Clustered low rank approximation of graphs in information science applications

Berkant Savas* Inderjit S. Dhillon*†

Abstract

In this paper we present a fast and accurate procedure called *clustered low rank matrix approximation* for massive graphs. The procedure involves a fast clustering of the graph and then approximates each cluster separately using existing methods, e.g. the singular value decomposition, or stochastic algorithms. The cluster-wise approximations are then extended to approximate the entire graph. This approach has several benefits: (1) important community structure of the graph is preserved due to the clustering; (2) highly accurate low rank approximations are achieved; (3) the procedure is efficient both in terms of computational speed and memory usage; (4) better performance in problems from various applications compared to standard low rank approximation. Further, we generalize stochastic algorithms to the clustered low rank approximation framework and present theoretical bounds for the approximation error. Finally, a set of experiments, using large scale and real-world graphs, show that our methods outperform standard low rank matrix approximation algorithms.

Keywords: Graph mining; low rank matrix approximation; clustering; stochastic algorithms; dimension reduction.

1 Introduction

Graphs are important modeling objects in many modern applications. Examples include: (1) analysis of online social networks such as Facebook, MySpace, LiveJournal, YouTube; (2) analysis and search on the Internet; (3) analysis of complex networks from biological systems; (4) image segmentation, etc. In all of these applications the edges of the graphs represent connections or relations or links between the vertices. The vertices could be interacting users, internet webpages, genes or pixels in a image. There are various problems of interest in these applications. A few examples are graph clustering [24, 3, 8], computation of different proximity measures (e.g. the Katz measure, escape probability, rooted page rank), link prediction [17, 20, 29, 28, 26], and modeling behavior in cellular networks [2, 22]. It is often the case that these graphs are massive in size involving over hundreds of millions of vertices. An important tool for analysis and interpretation of the data is the low rank approximation of the adjacency matrices or graph Laplacians related to the graphs at hand.

*Institute for Computational Engineering and Sciences, The University of Texas at Austin. Email: berkant@cs.utexas.edu.

†Department of Computer Science, The University of Texas at Austin. Email: inderjit@cs.utexas.edu.

In this paper we make three distinct contributions. We introduce a novel framework called *clustered low rank matrix approximation* for massive graphs. By first partitioning the vertices into a set of disjoint clusters with a fast procedure we are able to preserve important structural information of the original graph. Then we compute a low rank approximation of each cluster independently. Finally the different cluster-wise approximations are combined using an optimal projection step to obtain a low rank approximation of the entire graph, thus including connections or edges between vertices from different clusters. Recently, stochastic algorithms have been developed that use randomness to obtain low rank approximations of a given matrix [16, 10, 23]. The resulting approximations are not optimal, but the algorithms are fast and simple to implement. In addition, the approximation errors can be bounded due to their stochastic nature. As a second contribution of this paper, we extend stochastic methods to the clustered low rank approximation framework and prove corresponding theoretical bounds for the approximation errors. As a third contribution, we show that the proposed methods perform very well in practice.

The combination of clustering and low rank approximation gives a much better approximation of the original graph. In particular, information from every cluster is extracted regardless of their relative sizes. Different clusters of graphs have distinct meanings and are usually of varying sizes, and so it is beneficial to have information on all of them. A standard low rank computation is likely to only extract information from the largest or a few dominant clusters, thus filtering out smaller ones completely. In a specific web graph example, with over 400,000 vertices, the truncated *singular value decomposition* [14] (SVD) of the entire matrix gives an approximation with relative error as high as 95%, whereas our clustered algorithms result in a relative error of only 68%, see Figure 1. The ranks for the two low rank approximation are different but both approximations use the same amount of memory. In addition to this drastic improvement in accuracy, the proposed methods have usually faster execution time than state-of-the-art methods for low rank matrix approxi-

mation. The improvement in time is credited to the several but much smaller cluster-wise computations. Since the clusters are disjoint, the cluster-wise computations are independent and can be performed easily in parallel on modern architectures, such as multi-core machines. Note that the above mentioned relative errors could be considered high. But sparse matrices from information science applications, e.g. web link matrices, adjacency matrices for social networks and term-document matrices, do not have a clear low rank structure, thus can not be approximated with very low relative error, say 10^{-3} or 10^{-6}. Nonetheless, low rank approximations in these applications are extremely useful for many different reasons.

The structure of the paper is as follows. In Section 2 we introduce clustering and stochastic algorithms for low rank approximation. In Section 3 we present the framework for clustered low rank matrix approximation. We further state and prove bounds for stochastic algorithms within the clustered low rank approximation framework. Section 4 contains a set of experiments that highlight the advantages and differences of the proposed methods in comparison to existing methods. Finally in Section 5 we state our conclusion based on the proposed methods and presented experiments.

2 Preliminaries: graph clustering and stochastic algorithms

2.1 Graph clustering. A graph $G = (\mathcal{V}, \mathcal{E})$ is characterized by a set of vertices $\mathcal{V} = \{1, \cdots, m\}$ and a set of edges $\mathcal{E} = \{e_{ij} | i, j \in \mathcal{V}\}$. Let $A = [a_{ij}]$ be the corresponding $m \times m$ adjacency matrix such that a_{ij} represents the weight of edge e_{ij}. If there is no edge between vertices i and j then $a_{ij} = 0$. In modern applications, it is common that the number of vertices is large giving rise to massive adjacency matrices. An important problem in applications involving graphs is clustering of the vertices $\mathcal{V}$ into c disjoint sets $\mathcal{V}_i$, $i = 1, \cdots, c$, i.e. $\mathcal{V} = \cup_{i=1}^{c} \mathcal{V}_i$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ when $i \neq j$. There are objective functions, based on various graph cuts, e.g. *ratio cut* [15], and *normalized cut* [24], that measure the quality of a given clustering. It has been shown that these objective functions are NP-hard [31]. A common approach to tackle this difficulty is based on spectral relaxation [24, 21]. Solutions of the relaxed problem require calculation of a number of eigenvectors of the graph Laplacian. The unnormalized graph Laplacian is defined as $L = D - A$ and the normalized graph Laplacian as $\bar{L} = I - D^{-1/2} A D^{-1/2}$, where the diagonal degree matrix D is given by $d_{ii} = \sum_{i=1}^{m} a_{ij}$. Due to the high computational complexity of finding eigenvectors of large matrices, research has been done in finding scalable and often more accurate alternatives to spectral graph clustering algorithms. Examples include now widely used software packages GRACLUS [8], and METIS [1], which directly minimize various graph partition objectives using multilevel approaches without computing eigenvectors. Graph clustering is also closely related to k-means clustering. In fact the two different approaches can be viewed in a single unified framework [8]. For further details on clustering the reader may consider the survey [12].

Graph clustering in itself is not addressed in this paper, but we rely on fast and multilevel clustering algorithms for massive graphs, e.g. GRACLUS and METIS. For the remainder of the paper we assume that for a given adjacency matrix A we have a partitioning of the vertices $\mathcal{V}$ into c disjoint sets $\mathcal{V}_1, \cdots, \mathcal{V}_c$, with $m_i = |\mathcal{V}_i|$. Without loss of generality, we can assume that the vertices in $\mathcal{V}_1, \ldots, \mathcal{V}_c$ are sorted in strictly increasing order. Then the adjacency matrix will have the following form

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}, \tag{2.1}$$

where each diagonal block A_{ii}, $i = 1, \ldots, c$, is an $m_i \times m_i$ matrix that can be considered as a local adjacency matrix for cluster i. The off-diagonal $m_i \times m_j$ blocks A_{ij} with $i \neq j$, contain the set of edges between vertices belonging to different clusters.

In perfectly clusterable graphs, the off-diagonal blocks will not contain any edges, thus yielding $A_{ij} = 0$, and the graph will consist of c disconnected components. In a realistic scenario with a graph forming good clusters most of the edges will be contained within the diagonal blocks A_{ii}, while the off-diagonal blocks A_{ij} will contain only a few edges. We would like to remark that not all graphs contain natural clusters, but many graphs are clusterable to some extent [18, 19]. This is the case for graphs arising in many real-world applications, for example, see Table 1.

We will give a comparison with other methods that involving clustering in Section 3.4 after we have presented our work.

2.2 Stochastic algorithms for low rank matrix approximation. In recent years, stochastic algorithms have been employed for computing low rank matrix approximations [16, 10, 5, 23]. The benefits of these algorithms is their simple implementation, applicability on large scale problems, and existence of theoretical bounds for the approximation errors. The stochastic algorithms use randomness to construct a matrix Y that approxi-

Algorithm 1 Stochastic algorithm for low rank approximation [16].

Input: An $m \times m$ matrix A, target rank k, oversampling parameter $p \geq 1$.

Output: An orthonormal $m \times (k+p)$ matrix Q that approximates the $k+p$ dimensional dominant subspace of the range of A.

1: Generate an $m \times (k+p)$ random matrix Ω.
2: Compute $Y = A\Omega$.
3: Compute an orthonormal matrix Q whose columns form a basis for the range of Y.

mates the dominant subspace of the range[1] of a given matrix A. In general, A may be rectangular but we will focus on square (but possibly non-symmetric) matrices in this paper. For a given $m \times m$ matrix A and a target rank k in the approximation, stochastic methods generate an $m \times (k+p)$ standard Gaussian matrix[2] Ω, where p is a small oversampling parameter (typically set to 5–10). Multiplying A with the random matrix Ω we obtain $Y = A\Omega$. An orthonormal matrix Q is constructed from Y (e.g. using the QR decomposition) as a basis for the range of Y. These steps are presented in Algorithm 1. The corresponding low rank approximation is given by $A \approx \hat{A} = QQ^{\mathsf{T}}A$. By computing the SVD of $Q^{\mathsf{T}}A = \bar{W}\bar{\Sigma}\bar{V}^{\mathsf{T}}$ we get $\hat{A} = (Q\bar{W})\bar{\Sigma}\bar{V}^{\mathsf{T}} \equiv \bar{U}\bar{\Sigma}\bar{V}^{\mathsf{T}}$, which approximates the truncated SVD of A. We will now present a few theorems that bound the norm of the approximation error $\|A - \hat{A}\| = \|(I - QQ^{\mathsf{T}})A\|$ deterministically and in expectation due to the stochastic nature of the algorithm. In Section 3.2 we will present generalizations of these theorems within our clustered low rank approximation framework (to be presented in Section 3).

Let the full SVD of A be given by

$$A = U\Sigma V^{\mathsf{T}} = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^{\mathsf{T}} \\ V_2^{\mathsf{T}} \end{bmatrix}, \tag{2.2}$$

where the singular values of A are partitioned into $\Sigma_1 = \mathrm{diag}(\sigma_1, \cdots, \sigma_k)$ and $\Sigma_2 = \mathrm{diag}(\sigma_{k+1}, \cdots, \sigma_m)$. The matrices U and V are partitioned accordingly and contain the left and right singular vectors, respectively. Introduce also

$$\Omega_1 = V_1^{\mathsf{T}}\Omega, \quad \text{and} \quad \Omega_2 = V_2^{\mathsf{T}}\Omega, \tag{2.3}$$

for a given $m \times (k+p)$ matrix Ω. We have the following deterministic and probabilistic bounds.

[1]The range of A is the subspace $\{y \mid y = Ax,\ x \in \mathbf{R}^m\}$.

[2]Standard Gaussian matrix refers to a matrix with entries that are iid and normally distributed with zero mean and standard deviation of one.

THEOREM 2.1. ([5, LEM. 4.2]) *Let us be given an $m \times m$ matrix A, a target rank k, and oversampling parameter $p > 1$. For a given $m \times (k+p)$ matrix Ω compute $Y = A\Omega$. Let P_Y be the orthogonal projector on the range space of Y. Let the SVD of A be as in (2.2) and Ω_1, Ω_2 as in (2.3). Assume that Ω_1 has full rank. Then the approximation error is bounded as $\|(I - P_Y)A\|_*^2 \leq \|\Sigma_2\|_*^2 + \|\Sigma_2\Omega_2\Omega_1^{\dagger}\|_*^2$, where $\|\cdot\|_*$ denotes either the spectral norm or the Frobenius norm, and $\Omega_1^{\dagger}$ is the pseudo inverse of Ω_1.*

THEOREM 2.2. ([16, THM. 10.5 AND THM. 10.6]) *Let Ω be an $m \times (k+p)$ standard Gaussian matrix. With the notation as in Theorem 2.1 we have*

$$\mathbb{E}\|(I - P_Y)A\|_F \leq \Big(1 + \frac{k}{p+1}\Big)^{1/2} \|\Sigma_2\|_F,$$

$$\mathbb{E}\|(I - P_Y)A\|_2 \leq \Big(1 + \frac{\sqrt{k}}{\sqrt{p-1}}\Big)\|\Sigma_2\|_2 + \frac{\mathsf{e}\sqrt{k+p}}{p}\|\Sigma_2\|_F.$$

A simple but important modification, namely to compute $Y = (AA^{\mathsf{T}})^q A\Omega$ with $q > 0$, to step 2 in Algorithm 1 gives a considerable improvement in the low rank approximation, in particular when the decay of the singular values of A is slow. The introduced power parameter q is small and usually $q \lesssim 3$. This modification, with powers of AA^{T}, is closely related to block Krylov methods [27, Chp. 4, Sec. 3.3].

3 Contributions

3.1 Clustered low rank matrix approximation. We are interested in a low rank approximation of a given matrix A arising from a graph. It is important to observe that we are not interested in the best approximation of specified rank, which is given by the truncated SVD, but rather an approximation that has comparable memory requirements, is efficient to compute and that preserves important structure of the matrix. We will motivate the clustered low rank approximation by considering a special case. Let A be the adjacency matrix of a graph that consists of two perfect clusters (two connected components). Then we can write

$$A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix},$$

where A_{11} and A_{22} are (informally) the two clusters. The best rank-k approximation of A is given by $U_k\Sigma_k V_k^{\mathsf{T}}$ where U_k and V_k contain the first k left and right singular vectors, and Σ_k contains the top k singular values [11, 14]. Given the block structure of A, it is

clear that many elements of U_k and V_k will be zero[3]. For example, if $\sigma_1(A_{22}) < \sigma_k(A_{11})$, then the best low rank approximation of A has the form

$$U_k \Sigma_k V_k^\mathsf{T} = \begin{bmatrix} \bar{U}_k \\ 0 \end{bmatrix} \bar{\Sigma}_k [\bar{V}_k^\mathsf{T} \; 0],$$

where $\bar{U}_k \bar{\Sigma}_k \bar{V}_k^\mathsf{T}$ is the truncated SVD of A_{11}. No information from A_{22} is extracted. Of course, in the general case, the top k singular values might originate both from A_{11} and A_{22}. But, there will still be a non-zero pattern in both U_k and V_k, depending on which cluster the corresponding singular value belongs to.

In clustered low rank approximation we first extract the clusters and then compute low rank approximations of each cluster. In this example we may compute two rank-k truncated SVD approximations; $A_{11} \approx \bar{U}_k \bar{\Sigma}_k \bar{V}_k^\mathsf{T}$ and $A_{22} \approx \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^\mathsf{T}$. We could also use the stochastic algorithms on each block separately to obtain different (than the truncated SVD) low rank approximations. Combining the two approximations we get

$$A \approx \begin{bmatrix} \bar{U}_k & 0 \\ 0 & \tilde{U}_k \end{bmatrix} \begin{bmatrix} \bar{\Sigma}_k & 0 \\ 0 & \tilde{\Sigma}_k \end{bmatrix} \begin{bmatrix} \bar{V}_k & 0 \\ 0 & \tilde{V}_k \end{bmatrix}^\mathsf{T}. \tag{3.4}$$

Clearly, this is no longer a rank-k approximation of A but rather a rank-$(2k)$ approximation. A key observation is that the storage requirements is basically the same as for a rank-k approximation, since the zero off-diagonal blocks in the matrices of singular vectors of the right hand side of (3.4) are not stored.

In the general case we do not have a perfect clustering, resulting in non-zero off-diagonal blocks in A. It turns out that, for many matrices originating from real-world applications, in particular social networks and graphs from web and image data, the off-diagonal blocks are relatively small (e.g. in norm or number of non-zeros) compared to the diagonal blocks due to the clusterability of the data. Thus we can use the same principle as above to obtain a low rank approximation. Formally, we have

$$\begin{aligned} A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} &\approx \bar{U} \bar{S} \bar{V}^\mathsf{T} \\ &\equiv \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}^\mathsf{T}, \end{aligned} \tag{3.5}$$

where U_1 and V_1 are obtained from a low rank approximation of A_{11}, and similarly, U_2 and V_2 are obtained from a low rank approximation of A_{22}. Since U_1, V_1, U_2 and V_2 are orthonormal, it follows that $\bar{U}$ and $\bar{V}$ are orthonormal as well. Then it is easy to show that the optimal $\bar{S}$ (in a least squares sense) to approximate A is given by $\bar{S} = \bar{U}^\mathsf{T} A \bar{V}$ or $S_{ij} = U_i^\mathsf{T} A_{ij} V_j$ for $i, j = 1, 2$.

Another motivation to have block diagonal $\bar{U}$ and $\bar{V}$ matrices in a low rank approximation is given by the following observation. Assume we have a low rank approximation $A \approx U \Sigma V^\mathsf{T}$, e.g. the truncated SVD, split U and V in two parts:

$$U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}, \qquad V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}.$$

It follows that

$$A \approx U \Sigma V^\mathsf{T} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma & \Sigma \\ \Sigma & \Sigma \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}^\mathsf{T} = \bar{U} \bar{\Sigma} \bar{V}^\mathsf{T}$$

Clearly, this is not the best low rank approximation of A in terms of the subspaces of $\bar{U}$ and $\bar{V}$. The approximation can be improved by replacing $\bar{\Sigma}$ with $\bar{S} = \bar{U}^\mathsf{T} A \bar{V}$.

3.2 Stochastic algorithms for clustered low rank approximation. In this section will formally state and prove our main theorem for clustered low rank approximation using the stochastic algorithm presented in Section 2.2.

THEOREM 3.1. *Let A be an $m \times m$ matrix that is partitioned in $c \times c$ blocks as in (2.1). Further let $Y^{(i)}$ be given $m_i \times k_i$ matrices and denote by $P_{Y^{(i)}}$ the (unique) orthogonal projector on the range space of $Y^{(i)}$. Let*

$$Y = \operatorname{diag}\left(Y^{(1)}, \cdots, Y^{(c)}\right) = \begin{bmatrix} Y^{(1)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Y^{(c)} \end{bmatrix}$$

and denote P_Y to be the orthogonal projector on the range space of Y. Then we have

$$\|(I - P_Y)A\|_F^2 = \sum_{i,j=1}^{c} \|(I - P_{Y^{(i)}})A_{ij}\|_F^2,$$

$$\|(I - P_Y)A\|_2^2 \leq \sum_{i,j=1}^{c} \|(I - P_{Y^{(i)}})A_{ij}\|_2^2.$$

Proof. We will give the proof for the spectral norm bound. The equality for the Frobenius norm bound is straightforward. First, observe that P_Y is block diagonal, i.e.

$$P_Y = \operatorname{diag}(P_{Y^{(1)}}, \cdots, P_{Y^{(c)}}) = \begin{bmatrix} P_{Y^{(1)}} & & \\ & \ddots & \\ & & P_{Y^{(c)}} \end{bmatrix}.$$

[3] Unless we are in a degenerate situation where some singular values of A_{11} and A_{22} are equal, in which case a sparse basis of the corresponding invariant subspace can be chosen.

Then $\|(I-P_Y)A\|_2^2 = \|A^\mathsf{T}(I-P_Y)A\|_2$. Introduce $B = A^\mathsf{T}(I-P_Y)A$ and consider its block structure,

$$\begin{bmatrix} B_{11} & \cdots & B_{1c} \\ \vdots & \ddots & \vdots \\ B_{1c}^\mathsf{T} & \cdots & B_{cc} \end{bmatrix} \equiv \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}^\mathsf{T} \cdot \begin{bmatrix} I-P_{Y^{(1)}} & & \\ & \ddots & \\ & & I-P_{Y^{(c)}} \end{bmatrix} \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}.$$

Since B clearly is symmetric and positive semidefinite, using Proposition 8.3 in [16] and Theorem 4.2.1 in [14] it follows that

$$\|B\|_2 \le \sum_{j=1}^{c} \|B_{jj}\|_2. \tag{3.6}$$

Further, it is easy to show that the diagonal blocks[4] are given by $B_{jj} = \sum_{i=1}^{c} A_{ij}^\mathsf{T}(I-P_{Y^{(i)}})A_{ij}$. We then use the triangle inequality on $\|B_{jj}\|_2$ to obtain

$$\begin{aligned} \|B_{jj}\|_2 &\le \sum_{i=1}^{c} \|A_{ij}^\mathsf{T}(I-P_{Y^{(i)}})A_{ij}\|_2 \\ &= \sum_{i=1}^{c} \|(I-P_{Y^{(i)}})A_{ij}\|_2^2. \end{aligned}$$

Combining this bound with (3.6) completes the proof.

Theorem 3.1 is valid for any partitioning of A and any matrices $Y^{(i)}$. We will now use the stochastic algorithms from Section 2.2 to obtain low rank approximations of each diagonal block A_{ii}. Recall that after clustering most of the weight of A is within the diagonal blocks/clusters. Let each diagonal block A_{ii} have the full SVD

$$\begin{aligned} A_{ii} &= U^{(i)}\Sigma^{(i)}(V^{(i)})^\mathsf{T} \\ &= [U_1^{(i)}\ U_2^{(i)}] \begin{bmatrix} \Sigma_1^{(i)} & 0 \\ 0 & \Sigma_2^{(i)} \end{bmatrix} [V_1^{(i)}\ V_2^{(i)}]^\mathsf{T}, \end{aligned} \tag{3.7}$$

for $i = 1, \cdots, c$. We assume that each $\Sigma_1^{(i)}$ is a $k_i \times k_i$ diagonal matrix with the top k_i singular values. Introduce $m_i \times (k_i + p_i)$ matrices $\Omega^{(i)}$ and set $Y^{(i)} = A_{ii}\Omega^{(i)}$. Let also

$$\Omega_1^{(i)} = (V_1^{(i)})^\mathsf{T}\Omega^{(i)}, \quad \Omega_2^{(i)} = (V_2^{(i)})^\mathsf{T}\Omega^{(i)}, \tag{3.8}$$

for $i = 1, \cdots, c$.

[4]Expressions for the off-diagonal blocks B_{ij} are omitted since they are not used in the analysis.

THEOREM 3.2. (CLUSTERED ERROR BOUND) *Let A be a given $m \times m$ matrix with the block partitioning given in (2.1). Introduce the SVD of A_{ii} and a partitioning of the corresponding $\Sigma^{(i)}$ as in (3.7). Let k_i be a target rank for A_{ii} and p_i a corresponding oversampling parameter such that $k_i + p_i \le m_i$. Introduce matrices $\Omega^{(i)} \in \mathbb{R}^{m_i \times (k_i+p_i)}$, form $Y^{(i)} = A_{ii}\Omega^{(i)}$ and $Y = \mathrm{diag}(Y^{(1)}, \cdots, Y^{(c)})$. Form also $\Omega_1^{(i)}$ and $\Omega_2^{(i)}$ according to (3.8) and assume each $\Omega_1^{(i)}$ has full rank. Then the approximation error is bounded by*

$$\begin{aligned} \|(I-P_Y)A\|_*^2 &\le \sum_{i,j=1}^{c} \|(I-P_{Y^{(i)}})A_{ij}\|_*^2 \\ &\le \sum_{i=1}^{c} \left(\|\Sigma_2^{(i)}\Omega_2^{(i)}(\Omega_1^{(i)})^\dagger\|_*^2 + \|\Sigma_2^{(i)}\|_*^2 \right) + \\ &\qquad \sum_{i,j=1,\ i\neq j}^{c} \|A_{ij}\|_*^2, \end{aligned} \tag{3.9}$$

where the norm $\|\cdot\|_*$ denotes either the spectral or the Frobenius norm.

Proof. This bound follows trivially from the previously established results. The first inequality is the result of Theorem 3.1. In (3.9) we have used Theorem 2.1 on each term in the summation that involves a diagonal block A_{ii}. For off-diagonal blocks $\|(I-P_{Y(i)})A_{ij}\|_*^2 \le \|A_{ij}\|_*^2$, since $I - P_{Y^{(i)}}$ are orthogonal projectors.

It is now straightforward to bound the expectation of the approximation error, which we state without proof.

THEOREM 3.3. *Let $\Omega^{(i)}$ be $m_i \times (k_i + p_i)$ standard Gaussian matrices. Using the notation introduced in Theorem 3.2 we have*

$$\begin{aligned} &\mathbb{E}\|(I-P_Y)A\|_F \le \\ &\qquad \Big(\sum_i \Big(1 + \frac{k_i}{p_i+1}\Big)\|\Sigma_2^{(i)}\|_F^2 + \sum_{i\neq j} \|A_{ij}\|_F^2 \Big)^{1/2}, \\ &\mathbb{E}\|(I-P_Y)A\|_2 \le \sum_{i\neq j} \|A_{ij}\|_2 + \\ &\qquad \sum_i \Big(\Big(1 + \frac{\sqrt{k_i}}{\sqrt{p_i-1}}\Big)\|\Sigma_2^{(i)}\|_2 + \frac{\mathsf{e}\sqrt{k_i+p_i}}{p_i}\|\Sigma_2^{(i)}\|_F \Big). \end{aligned}$$

Our randomized clustered low rank approximation framework is summarized in Algorithm 2.

Observe that we can compute a deterministic clustered low rank approximation by taking U_i and V_i as the truncated left and right singular matrices of A_{ii}. In this case the approximation of the diagonal blocks is optimal, but the resulting approximation on the entire A is not optimal. In our experimental results, e.g., in Figure 2, we occasionally observed that randomized methods resulted in a slightly better approximation than the

Algorithm 2 Clustered and randomized low rank matrix approximation.

Input: An $m \times m$ matrix A, clustering of the vertices $\mathcal{V}_1, \cdots, \mathcal{V}_c$, target ranks k_i, oversampling parameters $p_i \geq 1$ for each cluster A_{ii}, and optionally a small power parameter $q > 0$.

Output: Orthonormal $m_i \times (k_i+p_i)$ matrices U_i and V_i, and $(k_i+p_i)\times(k_j+p_j)$ matrices S_{ij} for $i, j = 1, \cdots, c$ to approximate A as in (a generalized expression of) (3.5).

1: **for** $i = 1$ to c **do**
2: Generate $\Omega^{(i)}$, an $m_i \times (k_i+p_i)$ standard Gaussian matrix.
3: Compute $Y^{(i)} = A_{ii}\Omega^{(i)}$ or $Y^{(i)} = (A_{ii}A_{ii}^{\mathsf{T}})^q A_{ii}\Omega^{(i)}$.
4: Compute U_i as an orthonormal basis for the range of $Y^{(i)}$.
5: Compute V_i as an orthonormal basis for the range of $A_{ii}^{\mathsf{T}}U_i$.
6: **end for**
7: Compute $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ for $i, j = 1, \cdots, c$.

one obtained from truncated SVDs of A_{ii}. In addition we also observed that randomized algorithms tended to capture slightly more information from the off-diagonal blocks.

The bounds in the randomized approach of Theorem 2.2 clearly depend on the singular values of A. If there is a gap between the singular values σ_k and σ_{k+1} or the singular values have fast decay then the error bounds will be close to the optimal error bound for a rank-k approximation. One can observe that in the case of A having rank-k, then $Y = A\Omega$ will yield a subspace that spans the range space of A. On the other hand, if the singular values of A have slow decay, then the bounds due to the randomized approach will be practically useless. Nonetheless, these randomized algorithms perform well, in particular when used with the power approach described at the end of Section 2.2. See the experimental results in Section 4. The same is true for the clustered randomized approach. The bounds depend on the singular values of each cluster A_{ii}, but also on the amount of information that is in the off-diagonal blocks A_{ij}, $i \neq j$. We can make two observations in the clustered approach: (1) we get many more singular values, ck in total[5] compared to k singular values in the non-clustered case, (2) the bounds are increased by the presence of the off-diagonal blocks in (3.9). If we have a matrix A that forms "good" clusters, then the impact of the off-diagonal part will be smaller than the reduction that is obtained due to the inclusion of many more singular values of the diagonal blocks and ultimately give lower bounds. Also for the clustered case we observe very good performance, in particular when using a power method, i.e. $Y_i = (A_{ii}A_{ii}^{\mathsf{T}})^q\Omega_i$ as in step 3 in Algorithm 2.

[5]Here we assume for simplicity that each cluster A_{ii} is approximated with a rank-k approximation and we have c clusters. Using different ranks k_i for the different clusters ck would be replaced with $k_1 + \cdots + k_c$.

3.3 A few comparisons, observations and extensions. In this section we will discuss some of the properties of the clustered approach and differences compared to traditional approaches. Assume we have a rank-k approximation of an $m \times m$ matrix $A \approx USV^{\mathsf{T}}$ and rank-k approximation of each $A_{ii} \approx U^{(i)}S^{(i)}V^{(i)}$ that results in a rank-(ck) approximation of $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ where $\bar{U} = \mathrm{diag}(U^{(1)}, \cdots, U^{(c)})$, $\bar{V} = \mathrm{diag}(V^{(1)}, \cdots, V^{(c)})^{\mathsf{T}}$ and c is the number of clusters. Observe that there is a big difference in the ranks (k verses ck) for the two approximations. Here we focus on the memory usage in terms of number of entries to represent the approximation. Assuming the middle matrices S and S_{ii} are not diagonal, the non-clustered case uses $2mk + k^2$ entries while the clustered case uses $2mk + c^2k^2$ entries. For large scale problems the dominant part is the first term, which is the same in both expressions. But clearly when the number of clusters is large then the c^2k^2 term may become a concern.

Depending on the data set at hand, there are two obvious modifications we can make to save storage. First alternative is to omit all off-diagonal blocks A_{ij} with $i \neq j$ and only approximate the diagonal blocks A_{ii}. The size of the storage then becomes $2mk + ck^2$. Although this is a viable alternative, no interaction between the clusters can be captured. In our experiments, we have observed that the contribution from the off-diagonal blocks ranges from 3–25%, depending on how good clusters the graph forms. The second alternative is to simply store/save A_{ij} instead of computing and storing the corresponding $S_{ij} = (U^{(i)})^{\mathsf{T}} A_{ij} V^{(j)}$, which are dense. If the graph forms clusters with little interaction in between, and k is relatively large then the storage space for S_{ij} may be larger than for A_{ij}.

An important step in this framework is the existence of fast clustering algorithms. We have used both GRACLUS and METIS [8, 1] for clustering. They minimize different objective functions, and thus result in slightly different clusterings, in particular with respect to cluster sizes. However we have seen only small differences in the goodness of the resulting approximations. An interesting problem would be to formulate the clustering objective with the goal of achieving better low rank approximations.

The time complexity of the clustered low rank approximation is lower (or comparable) to non-clustered approaches. The core algorithms (truncated SVD or stochastic methods) for the low rank computation are the same in both cases. The difference is that in the proposed method with c clusters we compute c much smaller low rank approximations. In addition, since the clusters are disjoint the computations on each block are independent and can easily be performed in parallel on modern multicore machines.

3.4 Related work. There are a few publications that are similar in spirit to our proposed methods in the sense of using clustering in low rank approximations.

In Dhillon and Modha [9] *spherical k-means* clustering is applied to an $m \times n$ term document matrix A so that the columns (documents) of A are clustered into k clusters. Permuting the columns, according to cluster belongings, yields $AP = [A_1 \ \cdots \ A_k]$, where A_i denote the associated block-wise clusters. From each cluster a *concept vector* is derived that is the centroid of the column vectors from that cluster. The concept vectors are later used as a basis to obtain a low rank approximation of the term document matrix A.

In [7, 13] clustering is used to partition either the rows or columns of an $m \times n$ matrix A. Similarly as in [9] the data matrix is arranged into block-rows or block-columns according to cluster belongings. Then each cluster is approximated with a rank-1 or rank-k_i approximation using the truncated SVD. The vectors from the cluster-wise approximations are later used to compute an approximation for the entire data matrix A. These methods are used in information retrieval applications in high dimensional spaces, e.g. similarity search of patterns, content based retrieval from databases. Gallopoulos and Zeimpekis [13] also extend this clustered low rank approximation into clustered latent semantic indexing.

In a similar setting, Zhang and Zha [33] consider structural and perturbation analysis of truncated SVDs for column partitioned matrices. Necessary and sufficient conditions are given in order to reconstruct the truncated SVD of the original data matrix A from the truncated SVDs of its block-column-wise partitioning.

Boutsidis et al. [6] use clustering in the setting of column subset selection problem. Column subset selection problems are important in large scale computations when the data matrix A is streaming in a way that it is impossible or impractical to store it entirely. An approach to deal with this issue is to select certain number of columns of A and use them to compute the factors of a low rank approximation. In clustered subset selection the columns of A are clustered into l clusters, and then a certain number of columns from each cluster are selected as representatives. All selected columns are then used to obtain a low rank approximation of A.

Zhang and Kwok [32] present a procedure called clustered Nyström method. In the Nyström method approximations of the eigenfunctions to a kernel for an integral equation are sought. As in the column subset selection problem, the approach here is to chose a subset[6] of samples (called landmark points) and compute the eigenvectors of the sampled kernel matrix. The clustering of the sample points is based on the closest landmark point and an error analysis is presented for the sub-kernel matrices. The error analysis is then extended to the entire kernel. The applications in [32] are large scale manifold learning and dimensionality reduction.

There are important differences in our clustered low rank approximation with respect to the methods mentioned above. In our approach we have a clustering of the columns and rows simultaneously. This has the effect that the low rank approximations of each cluster, $U_i \Sigma_i V_i^{\mathsf{T}} \approx A_{ii}$, can be combined to approximate the entire $m \times m$ matrix A with $U = \mathrm{diag}(U_1, \cdots, U_c)$ and $V = \mathrm{diag}(V_1, \cdots, V_c)$, which are orthonormal by construction, i.e. $U^{\mathsf{T}}U = I$ and $V^{\mathsf{T}}V = I$. In the block-column-wise approximations used in [9, 7, 13, 33, 6] the left singular vectors U_i are obtained by approximating the block-columns A_i, then the approximation of A is obtained in terms of $\tilde{U} = [U_1, \cdots, U_c]$, which is not orthonormal and has to be dealt with explicitly in order to compute the core matrix in the approximation. A difference compared to the column subset selection methods, which use only part of the data to compute the low rank approximation, is that we use the entire matrix A to compute the clustered low rank approximation. We use the diagonal blocks A_{ii} to compute the U_i and V_i, but at the next step the core matrix S is generated using the off-diagonal blocks as well. A very important consequence of the block structure of $U = \mathrm{diag}(U_1, \cdots, U_c)$ is that we save in storage since only the non-zero parts are stored. On the other hand, in the related methods described above, there is no zero structure in the resulting basis matrices that allows memory storage savings, i.e. $\tilde{U} = [U_1, \cdots, U_c]$ is dense. A difference to the Nyström method is that, there it is assumed that the entire matrix is not accessible, and the kernel matrix is symmetric and positive definite. In our case the matrix A is square, sparse, not necessarily symmetric and in general indefinite. In addition we are not attempting to compute eigenvectors of A but rather to have a structured low rank approximation.

[6] The subset of samples is usually chosen randomly.

Table 1: Basic statistics of the three data sets. We also include clustering related information for a few cases.

Data set	# of vertices	# of edges	# of clusters	clustering method	edges within clusters
Image graph	10,000	1,091,910	10	GRACLUS	70.4%
Google web graph	434,818	3,419,124	40	GRACLUS	97.7%
LiveJournal graph	3,828,682	65,825,429	61	METIS	68.9%
LiveJournal graph	3,828,682	65,825,429	117	GRACLUS	66.3%

Table 2: We used three methods to compute low rank approximations. Each method has a non-clustered version and a clustered version, indicated with the additional "-C".

Name	Method
svds	truncated SVD of A
randAlg	$Y = A\Omega$
randAlgPow	$Y = (AA^{\mathsf{T}})^q A\Omega$
svds-C	truncated SVDs of A_{ii}
randAlg-C	$Y_i = A_{ii}\Omega_i$
randAlgPow-C	$Y_i = (A_{ii}A_{ii}^{\mathsf{T}})^q A_{ii}\Omega_i$

4 Experimental results

4.1 Data set, algorithms and experimental setup. We have conducted a large set of experiments on graph data from numerous and diverse applications. The presented experiments are based on: (1) Google web graph representing links between internet pages [18]; (2) a social network graph representing relationships among LiveJournal users [4, 18]; and (3) a graph obtained from an image segmentation problem. In all experiments we used only the largest connected component of each graph. Some basic statistics of the largest components of each data set are presented in Table 1. Note the large size of the LiveJournal graph.

Three different methods were used to compare the clustered verses the non-clustered approaches. These methods are: (1) truncated SVD; (2) stochastic method based on Algorithm 1; and (3) stochastic method with power, also based on Algorithm 1 but with step 2 replaced with $Y = (AA^{\mathsf{T}})^q A\Omega$. See Table 2 for a description. In the bottom second half of Table 2 we use the same set of algorithms but on each cluster A_{ii} separately. The outputs Y_i are orthonormalized and combined as described in Section 3.1 to obtain a low rank approximation of A.

Given a low rank approximation $\hat{A} = USV^{\mathsf{T}}$ of an adjacency or normalized adjacency matrix A, obtained by any of the methods in Table 2, we present the relative error based on the Frobenius norm (in %),

$$\|A - \hat{A}\|_F / \|A\|_F = \left(\|A\|_F^2 - \|S\|_F^2\right)^{1/2} / \|A\|_F.$$

since it can be computed without explicitly forming the approximation $\hat{A}$. This is not the case for the relative error in the spectral norm. In each experiment (both clustered and unclustered) we used the target ranks $k = 20, 50, 100, 150, 200$. Each rank corresponds to a point on a given curve in the plots. The oversampling and power parameters (where applicable) were set to $p = 10$ and $q = 2$. Observe that the effective rank of the approximation in the stochastic methods is $k + p$ for the non-clustered methods and $c(k + p)$ for the clustered methods. We will now present and discuss a few but highly representative plots illustrating that the clustered approaches drastically improve the quality of the low rank approximations compared to non-clustered methods.

4.2 Performance with respect to number of clusters. In Figure 1 we present the relative errors for the Google web graph and LiveJournal graph, when the low rank approximations are computed with truncated SVD (svds and svds-C). Clearly clustering and increasing the number of clusters substantially improves the quality of the approximation. Note that, in all relative error plots, the x-axis represents the memory usage (in terms of number of double precision numbers) and not the rank of the approximation. We can further observe that for a given target rank, increasing the number of clusters also increases the memory usage.

One may remark that the relative errors are not very small, but this is expected of sparse matrices from information retrieval applications, these approximations and dimensionality reduction schemes are still very useful. What we emphasize is the improvement of the clustered algorithms compared to the truncated SVD, which is optimal with respect to rank. But in our clustered case we obtain substantial improvements when comparing with respect to memory usage.

4.3 Comparison between different randomized algorithms. In Figure 2 we show performance of all six algorithms from Table 2 for a specific number of clusters. A first observation again is that all clustered methods, including the stochastic ones, give a much better performance than non-clustered methods. A second

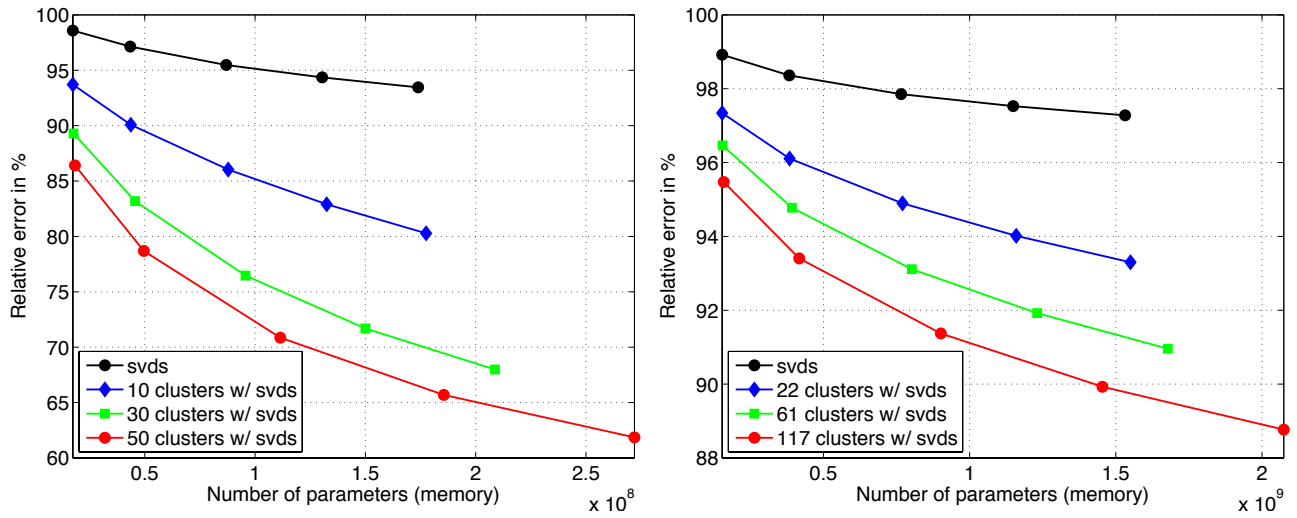

Figure 1: Relative error for the Google web graph (left) and the LiveJournal graph (right) when varying the number of clusters, indicated in the legend of corresponding graph. Clearly, for a specific target rank, increasing the number or clusters substantially improves the approximation. Note also that the proposed clustered methods significantly outperform the SVD in terms of the relative error presented with respect to memory consumption (in contrast the SVD is optimal with respect to the rank of the approximation). In the legend of the plots svds denotes a truncated SVD computation.

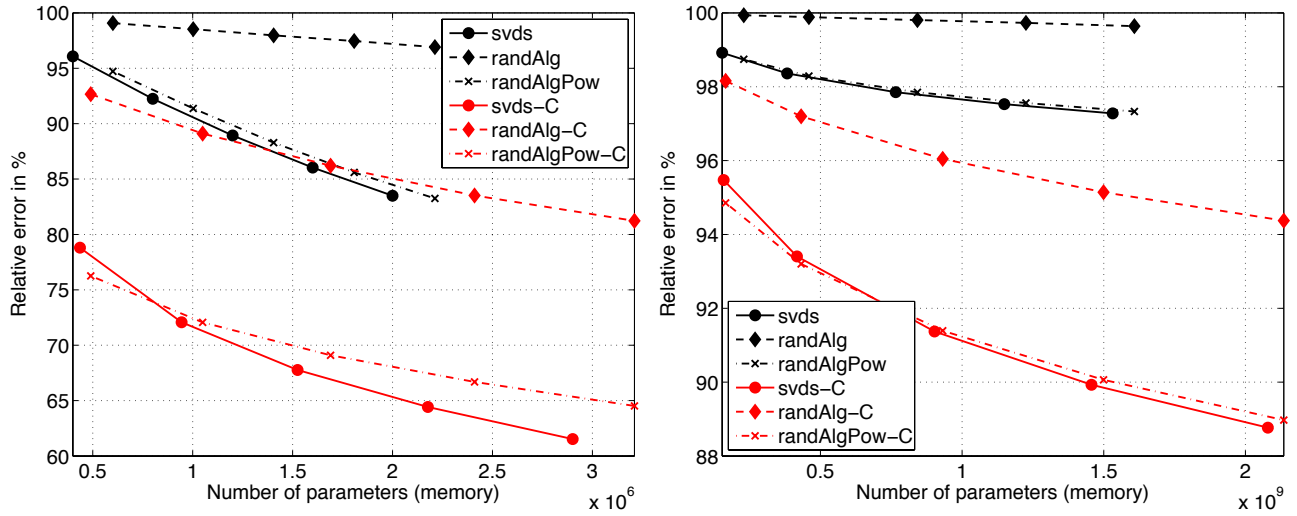

Figure 2: Relative error for the image graph (left) and the LiveJournal graph (right). In the left plot we approximated the normalized adjacency matrix $D^{-1/2}AD^{-1/2}$. The number of clusters are 10 for the image graph and 117 for the LiveJournal graph—both were obtained using GRACLUS. See Table 2 for a description of the different abbreviations.

observation is that the randAlgPow and svds methods perform similarly. The same conclusion can be drawn for the corresponding clustered versions, randAlgPow-C and svds-C. In addition, we see that the stochastic algorithm with power occasionally gives better result than the truncated SVD. Lastly, we see that the randAlg and randAlg-C methods do not perform as well as the other methods, indicating the importance of using powers, at the expense of additional computational time.

4.4 Timing comparisons. In Figure 3 we present two timing plots. The left plot of Figure 3 presents CPU

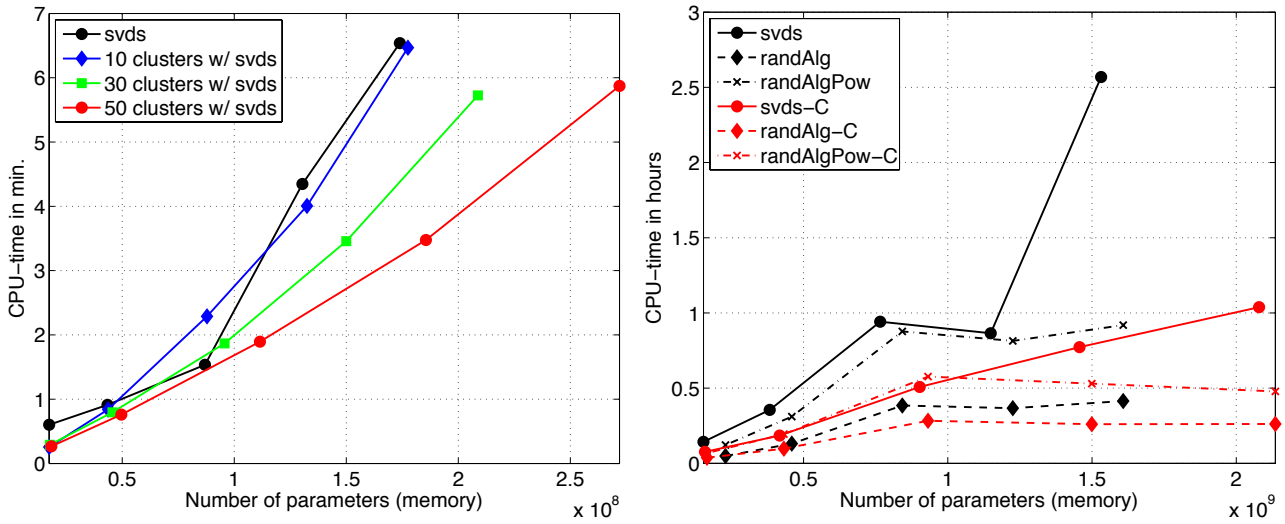

Figure 3: Left panel shows CPU run times for experiments with the Google web graph, when varying the number of clusters. Right panel shows CPU times for the LiveJournal experiments with 117 clusters and across all six methods. In all curves a dot corresponds to one target rank of $k = 20, 50, 100, 150, 200$. See Table 2 for a description of the different abbreviations.

times for the experiments in the left plot in Figure 1. Here we see that (roughly) increasing the number of clusters reduces the computational time. The is not strange since it is to be expected that the aggregated time for computing SVDs of many small matrices is smaller than computing the SVD of a single big matrix. The right plot of Figure 3 corresponds to timings for the LiveJournal experiments in Figure 2 (right plot). Also in this plot we can observe that the clustered methods are faster than the non-clustered ones. This is the case for the SVD based methods as well as the stochastic methods. Note that our timings include the time taken by the fast clustering procedures (GRACLUS or METIS).

4.5 Effect of different clusterings. Optimal clustering of a graph into c clusters is an NP-hard problem. One usually deals with this by relaxing the combinatorial problem. As a result various sub-optimal and approximate solutions are obtained. In addition, running a clustering algorithm several times yields slightly different clusters. One may ask if the clustered low rank approximations are sensitive to this kind of instabilities in the clustering step. In our experimental results, we observed minimal sensitivity to the clustering step. Although we may get different clusterings (from different runs), the changes are small and in particular the fraction of non-zeros that fall within the clusters are very similar. For example, in five different runs to partition the Google web graph into 20 clusters the mean and variance of the number of links within the clusters was 97.9% and 0.016, respectively. The corresponding relative errors in a rank-100 approximation with svds-C had mean 80.35% and variance 0.003. The variance was similar for the relative errors in the other low rank approximations. Thus, the quality of the low rank approximations appears to change only slightly with different clusterings.

4.6 Effect of clusterability. Many graphs arising in real-world applications are clusterable. From Table 1 we see that the Google web graph forms very good clusters, with only a small fraction, less than 3%, of the non-zeros in the off-diagonal blocks. For the LiveJournal graph, more than 30% of the non-zeros are contained outside the clusters (diagonal blocks). Thus, the LiveJournal graph is not as clusterable as the Google graph. Nonetheless, we still see substantial improvements by introducing clustering in the low rank approximation. The achieved relative errors in the LiveJournal experiments are higher than the Google graph experiments. The main reason for this is that the LiveJournal graph is much bigger, see Table 1. The off-diagonal blocks A_{ij}, i.e. $i \neq j$, of A are not involved when computing the cluster-wise low rank approximations. But they are involved in forming the approximation, see (3.5). The additional information obtained from the off-diagonal block in the approximation, ranged from about 3% to 25% depending on the data set, number of clusters used and the rank in the approximation.

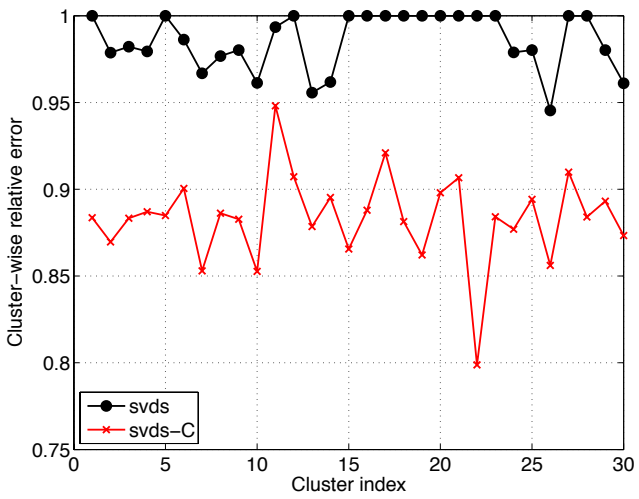

Figure 4: Information captured in a regular and clustered low rank approximations from the different clusters. The cluster-wise relative error in the y-axis is $\|A_{ii} - \hat{A}_{ii}\|_F / \|A_{ii}\|_F$.

4.7 Information content in SVD from clusters. One of the motivations to the clustered approach for low rank approximation was that the optimal (with respect to rank) truncated SVD captures information mostly from a few dominating clusters. Of course this is not desirable if one values information from all clusters. In Figure 4 we present an experiment that validates this motivation. We computed the truncated SVD of the Google web graph and calculated how much information this approximation captures from cluster A_{ii}. The number of clusters was $c = 30$ and the rank in the approximation was $k = 20$. For comparison we present a similar calculation for a clustered low rank approximation. Each data point in the plot represents the cluster-wise relative error $\|A_{ii} - \hat{A}_{ii}\|_F / \|A_{ii}\|_F$, where $\hat{A}_{ii}$ is an approximation of cluster A_{ii} obtained either with svds or svds-C. Clearly there are quite a few clusters in A from which almost no information is captured in the truncated SVD of A, for example, as seen in Figure 4, no information at all is captured for clusters 15 to 23, whereas the clustered low rank approximation has a certain level of information from each cluster. In addition, the clustered low rank approximations captures more informations from the dominant cluster compared to the truncated SVD.

4.8 Applications. We have used our proposed clustered low rank approach in two different applications. One is link prediction in large scale dynamic social networks, e.g. LiveJournal and MySpace, with several millions vertices and up to about hundred million edges, see [25] for more details. The second application is affiliation recommendation. Here we have an affiliation network, denoted $\mathcal{A}$, between a set of users and a set of groups. The task is to give users group recommendations that they would be interested in joining. In addition to the affiliation network there is a social network $\mathcal{S}$ between the users themselves. In [30] the clustering is employed in the task of group recommendation using both the affiliation network and the social network. The point we want to emphasize is that for both of these applications, using the clustered low rank application improves the link prediction and recommendation performance compared to methods that do not use clustering. A clear advantage of our proposed approach is that it allows us to easily handle social networks that are very large—millions of vertices and hundreds of millions of edges. On the other hand, traditional link prediction and affiliation recommendation algorithms do not scale to such large data sets.

5 Summary and future work

In this paper we presented a method called clustered low rank approximation of graphs. The procedure is to cluster a graph into c clusters and then compute a low rank approximation each cluster independently. The cluster-wise approximations are combined to obtain a low rank approximation of the entire adjacency matrix of the graph. We also introduced a stochastic algorithm for computing clustered low rank approximations and derived deterministic error bounds as well as expected error bounds due to the randomness. A set of experiments highlight the benefits of the clustering approach. In particular, when we consider the relative error versus the memory usage, then our clustered low rank approximation algorithms outperform the traditional ones by a large margin.

In the future we plan to develop software on parallel architectures, such as multicore machines, where we expect our parallelizable algorithms to yield significantly faster computational speeds on massive graphs with 10–100 millions of vertices. We also plan on extending the clustered low rank approach to rectangular sparse matrices.

Acknowledgments

This research was supported by NSF grants CCF-0916309 and IIS-0713142. Berkant Savas is supported by a postdoctoral scholarship from the Swedish Research Council and the Institute for Computational Engineering and Sciences, UT Austin. The authors also thank Professor Yin Zhang at the Department of Computer Science, UT Austin, for valuable discussions.

References

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

[2] T. Aittokallio and B. Schwikowski. Graph-basedmethods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7:243–255, 2006.

[3] F. R. Bach and M. I. Jordan. Learning spectral clustering. In *NIPS*, 2004.

[4] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD*, pages 44–54, 2006.

[5] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009.

[6] C. Boutsidis, J. Sun, and N. Anerousis. Clustered subset selection and its applications on IT service metrics. In *ACM CIKM*, pages 599–608, 2008.

[7] V. Castelli, A. Thomasian, and C.-S. Li. CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces. *IEEE TKDE*, 15(3):671–685, 2003.

[8] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE TPAMI*, 29(11):1944–1957, 2007.

[9] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1/2):143–175, 2001.

[10] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SICOMP*, 36(1):158–183, 2006.

[11] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[12] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[13] E. Gallopoulos and D. Zeimpekis. CLSI: A flexible approximation scheme from clustered term-document matrices. In *SDM*, 2005.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Third edition, 1996.

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

[16] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. Tech. rep. 2009–05, App. and Comp. Math, California Institute of Technology, 2009.

[17] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

[18] 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*, 6(1):29–123, 2009.

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

[20] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[22] S. Parthasarathy, S. Tatikonda, and D. Ucar. *A Survey of Graph Mining Techniques for Biological Datasets*, chapter 18, pages 547–580. Springer, 2010.

[23] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIMAX*, 31(3):1100–1124, 2009.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22:888–905, 2000.

[25] H. H. Song, B. Savas, T. W. Cho, V. Dave, Z. Lu, Y. Dhillon, Y. Zhang, and L. Qiu. Clustered embedding of massive online social networks. In preparation, UT Austin, 2010.

[26] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *IMC SIGCOMM* , pages 322–335, 2009.

[27] G.W. Stewart. *Matrix Algorithms*, volume II: Eigensystems. SIAM, 2001.

[28] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS*, 2004.

[29] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *ACM SIGKDD*, pages 747–756, 2007.

[30] V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. S. Dhillon. Scalable affiliation recommendation using auxiliary networks. *To appear in ACM TIST*, 2010.

[31] D. Wagner and F. Wagner. Between min cut and graph bisection. In *MFCS*, pages 744–750, 1993.

[32] K. Zhang and J. T. Kwok. Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE TNN*, 21:1576 –1587, 2010.

[33] Z. Zhang and H. Zha. Structure and perturbation analysis of truncated SVDs for column-partitioned matrices. *SIMAX*, 22(4):1245–1262, 2001.