SIAM J. MATRIX ANAL. APPL.
Vol. 37, No. 4, pp. 1531–1555




# CLUSTERED MATRIX APPROXIMATION*

BERKANT SAVAS† AND INDERJIT S. DHILLON‡

**Abstract.** In this paper we develop a novel clustered matrix approximation framework, first showing the motivation behind our research. The proposed methods are particularly well suited for problems with large scale sparse matrices that represent graphs and/or bipartite graphs from information science applications. Our framework and resulting approximations have a number of benefits: (1) the approximations preserve important structure that is present in the original matrix; (2) the approximations contain both global-scale and local-scale information; (3) the procedure is efficient both in computational speed and memory usage; and (4) the resulting approximations are considerably more accurate with less memory usage than truncated SVD approximations, which are optimal with respect to rank. The framework is also quite flexible as it may be modified in various ways to fit the needs of a particular application. In the paper we also derive a probabilistic approach that uses randomness to compute a clustered matrix approximation within the developed framework. We further prove deterministic and probabilistic bounds of the resulting approximation error. Finally, in a series of experiments we evaluate, analyze, and discuss various aspects of the proposed framework. In particular, all the benefits we claim for the clustered matrix approximation are clearly illustrated using real-world and large scale data.

**Key words.** matrix approximation, dimensionality reduction, low rank matrix approximation, probabilistic algorithms, graph mining, social network analysis, clustering, coclustering

**AMS subject classifications.** 15A23, 65F50, 05C50, 91D30, 91C20, 65F30, 15A99

**DOI.** 10.1137/15M1042206

## 1. Introduction.

### 1.1. Motivation.

A fundamental problem in numerous and diverse scientific applications is the problem of approximating a given matrix $A \in \mathbb{R}^{m\times n}$ by another matrix $\hat{A}$ of lower rank. The problem of best rank-$k$ matrix approximation

$$\min_{\operatorname{rank}(\hat{A})=k} \|A - \hat{A}\|_F$$

has been studied extensively in the literature, and it is well known that truncating the *singular value decomposition* (SVD) of $A$ solves this problem [16]. The solution may be written

$$\hat{A} = U_k \Sigma_k V_k^{\mathsf{T}}, \tag{1.1}$$

where $U_k \in \mathbb{R}^{m\times k}$, $V_k \in \mathbb{R}^{n\times k}$ are orthonormal matrices containing the left and right singular vectors, and $\Sigma_k \in \mathbb{R}^{k\times k}$ is a diagonal matrix with the $k$ largest singular values. A key property of the SVD is that it gives the means to analyze, interpret, and understand the original data in terms of globally optimal factors. This is a direct

*Received by the editors October 5, 2015; accepted for publication (in revised form) by P. Drineas August 4, 2016; published electronically October 27, 2016.
http://www.siam.org/journals/simax/37-4/M104220.html
**Funding:** The work of the first author was supported by Vetenskapsrådet, 2008:7145. The work of the second author was supported by the National Science Foundation (NSF), CCF-1320746, CCF-1117055.

†Department of Science and Technology, Linköping University, Norrköping 601 74, Sweden (berkant.savas@liu.se).
‡Department of Computer Science, The University of Texas at Austin, Austin, TX 78712-1188 (inderjit@cs.utexas.edu).

consequence of writing

$$\hat{A} = U_k \Sigma_k V_k^\mathsf{T} = \sigma_1 u_1 v_1^\mathsf{T} + \cdots + \sigma_k u_k v_k^\mathsf{T},$$

i.e., a sum of rank-1 matrices in terms of outer products between singular vectors weighted with the corresponding singular values.

When the matrix $A$ represents a graph, then graph partitioning and clustering analysis reveals important structural information of the underlying data. A distinction in the clustering analysis from the globally optimal factors in the SVD is that clusters are local in nature.

In this paper we will present novel low rank matrix approximation methods that incorporate the clustering information of the original data. A key feature and benefit of the resulting clustering-based methods is that they preserve local structural information.

**Metric for quality based on memory usage.** Relative reconstruction error versus the rank of the approximation is an often used metric for approximation quality. However, the main bottleneck when scaling up algorithms in applications with large amounts of data is often memory consumption and not the rank of the approximation. Then it becomes more interesting to consider relative error versus memory usage as a metric for quality. In this setting, our method produces a considerably more accurate matrix approximation with memory usage that is the same as or less than that of the truncated SVD approximation, which is optimal with respect to rank. We will illustrate this by considering a small example. There are a number of other benefits as well, which will be discussed in detail later on.

*Example* 1.1. The karate club network [40], illustrated in the left panel of Figure 1, is a well-known network example widely used to test various network analysis methods. The network models friendship relations between the 34 members of the club and is represented by a symmetric matrix $A \in \mathbb{R}^{34\times 34}$ with entries $a_{ij} = 1$ if nodes (members) $i$ and $j$ are connected, and $a_{ij} = 0$ otherwise.

Let $V_k \Lambda_k V_k^\mathsf{T}$ be the best rank-$k$ approximation of $A$, obtained using the *spectral decomposition*,[1] and calculate the relative reconstruction error

$$\frac{\|A - V_k \Lambda_k V_k^\mathsf{T}\|_F}{\|A\|_F}$$

for $k = 1, 2, \ldots, 5$. The memory usage (in floating point numbers) for each rank-$k$ approximation, accounting for symmetry, is simply $34 \cdot k + k$. The resulting approximation error is plotted against memory usage in the right panel of Figure 1. In the same figure, we have plotted the relative reconstruction errors for approximations obtained with our method based on partitioning the graph into three clusters (details given later). The clustering is shown in Figure 1 and obtained using spectral graph partitioning [10, 28].

The results shown in Figure 1 are clear; approximations obtained with our method are more accurate by a large margin than truncated spectral approximation. Observe that the comparison is made with respect to memory consumption and not rank of the approximation. Consider in particular the approximation error for $V_4 \Lambda_4 V_4^\mathsf{T}$ (fourth data point); this approximation uses 140 floating point numbers and has

[1] For symmetric matrices the spectral factorization may be used to compute the best rank-$k$ approximation.

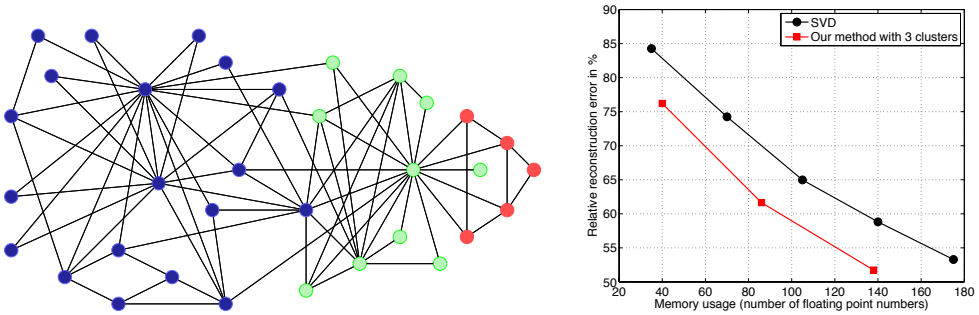


FIG. 1. *Left: The karate club network partitioned into three clusters using spectral graph partitioning. Cluster affiliation is indicated by the coloring of the nodes. Right: Relative reconstruction error versus memory consumption. We see that our method produces considerably lower reconstruction error than the spectral approximation, which is optimal with respect to rank. Observe in particular that the two lower data points in our method give more accurate approximations with less memory usage than the truncated spectral approximations.*

58.8% reconstruction error. Compare this with results of the third approximation (third data point) from our method. This approximation uses 138 floating point numbers and has 51.7% reconstruction error. A similar comparison can be made with the rank-3 spectral approximation and the second approximation from our method. In this case we have 65% reconstruction error using 105 floating point numbers for the spectral solution to be compared with 61.6% reconstruction error using 86 floating point numbers.

The algorithms we propose in this paper may be used in a wide range of applications, in particular those in information science. A few examples are *information retrieval using latent semantic indexing* [11, 4], *link prediction*, and *affiliation recommendation in social networks* [26, 22, 27, 35, 34, 36, 32]. A particular interest in network applications is to analyze network features as *centrality*, *communicability*, and *betweenness*. These and similar features may be expressed as a matrix function $f(A)$ in terms of the network's adjacency matrix $A$ [17]. Often, these functions contain matrix powers $A^p$. For small networks these powers can be computed explicitly; however, for large scale networks, computing $A^p$ is not practical due (mostly) to memory constraints. Low rank approximations provide the means to approximate these quantities and scale up algorithms. Using $A \approx VDV^\mathsf{T}$ with $V^\mathsf{T}V = I$, we can employ the approximation

$$A^p \approx \left(VDV^\mathsf{T}\right)^p = VD^pV^\mathsf{T},$$

where the power of a relatively small matrix $D$ is taken.

**1.2. Contributions.** There are three main contributions in this paper:

We propose a general and flexible framework for clustered matrix approximation methods. The methods can be applied on square (symmetric and nonsymmetric) and rectangular matrices and involve four steps: (1) a clustering or coclustering step so that the rows and columns of a given matrix are partitioned into a number of groups; (2) reordering the rows and columns according to cluster affiliation and extracting sufficiently dense blocks; (3) computing low rank approximations of these dense blocks; and (4) combining the blockwise approximations into an approximation of the entire matrix.

Computing truncated SVD approximations is relatively expensive, and in some circumstances one may wish to trade off computation time against some slight loss in

accuracy. Probabilistic algorithms for matrix approximation [21] give the means for this kind of trade-off. In this paper we develop and detail the use of such algorithms in the clustered low rank approximation framework. We also derive a few deterministic and probabilistic approximation error bounds.

An extensive and systematic set of experiments constitute the last contribution of this paper. Here, we investigate a number of aspects of the proposed methods that are important from both a practical and a theoretical point of view. The experiments clearly illustrate the benefits of the presented clustered matrix approximation framework.

**1.3. Outline.** The outline of the paper is as follows. In section 2 we present background material that serves as a foundation for the development of our methods. Section 3 contains the following main contributions of this paper: development of general clustered low rank matrix approximation methods that are applicable to both square and rectangular matrices, and derivation of deterministic and probabilistic error bounds. Section 4 contains an extensive set of numerical experiments that evaluate the proposed clustered low rank approximation methods using real-world and large scale data sets. In section 5 we present some related work. Finally, in section 6 we present our conclusions.

**1.4. Notation.** Matrices will be denoted by capital roman letters, e.g., $A$, $U$, $V$, $A_i$, or $A_{ij}$. Lowercase letters in the middle of the alphabet, e.g., $i$, $k$, $m$, $n$, will (often) denote subscript integers. Calligraphic letters will denote sets, e.g., $\mathcal{V}$. For a given matrix $U$, its column space will be denoted by range$(U)$. We define diag$(A_1, \ldots, A_k)$ as the $k \times k$ block matrix with $A_1$ to $A_k$ as diagonal blocks. We will use orth$(X)$ to denote[2] an orthonormal basis for range$(X)$. Additional notation will be described as it is introduced.

**2. Preliminaries.** The methods we develop in this paper rely on a number of concepts: efficient graph clustering; bipartite graph coclustering; low rank matrix approximations; and stochastic methods for low rank matrix approximations. In this section we will introduce these concepts and related background that is necessary to develop and present our framework.

**2.1. Graph clustering and bipartite graph coclustering.** A key step in the algorithms we develop is to extract (local) structural information of a given matrix. By considering a square $m \times m$ matrix $A = [a_{ij}]$ as a graph's adjacency matrix, we can obtain (structural) cluster information by partitioning the graph's vertices. Formally, a graph $G = (\mathcal{V}, \mathcal{E})$ is characterized by a set of vertices $\mathcal{V} = \{\nu_1, \ldots, \nu_m\}$ and a set of edges $\mathcal{E} = \{e_{ij} \mid \nu_i, \nu_j \in \mathcal{V}\}$. Elements $a_{ij}$ represent the edge weighs $e_{ij}$. If there is no edge between $\nu_i$ and $\nu_j$, then $a_{ij} = 0$. The clustering problem amounts to partitioning the vertices $\mathcal{V}$ into $c$ disjoint sets $\mathcal{V}_1, \ldots, \mathcal{V}_c$. Extensive research has been conducted to develop the theory [39, 20, 31, 37, 28, 13, 18, 38] (to mention a few) and efficient graph clustering software packages such as GRACLUS [13] and METIS [1]. In modern applications, it is common that the number of vertices is large, giving rise to massive (sparse) adjacency matrices.

From now on, we assume that we can partition the graph and obtain $c$ disjoint sets $\mathcal{V}_1, \ldots, \mathcal{V}_c$, with $m_i = |\mathcal{V}_i|$. Without loss of generality, we can assume that vertices in

[2]For $X$ with full column rank we may use the QR factorization $X = QR = [Q_1\ Q_2]\left[\begin{smallmatrix} R_1 \\ 0 \end{smallmatrix}\right]$ and set orth$(X) = Q_1$.

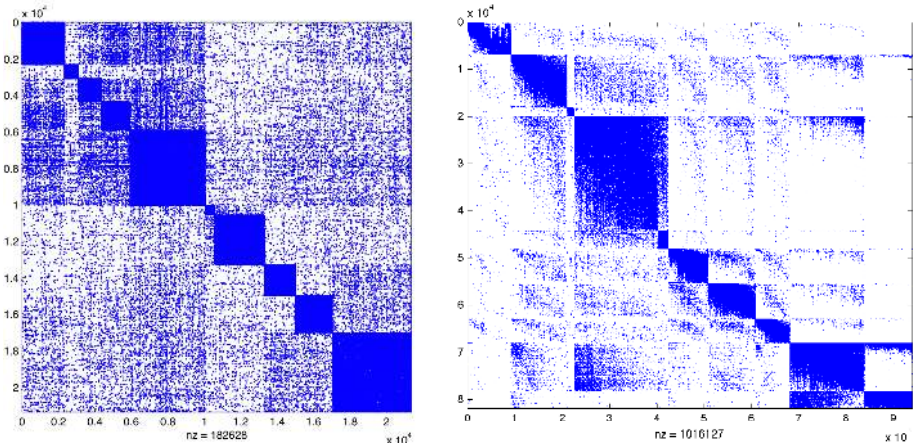


FIG. 2. *Left: Spy plot of the clustering structure of the arXiv condensed matter collaboration network* [23]. *The graphs contains* 21,363 *vertices and* 182,628 *edges.* 79.8% *of the edges are within the ten diagonal blocks. Right: Spy plot of the coclustering structure of the Notre Dame bipartite graph between* 81,823 *movies and* 94,003 *actors* [3]. *The associated matrix has* 1,016,127 *nonzeros, and* 83.4% *of the nonzeros are contained within the ten diagonal blocks.*

$\mathcal{V}_1, \ldots, \mathcal{V}_c$ are sorted in strictly increasing order. Then the matrix will have the 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 $A_{ii}$ is an $m_i \times m_i$ matrix that may be considered as a local adjacency matrix for cluster $i$. The off-diagonal $m_i \times m_j$ blocks $A_{ij}$ contain the set of edges between vertices belonging to different clusters.

A graph consisting of $c$ disconnected components will give a perfect clustering. The resulting block partitioning of $A$ will contain nonzero elements only in the diagonal blocks $A_{ii}$. In a realistic scenario, however, 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 only contain a small fraction of the edges.[3] An example of block partitioning revealing the underlying cluster structure of a graph is given in the left panel of Figure 2. In section 4.2 and Table 1 we give a more detailed presentation regarding clustering structure in the data.

Similarly, a rectangular $m \times n$ matrix $B = [b_{ij}]$ may be used to represent a bipartite graph. Consequently, coclustering [12, 43] may be used to extract corresponding structural information. Formally, a bipartite graph $G = (\mathcal{R}, \mathcal{C}, \mathcal{E})$ is characterized by two sets of vertices, $\mathcal{R} = \{r_1, \ldots, r_m\}$ and $\mathcal{C} = \{c_1, \ldots, c_n\}$, and a set of (undirected) edges, $\mathcal{E} = \{e_{ij} \mid r_i \in \mathcal{R},\ c_j \in \mathcal{C}\}$. The elements $b_{ij}$ represent the edge weights $e_{ij}$ and, if $b_{ij} = 0$, then there is no edge between $r_i$ and $c_j$. Assuming we partition $\mathcal{R}$ into $r$ row clusters and $\mathcal{C}$ into $c$ column clusters, the coclustering will yield disjoint row sets $\mathcal{R}_1, \ldots, \mathcal{R}_r$ and disjoint column sets $\mathcal{C}_1, \ldots, \mathcal{C}_c$. Again, without loss of generality, we

[3] We would like to remark that not all graphs contain natural clusters, but many graphs are clusterable to some extent [25, 24]. This is the case for graphs arising in many real-world applications.

may rearrange the row and column vertices according to cluster affiliation to obtain

$$B = \begin{bmatrix} B_{11} & \cdots & B_{1c} \\ \vdots & \ddots & \vdots \\ B_{r1} & \cdots & B_{rc} \end{bmatrix}, \tag{2.2}$$

where block $B_{ij}$ contains edges between row vertices of $\mathcal{R}_i$ and column vertices of $\mathcal{C}_j$. The right panel of Figure 2 shows the coclustering structure of a bipartite graph. It is clear here as well that the diagonal blocks are much denser than off-diagonal blocks.

We would like to remark that the coclustering methods in [12, 43] result in the same number of clusters for $\mathcal{R}$ as well as for $\mathcal{C}$. However, a straightforward modification[4] can be employed to obtain different numbers of clusters in $\mathcal{R}$ and $\mathcal{C}$. Alternatively, the block structure in (2.2) may be extracted by casting the problem as a regular graph partitioning problem. One may either consider the (symmetric) adjacency matrix

$$\begin{bmatrix} 0 & B^{\mathsf{T}} \\ B & 0 \end{bmatrix} \tag{2.3}$$

of the bipartite graph $G$ or form symmetric similarity matrices $A_1 = BB^{\mathsf{T}}$ and $A_2 = B^{\mathsf{T}}B$, and subsequently apply regular graph partitioning algorithms on these matrices to obtain independent row and column clusterings.

**2.2. Probabilistic methods for low rank matrix approximation.** In recent years, randomized algorithms have been employed for computing low rank matrix approximations [21, 15, 7, 29]. These algorithms have several benefits: they produce remarkably good results; they are simple to implement; they are applicable on large scale problems; and they have theoretical bounds for the approximation errors. The algorithms use randomness to construct a matrix $Y$ that approximates a low-dimensional dominant subspace of range$(A)$.

For a given $m \times n$ matrix $A$ and a target rank $k$ in the approximation, probabilistic methods generate an $n \times (k + p)$ standard Gaussian matrix[5] $\Omega$, where $p$ is a small oversampling parameter (typically set to 5–10). Multiplying $A$ with the random matrix $\Omega$, we obtain $Y = A\Omega$. Subsequently an orthonormal basis is calculated by $Q = \text{orth}(Y)$. 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 two theorems that bound the norm of the approximation error $\|A - \hat{A}\| = \|(I - QQ^{\mathsf{T}})A\|$ deterministically and in expectation due to the randomized nature of the algorithm. In section 3.3 we will present generalizations of these theorems within our clustered low rank approximation framework.

Let the full SVD of $A$ be given by

$$A = U\Sigma V^{\mathsf{T}} = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^{\mathsf{T}} \\ V_2^{\mathsf{T}} \end{bmatrix}, \tag{2.4}$$

where the singular values of $A$ are partitioned into $\Sigma_1 = \text{diag}(\sigma_1, \ldots, \sigma_k)$ and $\Sigma_2 = \text{diag}(\sigma_{k+1}, \ldots, \sigma_n)$. The matrices $U$ and $V$ are partitioned accordingly. Introduce

[4] For example, by running the $k$-means algorithm independently and with a different number of clusters on the two blocks of equation (12) in [12].

[5] "Standard Gaussian matrix" refers to a matrix with entries that are i.i.d. and normally distributed with zero mean and standard deviation of one.

**Algorithm 1** Randomized range finder [21].

**Input:** An $m \times n$ 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 range$(A)$.

1: Generate an $n \times (k+p)$ random matrix $\Omega$.
2: Compute $Y = A\Omega$.
3: Compute $Q = \mathrm{orth}(Y)$.

also

$$\Omega_1 = V_1^{\mathsf{T}}\Omega \quad \text{and} \quad \Omega_2 = V_2^{\mathsf{T}}\Omega \tag{2.5}$$

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

THEOREM 2.1 (see [7, Lem. 4.2]). *Let us be given an $m \times n$ matrix $A$, a target rank $k$, and oversampling parameter $p > 1$. For a given $n \times (k+p)$ matrix $\Omega$ compute $Y = A\Omega$. Let $P_Y$ be the orthogonal projector onto range$(Y)$. Let the SVD of $A$ be as in (2.4), and let $\Omega_1$, $\Omega_2$ be as in (2.5). Assume that $\Omega_1$ 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 pseudoinverse of $\Omega_1$.*

THEOREM 2.2 (see [21, Thms. 10.5 and 10.6]). *Let $\Omega$ be an $n \times (k+p)$ standard Gaussian matrix. With the notation as in Theorem 2.1 we have*

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

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

*where $\mathsf{e}$ is the base of the natural logarithm.*

A simple but important modification to step 2 in Algorithm 1, namely, to compute $Y = (AA^{\mathsf{T}})^q A\Omega$ with integer $q > 0$, 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 $q \lesssim 3$. The modification with powers of $AA^{\mathsf{T}}$ is closely related to the power or subspace iteration; see [21, Alg. 4.3] and [5, sec. 3.3.5].

**3. Clustered low rank matrix approximations.** The main goal of this section is to develop a framework for memory efficient matrix approximations that preserve important structural information of the data. In section 3.1, we initiate our presentation by extending the clustered low rank approximation of square matrices (graphs) [30] to rectangular matrices (bipartite graphs). The extension is straightforward and will serve as a foundation for the content of sections 3.2 and 3.3.

**3.1. Diagonal dense block structure.** Let the matrix $A \in \mathbb{R}^{m \times n}$ have the following $c \times c$ block structure:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}, \tag{3.1}$$

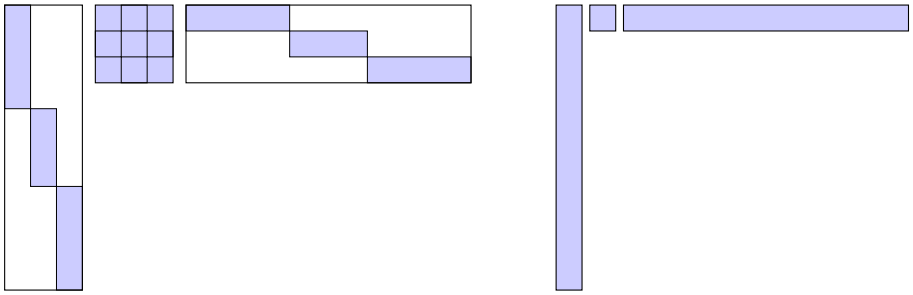

FIG. 3. *Graphical comparison between the clustered matrix approximation* $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ *in* (3.3) *using* $c = 3$ *and a regular truncated SVD approximation in* (1.1). *The memory usage in* $\bar{U}$ *and* $\bar{V}$ *is restricted only to the diagonal (shaded) blocks.*

where $A_{ij} \in \mathbb{R}^{m_i \times n_j}$. For a square matrix the block partitioning is given from a clustering of the associated graph. If $A$ is rectangular, the block partitioning is obtained by coclustering the associated bipartite graph. We assume for both cases that the diagonal blocks $A_{ii}$ are much denser in terms of nonzero entries than the off-diagonal blocks $A_{ij}$, as in Figure 2. In section 3.2 we will generalize this to block partitionings where off-diagonal blocks may also be dense. Compute low rank approximations of the diagonal blocks using the truncated SVD[6]

$$A_{ii} \approx \hat{A}_{ii} = U_i \Sigma_i V_i^{\mathsf{T}} \quad \text{with } \operatorname{rank}(A_{ii}) = k_{ii}, \text{ and } i = 1, \ldots, c. \tag{3.2}$$

We can then construct an approximation of $A$ as

$$A \approx \operatorname{diag}(U_1, \ldots, U_c) \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix} \operatorname{diag}(V_1, \ldots, V_c)^{\mathsf{T}} \equiv \bar{U}\bar{S}\bar{V}^{\mathsf{T}}, \tag{3.3}$$

where $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$. This choice of $\bar{S}$ yields the optimal approximation of $A$ in the least squares sense for the given orthonormal $\bar{U} = \operatorname{diag}(U_1, \ldots, U_c)$ and $\bar{V} = \operatorname{diag}(V_1, \ldots, V_c)$. For example, with $c = 3$ clusters we obtain

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \approx \begin{bmatrix} U_1 & 0 & 0 \\ 0 & U_2 & 0 \\ 0 & 0 & U_3 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix}^{\mathsf{T}}.$$

Observe that off-diagonal blocks are approximated as well: $A_{ij} \approx U_i S_{ij} V_j^{\mathsf{T}}$. These approximations are probably not good since they use $U_i$ and $V_j$ that capture information from diagonal blocks. This, however, is not a problem since off-diagonal blocks contain little information. In the ideal case we would have $A_{ij} = 0$, and this is almost the case for matrices from many real-world applications. We will also address this in the next section. Figure 3 shows a graphical illustration of the structure of the clustered low rank approximation using three clusters in comparison with the structure of the regular truncated SVD.

All steps in the process are described in Algorithm 2.

For cases with a small number of clusters $c$ and small to moderate $k_{ii}$, most of the memory usage in the clustered matrix approximation is consumed by $\bar{U}$ and $\bar{V}$.

[6] Clearly, if $A_{ii}$ are symmetric, then the spectral factorization should be used in this step.

**Algorithm 2** Clustered matrix approximation with diagonal block structure.

**Input:** $A$, number of clusters $c$, and ranks $k_{11}, \ldots, k_{cc}$.

**Output:** $U_i$, $V_j$, $S_{ij}$ for $i, j = 1, \ldots, c$.

1: Partition $A$ into $c \times c$ blocks by using a clustering or coclustering algorithm.
2: Reorder rows and columns of $A$ according to their cluster belonging in order to obtain a matrix with a block structure as in (3.1).
3: Compute low rank approximations of the diagonal blocks according to (3.2).
4: Set $S_{ii} = \Sigma_i$ and compute $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ when $i \neq j$ to obtain $\bar{S}$.

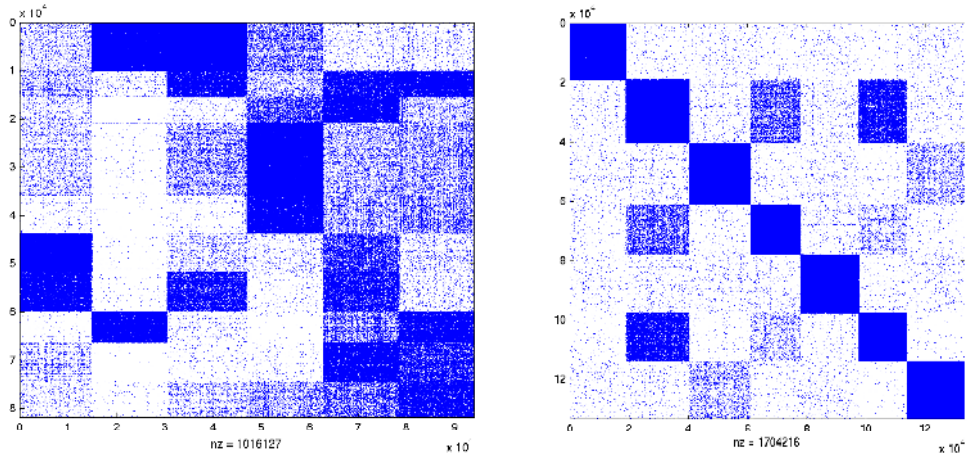


FIG. 4. *Left: Cluster structure of a rectangular matrix (bipartite graph) with* 10 *row clusters and* 6 *column clusters that are obtained independently. Right: Clustering of a matrix (graph) into* 7 *clusters. In both panels there is more than one dense block in a block row or block column.*

In such cases it makes little sense to set some or all off-diagonal $S_{ij} = 0$ in order to achieve additional memory savings. However, for cases with a large number of clusters $c$ and large ranks $k_{ii}$ the memory consumption of $\bar{S}$ may become a significant fraction of the total memory usage. This might motivate setting some off-diagonal $S_{ij} = 0$. Observe that setting all $S_{ij} = 0$ in (3.3) would result in only approximating the diagonal blocks $A_{ii}$, and in effect discarding all off-diagonal $A_{ij}$.

**3.2. Nondiagonal dense block structure.** Analysis of clustering results reveals structure of the data in terms of dense diagonal blocks, but often dense off-diagonal blocks are revealed as well. A more general kind of block structure is revealed when coclustering the rows and columns of a rectangular matrix independently, in particular when using a different number of row and column clusters. Both of these scenarios are illustrated in Figure 4.

The aim now is to construct a low rank matrix approximation $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ that contains structural information and can be stored efficiently. Using block diagonal $\bar{U} = \mathrm{diag}(U_1, \ldots, U_r)$ and $\bar{V} = \mathrm{diag}(V_1, \ldots, V_c)$, we can achieve both of these goals. In order to preserve the structure of the data, we must allow for all (sufficiently) dense blocks in the block partitioning to contribute to the approximation. Thus, every $U_i$ must contain column space contribution from all the dense blocks in the $i$th block row and, similarly, every $V_j$ must contain row space contribution from all the dense blocks in the $j$th block column. Clearly, these requirements are fulfilled

for the approximation in (3.3). In the following, we will determine orthonormal $U_i$ and $V_j$, leading to orthonormal $\bar{U}$ and $\bar{V}$. As a consequence we can calculate $\bar{S}$ with matrix multiplications only. However, with small changes, it is possible to formulate the entire process with blocks $U_i$ and $V_j$ that are not orthonormal.

**3.2.1. Specific example.** Partition a matrix $A$ into a $3 \times 4$ block structure

$$A = \begin{bmatrix} [A_{11}] & A_{12} & [A_{13}] & A_{14} \\ A_{21} & A_{22} & A_{23} & [A_{24}] \\ A_{31} & [A_{32}] & [A_{33}] & A_{34} \end{bmatrix}. \tag{3.4}$$

Assume that blocks $A_{11}$, $A_{13}$, $A_{24}$, $A_{32}$, $A_{33}$ (explicitly in brackets in (3.4)) are considered to be dense. Introduce the set $\mathcal{S} = \{(1,1),(1,3),(2,4),(3,2),(3,3)\}$ with pairwise integers that indicate the dense blocks. We will compute low rank approximations

$$A_{ij} \approx \hat{A}_{ij} = U_{ij}\Sigma_{ij}V_{ij}^{\mathsf{T}}, \text{ with } \operatorname{rank}(A_{ij}) = k_{ij}, \quad (i,j) \in \mathcal{S}, \tag{3.5}$$

and use them to obtain a clustered low rank approximation of the form

$$A \approx \begin{bmatrix} U_1 & 0 & 0 \\ 0 & U_2 & 0 \\ 0 & 0 & U_3 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \end{bmatrix} \begin{bmatrix} V_1 & 0 & 0 & 0 \\ 0 & V_2 & 0 & 0 \\ 0 & 0 & V_3 & 0 \\ 0 & 0 & 0 & V_4 \end{bmatrix}^{\mathsf{T}} \equiv \bar{U}\bar{S}\bar{V}^{\mathsf{T}},$$

where the $U_i$ and $V_i$ are orthonormal and the $S_{ij}$ are determined in an optimal least squares sense. Using results from (3.5) we compute or set

$$U_1 = \operatorname{orth}([U_{11}\ U_{13}]), \quad U_2 = U_{24}, \quad U_3 = \operatorname{orth}([U_{32}\ U_{33}])$$

to obtain $\bar{U}$, and similarly blocks of $\bar{V}$ are given by

$$V_1 = V_{11}, \quad V_2 = V_{32}, \quad V_3 = \operatorname{orth}([V_{13}\ V_{33}]), \quad V_4 = V_{24}.$$

Observe that all dense blocks directly contribute information to the approximation: $U_1$ contains information from both $A_{11}$ and $A_{13}$; $U_3$ contains information from both $A_{32}$ and $A_{33}$; $V_3$ contains information from both $A_{13}$ and $A_{33}$. Given $U_1$, $U_2$, $U_3$, and $V_1, \ldots, V_4$, optimal $S_{ij}$ are obtained, as previously, with $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$.

*Remark* 3.1. Observe that one may consider computing, e.g., $U_1$ from a single SVD of $[A_{11}\ A_{13}]$ instead of two separate SVDs of $A_{11}$ and $A_{13}$, respectively. This alternative approach, however, does not necessarily take into account the structure of the data that we want to preserve. For example, an SVD approximation of $[A_{11}\ A_{13}]$ may contain contributions only from $A_{11}$. The procedure we presented above will always extract a certain amount of information from $A_{11}$ and a certain amount of information from $A_{13}$, thus preserving the inherent clustering structure of the data.

**3.2.2. General description.** We will now state the clustered low rank approximation with arbitrary $r$ and $c$. Let an $m \times n$ matrix $A$ be partitioned into $r \times c$ blocks,

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{r1} & \cdots & A_{rc} \end{bmatrix}. \tag{3.6}$$

Let $\mathcal{S}$ denote a set with pairwise indices that specify the dense blocks of $A$. Each $A_{ij}$ with $(i,j) \in \mathcal{S}$ will make a direct contribution in the approximation of $A$. We introduce $\mathcal{R}_i = \{(i, c_{i,1}), \ldots, (i, c_{i,|\mathcal{R}_i|})\}$ with dense block indices from the $i$th block row, and similarly $\mathcal{C}_j = \{(r_{j,1}, j), \ldots, (r_{j,|\mathcal{C}_j|}, j)\}$ with dense block indices from the $j$th block column. Clearly, it holds that $\mathcal{S} = \cup_i \mathcal{R}_i = \cup_j \mathcal{C}_j$. Prescribe now the ranks $k_{ij}$ and compute low rank approximations

$$A_{ij} \approx \hat{A}_{ij} = U_{ij}\Sigma_{ij}V_{ij}^{\mathsf{T}}, \quad (i,j) \in \mathcal{S}, \quad \text{with } \operatorname{rank}(\hat{A}_{ij}) = k_{ij} \tag{3.7}$$

using the truncated SVD. The next step is to compute blocks in $\bar{U} = \operatorname{diag}(U_1, \ldots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \ldots, V_c)$ according to

$$U_i = \operatorname{orth}([U_{ic_{i,1}} \ \cdots \ U_{ic_{i,|\mathcal{R}_i|}}]), \qquad \text{where } (i,c_{i,1}), \ldots, (i,c_{i,|\mathcal{R}_i|}) \in \mathcal{R}_i, \tag{3.8}$$

$$V_j = \operatorname{orth}([V_{r_{j,1}j} \ \cdots \ V_{r_{j,|\mathcal{C}_j|}j}]), \qquad \text{where } (r_{j,1},j), \ldots, (r_{j,|\mathcal{C}_j|},j) \in \mathcal{C}_j. \tag{3.9}$$

The clustered approximation then takes the form $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ or, in block form,

$$\begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{r1} & \cdots & A_{rc} \end{bmatrix} \approx \operatorname{diag}(U_1, \ldots, U_r) \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{r1} & \cdots & S_{rc} \end{bmatrix} \operatorname{diag}(V_1, \ldots, V_c)^{\mathsf{T}}. \tag{3.10}$$

The blocks of $\bar{S}$ are determined by $S_{ij} = \Sigma_{ij}$ when $(i,j) \in \mathcal{S}$ and $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ otherwise. The blockwise approximations become $A_{ij} = U_i S_{ij} V_j^{\mathsf{T}}$ for all $i$ and $j$. The entire process is described in Algorithm 3. It is clear that (3.10) generalizes the clustered matrix approximation with diagonal blocks in section 3.1.

**Algorithm 3** Clustered matrix approximation with nondiagonal block structure.

**Input:** $A$, number of row clusters $r$, number of column clusters $c$,

**Output:** Block matrices that form the clustered low rank approximation: $U_1, \ldots, U_r$, $V_1, \ldots, V_c$, and $S_{ij}$ with $i = 1, \ldots, r$ and $j = 1, \ldots, c$.

1: Partition $A$ into $r \times c$ blocks using a clustering or coclustering algorithm.
2: Determine the dense blocks $A_{ij}$ and store their indices $(i,j)$ in $\mathcal{S}$.
3: Determine ranks $k_{ij}$ in the low rank approximations of $A_{ij}$ for $(i,j) \in \mathcal{S}$.
4: Compute low rank approximation using the truncated SVD according to (3.7).
5: Compute the diagonal blocks in $\bar{U} = \operatorname{diag}(U_1, \ldots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \ldots, V_c)$ according to (3.8) and (3.9), respectively.
6: Set $S_{ij} = \Sigma_{ij}$ when $(i,j) \in \mathcal{S}$ and compute $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ otherwise.

*Remark* 3.2. There are important aspects in the current presentation that we do not address. For example, how does one choose the number of row clusters $r$ and number of column clusters $c$? Given this, how does one determine if a given block is sufficiently dense? And, subsequently, what ranks $k_{ij}$ should be used in the blockwise SVD approximations? Different heuristics may be employed to address these questions, and it is likely that this will depend on the end application and on the particular way the clustered matrix approximation will be used in order to solve a problem. It is also possible to address these tuning aspects in a more stringent way by considering and optimizing some objective measure that involves aspects of the clustered matrix approximation, e.g., total memory consumption and performance of approximation in the end application. Regardless of this, we show in section 4 that

even simple strategies result in considerable benefits from numerical, theoretical, and computational points of view.

*Remark* 3.3. Clearly, the memory requirements for the clustered matrix approximation $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$, which is given in block form in (3.10), depend on the dimensions of $A$, the number of row and column clusters, the number of dense blocks, and the ranks $k_{ij}$. It may be the case that the factors $\bar{U}$, $\bar{S}$, and $\bar{V}$ require more memory storage than the memory needed to store the original (sparse) matrix $A$. We would like to clarify that it is not our objective to develop matrix approximations that use less memory than the original matrix. The intention is to produce memory efficient matrix approximations for dimensionality reduction purposes. In subsequent steps, these matrix approximations make it possible to scale up various algorithms while preserving important structure and information from the original matrix.

**3.3. Randomized clustered matrix approximation.** The methods we presented in sections 3.1 and 3.2 compute best rank-$k_{ij}$ approximations of the dense blocks $A_{ij}$. This is not necessary since the overall approximation of $A$ is not optimal in terms of rank. We may in fact compute approximations using any method that fits in the context. In particular, we may employ probabilistic algorithms. We will now describe the randomized clustered method for the general nondiagonal dense block structure case, and then derive bounds for the approximation error.

**3.3.1. Algorithm.** In Algorithm 1, an approximation of the dominant column space of a matrix is computed. This is done by $Y = A\Omega$, where $\Omega$ is a random Gaussian matrix. In this scenario without clustering, $Y$ uniquely determines the approximation as $A \approx QQ^{\mathsf{T}}A$, where $Q = \mathrm{orth}(Y)$. Subsequently, an approximation to the SVD may be computed from which the corresponding optimal row space is obtained. In other words, given $Y$, both the column space and the row space of the approximation are determined. In the clustered setting we need to adopt this relation.

Let an $m \times n$ matrix $A$ be partitioned into $r \times c$ blocks $A_{ij}$, with dimensions $m_i \times n_j$, as shown in (3.6). Our aim is to construct a clustered matrix approximation $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$, where both $\bar{U}$ and $\bar{V}$ are block diagonal, orthonormal, and obtained using a probabilistic approach. As previously, introduce the following sets: $\mathcal{S}$ containing index pairs for all dense blocks; $\mathcal{R}_i \subset \mathcal{S}$ with index pairs for block row $i$; and $\mathcal{C}_j \subset \mathcal{S}$ with index pairs for block column $j$. Introduce now Gaussian matrices $\Omega^{(ij)} \in \mathbb{R}^{n_j \times (k_{ij}+p_{ij})}$ with target ranks $k_{ij}$ and oversampling parameters $p_{ij}$. Compute

$$Y_{ij} = A_{ij}\Omega^{(ij)} \quad \text{or} \quad Y_{ij} = (A_{ij}A_{ij}^{\mathsf{T}})^q A_{ij}\Omega^{(ij)}, \qquad (i,j) \in \mathcal{S}, \tag{3.11}$$

where $q$ is a small integer. By forming

$$Y_i = [Y_{ic_{i,1}} \ \cdots \ Y_{ic_{i,|\mathcal{R}_i|}}], \quad (i,c_{i,1}), \ldots, (i,c_{i,|\mathcal{R}_i|}) \in \mathcal{R}_i, \quad \text{for} \quad i = 1, \ldots, r,$$

we consider the approximation $A \approx P_{\bar{Y}}A$, where $\bar{Y} = \mathrm{diag}(Y_1, \ldots, Y_r)$ and $P_{\bar{Y}}$ is the orthogonal projection onto $\mathrm{range}(\bar{Y})$. Using $\bar{Q} = \mathrm{orth}(\bar{Y})$ we can express the approximation as $A \approx \bar{Q}(\bar{Q}^{\mathsf{T}}A)$. It follows that $\bar{Q}$ has the same block diagonal structure as $\bar{Y}$. However, the matrix $\bar{Q}^{\mathsf{T}}A$, representing the associated row space, will not have a block diagonal structure. Consequently the approximation $A \approx P_{\bar{Y}}A = \bar{Q}(\bar{Q}^{\mathsf{T}}A)$ will not be memory efficient.

A block diagonal matrix for the row space of $A$ may be obtained as follows. Using $Q_{ij} = \mathrm{orth}(Y_{ij})$, we get the approximations

$$A_{ij} \approx Q_{ij}Q_{ij}^{\mathsf{T}}A_{ij} \equiv Q_{ij}Z_{ij}^{\mathsf{T}}, \quad (i,j) \in \mathcal{S},$$

**Algorithm 4** Randomized clustered matrix approximation.

**Input:** $A$, number of row clusters $r$, number of column clusters $c$.

**Output:** Block matrices that form the clustered low rank approximation: $U_1, \ldots, U_r$, $V_1, \ldots, V_c$, and $S_{ij}$ with $i = 1, \ldots, r$ and $j = 1, \ldots, c$.

1: Partition $A$ into $r \times c$ blocks using a clustering or coclustering algorithm.
2: Determine the dense blocks $A_{ij}$ and store their indices $(i, j)$ in the set $\mathcal{S}$.
3: Determine blockwise target ranks $k_{ij}$, oversampling parameters $p_{ij}$, and possibly a power parameter $q$.
4: Generate Gaussian random matrices $\Omega^{(ij)} \in \mathbb{R}^{n_j \times (k_{ij}+p_{ij})}$ and compute $Y_{ij}$ according to (3.11).
5: Compute $Q_{ij} = \mathrm{orth}(Y_{ij})$ with $(i, j) \in \mathcal{S}$.
6: Compute the row space matrices $Z_{ij} = A_{ij}^{\mathsf{T}} Q_{ij}$ with $(i, j) \in \mathcal{S}$.
7: Compute $U_1, \ldots, U_r$ and $V_1, \ldots, V_c$ according to (3.12) and (3.13), respectively.
8: Compute the blocks of $\bar{S}$ with $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ for all $i$ and $j$.

where $Z_{ij} = A_{ij}^{\mathsf{T}} Q_{ij}$ spans the row space of $A_{ij}$. We define the probabilistic clustered approximation to be $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$, where orthonormal $\bar{U} = \mathrm{diag}(U_1, \ldots, U_r)$ and $\bar{V} = \mathrm{diag}(V_1, \ldots, V_c)$ are obtained from

$$U_i = \mathrm{orth}([Q_{ic_{i,1}} \ \cdots \ Q_{ic_{i,|\mathcal{R}_i|}}]), \quad (i, c_{i,1}), \ldots, (i, c_{i,|\mathcal{R}_i|}) \in \mathcal{R}_i, \quad i = 1, \ldots, r, \tag{3.12}$$

$$V_j = \mathrm{orth}([Z_{r_{j,1}j} \ \cdots \ Z_{r_{j,|\mathcal{C}_j|}j}]), \quad (r_{j,1}, j), \ldots, (r_{j,|\mathcal{C}_j|}, j) \in \mathcal{C}_j, \quad j = 1, \ldots, c. \tag{3.13}$$

Then the optimal $\bar{S}$ is given by $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ for all $i$ and $j$. The entire process is presented in Algorithm 4.

**3.3.2. Analysis and error bounds.** The main theoretical results of this section are Theorems 3.4 and 3.5. We will first introduce necessary variables to conduct the analysis. Recall that $\mathcal{S}$ contains the set of index pairs indicating the dense blocks of $A$. Let $\mathcal{T}$ contain all of the remaining index pairs. Clearly $\mathcal{S}$ and $\mathcal{T}$ are disjoint and $\mathcal{S} \cup \mathcal{T} = \{(i, j) \mid i = 1, \ldots, r, \ j = 1, \ldots, c\}$. Let each block $A_{ij}$ with $(i, j) \in \mathcal{S}$ have the full SVD

$$A_{ij} = U^{(ij)}\Sigma^{(ij)}(V^{(ij)})^{\mathsf{T}} = [U_1^{(ij)} \ U_2^{(ij)}] \begin{bmatrix} \Sigma_1^{(ij)} & 0 \\ 0 & \Sigma_2^{(ij)} \end{bmatrix} [V_1^{(ij)} \ V_2^{(ij)}]^{\mathsf{T}}. \tag{3.14}$$

We have partitioned the SVD as in (2.4) so that $\Sigma_1^{(ij)}$ contains the top $k_{ij}$ singular values. $U^{(ij)}$ and $V^{(ij)}$ are partitioned accordingly. Introduce $n_j \times (k_{ij} + p_{ij})$ matrices $\Omega^{(ij)}$ and let $Y_{ij} = A_{ij}\Omega^{(ij)}$. Define further

$$\Omega_1^{(ij)} = (V_1^{(ij)})^{\mathsf{T}}\Omega^{(ij)}, \qquad \Omega_2^{(ij)} = (V_2^{(ij)})^{\mathsf{T}}\Omega^{(ij)}, \quad (i, j) \in \mathcal{S}. \tag{3.15}$$

In the following analysis we will consider two different approximations. The first one is given by

$$A \approx \hat{A} = \bar{U}\bar{S}\bar{V}^{\mathsf{T}} \equiv \bar{U}(\bar{U}^{\mathsf{T}} A \bar{V})\bar{V}^{\mathsf{T}},$$

or, equivalently, by considering each block separately,

$$A_{ij} \approx \hat{A}_{ij} = U_i S_{ij} V_j^{\mathsf{T}} \equiv U_i(U_i^{\mathsf{T}} A_{ij} V_j)V_j^{\mathsf{T}} \quad \forall (i, j), \tag{3.16}$$

where $\bar{U} = \mathrm{diag}(U_1, \ldots, U_r)$, $\bar{V} = \mathrm{diag}(V_1, \ldots, V_c)$, and $S_{ij}$ are computed according to Algorithm 4. Observe that this low rank approximation is valid for all the blocks.

In addition, for $(i,j) \in \mathcal{S}$, we have low rank approximation of $A_{ij}$ in terms of $Y_{ij}$ given by

$$A_{ij} \approx \tilde{A}_{ij} = P_{Y_{ij}} A_{ij} \equiv \tilde{U}^{(ij)} \tilde{\Sigma}^{(ij)} (\tilde{V}^{(ij)})^{\mathsf{T}}, \qquad (i,j) \in \mathcal{S}, \tag{3.17}$$

where $P_{Y_{ij}}$ is the orthogonal projector onto range$(Y_{ij})$. In (3.17) we also introduce the SVD of $\tilde{A}_{ij}$. Note that approximations in (3.16) are valid for all blocks, while those in (3.17) are valid only for the dense blocks. It is clear that range$(\tilde{U}^{(ij)}) \subseteq$ range$(U_i)$ as well as range$(\tilde{V}^{(ij)}) \subseteq$ range$(V_j)$ when $(i,j) \in \mathcal{S}$. We conclude this by observing that all dense blocks from the $i$th block row of $A$ contribute to $U_i$, while only $A_{ij}$ contributes to $\tilde{U}^{(ij)}$. Similarly, all dense blocks in the $j$th block column contribute to $V_j$, while only $A_{ij}$ contributes to $\tilde{V}^{(ij)}$. It follows that $\hat{A}_{ij}$ is a better approximation than $\tilde{A}_{ij}$ for all $(i,j) \in \mathcal{S}$. Now we state the first theorem.

THEOREM 3.4. *Let $A$ be a given $m \times n$ matrix with an $r \times c$ block partitioning as in* (3.6). *Introduce the SVDs of $A_{ij}$ and a partitioning of the corresponding $\Sigma^{(ij)}$ as in* (3.14). *Let $\mathcal{S}$ be a set of pairwise indices, so that indices of at least one block from every block row and block column are present. For $(i,j) \in \mathcal{S}$, let $k_{ij}$ be a target rank for $A_{ij}$ and $p_{ij}$ a corresponding oversampling parameter such that $k_{ij} + p_{ij} \leq \min(m_i, n_j)$. Introduce matrices $\Omega^{(ij)} \in \mathbb{R}^{n_j \times (k_{ij}+p_{ij})}$, form $\Omega_1^{(ij)}$ and $\Omega_2^{(ij)}$ according to* (3.15), *and assume each $\Omega_1^{(ij)}$ has full rank. Compute the approximation $\hat{A} = \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ of $A$ according to Algorithm* 4. *Then the approximation error is bounded by*

$$\|A - \hat{A}\|_*^2 \leq \sum_{(i,j)\in\mathcal{S}} \left( \|\Sigma_2^{(ij)} \Omega_2^{(ij)} (\Omega_1^{(ij)})^{\dagger}\|_*^2 + \|\Sigma_2^{(ij)}\|_*^2 \right) + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_*^2,$$

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

*Proof.* The proof is somewhat cumbersome but straightforward. A number of inequalities lead to the bound. We will write those out and explain them one by one.

$$\begin{aligned}
\|A - \hat{A}\|_*^2 &= \|A - \bar{U}\bar{S}\bar{V}^{\mathsf{T}}\|_*^2 \\
&\leq \sum_{i,j=1}^{r,c} \|A_{ij} - \hat{A}_{ij}\|_*^2 = \sum_{i,j=1}^{r,c} \|A_{ij} - U_i S_{ij} V_j^{\mathsf{T}}\|_*^2 && (3.18)\\
&\leq \sum_{(i,j)\in\mathcal{S}} \|A_{ij} - \tilde{U}^{(ij)} \tilde{\Sigma}^{(ij)} (\tilde{V}^{(ij)})^{\mathsf{T}}\|_*^2 + \sum_{(i,j)\in\mathcal{T}} \|A_{ij} - U_i S_{ij} V_j^{\mathsf{T}}\|_*^2 && (3.19)\\
&= \sum_{(i,j)\in\mathcal{S}} \|(I - P_{Y_{ij}}) A_{ij}\|_*^2 + \sum_{(i,j)\in\mathcal{T}} \|A_{ij} - U_i S_{ij} V_j^{\mathsf{T}}\|_*^2 && (3.20)\\
&\leq \sum_{(i,j)\in\mathcal{S}} \left( \|\Sigma_2^{(ij)} \Omega_2^{(ij)} (\Omega_1^{(ij)})^{\dagger}\|_*^2 + \|\Sigma_2^{(ij)}\|_*^2 \right) + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_*^2. && (3.21)
\end{aligned}$$

1. The inequality in (3.18) is due to the block partitioning of the residual. In the Frobenius norm case we have equality. In the following equality we have used that $\hat{A}_{ij} = U_i S_{ij} V_j^{\mathsf{T}}$.
2. In (3.19) we split the summation into two parts: one with $(i,j) \in \mathcal{S}$, which corresponds to the dense $A_{ij}$, and one with $(i,j) \in \mathcal{T}$, which corresponds to the remaining blocks. Using $\hat{A}_{ij}$ and $\tilde{A}_{ij} = \tilde{U}^{(ij)} \tilde{\Sigma}^{(ij)} (\tilde{V}^{(ij)})^{\mathsf{T}}$, the inequality follows from

$$\|A_{ij} - U_i S_{ij} V_j^{\mathsf{T}}\|_* = \|A_{ij} - \hat{A}_{ij}\|_* \leq \|A_{ij} - \tilde{A}_{ij}\|_* = \|A_{ij} - \tilde{U}^{(ij)} \tilde{\Sigma}^{(ij)} (\tilde{V}^{(ij)})^{\mathsf{T}}\|_*.$$

Recall that range($\tilde{U}^{(ij)}$) $\subseteq$ range($U_i$) and range($\tilde{V}^{(ij)}$) $\subseteq$ range($V_j$).

3. In (3.20) we use the results of (3.17).
4. Finally, in (3.21) we use

$$|(I - P_{Y_{ij}})A_{ij}\|_*^2 \leq \|\Sigma_2^{(ij)}\Omega_2^{(ij)}(\Omega_1^{(ij)})^\dagger\|_*^2 + \|\Sigma_2^{(ij)}\|_*^2, \qquad (i,j) \in \mathcal{S},$$

which is proved in [21, Lem. 4.2]. In the summation with $(i,j) \in \mathcal{T}$ we have removed the approximating term, as the corresponding $A_{ij}$ blocks are not dense and consequently should have small norms. ∎

Given the deterministic error bound of Theorem 3.4 we can state the theorem with respect to expected error.

THEOREM 3.5. *Using the notation introduced in Theorem* 3.4*, where now* $\Omega_{ij}$ *are standard Gaussian matrices, the expected errors in the Frobenius and spectral norms are bounded by*

$$\mathbb{E}\|A - \hat{A}\|_F \leq \left( \sum_{(i,j)\in\mathcal{S}} \left(1 + \frac{k_{ij}}{p_{ij}-1}\right) \|\Sigma_2^{(ij)}\|_F^2 + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_F^2 \right)^{1/2}, \tag{3.22}$$

$$\mathbb{E}\|A - \hat{A}\|_2 \leq \sum_{(i,j)\in\mathcal{S}} \left( \left(1 + \frac{\sqrt{k_{ij}}}{\sqrt{p_{ij}-1}}\right) \|\Sigma_2^{(ij)}\|_2 + \frac{\mathsf{e}\sqrt{k_{ij}+p_{ij}}}{p_{ij}} \|\Sigma_2^{(ij)}\|_F \right) + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_2. \tag{3.23}$$

*Proof.* Both bounds for the expected error norm follow from Theorem 3.4 and from analysis similar to that in the proofs of Theorems 10.5 and 10.6 in [21]. ∎

We will now extend the spectral norm version of the deterministic and probabilistic error bounds of Theorems 3.4 and 3.5.

COROLLARY 3.6. *Let* $Y_{ij} = (A_{ij}A_{ij}^\mathsf{T})^q A_{ij}\Omega^{(ij)}$ *when* $(i,j) \in \mathcal{S}$ *with integer power parameter* $q > 0$ *as in* (3.11)*. Using the notation introduced in Theorem* 3.4*, the following deterministic error bound holds:*

$$\|A - \hat{A}\|_2^2 \leq \sum_{(i,j)\in\mathcal{S}} \left( \left( \|\Omega_2^{(ij)}(\Omega_1^{(ij)})^\dagger\|_2^2 + 1 \right)^{1/(2q+1)} \|\Sigma_2^{(ij)}\|_2^2 \right) + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_2^2.$$

COROLLARY 3.7. *Using the notation introduced in Corollary* 3.6 *and Theorem* 3.4*, the following expected error bound holds:*

$$\mathbb{E}\|A - \hat{A}\|_2 \leq \sum_{(i,j)\in\mathcal{S}} \left( \left(1 + \frac{\sqrt{k_{ij}}}{\sqrt{p_{ij}-1}}\right) \left\| \left(\Sigma_2^{(ij)}\right)^{2q+1} \right\|_2 + \frac{\mathsf{e}\sqrt{k_{ij}+p_{ij}}}{p_{ij}} \left\| \left(\Sigma_2^{(ij)}\right)^{2q+1} \right\|_F \right)^{1/(2q+1)} + \sum_{(i,j)\in\mathcal{T}} \|A_{ij}\|_2.$$

*Proof.* Both error bounds are direct consequences of Theorem 3.4 and [21, Thms. 9.1 and 9.2] for the deterministic bound and [21, Cor. 10.10] for the probabilistic bound. ∎

**4. Experiments and discussion.** In the following series of experiments we will compare various results of the developed framework with results from the truncated SVD. We will focus on the following key properties, which we think are of general importance both from a practical and a theoretical point of view: quality of the approximation; clustering structure that is preserved in the approximation; computational efficiency in terms of memory and time; row and column subspace properties.

**4.1. Data sets, algorithms, and experimental setup.** The presented experiments are based on the Notre Dame actors data [3], which is a bipartite graph representing links between movies and actors from the Internet Movie Database,[7] and LiveJournal data, which is a directed social network graph representing relationships among LiveJournal users [2, 25]. In all experiments we used the largest connected component[8] of each graph. The preprocessing resulted in a 81,823 × 94,003 matrix with 1,016,127 nonzeros for the Notre Dame data and a 3,828,682 × 3,828,682 nonsymmetric matrix with 65,825,429 nonzeros for the LiveJournal data. In both graphs the links are unweighted, giving all nonzero entries equal to one.

We will have three different SVD-based methods for computing approximations: (1) regular truncated SVD; (2) Algorithm 2—clustered approximation with diagonal block structure; (3) Algorithm 3—clustered approximation with nondiagonal block structure. Replacing the SVD computations with the probabilistic methods described in section 3.3 yields three more methods for computing matrix approximations. Implementation of all presented algorithms is provided in this paper's supplementary material (local/web 11.0MB).

Given a matrix $A$ and its low rank approximation $\hat{A} = USV^{\mathsf{T}}$, obtained by any of the methods we have discussed, we will measure the approximation accuracy using the relative error in the Frobenius norm,

$$\|A - \hat{A}\|_F / \|A\|_F = \left(\|A\|_F^2 - \|S\|_F^2\right)^{1/2} / \|A\|_F. \tag{4.1}$$

We see that the norm of the residual can be computed without explicitly forming $\hat{A}$.

**4.2. Clustering properties of the data.** The first step in our framework is to partition rows and columns into disjoint sets. Let $A$ be an $m \times n$ matrix with $r \times c$ block structure. Let $|A|$ denote the number of nonzero entries in $A$; then the fraction of nonzeros contained in the diagonal blocks becomes $\phi_d = \sum_i |A_{ii}|/|A|$. Similarly if $\mathcal{S}$ denotes the set of pairwise indices for dense blocks in the nondiagonal case, then $\phi_{\mathcal{S}} = \sum_{(i,j)\in\mathcal{S}} |A_{ij}|/|A|$ is the fraction of nonzero entries of $A$ contained in all dense blocks. We consider a block $A_{ij}$ to be dense if $|A_{ij}|/|A| \geq \tau$, where $\tau$ is a threshold value. Three different clusterings are performed on each data set using GRACLUS or METIS. The Notre Dame data was clustered using the matrix in (2.3). Information related to the clustering and introduced quantities is presented in Table 1. It is not our intention to compare different clustering algorithms, but rather to indicate that our framework can be used with different clustering methods and different objectives. The particular choice of clustering algorithm will likely depend on the domain area from which the data originates.

In the left and middle panels of Figure 5 we illustrate the dense blocks of $A$ for two cases from Table 1. $A$ is structured as in (3.6). The right panel of Figure 5 shows the fraction of nonzeros of all blocks in a $61 \times 61$ clustering of the LiveJournal matrix.

[7] http://www.imdb.com/

[8] For a bipartite graph represented by $B$, both $B^{\mathsf{T}}B$ and $BB^{\mathsf{T}}$ consist of a single component.

TABLE 1

*Clustering related information. $r$ and $c$ are the number of row and column clusters, respectively; $\phi_d$ is the fraction of nonzeros within the diagonal blocks; $\tau$ is the threshold used to determine the dense blocks; $|\mathcal{S}|$ is the number of dense blocks; $r \times c$ is the total number of blocks; $\phi_{\mathcal{S}}$ is the fraction of nonzeros within the dense blocks; and the last column indicates the clustering method used.*

| Data set | $r = c$ | $\phi_d$ in % | $\tau$ in % | $\lvert\mathcal{S}\rvert$ | $r \times c$ | $\phi_{\mathcal{S}}$ in % | Clustering |
|---|---|---|---|---|---|---|---|
| Notre Dame | 10 | 83.4 | 0.5 | 18 | 100 | 92.5 | METIS |
| | 20 | 75.3 | 0.3 | 45 | 400 | 90.5 | METIS |
| | 30 | 69.4 | 0.2 | 76 | 900 | 85.8 | METIS |
| LiveJournal | 22 | 76.9 | 0.2 | 37 | 484 | 84.3 | GRACLUS |
| | 61 | 69.3 | 0.05 | 134 | 3,761 | 79.3 | GRACLUS |
| | 117 | 66.3 | 0.04 | 222 | 13,689 | 75.2 | GRACLUS |

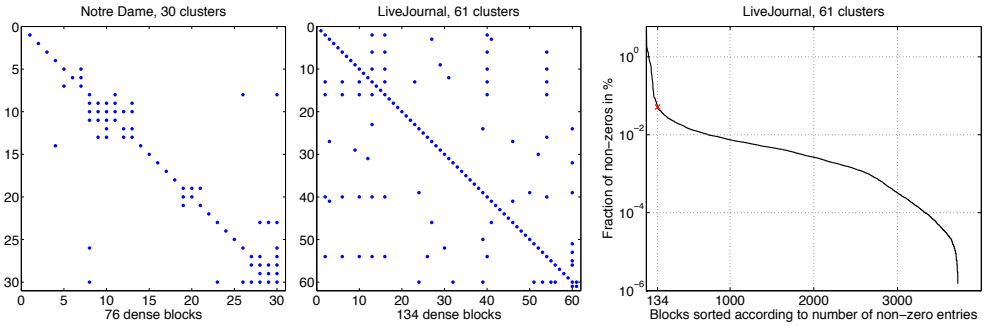


FIG. 5. *Left: Dense blocks of a $30 \times 30$ clustering of the Notre Dame matrix using $\tau = 0.2$. Middle: Dense blocks in a $61 \times 61$ clustering of the LiveJournal matrix using $\tau = 0.05$. Right: Fraction of not (in descending order) of all blocks in the $61 \times 61$ clustering of the LiveJournal matrix. Clearly, most blocks contain a very small fraction of nonzeros. The $\times$ mark at $x = 134$ indicates the last block considered to be dense.*

It is clear from the shape of the curve that using a smaller threshold $\tau$ (than the one corresponding to the $\times$ mark at $x = 134$) will considerably increase the number of dense blocks, resulting in a clustered matrix approximation that will use much more memory. Furthermore, the fact that a small number of blocks have a larger fraction of nonzeros is an indication of structure present in the network. This structure is preserved in the clustered low rank approximation.

**4.3. Approximation quality.** We will now compare quality of approximations obtained using our framework and with truncated SVD approximations using (1.1).

**LiveJournal.** In Figure 6 we present results for the LiveJournal data. The left panel shows results using Algorithm 2 for all three clustering cases from Table 1. Let $\mathcal{K}_{\text{LJ}} = \{25, 50, 100, 150, 200\}$. For each clustering case we compute five approximations using $k_{ii} \in \mathcal{K}_{\text{LJ}}$, and in each approximation we use the same $k_{ii}$ in all diagonal blocks $A_{ii}$. In the right panel we use Algorithm 3 with the clustering and corresponding threshold values from Table 1. Again, for each clustering case we compute five approximations using $k_{ij} \in \mathcal{K}_{\text{LJ}}$, and in each approximation we use the same $k_{ij}$ for all dense blocks. We also compute five regular truncated SVD approximations using (1.1) with $k \in \mathcal{K}_{\text{LJ}}$. Each curve corresponds to a specific clustering case, and each mark on a curve corresponds to a particular rank. Note that the $x$-axis in the plots represents memory usage (in terms of floating point numbers) and not the rank of the approximation. Clearly, clustering substantially improves the quality of the approximation! Additional improvement is seen by increasing the number of clusters! We

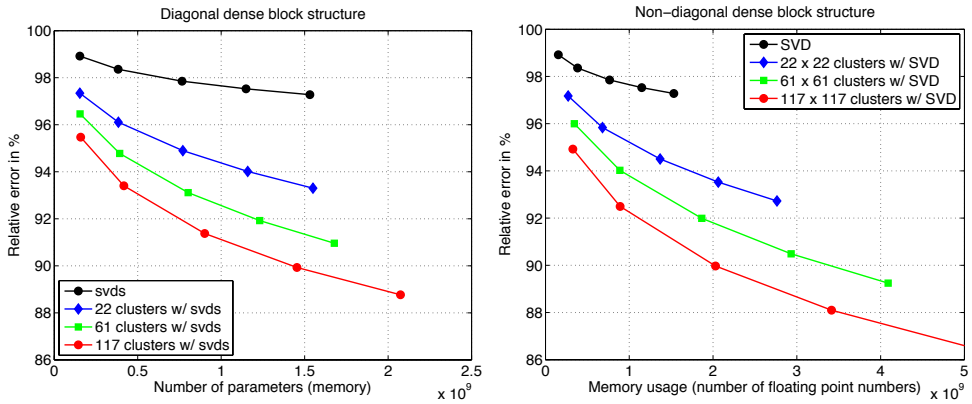


FIG. 6. *Relative errors for the LiveJournal matrix approximations. The left panel shows results for the clustered low rank approximation using only the diagonal blocks, while the right panel uses a nondiagonal dense block structure.*

also see that increasing the number of clusters increases the memory usage. This is particularly obvious for the nondiagonal clustered approximation in the right panel. A closer examination reveals that approximations with Algorithm 2 are a bit more memory efficient than approximations with Algorithm 3. This is to be expected since many nondiagonal dense blocks have a relatively small fraction of nonzeros compared to the diagonal blocks; thus approximating these "smaller" blocks with the same rank as the "heavier" blocks is costly. On the other hand Algorithm 3 is better in terms of preserving the structure of the original matrix. Using different ranks for different blocks could be a better approach to finding a balance between memory usage and desired level of structure preservation.

**Notre Dame.** In Figure 7 we show results of experiments for the Notre Dame matrix. The setting is similar. In the left panel we use Algorithm 2, and in the right panel we use Algorithm 3, with the three clustering cases from Table 1 for each algorithm. For each clustering case we compute 10 approximations using $k_{ij} \in \mathcal{K}_{\text{ND}} = \{10, 20, \ldots, 100\}$, and for each approximation we use the same $k_{ij}$ for all dense blocks. In truncated SVD approximations we use ranks $k \in \mathcal{K}_{\text{R}} = \{25, 50, 100, 150, \ldots, 500\}$. Also in this case, clustering significantly improves the quality of the approximation compared to the truncated SVD approximation. Although increasing the number of clusters does give better approximations, the benefit seems to stagnate. For example, there is only slight improvement in Algorithm 3 (right panel) when increasing the clustering from $20 \times 20$ to $30 \times 30$. Also in this case one may benefit by using different ranks for different dense blocks in order to balance approximation quality, memory consumption, and structure preservation.

**4.4. Blockwise information content.** We claimed previously that the truncated SVD captures information from a few dominating clusters. Of course, this is not desirable if cluster structure of the matrix is valuable. In Figure 8 we present two experiments that validate this claim. Let $A$ be partitioned as in (3.6). We will investigate and compare the blockwise relative errors $\|A_{ij} - \hat{A}_{ij}\|_F / \|A_{ij}\|_F$, where $\hat{A}_{ij}$ is obtained by either the truncated SVD or Algorithm 3.

**Notre Dame.** For the Notre Dame matrix we used a rank-500 truncated SVD yielding 94.4% in relative error using about $8.8 \times 10^7$ floating point numbers (this

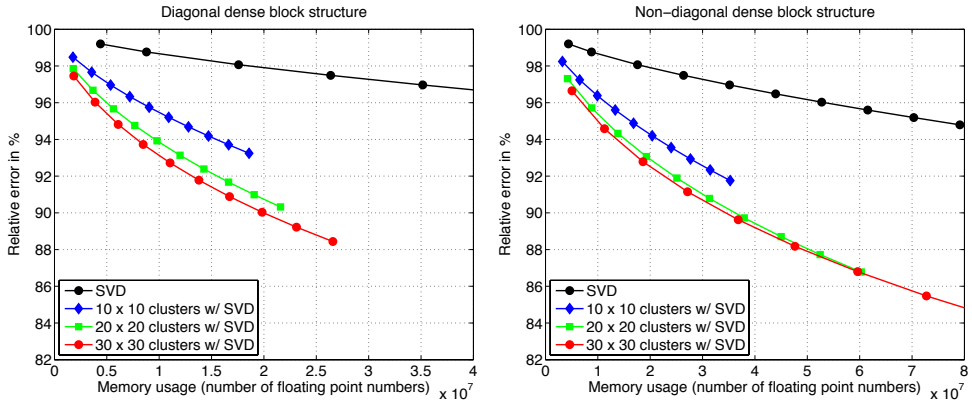


FIG. 7. *Relative errors for the Notre Dame matrix. Left panel shows results for the diagonal clustered approximation, and the right panel shows the nondiagonal dense block structure.*

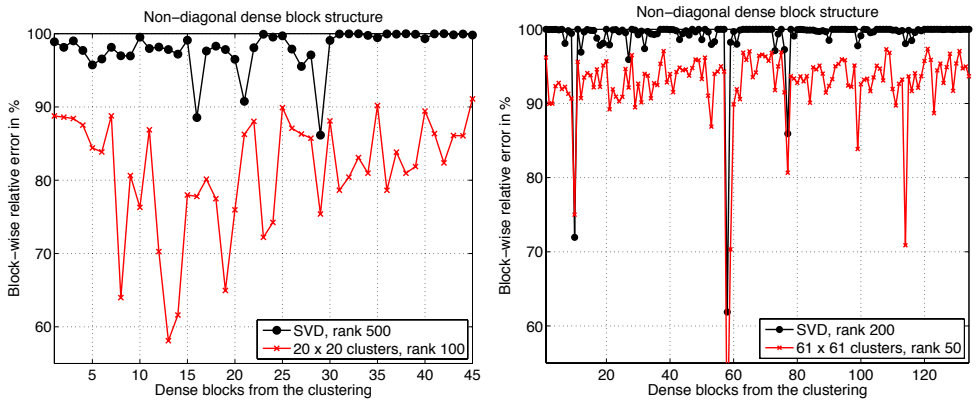


FIG. 8. *Information captured from the dense blocks in a regular and clustered matrix approximation. The blockwise relative error is* $\|A_{ij} - \hat{A}_{ij}\|_F / \|A_{ij}\|_F$ *for* $(i,j) \in \mathcal{S}$ *for the Notre Dame matrix (left) and the LiveJournal matrix (right).*

last entry is off the chart in Figure 7). Algorithm 3 uses the $20 \times 20$ clustering from Table 1, giving 45 dense blocks. Each dense block is approximated with a rank-100 truncated SVD, achieving about 87% in overall relative error with $6 \times 10^7$ (about 30% less memory) floating point numbers. We see in Figure 8 that the regular SVD achieves about 90% relative error in three of the 45 dense blocks, while the relative error for the remaining blocks is 96–100%. Algorithm 3, on the other hand, achieves about 90% or less in relative error from all dense blocks, and mean relative error of 81%—a substantial improvement from the 98% in mean relative error for the regular truncated SVD approximation.

**LiveJournal.** For the LiveJournal matrix we used a rank-200 truncated SVD giving 97.3% in relative error with $1.5 \times 10^9$ floating point numbers. In Algorithm 3 we use the $61 \times 61$ clustering from Table 1, giving 134 dense blocks. Each dense block was approximated with a rank-50 truncated SVD. The resulting approximation gives 94% in overall relative error using $0.9 \times 10^9$ (about 40% less memory) floating point numbers. The right panel of Figure 8 shows a similar result. The regular truncated

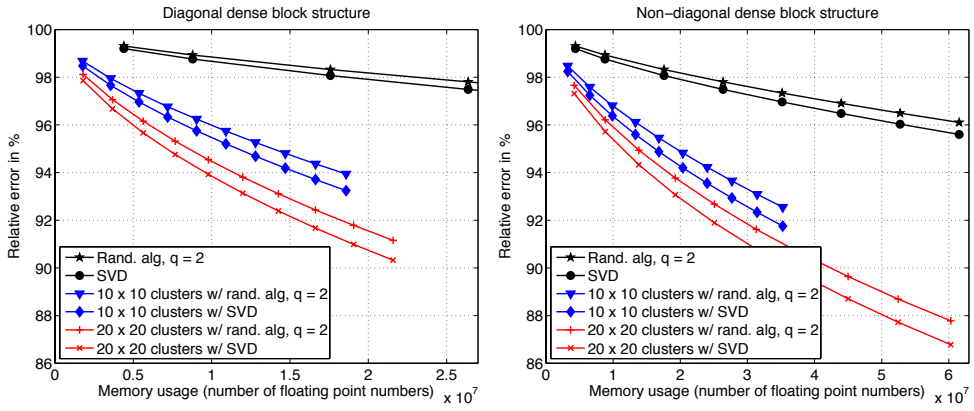


FIG. 9. *Relative errors for the Notre Dame matrix approximations. The left panel shows results for the clustered matrix approximation using diagonal dense block structure, while the right panel uses a nondiagonal dense block structure.*

SVD achieves good approximation for three blocks, while the remaining 131 dense blocks are hardly approximated at all. Our method again captures a considerable amount of information from each dense block, resulting in 92.6% mean relative error over all dense blocks. The corresponding mean relative error for the regular truncated SVD approximation is 99.0%.

**4.5. Performance of probabilistic clustered algorithms.** We will now compare the performance of probabilistic clustered methods with Algorithms 2 and 3 as well as nonclustered methods. We will experiment with the Notre Dame matrix using the $10 \times 10$ and $20 \times 20$ clusterings from Table 1. In the left panel of Figure 9 we use Algorithm 2 and a probabilistic version of it, in which the blockwise SVD approximations are replaced with probabilistic approximations. In the right panel we use Algorithms 3 and 4. All clustered methods use $k_{ij} \in \mathcal{K}_{\mathrm{ND}}$ when approximating the dense blocks. In all probabilistic methods the power parameter $q = 2$. We do not consider $q = 0$ as this case gives considerably higher relative errors [30]. In both panels we also present nonclustered approximations with ranks $k \in \mathcal{K}_{\mathrm{R}}$ using the truncated SVD and a corresponding probabilistic method.

Again, all clustered methods, including the probabilistic ones, give a much better approximation than nonclustered methods. We see that the SVD-based approximations give better accuracy (lower error rate), but for a higher computational price, as will be shown in section 4.6. However, the difference between a probabilistic and the corresponding SVD-based approach can be made smaller by using a higher power parameter, e.g., $q = 4$. The computational amount will increase slightly, but it will still be faster than the SVD-based approach.

**4.6. Timing comparisons.** In Figure 10 we present two plots with timing results using the *cputime*-function in MATLAB for the experiments in Figure 9. Several observations can be made: (1) We see in both panels that increasing the number of clusters in the SVD-based methods reduces the computational time. Thus, computing SVDs of many small matrices is faster than computing the SVD of a single big matrix. (2) The execution time for all probabilistic methods is considerably faster than those of SVD-based methods. (3) There are very small timing differences in the probabilistic methods when considering different numbers of clusters. (4) We would like to

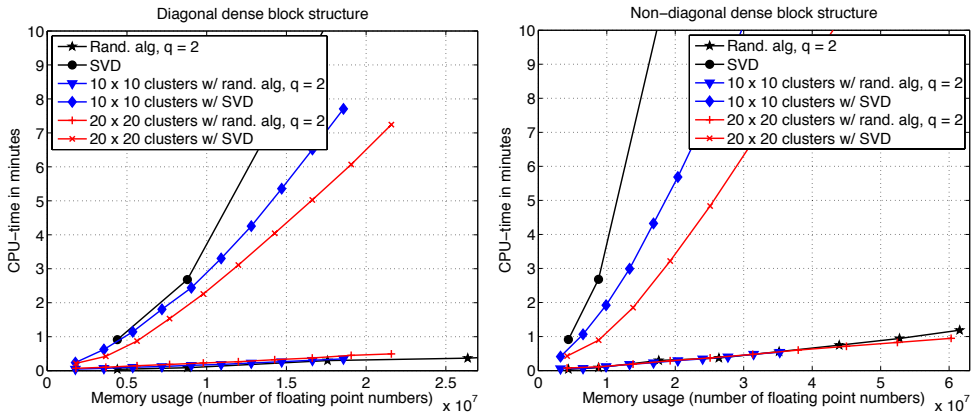


FIG. 10. *Timing results corresponding to the experiments in Figure* 9.

point out that these timings include the time taken by the fast clustering procedures, which for these experiments is about two to three seconds.

**4.7. Principal angles.** Assume we have a truncated SVD approximation $A \approx U\Sigma V^{\mathsf{T}}$ and a clustered approximation $A \approx \bar{U}\bar{\Sigma}\bar{V}^{\mathsf{T}}$ obtained with one of our methods. It is very relevant to ask, how close is range($\bar{U}$) to range($U$)? The question can be answered by examining the singular values $\sigma_i$ of $U^{\mathsf{T}}\bar{U}$, since $\sigma_i = \cos(\theta_i)$ are cosines of the principal angles between the two subspaces [6]. In Figure 11 we present cosines of the principal angles for several experiments on both the Notre Dame matrix (left panel) and the LiveJournal matrix (right panel). The subspaces are close to each other if many $\sigma_i$ are close to one. We use the same clustering cases as previously. Additional information for each case, e.g., relative errors and memory usage, can be obtained from previously presented figures.

**Notre Dame.** For this case, $U$ is obtained from a rank-500 truncated SVD approximation. We have six different settings for the computation of $\bar{U}$ that are specified in the figure legend. It is easy to verify that each of the following claims brings range($\bar{U}$) closer to the range($U$) in significant jumps: (1) increasing the number of clusters; (2) increasing the rank in the blockwise approximations; and (3) using a nondiagonal block structure instead of diagonal blocks only. For example, to verify the third claim, compare the two green curves, or the two blue curves (color figure available online).

**LiveJournal.** The situation is similar for the LiveJournal matrix. Here, we use $U$ from a rank-200 truncated SVD approximation. For $\bar{U}$ in the clustered approximation we only use Algorithm 3 (nondiagonal block structure). We compute one approximation for each clustering setting from Table 1, and in each approximation we use $k_{ij} = 50$ in the blockwise approximations. From Figure 6 we see that all three clustered approximations have about the same memory usage as a rank-100 truncated SVD approximation. Thus, $\bar{U}$ uses about half the memory required for $U$. We observe that in all three clustered approximations range($\bar{U}$) is very close to a 170-dimensional subspace of range($U$). Also in this case, increasing the number of clusters produces range($\bar{U}$) that approach range($U$).

Evidently, our framework produces matrix approximations of $A$ with range($\bar{U}$) very close to range($U$), which is the dominant subspace for the columns space of $A$.

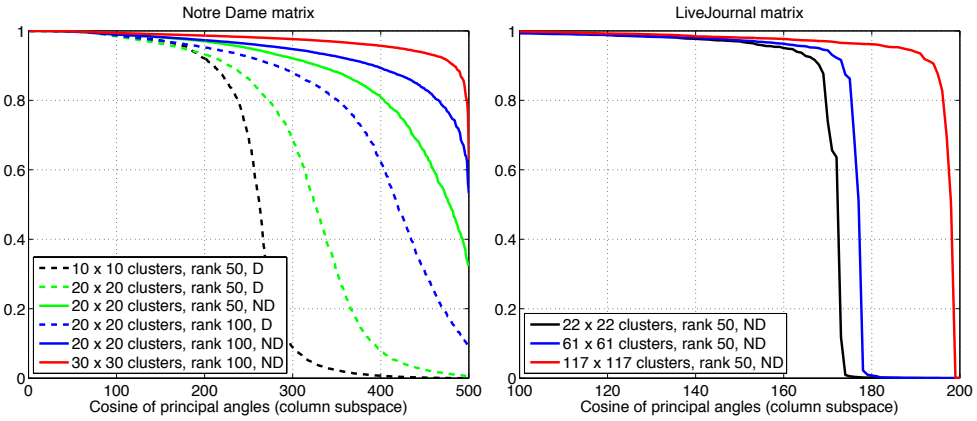


FIG. 11. *Left: Cosine of the principal angles for the Notre Dame matrix for six different experiments. Right: Cosine of the principal angles for the LiveJournal matrix. "D" in the description indicates diagonal dense structure, and "ND" indicates a nondiagonal dense block structure. Also note that the x-axis for the right panel starts at* 100*, as all three curves are very close to* 1 *in the omitted part. (Color available online.)*

Consequently, the clustered matrix approximations more or less contain the truncated SVD approximation! Experiments with $V$ and $\bar{V}$, which approximate the row space of $A$, show a similar behavior and lead to the same conclusions.

**5. Related work.** In a recent publication Savas and Dhillon [30] introduced a first approach to clustered low rank approximation of graphs (square matrices) in information science applications. Their approach has proven to perform exceptionally well in a number of applications [34, 32, 36]. Subsequently a multilevel clustering approach was developed in order to speed up the computation of the dominant eigenvalues and eigenvectors of massive graphs [33]. In comparison with [30], the current presentation extends the clustered matrix approximation approach in several ways. (1) The clustered matrix approximation with dense diagonal block structure, i.e., when $r = c$, can now be applied on rectangular matrices or bipartite graphs. (2) For both graphs and bipartite graphs (square and rectangular matrices) we may use a different number of row and column clusters, i.e., $r \neq c$, and allow for contribution from off-diagonal blocks to the matrix approximation. The resulting matrix approximation has the potential to preserve even more structural information from the original matrix. (3) We have also developed probabilistic algorithms for the entire clustered matrix approximation framework. In addition, we have derived several new theoretical results with deterministic and probabilistic error bounds.

Clustering has previously been used in similar ways to obtain low rank matrix approximations. Dhillon and Modha [14] use *spherical k-means* clustering on $m \times n$ term document matrix $A$ so that the columns (documents) of $A$ are clustered into $k$ clusters. From each cluster a *concept vector* is derived which is later used as a basis to obtain a low rank approximation of $A$. In [9, 41], similarly to [14], clustering is used to partition either the rows or columns of an $m \times n$ matrix $A$. An approximation of the data matrix $A$ is obtained using rank-1 or rank-$k_i$ truncated SVD approximations of each row or column cluster. In Zeimpekis and Gallopoulos [41] and Gao and Zhang [19], the authors use similar clustering strategies to extended low rank approximation methods to clustered latent semantic indexing. In [42] dimensional-

ity reduction is obtained via class representatives, where a clustering of the matrix columns is performed and from each cluster a number of representative vectors are extracted. An orthonormal basis is computed using these representatives in order to approximate $A$. In a similar setting, Zhang and Zha [45] 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, Sun, and Anerousis [8] use clustering for the column subset selection problem when dealing with streaming matrix data. In a similar approach with data sampling, Zhang and Kwok [44] present a clustered Nyström method for large scale manifold learning applications, where the authors approximate the eigenfunctions of associated integral equation kernels.

An important difference of our method, in comparison with the approximations in [14, 9, 41, 45, 8], is that we cluster rows and columns simultaneously. In addition $U = \mathrm{diag}(U_1, \ldots, U_r)$ and $V = \mathrm{diag}(V_1, \ldots, V_c)$, which are used to approximate $A$, are sparse and orthogonal by construction. As a consequence of the block structure of $U$ and $V$ we save in storage since only the nonzero blocks are stored. The basis matrices in the related methods have no zero structure that allows memory storage savings. In column subset selection methods and the Nyström method, only part of the data is used to compute an approximation of $A$, but in the clustered low rank approach all entries of $A$ influence the approximation.

**6. Conclusions.** In this paper we have developed a framework for matrix approximations that preserve the important structure of the underlying data. The structural information of a matrix $A$ is extracted by a (co)clustering algorithm that leads to a block partitioning of $A$. For matrices arising from a wide range of applications, only a small fraction of the blocks is dense, which thus contains a sufficient amount of information. By explicitly computing approximations of all dense blocks we preserve the structural (cluster) information of the data. Subsequently, we extend the blockwise approximations to an approximation for the entire matrix $A$. We have also developed a probabilistic approach within our framework that uses randomness to compute the clustered matrix approximation. For the probabilistic algorithms we proved deterministic and probabilistic bounds for the norm of the approximation errors. The clustered matrix approximation has the form $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ with orthonormal and block diagonal $\bar{U}$ and $\bar{V}$.

Based on a series of experiments we have made a number of observations that highlight the benefits of our framework. We conclude that, using a fixed amount of memory, our approach produces substantially more accurate approximations than truncated SVD approximations, which are optimal with respect to rank; our algorithm is faster (all steps included) than the corresponding truncated SVD algorithm; a block-by-block comparison reveals that, in our method, a significant amount of information is captured from all dense blocks, whereas the truncated SVD captures information from only a few dominant blocks; in addition to higher accuracy, higher memory efficiency, shorter execution times, and structure preservation in our method, subspace analysis reveals that the corresponding truncated SVD approximation is almost entirely contained in the clustered approximation.

## REFERENCES

[1] A. Abou-Rjeili and G. Karypis, *Multilevel algorithms for partitioning power-law graphs*, in IEEE International Parallel & Distributed Processing Symposium (IPDPS), 2006.

[2] 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 International Conference on Knowledge Discovery and Data Mining (KDD '06), ACM, New York, 2006, pp. 44–54.
[3] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.
[4] M. Berry, *Survey of Text Mining: Clustering, Classification, and Retrieval*, Springer, New York, 2003.
[5] Å. Björck, *Numerical Methods in Matrix Computations*, Springer, New York, 2015.
[6] Å. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Math. Comp., 27 (1973), pp. 579–594.
[7] C. Boutsidis, M. W. Mahoney, and P. Drineas, *An improved approximation algorithm for the column subset selection problem*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09), SIAM, Philadelphia, 2009, pp. 968–977, doi:10.1137/1.9781611973068.105.
[8] C. Boutsidis, J. Sun, and N. Anerousis, *Clustered subset selection and its applications on IT service metrics*, in Proceeding of the 17th ACM Conference on Information and Knowledge Management (CIKM '08), ACM, New York, 2008, pp. 599–608.
[9] V. Castelli, A. Thomasian, and C.-S. Li, *CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces*, IEEE Trans. Knowl. Data Engrg., 15 (2003), pp. 671–685.
[10] N. Cristianini, J. Shawe-Taylor, and J. S. Kandola, *Spectral kernel methods for clustering*, in Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 649–655.
[11] S. Deerwester, *Improving information retrieval with latent semantic indexing*, in Proceedings of the 51st ASIS Annual Meeting (ASIS '88) C. L. Borgman and E. Y. H. Pai, eds., Proc. ASIS Ann. Meeting 25, American Society for Information Science, 1988, pp. 36–40.
[12] I. S. Dhillon, *Co-clustering documents and words using bipartite spectral graph partitioning*, in Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, 2001, pp. 269–274.
[13] I. S. Dhillon, Y. Guan, and B. Kulis, *Weighted graph cuts without eigenvectors: A multilevel approach*, IEEE Trans. Pattern Anal. Mach. Intell., 29 (2007), pp. 1944–1957.
[14] I. S. Dhillon and D. S. Modha, *Concept decompositions for large sparse text data using clustering*, Machine Learning, 42 (2001), pp. 143–175.
[15] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183, doi:10.1137/S0097539704442696.
[16] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.
[17] E. Estrada and D. J. Higham, *Network properties revealed through matrix functions*, SIAM Rev., 52 (2010), pp. 696–714, doi:10.1137/090761070.
[18] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, *A survey of kernel and spectral methods for clustering*, Pattern Recognition, 41 (2008), pp. 176–190.
[19] J. Gao and J. Zhang, *Clustered SVD strategies in latent semantic indexing*, Inform. Process. Management, 41 (2005), pp. 1051–1063.
[20] L. Hagen and A. B. Kahng, *New spectral methods for ratio cut partitioning and clustering*, IEEE Trans. Computer-Aided Design Integrated Circuits Syst., 11 (1992), pp. 1074–1085.
[21] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, doi:10.1137/090771806.
[22] J. Kunegis and A. Lommatzsch, *Learning spectral graph transformations for link prediction*, in Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09), ACM, New York, 2009, pp. 561–568.
[23] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graph evolution: Densification and shrinking diameters*, ACM Trans. Knowl. Discov. Data, 1 (2007), 2.
[24] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, *Statistical properties of community structure in large social and information networks*, in Proceeding of the 17th International Conference on World Wide Web (WWW '08), ACM, New York, 2008, pp. 695–704.
[25] 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 Math., 6 (2009), pp. 29–123.
[26] D. Liben-Nowell and J. Kleinberg, *The link-prediction problem for social networks*, J. Amer. Soc. Inform. Sci. Technol., 58 (2007), pp. 1019–1031.

[27] Z. Lu, B. Savas, W. Tang, and I. S. Dhillon, *Supervised link prediction using multiple sources*, in Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM), 2010, pp. 923–928.

[28] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On Spectral Clustering: Analysis and an Algorithm*, in Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 849–856.

[29] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124, doi:10.1137/080736417.

[30] B. Savas and I. S. Dhillon, *Clustered low rank approximation of graphs in information science applications*, in Proceedings of the 11th SIAM International Conference on Data Mining (SDM), SIAM, Philadelphia, 2011, pp. 164–175.

[31] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intelligence, 22 (2000), pp. 888–905.

[32] D. Shin, S. Si, and I. S. Dhillon, *Multi-scale link prediction*, in Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM), ACM, New York, 2012, pp. 215–224.

[33] S. Si, D. Shin, I. S. Dhillon, and B. N. Parlett, *Multi-scale spectral decomposition of massive graphs*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, 2014, pp. 2798–2806.

[34] 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*, in Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, ACM, New York, 2012, pp. 331–342.

[35] X. Sui, T.-H. Lee, J. J. Whang, B. Savas, S. Jain, K. Pingali, and I. S. Dhillon, *Parallel clustered low-rank approximation of graphs and its application to link prediction*, in Languages and Compilers for Parallel Computing, H. Kasahara and K. Kimura, eds., Lecture Notes in Comput. Sci. 7760, Springer, Berlin, Heidelberg, 2013, pp. 76–95.

[36] V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. S. Dhillon, *Scalable affiliation recommendation using auxiliary networks*, ACM Trans. Intell. Syst. Technol., 3 (2011), pp. 3:1–3:20.

[37] D. Wagner and F. Wagner, *Between min cut and graph bisection*, in Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS '93), London, UK, 1993, Springer-Verlag, Berlin, Heidelberg, pp. 744–750.

[38] J. J. Whang, I. S. Dhillon, and D. F. Gleich, *Non-exhaustive, overlapping k-means*, in Proceedings of the 2015 SIAM International Conference on Data Mining (SDM), SIAM, Philadelphia, 2015, pp. 936–944, doi:10.1137/1.9781611974010.105.

[39] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 15 (1993), pp. 1101–1113.

[40] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. Anthropological Res., 33 (1977), pp. 452–473.

[41] D. Zeimpekis and E. Gallopoulos, *CLSI: A flexible approximation scheme from clustered term-document matrices*, in Proceedings of the 2005 SIAM International Conference on Data Mining (SDM), SIAM, Philadelphia, 2005, pp. 631–635, doi:10.1137/1.9781611972757.75.

[42] D. Zeimpekis and E. Gallopoulos, *Linear and non-linear dimensional reduction via class representatives for text classification*, in Proceedings of the Sixth International Conference on Data Mining (ICDM '06), IEEE Computer Society, 2006, pp. 1172–1177.

[43] H. Zha, C. Ding, M. Gu, X. He, and H. Simon, *Spectral relaxation for k-means clustering*, in Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 1057–1064.

[44] K. Zhang and J. T. Kwok, *Clustered Nyström method for large scale manifold learning and dimension reduction*, IEEE Trans. Neural Networks, 21 (2010), pp. 1576–1587.

[45] Z. Zhang and H. Zha, *Structure and perturbation analysis of truncated SVDs for column-partitioned matrices*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1245–1262, doi:10.1137/S0895479899357875.