com

Abstract — The explosive growth of social networks has created numerous exciting research opportunities. A central concept in the analysis of social networks is a proximity measure, which captures the closeness or similarity between nodes in the network. Despite much research on proximity measures, there is a lack of techniques to efficiently and accurately compute proximity measures for large-scale social networks. In this paper, we embed the original massive social graph into a much smaller graph, using a novel dimensionality reduction technique termed *Clustered Spectral Graph Embedding*. We show that the embedded graph captures the essential clustering and spectral structure of the original graph and allow a wide range of analysis to be performed on massive social graphs. Applying the clustered embedding to proximity measurement of social networks, we develop accurate, scalable, and flexible solutions to three important social network analysis tasks: proximity estimation, missing link inference, and link prediction. We demonstrate the effectiveness of our solutions to the tasks in the context of large real-world social network datasets: Flickr, LiveJournal, and MySpace with up to 2 million nodes and 90 million links.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Sociology*

General Terms

Algorithms, Human Factors, Measurement

Keywords

Social Network, Graph Clustering, Graph Embedding, Proximity Estimation, Missing Link Inference, Link Prediction

1. INTRODUCTION

Motivation. Social networks have gained tremendous popularity recently. Social networking sites, such as MySpace, Facebook, YouTube, Twitter and LiveJournal have each attracted tens of millions of visitors each month [32] and are among the most popular sites on today's Internet [3]. The explosive growth of social networks creates exciting research opportunities in network security (*e.g.*, fighting spam [12], defending against Sybil attacks [43, 45]), systems research (*e.g.*, constructing socially aware overlay networks [33] and systems), information technology (*e.g.*, improving Internet search [27] and content recommendation [5]), business (*e.g.*, fraud detection [8], viral marketing [15]), and social sciences (*e.g.*, modeling complex networks [4, 9]).

A central concept in the analysis of social networks is a *proximity measure*, which quantifies the closeness or similarity between nodes in a social network. Intuitively, socially "close" users are often more trustworthy (which is useful for fraud detection [8], spam mitigation [12], and Sybil attack defense [43, 45]), and tend to have similar interests (which is useful for improving Internet search [27] and content recommendation [5]). A variety of effective proximity measures have been proposed, such as the number of common neighbors, the Katz measure [16], rooted PageRank [25], and escape probability [42]. Despite their effectiveness, however, many proximity measures have high computational complexity and are considered prohibitive for large social networks [37, 42].

In our previous work [40], we made significant progress on scalable proximity estimation in large social networks. Its key observation is that for certain proximity measures, such as the Katz measure, while the complete proximity matrix (which specifies the proximity between all node pairs) is massive (with millions of rows and columns), it can be accurately approximated as the product of two factor matrices with much smaller size. In other words, the proximity matrix has a good *low-rank approximation*. Similar ideas based on low-rank approximation have also been successfully exploited in network coordinate systems to approximate delay between Internet hosts (*e.g.*, [26, 31]), and in network compressive sensing to infer missing elements of a network data matrix [46].

Challenges. Despite much progress in low-rank approximation of massive network matrices, three significant challenges remain: (i) *Accuracy*. Existing low-rank approximation techniques are only effective when the matrix of interest has a comparatively low rank. For example, the proximity estimation techniques in [40] use factor matrices with rank 60. Similarly, existing network coordinate systems typically use low-rank approximation with rank below 20. Unfortunately, given the massive scale and enormous complexity of social networks, such few dimensions may not capture enough information about the underlying social structure. Indeed, as shown in [40], proximity measures, such as rooted PageRank [25] and escape probability [42], have much higher intrinsic dimensionality and thus cannot be approximated accurately by existing techniques. (ii) *Scalability*. Existing techniques require computing a large number of rows and columns within the proximity matrix (also known as "landmarks") in order to derive the low-rank ap-

SIGMETRICS'12, June 11–15, 2012, London, England, UK.

proximation. Such preprocessing becomes increasingly expensive as the size of the network grows. Our results in Section 5.2 suggest that with just 2.5% of MySpace's user accounts, the preprocessing already takes over 5 hours. Further scaling up the method poses significant scalability challenges. (iii) *Flexibility.* As shown in [40], no proximity measure performs consistently well across different social networks. Their effectiveness is also sensitive to the choice of control parameters, which is difficult to tune for unclear network characteristics each social graph has. It is desirable to have a flexible technique that automatically learns the optimal parameter configuration and performs consistently better than other measures.

Approach and contributions. To address the above challenges, we develop a novel dimensionality reduction technique, called *clustered spectral graph embedding (CSGE)*, which embeds the original highly sparse but massive social graph into a dense but much smaller graph. In this paper we make the following contributions: (1) We show that CSGE improves performance in a wide range of social network analysis tasks. In particular, CSGE captures essential clustering and spectral structure of the original massive graph and improves proximity measure estimates, performance of link prediction, and missing link inference, even in cases when the considered matrix does not have a low-rank structure. (2) Social network analysis often involves functions of the adjacency matrix of a graph. We show that by incorporating CSGE into these functions, we can analyze and work with functions of huge graphs in a computationally efficient and scalable manner. In addition, the methods are memory efficient and can easily be implemented on a parallel computer. (3) We further generalize the CSGE approach by optimizing for model parameters with supervised learning, by accommodating asymmetric link structure, and by coping with dynamic networks that have incremental updates.

We use three large real-world social network datasets (Flickr, LiveJournal, and MySpace with up to 2 million nodes and 90 million links) to experimentally demonstrate the effectiveness of our CSGE based proximity estimation. In particular, we evaluate its accuracy, scalability and flexibility in the context of three important social network analysis tasks: (i) proximity estimation (*i.e.*, approximating well-known proximity measures proposed in the literature), (ii) *missing link inference* (*i.e.*, inferring the locations of unobserved links based on observed links), and (iii) *link prediction* (*i.e.*, predicting which node pairs will become connected based on past snapshots of the social network).

In the context of proximity estimation, our new technique results in nearly an order of magnitude speedup over the state-of-the-art proximity estimation techniques [40]. More importantly, with the same memory requirement, our technique is able to create approximations when the rank is an order of magnitude higher than previous methods. As a result, our technique results in dramatic improvement on the approximation accuracy of proximity measures, such as rooted PageRank and escape probability, which are not low-rank and thus cannot be approximated accurately by previous methods. In the context of missing link inference, our technique results in several-fold reduction in the false positive rate subject to the same false negative rate. In the context of link prediction, our technique yields a novel *supervised proximity measure* that significantly improves link prediction accuracy by learning optimal parameter configurations and consistently yielding better accuracy than the previously proposed proximity measures across the three social network datasets. These results clearly demonstrate the effectiveness of our approach.

Organization. The remainder of the paper is organized as follows. In Section 2, we provide background on proximity measures, spectral embedding and clustering. In Section 3, we present details on our clustered spectral graph embedding approach. In Section 4, we explore three important applications: proximity estimation, missing link inference, and link prediction, and then evaluate the effectiveness of our methods on these applications in Section 5. In Section 6, we survey the related work, and conclude in Section 7.

2. PRELIMINARIES

A social network can naturally model the explicit friendship or trust relationship among users in a social networking site. A social network can also be defined implicitly to quantify the interaction levels among users. For example, one can construct a social network based on the past (non-spam) email exchanges between users [12], or the traffic volumes transmitted between Internet hosts.

Formally, we denote a social network as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$ is a set of vertices, and $\mathcal{E} = \{e_{ij} \mid i, j \in \mathcal{V}\}$ is a set of edges. In particular, if there is an edge between vertex i and vertex j, then e_{ij} denotes the weight of this edge. The *adjacency matrix* A of the graph G is an $m \times m$ matrix with $m = |\mathcal{V}|$:

$$a_{ij} = A[i, j] = \begin{cases} e_{ij}, & \text{if there is an edge between } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases}$$

Below we define some of the most commonly used proximity measures, all of which can benefit from our new dimensionality reduction technique. We then introduce three important concepts – *graph embedding*, *spectral graph embedding*, and *graph clustering* – essential for understanding our graph embedding technique.

2.1 Proximity measures

Proximity measures are important for many social network applications. Most proximity measures can be divided into two broad categories: (1) direct measures that are based on shortest graph distances or maximum information flow between two nodes or node neighborhoods, *e.g.*, *common neighbors*; and (2) more sophisticated measures that include infinite sums over ensembles of all paths between two nodes, *e.g.*, *Katz measure* [16], *rooted PageRank* [25], and *escape probability* [42]. It has been shown that the path-ensemble based proximity measures capture much more information about the underlying network compared to the direct measures and are generally more effective at performing various tasks [25, 42]. The above measures are defined as follows:

Common neighbors. Let N_i be the neighbor set of vertex i. Then, the common neighbors proximity measure is $P_{\mathsf{cn}}[i, j] = |N_i \cap N_j|$. If the number of common neighbors is high between vertices i and j, then it is more likely they will get connected. For an *unweighted* graph (*i.e.*, $e_{ij} = 1$ for all edges), the common neighbor matrix for all vertex pairs is simply given by $P_{\mathsf{cn}} = A^2$.

Katz measure. Let $p_{ij}^{(k)}$ denote the number of paths of length k between vertices i and j. Then, the Katz measure is

$$P_{\mathsf{kz}}[i, j] = \sum_{k=1}^{\infty} \beta^k p_{ij}^{(k)},$$

where β is a damping parameter. A high value of the Katz measure between two vertices signifies a stronger relationship. Using the adjacency matrix A, we may write the Katz measure for all vertex pairs simultaneously as

$$P_{\mathsf{kz}} = \sum_{k=1}^{\infty} \beta^k A^k = (I - \beta A)^{-1} - I,$$

where I is an identity matrix and $\beta < 1/\|A\|$.

Rooted PageRank. The rooted PageRank measure (rooted at vertex i) is the probability of landing at vertex j in a random walk with a probability α of jumping to vertex i in each step, and with a probability $1-\alpha$ that the process continues to a random neighbor. As an irreducible, finite, and aperiodic Markov chain, the probabilities that rooted PageRank yield are stationary and converge to an equilibrium distribution regardless of the starting distribution.

Let D be the diagonal *degree matrix* given by $D[i,i] = \sum_j A[i,j]$. Let $T = D^{-1/2}AD^{-1/2}$ be the normalized adjacency matrix. The stationary probability of the rooted PageRank for all vertex pairs is

$$\begin{aligned} P_{\mathsf{rpr}} &= (1-\alpha)(I - \alpha D^{-1}A)^{-1} \\ &= (1-\alpha)D^{-1/2}(I-\alpha T)^{-1}D^{1/2} \\ &= (1-\alpha)D^{-1/2}\Big(\sum_{k=0}^{\infty}\alpha^k T^k\Big)D^{1/2}. \end{aligned}$$

Escape probability. The escape probability $P_{\mathsf{ep}}[i,j]$ from vertex i to j is the probability that a random walker (who started from i) will visit j before returning back to i. With α being the random jump probability, the measure is derivable from rooted PageRank by $P_{\mathsf{ep}}[i,j] = f(P_{\mathsf{rpr}}, i, j)$, where the function f is defined as

$$f(P,i,j) = \frac{(1-\alpha)P[i,j]}{P[i,i]P[j,j] - P[i,j]P[j,i]}. \tag{1}$$

2.2 Graph embedding

Let A be an $m \times m$ adjacency matrix of a graph. For simplicity, we assume that A is symmetric. In section 3.2 we will demonstrate how to extend the concepts to non-symmetric adjacency matrices. A *graph embedding* can bemathematically formalized as the decomposition

$$A_{m\times m} \approx U_{m\times r}L_{r\times r}U^{\mathsf{T}}_{m\times r} = ULU^{\mathsf{T}}, \tag{2}$$

where U is an $m \times r$ orthonormal matrix (*i.e.*, $U^{\mathsf{T}}U = I_r$ is an identity matrix), and L is an $r \times r$ matrix. U represents a basis for the *embedding subspace* and L represents the *embedded adjacency matrix* of the original graph. Since U is orthonormal, Eq. (2) can be applied to approximate any matrix power A^k by

$$A^k \approx (ULU^{\mathsf{T}})^k = UL^kU^{\mathsf{T}}. \tag{3}$$

As a special case, with $k = 2$, we get the frequently used common neighbor proximity measure $P_{\mathsf{cn}} = A^2 \approx UL^2U^{\mathsf{T}}$. Many functions of A can be approximated using a sum of matrix powers through the Taylor series expansion. Using Eq. (3) we can approximate these functions with corresponding functions on L. This can significantly reduce the computational cost as typically $r \ll m$ and most of the calculations will involve the $r \times r$ matrix L instead of the $m \times m$ matrix A. For example, using Eq. (2) and Eq. (3), we can approximate the Katz measure as

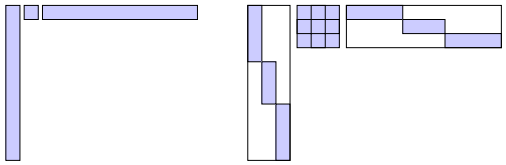
$$P_{\mathsf{kz}} \approx \sum_{k=1}^{\infty}\beta^k UL^kU^{\mathsf{T}} = U\left((I_r - \beta L)^{-1} - I_r\right)U^{\mathsf{T}}.$$

2.3 Spectral graph embedding

The best rank-r approximation of A, in terms of squared approximation error, is given by the r-dimensional *spectral graph embedding* (SGE):

$$A \approx U\Lambda U^{\mathsf{T}}, \tag{4}$$

where Λ is a diagonal matrix with the r largest (in magnitude) eigenvalues of A, and U contains the corresponding eigenvectors. Figure 1(a) shows an illustration of SGE.



(a) Regular SGE (b) CSGE

Figure 1: Illustration of (a) regular SGE $A \approx U\Lambda U^{\mathsf{T}}$ and (b) CSGE $A \approx VSV^{\mathsf{T}}$ in Eq. (7) with $c = 3$ clusters.

Over the past several decades, eigendecomposition and spectral graph embedding have been important research tools for achieving dimensionality reduction on large matrices and graphs. Although there are computationally efficient algorithms [20, 21] to compute the spectral embedding of large sparse matrices, they are still quite expensive on massive social networks with millions of nodes. As a result, SGE can only afford to work with a relatively small r, which may not capture sufficient social/network structure of very large social networks and thus yields poor approximation accuracy (see Section 5).

2.4 Graph clustering

Graph clustering in itself is not the focus of this paper, however, efficient clustering or partitioning algorithms are crucial to the methods we are proposing. Given a graph $G = (\mathcal{V}, \mathcal{E})$ there are various objective functions that measure the quality of the clustering, *e.g.*, *ratio cut* [14] and *normalized cut* [39]. Although these graph clustering objectives are NP-hard to optimize [44], there are several efficient clustering algorithms that often produce good quality results, *e.g.*, Graclus [10], METIS [1], SPAR [34], and "modularity" optimization [6].

Assume that we have a clustering of $G(\mathcal{V}, \mathcal{E})$ into c disjoint clusters specified by the vertex sets $\mathcal{V}_i$, $i = 1, \ldots, c$, *i.e.*, $\bigcup_{i=1}^{c}\mathcal{V}_i = \mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for all $i \neq j$. Let $m_i = |\mathcal{V}_i|$. Without loss of generality, we can assume that the vertices in $\mathcal{V}_1, \ldots, \mathcal{V}_c$ are in a strictly increasing order. Then the adjacency matrix A will have the following form

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}, \tag{5}$$

where each diagonal block A_{ii} is an $m_i \times m_i$ matrix, that can be considered as a local adjacency matrix for cluster i. The off-diagonal blocks A_{ij} ($i \neq j$) are $m_i \times m_j$ matrices that contain the set of edges with vertices belonging to two different clusters. In an ideal scenario with perfect clustering, the off-diagonal blocks will not contain any edges, so $A_{ij} = 0$, and the graph will have c connected components. On the other hand, in the more general scenario of A comprised of a single connected component with a clear clustering structure, a naive approximation of the original graph could be obtained through its diagonal blocks $A \approx \mathrm{diag}(A_{11}, \ldots, A_{cc})$. By further introducing low-rank approximations $A_{ii} \approx V_i\Lambda_i V_i^{\mathsf{T}}$, we have $A \approx \mathrm{diag}(V_1\Lambda_1V_1^{\mathsf{T}}, \ldots, V_c\Lambda_cV_c^{\mathsf{T}})$.

3. CLUSTERED SPECTRAL GRAPH EMBEDDING

In this section, we describe the baseline *clustered spectral graph embedding (CSGE)* and suggest extensions suitable for dealing with large, dynamic social networks. Then we analyze advantages of our proposed CSGE over regular spectral graph embedding (SGE) in terms of computational efficiency and accuracy.

3.1 Basic algorithm

Our baseline algorithm, *clustered spectral graph embedding* (CSGE) improves the efficiency and accuracy of approximating

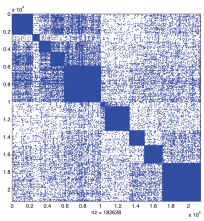


Figure 2: Sparsity pattern of an adjacency matrix with $c = 10$ clusters; 80% of the edges are within the diagonal blocks.

various proximity measures by effectively combining clustering with spectral graph embedding. Recall that A is the adjacency matrix of a graph. We will assume that the graph has been partitioned into c clusters and that the vertices are ordered so that the diagonal blocks A_{ii}, $i = 1, \cdots, c$, correspond to the local adjacency matrices for the different clusters as in Eq. (5). Since the block partitioning of A is obtained through clustering of the graph, it follows that most of the edges are within the diagonal blocks. Only a small fraction of the edges are between clusters and are consequently located in the off-diagonal blocks. Figure 2 shows the typical sparsity pattern of an adjacency matrix after a clustering step. In this particular case, we have clustered the graph into $c = 10$ clusters and 80% of the edges are within the diagonal blocks. Computing the best rank-r_i approximations through spectral graph embedding for every cluster (diagonal block), we get

$$A_{ii} \approx V_i \Lambda_i V_i^{\mathsf{T}}, \quad i = 1, \ldots, c, \tag{6}$$

where Λ_i is a diagonal matrix and contains the r_i largest eigenvalues of A_{ii} (in magnitude), and V_i is an orthonormal matrix with the corresponding eigenvectors. Due to the orthonormality of V_i it can be shown that the matrix $V = \mathrm{diag}(V_1, \ldots, V_c)$ is also orthonormal. We can now use this block-diagonal matrix V to obtain a graph embedding for the entire adjacency matrix A. The graph embedding may be written as $A \approx VSV^{\mathsf{T}}$. Since V is orthonormal, it follows that the optimal S, in least squares sense, is

$$S = V^{\mathsf{T}} A V = \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix},$$

where $S_{ij} = V_i^{\mathsf{T}} A_{ij} V_j$, for $i, j = 1, \ldots, c$. Using Eq. (6) we can verify that $S_{ii} = \Lambda_i$ are diagonal. The off-diagonal blocks S_{ij}, on the other hand, capture interactions between different clusters. We obtain the following approximation

$$A \approx VSV^{\mathsf{T}} = \tag{7}$$
$$\mathrm{diag}(V_1, \ldots, V_c) \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix} \mathrm{diag}(V_1, \ldots, V_c)^{\mathsf{T}},$$

which we call *clustered spectral graph embedding* (CSGE). For example, with $c = 3$ we can write

$$A \approx \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix}^{\mathsf{T}}.$$

Figure 1 shows an illustration of SGE compared to CSGE. Using analogous terminology SGE, we denote $V = \mathrm{diag}(V_1, \cdots, V_c)$ as a basis for the *clustered embedding subspace*, and S as the *embedded adjacency matrix* for CSGE. Note that with $c = 1$, CSGE becomes the regular SGE. In related preliminary evaluation, we explored the benefit of combining clustering with different local (within-cluster) low rank approximation schemes from the viewpoint of numerical accuracy on a few static graphs [38]. However, higher numerical accuracy in matrix approximations does not necessarily translate to benefits in end applications, such as link prediction. Additionally, social network analysis requires more than matrix approximation in order to accommodate scalability and flexibility for various algorithms. In the subsequent sections, we develop and validate CSGE in the context of large, dynamic social networks.

3.2 Extensions

Different social networks exhibit different link structure, density, and dynamic behavior as their purpose, characteristics, and demography vary. In this section, we propose the following extensions to the basic CSGE algorithm aiming to better accommodate various networks: (i) asymmetric adjacency matrix computation to flexibly accommodate networks with asymmetric link relationships, (ii) further embedding to scalably handle larger networks, (iii) incremental updates to cope with dynamic nature of social networks, and (iv) parallel/distributed computation to balance the computation load.

Asymmetric adjacency matrices. There are two possible strategies to cope with an asymmetric adjacency matrix A:

(1) When the fraction of asymmetric edges (*i.e.*, vertex pairs (i, j) such that $A[i, j] \neq A[j, i]$) is not high, we can simply derive the basis of the clustered embedding subspace (*i.e.*, the V matrix) using a symmetrized version of A, *e.g.*, $A_{\mathrm{sym}} = \frac{1}{2}(A + A^{\mathsf{T}})$ or $A_{\mathrm{sym}} = \max(A, A^{\mathsf{T}})$. We can then capture the asymmetry of A by solving $\min_S \|A - VSV^{\mathsf{T}}\|_F$ which yields $S = V^{\mathsf{T}} A V$, where $\|X\|_F = \sqrt{\sum_{ij} X[i, j]^2}$ is the Frobenius norm of X.

(2) Alternatively, we can apply singular value decomposition [13] to compute the best rank-r_i approximation for every cluster (diagonal block), yielding $A_{ii} \approx U_i \Sigma_i V_i^{\mathsf{T}}$ for $i = 1, \ldots, c$, where Σ_i is a diagonal matrix containing the r_i largest singular values of A_{ii}, U_i and V_i are orthonormal matrices with the corresponding left and right singular vectors, respectively. Due to the orthonormality of U_i and V_i, $U \triangleq \mathrm{diag}(U_1, \ldots, U_c)$ and $V \triangleq \mathrm{diag}(V_1, \ldots, V_c)$ are also orthonormal. We can then use the block-diagonal matrices U and V to obtain an asymmetric clustered graph embedding for the entire adjacency matrix A, which is written as $A \approx USV^{\mathsf{T}}$. The optimal S, in least squares sense, is

$$S = U^{\mathsf{T}} A V = \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix},$$

where $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$, for $i, j = 1, \ldots, c$. The asymmetric graph embedding $A \approx USV^{\mathsf{T}}$ has properties very similar to that of the symmetric graph embedding. For example, it can be applied to efficiently approximate matrix powers $A^k \approx (USV^{\mathsf{T}})^k = U(SV^{\mathsf{T}}U)^{k-1}SV^{\mathsf{T}}$.

Further embedding. With a large number of clusters, and larger ranks in the approximations, the size of S could become too large. To reduce memory usage, we can further compute SGE of the embedded adjacency matrix S. That is, we further approximate S by keeping the largest eigenvalues (in magnitude), *i.e.*, $S \approx \bar{Q}\bar{\Lambda}\bar{Q}^{\mathsf{T}}$, where $\bar{\Lambda}$ contains the largest eigenvalues of S and $\bar{Q}$ contains the corresponding eigenvectors. The combined approximation is then $A \approx V\bar{Q}\bar{\Lambda}Q^{\mathsf{T}}V^{\mathsf{T}}$. Note that we can directly derive SGE $S \approx \bar{Q}\bar{\Lambda}\bar{Q}^{\mathsf{T}}$ without having to explicitly compute the dense matrix S (which requires extra computation and memory). All we need is to treat $S = V^{\mathsf{T}} A V$ as an operator acting on a vector. Thus, for

a given vector v we compute $Sv = (V^\mathsf{T}AV)v = V^\mathsf{T}(A(Vv))$. Since V is block diagonal, and A is sparse, computing matrix-vector products with both V and A is efficient. State-of-the-art algorithms for sparse eigendecomposition (*e.g.*, [21, 22]) can then efficiently compute SGE $S \approx \bar{Q}\bar{\Lambda}\bar{Q}^\mathsf{T}$ using such matrix-vector products.

Incremental updates. Because many social networks are highly dynamic, it is desirable to cheaply update the graph embedding equation $A \approx VSV^\mathsf{T}$ under the new adjacency matrix $A' = A + \Delta_A$ when Δ_A has only few non-zero elements. A simple strategy is to keep V stable while updating S with

$$S' = V^\mathsf{T}A'V = V^\mathsf{T}(A + \Delta_A)V = S + V^\mathsf{T}\Delta_A V \triangleq S + \Delta_S, \quad (8)$$

where $\Delta_S = V^\mathsf{T}\Delta_A V$ contains the updates to S and can be efficiently computed due to the sparsity of Δ_A. The embedding for the new adjacency matrix A' is then $A' \approx VS'V^\mathsf{T}$.

Parallel/distributed computation. For the applications with pre-clustered input, CSGE is naturally suited for parallel/distributed computation. For example, consider the email exchange graph of a large corporation, which may consist of a number of organizations. Instead of requiring a separate clustering step, we can directly partition users based on their natural organizational boundaries. Each organization (*i.e.*, cluster) i can then derive V_i locally based on its internal email exchange subgraph A_{ii} according to $A_{ii} \approx V_i\Lambda_i V_i^\mathsf{T}$. Each pair of organizations i and j can jointly compute the block S_{ij} of the embedded adjacency matrix S based on their V_i, V_j and the inter-organization email exchange subgraph A_{ij}, which can again be locally monitored. The ability to support parallel/distributed computations can further improve the efficiency in parallel/distributed computational environments.

3.3 Advantages

Compared with the rank-r SGE $A \approx U\Lambda U^\mathsf{T}$, the CSGE $A \approx VSV^\mathsf{T}$ can achieve much higher accuracy while using a comparable amount of memory. In many cases, CSGE is also computationally more efficient. To simplify the discussion for CSGE, we use r-dimensional graph embeddings for each cluster A_{ii} (*i.e.*, $r_i = r$).

3.3.1 Efficiency

Memory usage of the final embedding. From the block-diagonal structure of V, we immediately observe that we have a cr-dimensional embedding in CSGE, while only an r-dimensional embedding in SGE. Thus, there are c times more columns in $V_{m\times cr}$ than in $U_{m\times r}$. However, $V_{m\times cr}$ has exactly the same $O(mr)$ memory usage as $U_{m\times r}$, because we only store the diagonal blocks V_i, and not the off-diagonal blocks, which are all zeros (see Figure 1). Comparing the embedded adjacency matrices S from CSGE and Λ from SGE, we see that the $cr \times cr$ matrix S is larger and dense, while the Λ is $r \times r$ and diagonal. Therefore, CSGE uses $O(mr + c^2r^2)$ total memory and SGE uses $O(mr + r)$. For large graphs with millions of vertices, the memory complexity is dominated by U or V because we have $m \gg r$ and $m \gg cr$. For example, in the LiveJournal dataset $m = 1,770,961$. With (typical values) $r = 100$ and $c = 50$, S accounts for only 12% of the total memory usage of CSGE. As shown in Section 3.2, we can further reduce the memory usage of S through another embedding. So the total memory usage of CSGE is comparable to that of SGE.

Memory usage for deriving the embedding. With state-of-the-art algorithms for computing a small number of eigenvectors of a large and sparse matrix (*e.g.*, [21,22]), SGE has a memory complexity of $O(m(r + p))$, where p is a user specified parameter and usually $p \approx r$. A smaller p requires less memory but the convergence becomes slower, and vice versa. For CSGE, the approximation of the different clusters is independent of one another and thus the computation of each cluster can be done using the entire memory space. Because the size of each cluster $m_i = |\mathcal{V}_i|$ may be orders of magnitude smaller than $m = \sum_{i=1}^{c} m_i$, we can compute the per-cluster spectral graph embedding with a much larger dimension than the embedding on the entire m. When $c = 10$, for example, the average cluster size would be about $m/10$. Allowing CSGE to fully utilize the entire memory space would increase the maximum computable dimensions from r to $10r$.

Computational efficiency. The time complexity for computing CSGE or SGE is dominated by the cost of eigendecomposition. State-of-the-art algorithms (*e.g.*, [21,22]) for sparse eigendecomposition are typically iterative; their time complexity for each iteration scales linearly with respect to the number of non-zeros in the input matrix, whereas the number of iterations required to achieve convergence depends on the gap between adjacent eigenvalues. Thus, it typically takes many more iterations to decompose the normalized adjacency matrix $T = D^{-1/2}AD^{-1/2}$ than the original adjacency matrix A because eigenvalues of T have smaller gaps.

It is easy to see that the number of non-zeros in the global adjacency matrix A is larger than the total number of non-zeros from all of the per-cluster adjacency matrices A_{ii}. Hence, the per-iteration cost for decomposing A is higher than the total per-iteration cost for decomposing all the A_{ii}. In addition, our experience suggests that it often takes a larger number of iterations for the global eigendecompositon to converge. As a result, it is often much faster to compute many r-dimensional graph embeddings of smaller matrices A_{ii} than it is for a single r-dimensional graph embedding computation of a large A. Our experimental results in Section 5 show that even after including the initial clustering time of the graph as well as the computation time for S, CSGE is still much faster than SGE. In the specific case when the normalized adjacency matrix T is used, CSGE is an order of magnitude faster than SGE. Moreover, with parallel/distributed computation of clusters detailed in Section 3.2, we can further improve the timing efficiency of CSGE.

3.3.2 Accuracy

An important consequence of explicitly focusing on each cluster of the network is that, using the same amount of memory, CSGE yields a significantly smaller residual than the residual for the regular SGE, *i.e.*, $\|A - VSV^\mathsf{T}\|_F < \|A - U\Lambda U^\mathsf{T}\|_F$, Recall that SGE is optimal with respect to the dimension of the embedding (or with respect to the rank in the approximation), but in terms of memory consumption, CSGE gives significantly better embedding (see Section 5).

The accuracy benefit of CSGE is most significant when we are able to transform (using permutations) a given adjacency matrix so that most of the non-zeros are contained within the diagonal blocks. This property is closely related to a graph forming good clusters. Many, if not all, real-world graphs and social networks exhibit this property of forming reasonably good clusters [23]. This is certainly the case for the datasets we consider in this paper.

Interestingly, even if the original graph does not form good clusters or if the clustering algorithm performs poorly, CSGE can still achieve a lower residual error than the regular SGE. The following theorem establishes this guarantee for the special case with $c = 2$ clusters [1].

THEOREM 1. *Let $A \approx U\Lambda U^\mathsf{T}$ be the r-dimensional SGE. Split U into any two parts $U^\mathsf{T} = [U_1^\mathsf{T}\ U_2^\mathsf{T}]$. Let $U_i = Q_iR_i$ be the QR decomposition [13] of U_i, where Q_i is orthonormal (i.e., $Q_i^\mathsf{T}Q_i =$*

[1] The proof of the theorem is provided in the technical report version of this paper [41].

I_r) and R_i is upper triangular. Let $V = \mathrm{diag}(Q_1, Q_2)$ and let $S = V^\mathsf{T} A V$. We have $\|A - VSV^\mathsf{T}\|_F \leq \|A - U\Lambda U^\mathsf{T}\|_F$.

The theorem can be easily generalized to the case with $c > 2$ clusters. Therefore, under any arbitrary clustering of the original graph, there always exists CSGE that has a lower residual error than the regular SGE.

4. APPLICATION TO SOCIAL NETWORKS

In this section, we describe how we apply CSGE to three significant social network analysis tasks. In proximity estimation, we provide detailed description on estimation techniques to a number of proximity measures. In missing link inference, we leverage CSGE in determining unobserved interactions in the well appreciated network analysis problem. And in link prediction, we develop novel parameter learning schemes aimed at automatically optimizing for the best parameter configuration for targeted networks.

4.1 Proximity estimation

In Section 2.1 we defined four proximity measures: common neighbors, Katz measure, rooted PageRank, and escape probability. Each of these proximity measures can be computed or approximated using three different approaches: (i) direct computation; (ii) approximation using SGE $A \approx U\Lambda U^\mathsf{T}$ from Eq. (4); and (iii) approximation using CSGE $A \approx VSV^\mathsf{T}$ from Eq. (7). We obtain 12 different proximity measures as summarized in Table 1. Note that following [40], the summation for the Katz measures are truncated to only include $k_{\max} = 6$ terms (*i.e.*, at most length-6 paths are taken into account). This truncation is necessary to compute the real proximity measures that do not use any graph embedding [40]; these serve as the ground truth for accuracy evaluation when we compare approximated measures from (ii) and (iii). In the rooted PageRank measures where we truncate the summation to only include $k_{\max} = 20$ terms, we have $T \triangleq D^{-1/2} A D^{-1/2} \approx \bar{U}\bar{\Lambda}\bar{U}^\mathsf{T}$ (from SGE) and $T \approx \bar{V}\bar{S}\bar{V}^\mathsf{T}$ (from CSGE). Finally, the escape probability measures are computed from the rooted PageRank measures using the function f defined in Eq. (1).

4.2 Missing link inference

Missing link inference aims to infer additional links that, while not directly visible, are likely to exist (based on directly observed links). Missing link inference falls into the general realm of compressive sensing, which aims to reconstruct missing data based on indirect, partial observations. Compressive sensing has many applications in networking [46] and beyond, and has attracted considerable research attention recently (*e.g.*, [7, 11, 35, 46]).

Problem definition. Let $G = (\mathcal{V}, \mathcal{E})$ be a social network with a binary adjacency matrix A. Let $\mathcal{O}$ be the set of observed edges or links and $\mathcal{M} = \mathcal{E} \setminus \mathcal{O}$ be the set of missing or unobserved links. Let $A_\mathcal{O}$ be the incomplete adjacency matrix containing the set of observed links. $A_\mathcal{O}$ is defined as

$$A_\mathcal{O}[i,j] = \begin{cases} A[i,j], & \text{if } (i,j) \in \mathcal{O}; \\ 0, & \text{otherwise.} \end{cases}$$

Let $A_\mathcal{M} = A - A_\mathcal{O}$. Given $A_\mathcal{O}$, the goal of missing link inference is to infer non-zeros in $A_\mathcal{M}$ as they correspond to the set of missing links in $\mathcal{M}$.

Inference algorithm. Despite much progress on compressive sensing, we are not aware of any existing compressive sensing algorithm that can scale to massive social networks with millions of vertices. Therefore, in this paper we explore the following simple but much more scalable heuristic. We first compute proximity measures based on the observed incomplete adjacency matrix $A_\mathcal{O}$ (*e.g.*, those given in Section 4.1), and then assume that the high values in these proximity measures will correspond to the set of missing links, *i.e.*, non-zeros in $A_\mathcal{M}$. The threshold for determining high proximity measure values can be varied to achieve different tradeoffs between false positives and false negatives.

4.3 Link prediction

Link prediction [25] refers to the task of predicting which vertex pairs in a social network will become connected. An accurate link predictor can provide valuable insights for constructing meaningful network evolution models [4, 9]. It also allows social networks to make high-quality recommendations on potential new friends, making it easier for individual users to expand their social neighborhood. Link prediction also has direct applications in cyber security. For example, it allows one to conjecture that certain individuals in a terrorist network are working together even though their interaction has not been directly observed [18]. In this section, we show how CSGE and supervised learning together can facilitate more accurate link prediction with no parameter configurations.

4.3.1 Problem specification

A natural setting for evolving social networks, or networks in general, is to introduce discrete time steps $t = 1, \cdots, t_{\max}$ at which "snapshots" $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ of the graph are taken. Denote the corresponding adjacency matrices with A_t. To ease the analysis and notation, we will restrict ourselves and only use $\mathcal{V}_1$ for all time steps, *i.e.*, $\mathcal{V}_k = \mathcal{V}_1 \triangleq \mathcal{V}$. It is clear that the evolution of the graph is incremental. In terms of the adjacency matrices, we can express this as $A_{t+1} = A_t + \Delta_t$, where Δ_t contains the edges or links that are formed between time t and $t+1$ (for simplicity, we assume no edges are removed).

In the link prediction problem, given A_t, we try to find the non-zeros in Δ_t as they correspond to newly formed links. The standard heuristic is to first compute some proximity measures based on A_t (*e.g.*, those given in Section 4.1) and then assume that the high scores in these proximity measures will correspond to new links, *i.e.*, non-zeros in Δ_t. In the following, we present several supervised proximity measures that explicitly target Δ_t and construct graph specific models for link prediction. To clarify the problem setting, we use three snapshots, A_1, A_2, and A_3, in the order of data collection time. The first two will be used to learn the model and the third one will be used to validate the obtained model. Extending to more than three snapshots is straightforward.

4.3.2 Link prediction models and framework

Link prediction models. The models we propose have generic prediction scores, obtained from a low dimensional embedding

$$P_* = W_* F_*(x) W_*^\mathsf{T},$$

where W_* represents the basis for a graph embedding, and $F_*(x)$ is a small matrix with the model parameters x that will be learned. In particular, we will consider the following two models that are characterized by the particular form of the embedded adjacency matrix $F_*(x)$ and the kind of graph embedding (spectral or clustered spectral) that is used.

- *Spectral learning.* In this model, we set $W_{\text{sl-sge}} = U_t$, where $A_t \approx U_t \Lambda_t U_t^\mathsf{T}$ is SGE of the adjacency matrix A_t. We then let the parameter matrix be of the form

$$F_{\text{sl-sge}}(x) = \mathrm{diag}(x_1, \cdots, x_r).$$

- *Clustered spectral learning.* Here we set $W_{\text{sl-csge}} = V_t$, where $A_t \approx V_t S_t V_t^\mathsf{T}$ is CSGE of the adjacency matrix A_t as

Abbreviation	Method name	Definition
cn	Common neighbor	$P_{\mathsf{cn}} = A^2$
cn-sge	Common neighbor with spectral graph embedding	$P_{\mathsf{cn\text{-}sge}} = U\Lambda^2 U^\mathsf{T}$
cn-csge	Common neighbor with clustered spectral graph embedding	$P_{\mathsf{cn\text{-}csge}} = VS^2V^\mathsf{T}$
kz	Katz measure	$P_{\mathsf{kz}} = \sum_{k=1}^{k_{\max}} \beta^k A^k$
kz-sge	Katz measure with spectral graph embedding	$P_{\mathsf{kz\text{-}sge}} = U\big(\sum_{k=1}^{k_{\max}} \beta^k \Lambda^k\big)U^\mathsf{T}$
kz-csge	Katz measure with clustered spectral graph embedding	$P_{\mathsf{kz\text{-}csge}} = V\big(\sum_{k=1}^{k_{\max}} \beta^k S^k\big)V^\mathsf{T}$
rpr	Rooted PageRank	$P_{\mathsf{rpr}} = (1-\alpha)D^{-1/2}\big(\sum_{k=0}^{k_{\max}} \alpha^k T^k\big)D^{1/2}$
rpr-sge	Rooted PageRank with spectral graph embedding	$P_{\mathsf{rpr\text{-}sge}} = (1-\alpha)D^{-1/2}\big(I + \bar{U}\big(\sum_{k=1}^{k_{\max}} \alpha^k \bar{\Lambda}^k\big)\bar{U}^\mathsf{T}\big)D^{1/2}$
rpr-csge	Rooted PageRank with clustered spectral graph embedding	$P_{\mathsf{rpr\text{-}csge}} = (1-\alpha)D^{-1/2}\big(I + \bar{V}\big(\sum_{k=1}^{k_{\max}} \alpha^k \bar{S}^k\big)\bar{V}^\mathsf{T}\big)D^{1/2}$
ep	Escape probability	$P_{\mathsf{ep}}[i,j] = f(P_{\mathsf{rpr}}, i, j)$
ep-sge	Escape probability with spectral graph embedding	$P_{\mathsf{ep\text{-}sge}}[i,j] = f(P_{\mathsf{rpr\text{-}sge}}, i, j)$
ep-csge	Escape probability with clustered spectral graph embedding	$P_{\mathsf{ep\text{-}csge}}[i,j] = f(P_{\mathsf{rpr\text{-}csge}}, i, j)$

Table 1: Description of the proximity measures.

in Eq. (7). The parameter matrix has the form

$$F_{\mathsf{sl\text{-}csge}}(x) = Q_t \operatorname{diag}(x_1, \cdots, x_{cr}) Q_t^\mathsf{T},$$

where orthogonal matrix Q_t is obtained from the full eigendecomposition $S_t = Q_t \Lambda_{S_t} Q_t^\mathsf{T}$. Recall that V_t is block diagonal and S_t is a dense matrix that captures interactions among all clusters. The number of parameters to be learned is cr, where c is the number of clusters and r is the embedding dimension.

Framework. Our basic framework for supervised link prediction consists of the following two phases:

1. *Supervised learning.* We compute the basis matrix W_* from either SGE or CSGE of the first snapshot A_1. We then learn the parameter matrix $F_*(x)$ by performing linear regression on $\Delta_1 = A_2 - A_1$ (which contains the links newly formed between snapshots A_1 and A_2) with respect to $F_*(x)$.
2. *Link prediction.* We compute the basis matrix $\bar{W}_*$ from either SGE or CSGE of the second snapshot A_2. We then use $P_* = \bar{W}_* F_*(x) \bar{W}_*^\mathsf{T}$ as the prediction score matrix to predict $\Delta_2 = A_3 - A_2$, which contains the set of new links that are formed between snapshots A_2 and A_3.

4.3.3 Supervised learning

Because different networks exhibit different link structures, performance of link prediction schemes vary to their characteristics ([40]). Moreover, because the characteristics tend to be unclear, it is not easy to pick one that will perform the best. To flexibly accommodate diverse networks, we develop a series of supervision schemes that seek for the most favorable model parameters.

Problem setup. At each time step t, there are three associated sets: the set of existing edges $\mathcal{E}_t$; a positive set $\mathcal{P}_t$ containing vertex pairs that form new links in the time interval $(t, t+1]$; and a negative set $\mathcal{N}_t$ containing vertex pairs without edges at time step $t+1$. Using adjacency matrices A_t and A_{t+1}, we can write

$$\begin{aligned}
\mathcal{E}_t &= \{(i,j) \mid A_t[i,j] \neq 0\},\\
\mathcal{P}_t &= \{(i,j) \mid A_t[i,j] = 0 \text{ and } A_{t+1}[i,j] \neq 0\},\\
\mathcal{N}_t &= \{(i,j) \mid A_t[i,j] = 0 \text{ and } A_{t+1}[i,j] = 0\}.
\end{aligned}$$

It is straightforward to see that $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \mathcal{P}_t$, and that the three sets $\mathcal{E}_t, \mathcal{P}_t, \mathcal{N}_t$ are mutually disjoint.

Supervised learning of model parameters. We perform linear regression on Δ_1 to learn the parameter matrix $F_*(x)$. Specifically, we learn x by solving the following least squares problem:

$$\min_x \sum_{(i,j)\in\mathcal{S}} \big([W_* F_*(x) W_*^\mathsf{T}]_{i,j} - \Delta_1[i,j]\big)^2, \tag{9}$$

where W_* is obtained from either SGE or CSGE of A_1, $F_*(x)$ is one of the parameter matrices from Section 4.3.2 (for time step $t=1$), $\Delta_1 = A_2 - A_1$ contains the new links, the notation $[\,\cdot\,]_{i,j}$ denotes the i,j entry of the argument, and $\mathcal{S}$ is a sample set (see discussions below). Care must be taken in order to solve this least squares problem efficiently. In the interest of brevity, we will defer the details of the solution process to the technical report [41].

Choice of the sample set $\mathcal{S}$. In an ideal case, the model parameters in the vector x should be learned only over the positive set $\mathcal{P}_1$ and the negative set $\mathcal{N}_1$, thus setting $\mathcal{S} = \mathcal{P}_1 \cup \mathcal{N}_1$. With this approach, the edges that already exist in $\mathcal{E}_1$, *i.e.*, the non-zeros in A_1, will not contribute to the learning process. The reason behind this choice of the sample set is that we only want to target links that will form during the next time interval and we do not want to target links that already exist. Unfortunately, for the social networks we consider, the choice $\mathcal{S} = \mathcal{P}_1 \cup \mathcal{N}_1$ would yield a sample set of the order $|\mathcal{V}|^2$ and this is practically impossible to work with. To make the problem manageable, we choose $\mathcal{S}$ to contain a fraction of $\mathcal{P}_1$ and a fraction of $\mathcal{N}_1$. In addition $|\mathcal{S}|$ should not only be large enough to capture the essence of the model, but also have a manageable size. In our experiments we have $|\mathcal{V}| \approx 2 \cdot 10^6$ and we choose $|\mathcal{S}| \approx 5 \cdot 10^5$ (see Section 5).

4.3.4 Link prediction

Different link prediction models are validated by predicting the new links in A_3. The prediction scores are based on

$$P_* = \bar{W}_* F_* \bar{W}_*^\mathsf{T},$$

where $\bar{W}_*$ is obtained from a SGE or CSGE based on A_2 (as opposed to A_1 which is used in the model learning step). Specifically, let SGE of A_2 be $U_2\Lambda_2 U_2^\mathsf{T}$, then the prediction scores for sl-sge can be written as

$$P_{\mathsf{sl\text{-}sge}} = U_2 F_{\mathsf{sl\text{-}sge}}(x) U_2^\mathsf{T} = U_2 \operatorname{diag}(x_1, \cdots, x_r) U_2^\mathsf{T},$$

where the parameter vector $x = [x_1, \cdots, x_r]^\mathsf{T}$ is learned by solving Eq. (9). Similarly, let CSGE of A_2 be $V_2 S_2 V_2^\mathsf{T}$, then the prediction scores for sl-csge are given by

$$P_{\mathsf{sl\text{-}csge}} = V_2 F_{\mathsf{sl\text{-}csge}}(x) V_2^\mathsf{T} = V_2 Q_2 \operatorname{diag}(x_1, \cdots, x_{cr}) Q_2^\mathsf{T} V_2^\mathsf{T},$$

where Q_2 is obtained from the eigendecomposition $S_2 = Q_2 \Lambda_{S_2} Q_2^\mathsf{T}$, and the parameter vector $x = [x_1, \cdots, x_{cr}]^\mathsf{T}$ is learned by solving the least squares problem in Eq. (9).

5. EVALUATION

In this section we present experimental results that evaluate accuracy, scalability, and flexibility of CSGE in the context of the three applications on massive scale social network datasets.

	c	$\mu = \sum_i m_i/c$	Links in A_{ii}	links in $A_{ij}, i \neq j$
Flickr	18	110,563	71.8%	28.2%
LiveJournal	17	106,241	72.5%	27.5%
MySpace	17	125,721	51.9%	48.1%

Table 2: Clustering results with Graclus; c is the number of clusters and μ is the average cluster size.

5.1 Dataset description

In our experiments, we use three large real-world online social networks with millions of nodes: Flickr (1,990,149 nodes, 41,302, 536 links), LiveJournal (1,770,961 nodes, 83,663,478 links) and MySpace (2,137,264 nodes, 90,333,122 links) From the dataset, we see that the majority (80%) of links in the positive set $\mathcal{P}_1$ are between user pairs who are two-hops away, whereas the fraction of new links between users who are 4 or more hops away is very small [2]. All users in the datasets are connected to at least one other user and for simplicity, we do not consider rare occasions of link deletions. Links in MySpace are undirected because it requires mutual agreement for users to become friends. Although Flickr and LiveJournal allow directed 'follower/followee' relations, the majority is symmetric. For simplicity, we make Flickr and LiveJournal undirected.

Flickr is a photo-sharing website, where users can connect to each other by indicating a relationship. This dataset was gathered by a breadth-first search on the graph starting from a few seed nodes. To allow most nodes to be discovered, we use the first few months as a bootstrap period, and create three snapshots when most nodes have been discovered and link growth has stabilized. Because of this crawling method, we observe that even though the snapshot dates are just ten days apart, there is 2% growth in the number of links.

LiveJournal is a blogging site, where members can become "fans" of other members. The dataset was collected by listening to the RSS server, which sends out recent update information. The statistics suggest that LiveJournal users are more active in forming links than users in other networks.

MySpace is a social networking site for people to interact with their acquaintances by posting on each other's personal pages. We collected the friendships of the first 10 million users. MySpace assigns user IDs chronologically. Since the first few million IDs we crawled are also the oldest IDs, they have already formed most of the links and are relatively dormant in forming new ones (*i.e.*, the fraction of new links is smaller than those of other networks).

5.2 Scalability

In this section, we compare various aspects of the computational efficiency of CSGE, SGE, and *proximity embedding* introduced in [40]. All benchmarks were done using an Intel Xeon$^{\text{tm}}$ E5440 machine with 32GB memory, running Ubuntu Linux Kernel v2.6.

Graph clustering: In our experiments we use both GRACLUS [10] and METIS [1] to partition the social networks. The two software packages produce different clusterings as they minimize different objective functions: METIS attempts to produce clusters with equal size, regardless of the inherent clustering structure of the network, whereas GRACLUS produces more balanced clusters. Both software produce good quality partitioning in a relatively short period of time. All experiments are conducted on the largest connected component of each graph, which results in only a very small fraction of users (5.1%) and links (0.5%) being discarded. Table 2 gives an example of clustering results on each data set. In this particular case the clustering was performed using GRACLUS in a recursive way until all cluster sizes were smaller than 1/10 of the original network size. We observe that more than 70% of the links in Flickr and LiveJournal are within the same clusters, while only 51.9% of the links in MySpace are within the same clusters.

[2] A detailed description of the three datasets is provided in our technical report [41].

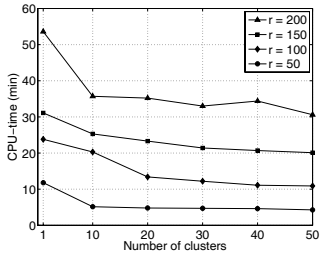


Figure 3: Comparison of eigendecomposition times (on LiveJournal).

Timing benchmarks: Approximation of proximity measures involves two stages: (1) *preparation* of the embedded matrix and (2) *querying* proximity of user pairs from the embedded matrix. We evaluate the timing performance of CSGE by comparing it against other algorithms for each of these stages.

In the first stage of preparing an embedded matrix, we compare CSGE against SGE and Proximity Embedding from [40]. In the case of CSGE, the preparation involves three sub-stages: (i) clustering; (ii) SGE of each cluster; and (iii) computation of the embedded adjacency matrix S. We compare CSGE and SGE for embedding both the adjacency matrix A and the normalized adjacency matrix $T = D^{-1/2}AD^{-1/2}$ in each of these sub-stages. For Proximity Embedding, which does not directly approximate A or T, we instead generate models to compute approximations of P_{kz} and P_{rpr} with relatively few (1,600) "landmark" nodes so that they are comparable to the embeddings of A and T, respectively.

Table 3 summarizes the preparation timing benchmarks for the three approximation algorithms for all three data sets. The number of clusters is set to be $c = 20$ and the rank in SGE and CSGE is set to be $r = r_i = 100$. We see that CSGE outperforms the other two algorithms in most cases; in the approximation of A, CSGE is up to 30% faster than the other algorithms. In the approximation of T, the timing difference between CSGE and SGE becomes even higher, resulting in over an order of magnitude difference. The convergence of iterative spectral methods is influenced by the size of the "gap" in the eigenvalues [21]. Because the eigenvalues of T are normalized to be between 1 and -1 with very small gaps between them, the computation times for T are expected to become much longer than that of A; therefore, the difference in timing performance becomes more dramatic. In an extreme case of Flickr dataset where all 100 dominant eigenvalues of T (and T_{ii}) are very close to 1, the computation of T by CSGE is 20% slower than the computation of rooted PageRank by Proximity Embedding due to the aforementioned reasons.

Among the three sub-stages of CSGE, the majority of time is spent in SGE for each of the A_{ii} and T_{ii}. Figure 3 further analyzes the SGE time of LiveJournal dataset with varying numbers of clusters and sizes of embedding dimensions within each cluster. As Flickr and MySpace exhibit similar trends, we omit them here for brevity. The timing results for $c = 1$ are simply SGE on the entire A (because A is a single connected component). The timings for $c > 1$ are the sum of SGE times for all A_{ii}. We observe from the graph that, for any given r, the aggregate times of cluster-wise approximations are significantly less than the SGE time without clustering. It is also evident that computational time generally decreases with increasing number of clusters. Furthermore, from these results, we can estimate the potential amount of time improvement when we parallelize the per-cluster SGE computation (as discussed in Sec. 3.2).

	Proximity Embedding		SGE		CSGE						
Dataset	Katz	Rooted PageRank	A	T	clustering	SGE of all A_{ii}	S	Total	SGE of all T_{ii}	S	Total
Flickr	216	276	19.9	2,821	1.4	12.0	2.3	**15.7**	327	3.1	**331.5**
LiveJournal	324	456	23.8	420	1.4	12.8	2.4	**16.6**	33.0	2.5	**36.9**
MySpace	330	588	22.8	546	2.6	14.4	2.9	**19.9**	48.1	3.5	**54.2**

Table 3: Comparison of preparation times ($c = 20$ and $r = r_i = 100$). All timings are in minutes.

	Direct Method		Proximity Embedding	SGE		CSGE		
Dataset	cn	kz	kz-prox.embed.	cn-sge	kz-sge	cn-csge	kz-csge	sl-csge
Flickr	15.9ms	8040ms	0.051ms	0.042ms	0.045ms	2.72ms	2.76ms	5.59ms
LiveJournal	23.5ms	14790ms	0.045ms	0.038ms	0.036ms	5.45ms	2.03ms	4.57ms
MySpace	24.5ms	16655ms	0.076ms	0.040ms	0.036ms	2.60ms	2.65ms	4.73ms

Table 4: Comparison of query times ($r = r_i = 100$, 0.6 million samples).

	A_{FL}	B	C
m	2,000,000	6,000,000	12,000,000
Numbers of links	40,000,000	239,000,000	598,000,000
c	18	52	104
Timing	22.6 min	116.3 min	391.2 min
Memory usage	74 MB	546 MB	1,197 MB

Table 5: Computational time and memory usage for CSGE. Network size m and number of links are approximate.

In the second stage of querying proximity of user pairs, we compare the average query times of 600,000 randomly sampled node pairs for the three embedding algorithms (*i.e.*, CSGE, SGE, and Proximity Embedding) as well as directly calculated common neighbor and Katz scores. We consider the direct calculation in the query stage but not in the preparation stage because the storage of exact proximity measures for all m^2 node pairs in preparation stage is prohibitive. From Table 4, we observe that all three embeddings are orders of magnitude faster than the direct calculations. While all three embedding algorithms exhibit a millisecond level of fast query time, CSGE is slightly slower than the other two. This is because CSGE is dependent on the size of its dense core matrix S. While the query time of CSGE is already several folds faster than that of Proximity Embedding, in a time-sensitive online application, the query time can be further improved by either using a smaller S or applying low-rank approximation on S (*c.f.*, further embedding in Sec. 3.2).

Scalability of graph embedding: While CSGE has more efficient precomputation, its major advantage is lower memory usage so that it can achieve higher accuracy for a fixed memory cost. In Section 3.3.1, we discussed the memory usage of CSGE. To show the scalability, both in computational time and memory efficiency, we present some measurements on even bigger (artificial) networks. Let A_{FL}, A_{LJ}, and A_{MS} be the adjacency matrices for Flickr, LiveJournal and MySpace, respectively. Then we form $B = \text{diag}(A_{\text{FL}}, A_{\text{LJ}}, A_{\text{MS}}) + A_{\text{Off}}$ where A_{Off} contains a small fraction of links in the off-diagonal part so that B becomes connected. We also form C in a similar way, $C = \text{diag}(B, B) + B_{\text{Off}}$. One may consider that the constructed datasets correspond to some real-world social networks. The approximate size, number of links, and number of clusters are presented in the upper half of Table 5.

The lower part of Table 5 shows the memory required to store $V = \text{diag}(V_1, \cdots, V_c)$ and the embedded adjacency matrix S. Corresponding computational times are also given. We fix $r_i = 100$ in all cases and examine the behavior by increasing the number of clusters c rather than the size of individual clusters. The memory and time required to create the embedded adjacency matrix S not only grows quadratically with the number of clusters c, but also with the rank r_i. Despite the fact that the size of S is inherently quadratic in c, CSGE is able to handle the largest network with 12 million users and 598 million links - with only 1.2 GB of memory. On the other hand, regular SGE is unable to load U for B network, confirming that S is indeed not a limiting factor of memory usage in CSGE, and that it has higher spatial scalability than SGE.

5.3 Proximity measure estimation

Next, we compare the accuracy of Katz, rooted PageRank, and escape probability when they are combined with SGE and CSGE. For the sake of brevity, we present results only for LiveJournal.

5.3.1 Evaluation methodology

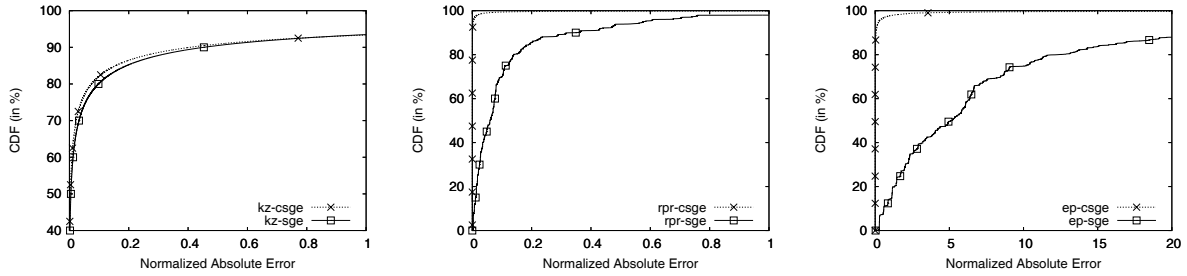
Since it is expensive to compute and store the actual proximity measures for all m^2 user pairs, we instead evaluate SGE and CSGE using sampled user pairs $\mathcal{S}$, which consists of 100 columns randomly chosen from the proximity matrix. This gives roughly 180,000,000 sample points. For each pair $(i, j) \in \mathcal{S}$ we let p_{ij} denote the 'true' value of a proximity measure and $\hat{p}_{ij}$ to be an 'estimate'. For example, we may have $p_{ij} = P_{\mathsf{kz}}[i, j]$ while $\hat{p}_{ij} = P_{\mathsf{kz\text{-}sge}}[i, j]$. The true proximity measures were computed for comparison purpose using the methodology outlined in [40]. Regarding the accuracy measure, we use the *normalized absolute error* $e_{ij} = |p_{ij} - \hat{p}_{ij}|/\mu$, where $\mu = \sum_{(i,j)\in\mathcal{S}} p_{ij}/|\mathcal{S}|$. We plot the *cumulative distribution function* (CDF) to compare the error.

In all Katz computations, we use $\beta = 0.0005$ and $k_{\max} = 6$. For computing the rooted PageRank and the escape probability, we use $\alpha = 0.15$ and $k_{\max} = 20$ (*c.f.*, Table 1). The embedding subspace dimensions in SGE and within each cluster in CSGE are set to $r = 100$. Clustering of the datasets is done on A while the graph embedding is based on A for Katz and $T = D^{-1/2}AD^{-1/2}$ for rooted PageRank and escape probability.

5.3.2 Accuracy evaluation

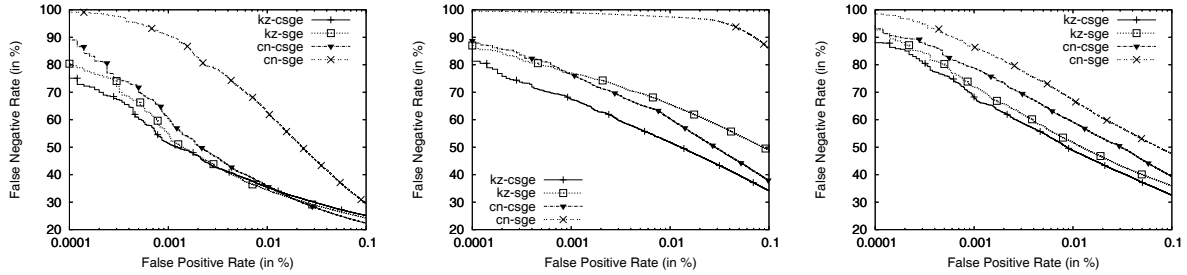
Approximating low-rank matrix. Figure 4 (a) plots the CDF of the normalized absolute errors in approximating Katz measure with $P_{\mathsf{kz\text{-}sge}}$ and $P_{\mathsf{kz\text{-}csge}}$. We make two observations: (1) for most samples, the error is small: with 85% of the node pairs having an error of 0.2 or less; and (2) the error for CSGE is lower than SGE with the gap in error as little as 2%. A likely reason for SGE and CSGE yielding similar performance is that P_{kz} has a low intrinsic dimensionality and that both SGE and CSGE have reached a point where $r = 100$ is enough for the low-rank approximation of Katz.

Approximating non low-rank matrix. In Figure 4 (b) we present the CDF of normalized absolute errors in approximating the rooted PageRank measure with $P_{\mathsf{rpr\text{-}sge}}$ and $P_{\mathsf{rpr\text{-}csge}}$. We observe that clustering gives a considerable improvement in the accuracy of the rooted PageRank measure. For over 95% of the samples, the normalized absolute error is less than 0.01. On the other hand, SGE exhibits a relatively higher error: above 0.6 error for 95% of the samples. It can be verified that the normalized adjacency matrix T, for which we compute SGE and CSGE, has a much higher intrinsic dimensionality than A. The improved accuracy in $P_{\mathsf{rpr\text{-}csge}}$ may be explained by the fact that 100 dimensional embeddings on



(a) Katz measure (b) Rooted PageRank (c) Escape Probability

Figure 4: Accuracy of proximity estimation for LiveJournal dataset.



(a) Flickr (b) LiveJournal (c) MySpace

Figure 5: Missing link inference accuracy for different proximity measures.

each cluster (CSGE) captures a much larger fraction of variance compared to an $r = 100$ embedding on the entire matrix (SGE). Figure 4 (c) shows the CDFs of normalized absolute error for escape probability measure $P_{\text{ep-sge}}$ and $P_{\text{ep-csge}}$. Again, using clustering significantly improves the accuracy. This improvement is to be expected because the escape probability is based on the rooted PageRank measure. While CSGE provides a normalized absolute error less than 0.1 for more than 95% of the samples, SGE exhibits a much higher error of over 20 for the same 95% of the samples.

Summary. Our proximity estimation evaluation shows that CSGE is not only effective in approximating proximity measures on A, but also performs well on matrices with high intrinsic dimension such as the normalized adjacency matrix T. Compared with SGE, CSGE can accurately approximate the rooted PageRank and escape probability despite the general difficulty in approximating these metrics.

5.4 Missing link inference

Here, we compare the amount of correctly inferred missing links among proximity measures with and without clustering.

5.4.1 Evaluation methodology

Metrics. The accuracy of link prediction is quantified by computing *false positive rates* (FPR $= \frac{\#\text{of incorrectly predicted friend links}}{\#\text{of non-friend links}}$) , and *false negative rates* (FNR $= \frac{\#\text{ of missed friend links}}{\#\text{ of new-friend links}}$) of all user pairs in a sample set. Note that the denominator of the FPR is the number of user pairs who are not friends. This number is usually very large, *e.g.*in MySpace, we have $2 \times 10^6 \times 2 \times 10^6 - 90 \times 10^6 \approx 10^{12}$. Since we are more interested in picking up small number of correct friendships (as opposed to finding as many new friendship links as possible), we present trade-off curves with an emphasis to small FPR area by displaying the x-axis in log-scale.

Experimental setup. From the first snapshots, A_1, of all three datasets, we randomly mask half of the links as missing $\mathcal{M}$. The adjacency matrix with the remaining links is considered as an incomplete adjacency matrix with observed links $A_{\mathcal{O}}$. The trade-off curve between FPR and FNR is plotted for all user pairs given the non-zero of $A_{\mathcal{M}}$ (*i.e.*, $A_{\mathcal{M}}$ being used as the ground truth).

5.4.2 Accuracy evaluation

In Figure 5, we present the performance of $P_{\text{cn-sge}}$, $P_{\text{cn-csge}}$, $P_{\text{kz-csge}}$, and $P_{\text{kz-csge}}$. We observe that using CSGE in both proximity measures consistently outperforms SGE in all three datasets. Comparing across different measures, we observe that Katz measure with CSGE generally performs the best. For instance, in LiveJournal, for a given FPR, $P_{\text{kz-csge}}$ yields 10% or less FNR than $P_{\text{kz-sge}}$.

5.5 Link prediction evaluation

In this section, we compare the link prediction accuracy of supervised learning combined with CSGE, supervised learning with SGE, and unsupervised proximity measures.

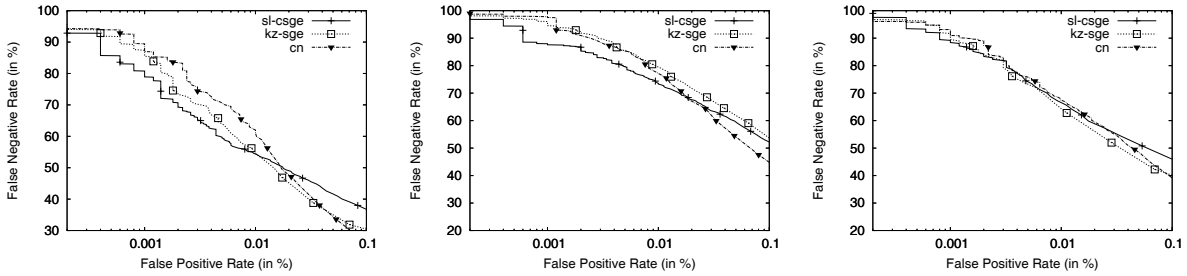
5.5.1 Evaluation methodology

Training and testing steps. From each dataset, we take three snapshots of social graphs, A_1, A_2, and A_3, with 1-2% (600,000 - 1,300,000) increment in links from the previous snapshot. To learn the model parameters in the two supervised models, we devise the training step to use a graph embedding of A_1 and minimize Eq. (9) by explicitly targeting newly formed links in A_2. In the next step of testing, we use an embedding of A_2 with the learned model parameters to predict new links in A_3. We validate the prediction against the ground truth of A_3.

Because of the size of our datasets, we randomly select a fraction of the positive and negative samples. Specifically, for training, we select 100,000 user pairs from the positive links $\mathcal{P}_1$, and 500,000 from the negative links $\mathcal{N}_1$. For testing, we pick a different sample set of the same size as before but now from $\mathcal{P}_2$ and $\mathcal{N}_2$.

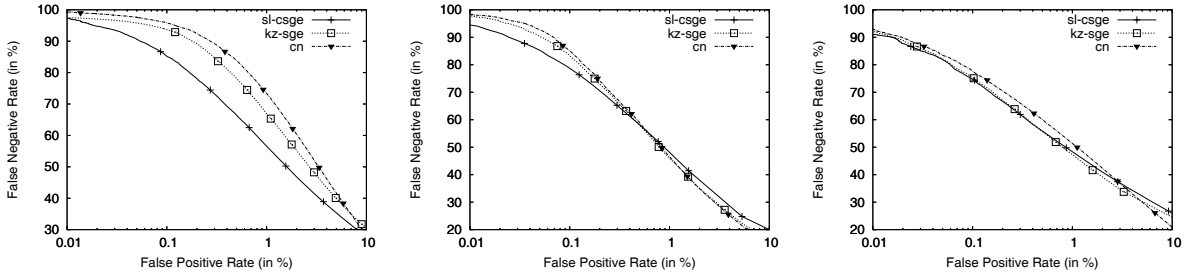
In a second experiment, we learn the model parameters based on a sample set of user pairs that are connected by *two hops*. This practical scenario focuses on link prediction for user pairs who are already close in the network (likely to be friends but not friends yet) and thus require proximity algorithms to consider only a small set of user pairs. These are user pairs that will form a triangle with a common friend, if they become friends.

Metrics. The accuracy of link prediction is quantified using the FPR and FNR introduced in Section 5.4. In the context of link prediction, the "true" and "estimated" links refer to the links in



(a) Flickr (b) LiveJournal (c) MySpace

Figure 6: Link prediction accuracy of different measures.



(a) Flickr (b) LiveJournal (c) MySpace

Figure 7: Link prediction accuracy with 2-hop scenario.

$\mathcal{P}_2$. As in missing link inference, the raw count of non-friend pairs in FPR is extremely large, and thus we present the performance measures with x-axis in log-scale.

Link predictors. We have conducted an extensive set of experiments, with numerous methods. In the interest of brevity, we present link prediction results based on three link models: P_{cn}, $P_{\text{kz-sge}}$ and the supervised $P_{\text{sl-csge}}$.

5.5.2 *Accuracy evaluation*

Clustered Spectral Graph Embedding. Figure 6 presents the link prediction accuracy of P_{cn}, $P_{\text{kz-sge}}$ and $P_{\text{sl-csge}}$. We see that, the spectral learning with CSGE performs the best in all datasets, followed by the Katz measure with SGE. For example, at an FPR of 0.001, spectral learning with CSGE reduces FNR by more than 10% in both Flickr and LiveJournal. In MySpace, the performance of clustered spectral learning is still better than the other two measures albeit by a smaller margin.

Two hop user pairs. In Figure 7, we evaluate link predictors for user pairs who are only two hops apart. Our spectral learning model again outperforms other measures by up to 20% in the Flickr dataset, and 10% in the LiveJournal dataset. For MySpace, the performance of Katz with SGE and spectral learning with CSGE is better than common neighbor by about 4%.

Note that there is no significant difference among predictors in MySpace. We speculate that this is an artifact of the data collection technique. Since this data set contains first 10 million users [40], these users have been in the network the longest time, and seem to already have a large number of friends, and thus are less active in creating new relationships. An indication of the claim is in the relatively small rate of link increase in MySpace (0.7% on average) compared to Flickr (1.2%) and LiveJournal (1.8%). Another possible reason is the large number of inter-cluster links in MySpace, as shown in Table 2. Taking into account that 48% of the links are outside the clusters, the benefit of the clustering approach may have diminished, as almost half the links in the network are not used when computing the cluster-wise embeddings.

Summary. Through the comparison between supervised and unsupervised learning methods, we verify that learning model parameters in a supervised way is indeed helpful in improving accuracy. Also, comparing the two supervised learning methods, we find sl-csge consistently out-performs sl-sge across different datasets.

6. RELATED WORK

Social network analysis. Traditionally, studies on social networks often focus on relatively small social networks (*e.g.*, [25] examine co-authorship networks with about 5000 nodes). The recent explosion of online social networks, however, has given rise to large-scale social networks with billions of links. A number of measurement studies characterize the topological structure, information propagation and user behavior of online social networks [2, 28, 29]. In [38], we explore the use of graph clustering with a few different within clusters dimensionality reduction schemes. In this paper, we develop the CSGE and demonstrate its effectiveness on large scale social network analysis tasks, and illustrate its ability to cope with temporal dynamics present in real world applications.

Proximity estimation. Given their importance in social network applications, proximity measures have received considerable research attention (*e.g.*, [16, 17, 25, 30, 36, 37, 42]). A number of proximity measures have been proposed, such as common neighbors, the Katz measure [16], rooted PageRank [25], and escape probability [42]. Despite their effectiveness in a multitude of social network applications (*e.g.*, fraud detection [8], viral marketing [15], and link prediction [25]), many existing proximity measures are computationally prohibitive for large social networks [37, 42]. In our previous work [40], we proposed scalable techniques for approximating proximity measures in online social networks based on the idea of low-rank approximation. In Section 5, we compare our new technique with [40] and show that the new approach is not only much more scalable, it is much more accurate when the proximity measures of interest have a high intrinsic dimensionality.

Link prediction. The problem with link prediction in social networks was first introduced in [24] and a number of proximity measures have been considered for link prediction in [24, 25]. A decision tree based link predictor that combines multiple proximity measures has been studied in [40]. There has also been some recent work on using supervised learning methods for link prediction

in diverse networks such as hyperlink and citation graphs [19]. As shown in Section 5.5, our supervised link prediction technique can achieve good performance in link prediction accuracy without requiring network-specific parameter configurations.

Missing link inference. The problem of missing link inference is closely related to link prediction and was also mentioned in [24,25]. Missing link inference falls into the general realm of compressive sensing, which has broad applicability in computer science [7, 11, 35, 46]. However, we are unaware of any existing compressive sensing techniques that can scale to massive social networks with millions of vertices. To this end, we focus on simple inference algorithms based on proximity measures, which are much more scalable and can directly benefit from our CSGE technique.

7. CONCLUSION

In this paper, we develop a novel dimensionality reduction technique termed *clustered spectral graph embedding (CSGE)* to embed a massive original graph into a much smaller, dense graph and show its potential in social network analysis. The existing techniques often require the underlying graph to have good low-rank approximations. CSGE, in contrast, can easily cope with massive graphs that have much higher intrinsic dimensionalities and improves both accuracy and computational efficiency.

On three large-scale real-world social network datasets (*i.e.*, Flickr, LiveJournal and MySpace) with up to 2 million vertices and 90 million edges, we employ CSGE to explore three important social network analysis tasks – for proximity estimation, we achieve up to an order of magnitude improvement in computation time and memory usage, and up to several orders of magnitude improvement in accuracy when the proximity measures of interest have high intrinsic dimensionality. For missing link inference, CSGE consistently yields better inference on unobserved links across different datasets. For link prediction, our novel *supervised learning* automatically learns parameter configurations optimal to target networks, achieving the best accuracy among proximity measures being compared.

Acknowledgments: This research is supported in part by NSF grants CCF-0916309 and CCF-1117009. We thank Vijay Erramilli and anonymous reviewers for their valuable comments.

8. REFERENCES

[1] A. Abou-Rjeili and G. Karypis. Multilevel algorithms for partitioning power-law graphs. In *Proc. of IEEE IPDPS*, 2006.

[2] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *WWW*, 2007.

[3] Alexa global top 500 sites. `http://www.alexa.com/site/ds/top_sites`.

[4] A. L. Barabasi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. *Physica A: Statistical Mechanics and its Application*, 2002.

[5] R. M. Bell, Y. Koren, and C. Volinsky. Chasing $1,000,000: How we won the Netflix Progress Prize. *Statistical Computing and Statistical Graphics Newsletter*, 2007.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 7(10), 2008.

[7] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[8] C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of interest. *Intelligent Data Analysis*, 6(3):211–219, 2002.

[9] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 2002.

[10] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.

[11] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, Apr. 2006.

[12] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. RE: Reliable Email. In *Proc. of NSDI*, 2006.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[14] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(9):1074–1085, Sep 1992.

[15] S. Hill, F. Provost, and C. Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 2006.

[16] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.

[17] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *ACM Trans. Knowl. Discov. Data*, 2007.

[18] V. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, Winter 2002.

[19] J. Kunegis and A. Lommatzsch. Learning spectral graph transformations for link prediction. In *ICML*, 2009.

[20] R. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Dept. of Comp. Sci., Aarhus Univ., 1998.

[21] R. Lehoucq, D. Sorensen, and C. Yang. *Arpack Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998.

[22] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996.

[23] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. of WWW*, pages 695–704, New York, NY, USA, 2008. ACM.

[24] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.

[25] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 2007.

[26] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of IMC*, 2004.

[27] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for Internet search. In *Proc. of HotNets-V*, 2006.

[28] A. Mislove, H. S. Koppula, K. Gummadi, P. Druschel, and B. Bhattacharjee. Growth of the Flickr social network. In *Proc. of WOSN*, 2008.

[29] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and S. Bhattacharjee. Measurement and analysis of online social networks. In *Proc. of IMC*, 2007.

[30] T. Murata and S. Moriyasu. Link prediction of social networks based on weighted proximity measures. In *Proc. of Int. Conf. Web Intelligence*, 2007.

[31] T. E. Ng and H. Zhang. Predicting internet network distance with coordinate-based appoaches. In *INFOCOMM*, 2002.

[32] Nielson Online. Fastest growing social networks for September 2008. `http://blog.nielsen.com/nielsenwire/wp-content/uploads/2008/10/press_release24.pdf`.

[33] J. A. Patel, I. Gupta, and N. Contractor. JetStream: Achieving predictable gossip dissemination by leveraging social network principles. In *Proc. of IEEE Network Computing and Applications (NCA)*, 2006.

[34] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. In *Proc. of ACM SIGCOMM 2010*.

[35] B. Recht, W. Xu, and B. Hassibi. Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization. In *Proc. of 47th IEEE Conference on Decision and Control*, Dec. 2008.

[36] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *NIPS*, 2002.

[37] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *Proc. of ICML '08*, 2008.

[38] B. Savas and I. S. Dhillon. Clustered low rank approximation of graphs in information science applications. In *SIAM Conference on Data Mining (SDM)*, pages 164–175, 2011.

[39] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[40] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *Proc. of IMC*, 2009.

[41] H. H. Song, B. Savas, T. W. Cho, V. Dave, Z. Lu, I. S. Dhillon, Y. Zhang, and L. Qiu. Clustered embedding of massive social networks. Technical Report TR–12–06, Department of Computer Science, U. of Texas at Austin, 2012.

[42] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *Proc. of KDD '07*, 2007.

[43] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based Sybil defenses. In *Proc. of ACM SIGCOMM*, 2010.

[44] D. Wagner and F. Wagner. Between min cut and graph bisection. In *Proc. of MFCS*, pages 744–750, London, UK, 1993. Springer-Verlag.

[45] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *Proc. of SIGCOMM*, 2006.

[46] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proc. of ACM SIGCOMM*, 2009.