CLUSTERED MATRIX APPROXIMATION

BERKANT SAVAS[*] AND INDERJIT S. DHILLON[†]

Abstract. In this paper we motivate and develop a novel clustered matrix approximation framework. 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 rank-wise optimal truncated SVD approximations; The framework is also quite flexible as it may be modified in various ways to fit the needs in 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, co-clustering.

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

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 [12]. 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 with the k largest singular values. A key property of the SVD is that it gives the means to analyse, interpret and understand the original data in terms of globally optimal factors. This is a direct consequence from 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 one 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

[*] Department of Science and Technology, Linköping University
Email: berkant.savas@liu.se.

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

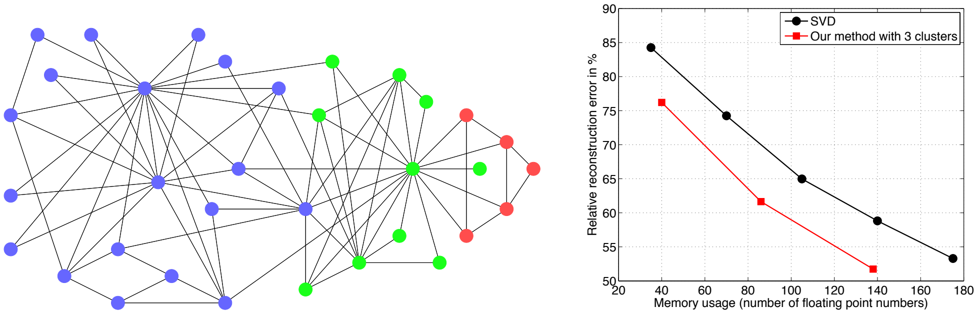

Fig. 1.1. *Left: The karate club network partitioned into three clusters using spectral graph partitioning. Cluster affiliation is indicated by the coloring of the nodes. Right: Relative reconstruction error versus memory consumption. We see that our method produces considerably lower reconstruction error than the rank-wise optimal spectral approximation. 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.*

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. Another key benefit of our method is that it produces a considerably more accurate matrix approximation than the truncated SVD approximation, for the same or less amount of memory usage. 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 [35], illustrated in the left panel of Figure 1.1, 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 $k = 1, 2, \dots, 5$. 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 1.1. In the same figure, we have plotted the relative reconstruction errors for approximations obtained with our method based on partitioning the graph in three clusters (details given later). The clustering is shown in Figure 1.1 and obtained using spectral graph partitioning [7, 23].

The results shown in Figure 1.1 are clear; approximations obtained with our method are more accurate by a large margin than truncated spectral approximation.

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

Observe that comparison is made with respect to memory consumption and not rank of the approximation. Consider in particular the approximation error for $V_4\Lambda_4V_4^\mathsf{T}$ (fourth data point), this approximation uses 140 floating point numbers and has 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 and in particular information scientific applications. A few examples are *information retrieval using latent semantic indexing* [8, 4], *link prediction* and *affiliation recommendation in social networks* [21, 17, 22, 30, 29, 31, 27]. 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 [13]. 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 to (mostly) memory constraints. Low rank approximations give 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 non-symmetric) and rectangular matrices, and involve four steps: (1) a clustering or co-clustering 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 block-wise 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 [16] 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 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 main contributions of this paper. These are: 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. Finally, in section 5 we present our conclusions.

1.4. Notation.

Matrices will be denoted with capital roman letters, e.g., A, U, V, A_i, or A_{ij}. Lower case 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$(A_1, \cdots, A_k)$ as the $k \times k$ block matrix with A_1 to A_k as diagonal blocks. We will use orth(X)[2] to denote 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 co-clustering; 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 co-clustering.

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 characterised by a set of vertices $\mathcal{V} = \{\nu_1, \cdots, \nu_m\}$ 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 ν_i and ν_j then $a_{ij} = 0$. The clustering problem amounts to partition the vertices $\mathcal{V}$ into c disjoint sets $\mathcal{V}_1, \ldots, \mathcal{V}_c$. Extensive research has been conducted to develop theory [34, 15, 26, 32, 23, 10, 14, 33] (to mention a few) and efficient software packages as GRACLUS [10], and METIS [1] for graph clustering. 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 obtained c disjoint sets $\mathcal{V}_1, \cdots, \mathcal{V}_c$, with $m_i = |\mathcal{V}_i|$. Without loss of generality, we can assume that vertices in $\mathcal{V}_1, \ldots, \mathcal{V}_c$ are sorted in strictly increasing order. Then, the 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} 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 non-zero 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

[2] Using the QR factorization $QR = [Q_1 \; Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = X$ we may set orth$(X) = Q_1$.

[3] We would like to remark that not all graphs contain natural clusters, but many graphs are clusterable to some extent [20, 19]. This is the case for graphs arising in many real-world applications.

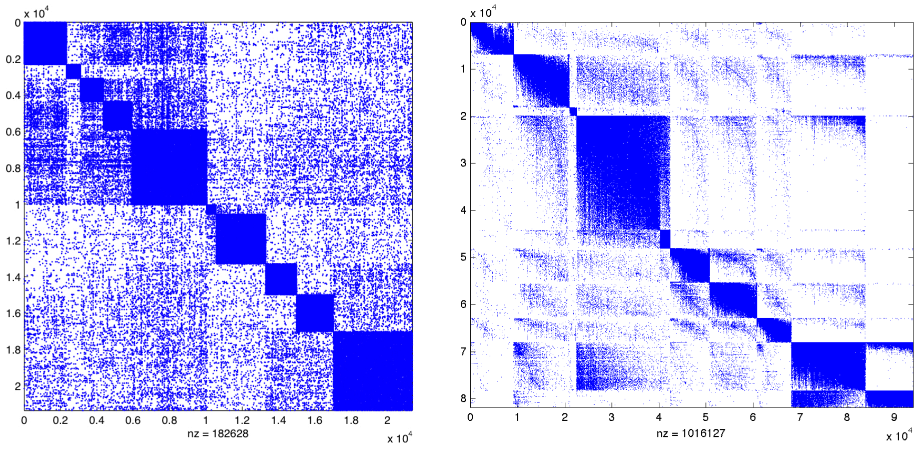

Fig. 2.1. *Left panel: Spy plot of the clustering structure of the arXiv condensed matter collaboration network [18]. The graphs contains 21,363 vertices and 182,628 edges. 79.8% of the edges are within the ten diagonal blocks. Right panel: Spy plot of the co-clustering structure of the Notre Dame bipartite graph between 81,823 movies and 94,003 actors [3]. The associated matrix has 1,016,127 non-zeros and 83.4% of the non-zeros are contained within the ten diagonal blocks.*

block partitioning revealing the underlying cluster structure of a graph is given in the left panel of Figure 2.1. In section 4.2 and Table 4.1 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, co-clustering [9, 36] may be used to extract a corresponding structural information. Formally, a bipartite graph $G = (\mathcal{R}, \mathcal{C}, \mathcal{E})$ is characterised by two sets of vertices: $\mathcal{R} = \{r_1, \cdots, r_m\}$ and $\mathcal{C} = \{c_1, \cdots, c_n\}$, 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 co-clustering will yield disjoint row sets $\mathcal{R}_1, \cdots, \mathcal{R}_r$ and disjoint columns sets $\mathcal{C}_1, \cdots, \mathcal{C}_c$. Again, without loss of generality, we 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 2.1 shows the co-clustering 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 co-clustering methods in [9, 36] 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 number of clusters in $\mathcal{R}$ and $\mathcal{C}$. Alternatively, the block structure in (2.2) may be extracted by casting the problem to a regular graph

[4] For example by running the k-means algorithm independently and with different number of clusters on the two blocks of equation (12) in [9].

Algorithm 1 Randomized range finder [16].

```
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 $Q = \text{orth}(Y)$.
```

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 [16, 11, 6, 24]. 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] Ω, where p is a small oversampling parameter (typically set to 5–10). Multiplying A with the random matrix Ω we obtain $Y = A\Omega$. 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 $\Sigma_1 = \text{diag}(\sigma_1, \cdots, \sigma_k)$ and $\Sigma_2 = \text{diag}(\sigma_{k+1}, \cdots, \sigma_n)$. The matrices U and V are partitioned accordingly. Introduce also

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

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

[5] 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 ([6, 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 Ω 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 Ω_1, Ω_2 as in (2.5). 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 $n \times (k+p)$ standard Gaussian matrix. With the notation as in Theorem 2.1 we have*

$$\mathbb{E}\|(I - P_Y)A\|_F \leq \Big(1 + \frac{k}{p-1}\Big)^{1/2} \|\Sigma_2\|_F,$$
$$\mathbb{E}\|(I - P_Y)A\|_2 \leq \Big(1 + \frac{\sqrt{k}}{\sqrt{p-1}}\Big)\|\Sigma_2\|_2 + \frac{\mathsf{e}\sqrt{k+p}}{p}\|\Sigma_2\|_F,$$

where 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 $q \lesssim 3$.

3. Clustered low rank matrix approximations.

The main goal of this section is to develop a framework for memory efficient matrix approximations that preserves important structural information of the data. In a recent publication Savas and Dhillon [25] 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 application [29, 27, 31]. Subsequently a multilevel clustering approach was developed in order to speed up the computation of the dominant eigenvalues and eigenvecotrs of massive graphs [28]. In section 3.1, we initiate our presentation by showing how to use the clustered low rank approximation on rectangular matrices (bipartite graphs) as well. 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:

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

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 co-clustering the associated bipartite graph. We assume for both cases that the diagonal blocks A_{ii} are much denser in terms of non-zero entries than the off-diagonal blocks A_{ij}, as in Figure 2.1. 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]

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

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

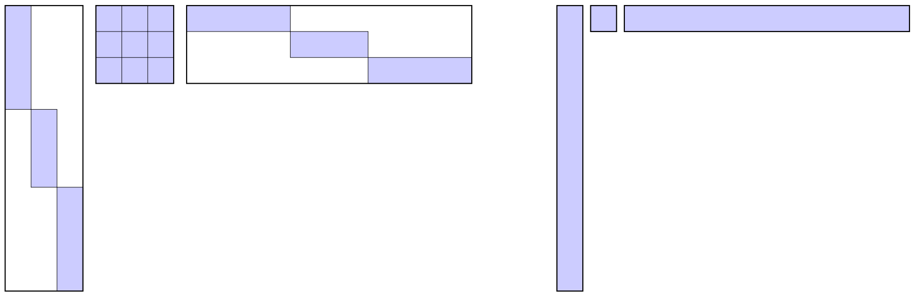

FIG. 3.1. *Graphical comparison between the clustered matrix approximation* $A \approx \bar{U}\bar{S}\bar{V}^{\mathsf{T}}$ *in* (3.3) *using* $c = 3$ *and a regular truncated SVD approximation in* (1.1)*. The memory usage in* $\bar{U}$ *and* $\bar{V}$ *is restricted only to the diagonal (shaded) blocks.*

Algorithm 2 Clustered matrix approximation with diagonal block structure

```
Input: $A$, number of clusters $c$, and ranks $k_{11}, \cdots, k_{cc}$.
Output: $U_i$, $V_j$, $S_{ij}$ for $i, j = 1, \cdots, c$.
  Partition $A$ into $c \times c$ blocks by using a clustering or co-clustering algorithm.
  Reorder rows and columns of $A$ according to cluster belonging to yield a block
    structure as in (3.1).
  Compute low rank approximations of the diagonal blocks according to (3.2).
  Set $S_{ii} = \Sigma_i$ and compute $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ when $i \neq j$ to obtain $\bar{S}$.
```

We can then construct an approximation of A as

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

where $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$. This choice of $\bar{S}$ yields the optimal approximation of A in least squares sense for the given orthonormal $\bar{U} = \operatorname{diag}(U_1, \cdots, U_c)$ and $\bar{V} = \operatorname{diag}(V_1, \cdots, V_c)$. 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 very 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 3.1 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.

3.2. Non-diagonal dense block structure. Analysis of clustering and co-clustering results reveals structure of the data in terms of dense diagonal blocks. However, when the data does not form good clusters, additional structure outside the

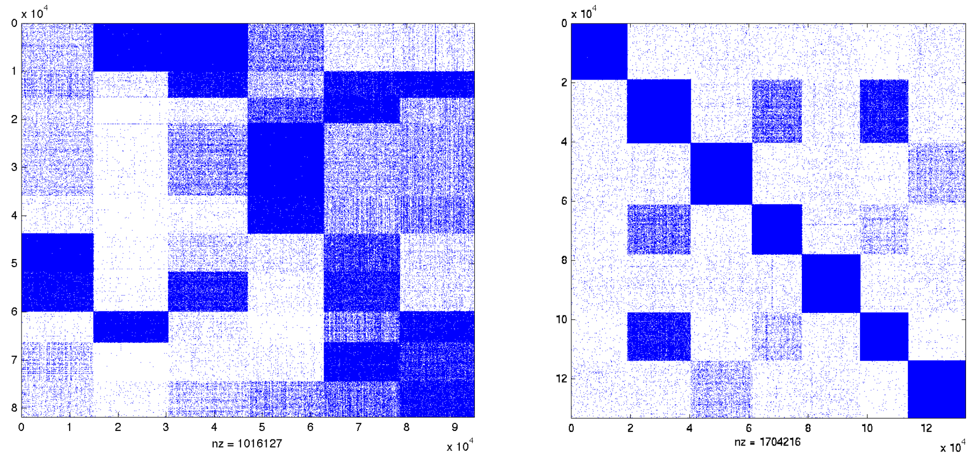

FIG. 3.2. *Left panel: Cluster structure of a rectangular matrix (bipartite graph) with 10 row clusters and 6 column clusters that are obtained independently. Right panel: Clustering of a matrix (graph) into 7 clusters. In both panels there is more than one dense block in a block row or block column.*

diagonal blocks is revealed as well. A more general kind of block structure is revealed when partitioning the rows and columns of a rectangular matrix independently, in particular when using different number of row and column clusters. Both of these scenarios are illustrated in Figure 3.2.

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 $\bar{U} = \operatorname{diag}(U_1, \cdots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \cdots, V_c)$ we can achieve both of these goals. In order to preserve 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 3×4 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 $V_1, \cdots, V_4$ optimal S_{ij} are obtained, as previously, with $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$.

Remark. 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 certain amount of information from A_{11} and 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 $\mathcal{R}_i = \{(i, c_{i,1}), \cdots, (i, c_{i,|\mathcal{R}_i|})\}$ with dense blocks indices from the i'th block row, and similarly $\mathcal{C}_j = \{(r_{j,1}, j), \cdots, (r_{j,|\mathcal{C}_j|}, j)\}$ with dense blocks 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. Next step is to compute blocks in $\bar{U} = \operatorname{diag}(U_1, \cdots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \cdots, V_c)$ according to

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

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

Algorithm 3 Clustered matrix approximation with non-diagonal block structure

```
Input: $A$, number of row clusters $r$, number of column clusters $c$,
Output: Block matrices that form the clustered low rank approximation: $U_1, \cdots, U_r$,
    $V_1, \cdots, V_c$, and $S_{ij}$ with $i = 1, \cdots, r$ and $j = 1, \cdots, c$.
Partition $A$ into $r \times c$ blocks using a clustering or co-clustering algorithm.
Determine the dense blocks $A_{ij}$ and store their indices $(i, j)$ in $\mathcal{S}$.
Determine ranks $k_{ij}$ in the low rank approximations of $A_{ij}$ for $(i, j) \in \mathcal{S}$.
Compute low rank approximation using the truncated SVD according to (3.7).
Compute the diagonal blocks in $\bar{U} = \operatorname{diag}(U_1, \cdots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \cdots, V_c)$
    according to equations (3.8) and (3.9), respectively.
Set $S_{ij} = \Sigma_{ij}$ when $(i, j) \in \mathcal{S}$ and compute $S_{ij} = U_i^{\mathsf{T}} A_{ij} V_j$ otherwise.
```

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

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

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 block-wise 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.

Remark. There are important aspects in the current presentation that we do not adress. For example, how to choose the number of row clusters r and number of column clusters c? Given this, how to determine if a given block is sufficiently dense? And subsequently, what ranks k_{ij} to use in the block-wise 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 optimising 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 point of view.

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 non-diagonal 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 Ω is a random Gaussian matrix. In this scenario without clustering, Y uniquely determines the approximation as $A \approx QQ^{\mathsf{T}}A$, where $Q = \operatorname{orth}(Y)$. 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 columns space and the row space of the approximation is 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 sets: $\mathcal{S}$ containing index paris for all dense blocks; $\mathcal{R}_i \subset \mathcal{S}$ with index paris for block row i; and $\mathcal{C}_j \subset \mathcal{S}$ with index paris 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

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

we consider the approximation $A \approx P_{\bar{Y}}A$, where $\bar{Y} = \operatorname{diag}(Y_1, \cdots, Y_r)$ and $P_{\bar{Y}}$ is the orthogonal projection onto range$(\bar{Y})$. Using $\bar{Q} = \operatorname{orth}(\bar{Y})$ we can express the approximation as $A \approx \bar{Q}(\bar{Q}^{\mathsf{T}}A)$. It follows that $\bar{Q}$ has the same blockdiagonal structure as $\bar{Y}$. However, the matrix $\bar{Q}^{\mathsf{T}}A$, representing the associated row space, will not have a blockdiagonal 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 $Q_{ij} = \operatorname{orth}(Y_{ij})$ we get the approximations

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

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 $\bar{U} = \operatorname{diag}(U_1, \cdots, U_r)$ and $\bar{V} = \operatorname{diag}(V_1, \cdots, V_c)$ are obtained from

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

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

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 3.1 and 3.2. 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 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 $n_j \times (k_{ij}+p_{ij})$ standard Gaussian matrices $\Omega^{(ij)}$ and let $Y_{ij} = A_{ij}\Omega^{(ij)}$. Define further

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

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

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 $\bar{U} = \mathrm{diag}(U_1, \cdots, U_r)$, $\bar{V} = \mathrm{diag}(V_1, \cdots, V_c)$, 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 $\mathrm{range}(Y_{ij})$. 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 $\mathrm{range}(\tilde{U}^{(ij)}) \subseteq \mathrm{range}(U_i)$ as well as $\mathrm{range}(\tilde{V}^{(ij)}) \subseteq \mathrm{range}(V_j)$ 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 3.1. *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 is 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 $k_{ij} + p_{ij} \le \min(m_i, n_j)$. 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*

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

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.

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

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 $\hat{A}_{ij}=U_iS_{ij}V_j^{\mathsf{T}}$.
2. In (3.19) we split the summation in two parts: one with $(i,j)\in\mathcal{S}$, that corresponds to the dense A_{ij}, and one with $(i,j)\in\mathcal{T}$, that 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
$$\|A_{ij}-U_iS_{ij}V_j^{\mathsf{T}}\|_* = \|A_{ij}-\hat{A}_{ij}\|_* \le \|A_{ij}-\tilde{A}_{ij}\|_* = \|A_{ij}-\tilde{U}^{(ij)}\tilde{\Sigma}^{(ij)}(\tilde{V}^{(ij)})^{\mathsf{T}}\|_*.$$
Recall that $\operatorname{range}(\tilde{U}^{(ij)})\subseteq\operatorname{range}(U_i)$ and $\operatorname{range}(\tilde{V}^{(ij)})\subseteq\operatorname{range}(V_j)$.
3. In (3.20) we use the results of (3.17).
4. Finally in (3.21) we use
$$|(I-P_{Y_{ij}})A_{ij}\|_*^2 \le \|\Sigma_2^{(ij)}\Omega_2^{(ij)}(\Omega_1^{(ij)})^{\dagger}\|_*^2+\|\Sigma_2^{(ij)}\|_*^2, \qquad (i,j)\in\mathcal{S},$$
which is proven in [16, Lemma 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 3.1 we can state the theorem with respect to expected error.

THEOREM 3.2. *Using the notation introduced in theorem 3.1, the expected error norm are bounded by*

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

$$\begin{aligned}
\mathbb{E}\|A-\hat{A}\|_2 \le \sum_{(i,j)\in\mathcal{S}}&\Big(\Big(1+\frac{\sqrt{k_{ij}}}{\sqrt{p_{ij}-1}}\Big)\|\Sigma_2^{(ij)}\|_2+\frac{\mathsf{e}\sqrt{k_{ij}+p_{ij}}}{p_{ij}}\|\Sigma_2^{(ij)}\|_F\Big)+\\
&\sum_{(i,j)\in\mathcal{T}}\|A_{ij}\|_2.
\end{aligned}$$

Proof. Both inequalities follow from a similar analysis as in [16]. □

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 a number of key properties we think are of general importance both from practical and theoretical point of view. These are: 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, 20]. In all experiments we used the largest connected component[8] of each graph. The preprocessing resulted in a $81,823 \times 94,003$ matrix with 1,016,127 non-zeros for the Notre Dame data and a $3,828,682 \times 3,828,682$ non-symmetric matrix with 65,825,429 non-zeros for the LiveJournal data. In both graphs the links are unweighted giving all non-zero 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 non-diagonal 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 publicly available.[9]

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

4.2. Clustering properties of the data. 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 non-zero entries in A, then the fraction of non-zeros 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 block in the non-diagonal case, then $\phi_{\mathcal{S}} = \sum_{(i,j)\in\mathcal{S}} |A_{ij}|/|A|$ is the fraction of non-zero entries of A contained in all dense blocks. We consider a block A_{ij} to be dense if $|A_{ij}|/|A| \geq \tau$, where τ is a threshold value. Three different clusterings is 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 are presented in Table 4.1.

In the left and middle panel of Figure 4.1 we illustrate the dense blocks of A for two cases from Table 4.1. A is structured as in (3.6). The right panel of Figure 4.1 shows the fraction of non-zeros of all blocks in a 61×61 clustering of the LiveJournal matrix. The fact that a small number of blocks have larger fraction of non-zeros illustrate the presence of structure in the network. This structure is preserved in the clustered low rank approximation.

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

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

[9] http://www.cs.utexas.edu/~berkant/cmapp/

TABLE 4.1

Clustering related information. Columns represent: r and c is the number of row and column clusters, respectively; ϕ_d is the fraction of non-zeros within the diagonal blocks; τ 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 non-zeros within the dense blocks; and clustering method.

Data set	$r = c$	ϕ_d in %	τ in %	$\|\mathcal{S}\|$	$r \times c$	$\phi_{\mathcal{S}}$ 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

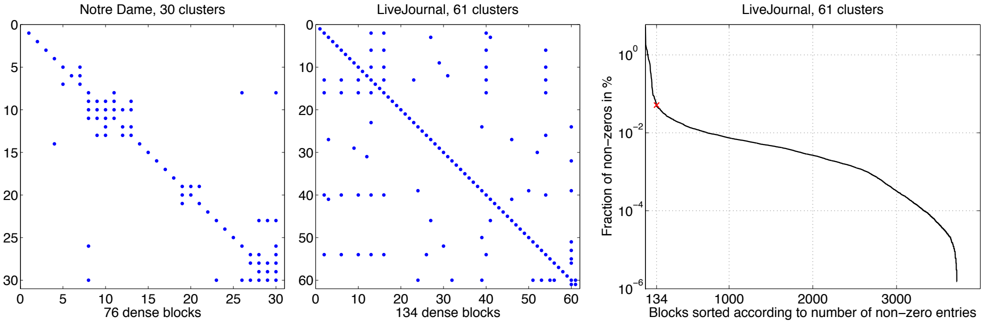

FIG. 4.1. *Left panel: dense blocks of a* 30×30 *clustering of the Notre Dame matrix using* $\tau = 0.2$. *Middle panel: dense blocks in a* 61×61 *clustering of the LiveJournal matrix using* $\tau = 0.05$. *Right panel: fraction of not (in descending order) of all blocks in the* 61×61 *clustering of the LiveJournal matrix. Clearly, most blocks contain very small fraction of non-zeros. The mark at* $x = 134$ *indicates the last block considered to be dense.*

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 4.2 we present results for the LiveJournal data. The left panel shows results using Algorithm 2 for all three clustering cases form Table 4.1. 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 4.1. 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 also see that increasing the number of clusters increases the memory usage. This is particularly obvious for the non-diagonal 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 non-diagonal dense blocks have relatively small fraction of non-zeros compared

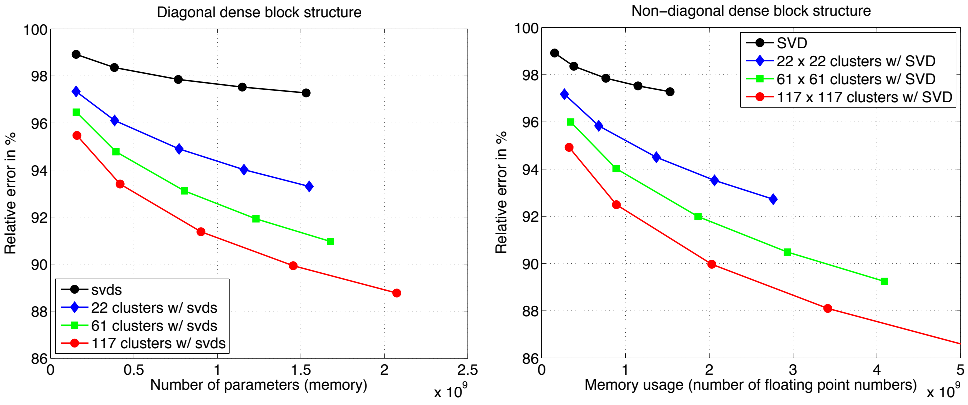

Fig. 4.2. *Relative errors for the LiveJournal matrix approximations. The left panel shows results for the clustered low rank approximation using only the diagonal blocks, while the right panel uses a non-diagonal dense block structure.*

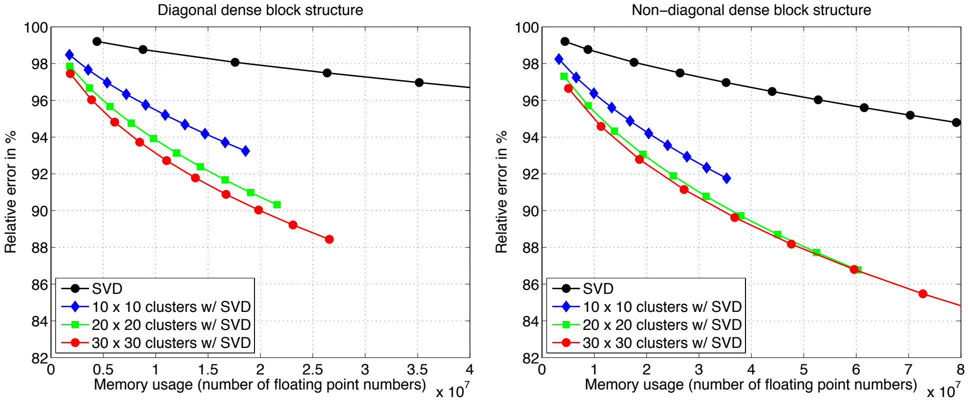

Fig. 4.3. *Relative errors for the Notre Dame matrix. Left panel shows results for the diagonal clustered approximation and the right panel shows the non-diagonal dense block structure.*

to the diagonal blocks, thus approximating these “smaller” blocks with the same rank as the more “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 find a balance between memory usage and desired level of structure preservation.

Notre Dame. In Figure 4.3 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 4.1 for each algorithm. For each clustering case we compute 10 approximations using $k_{ij} \in \mathcal{K}_{\mathrm{ND}} = \{10, 20, \cdots, 100\}$, and for each approximation we use the same k_{ij} for all dense blocks. In truncated SVD approximations we use ranks $k \in \mathcal{K}_{\mathrm{R}} = \{25, 50, 100, 150, \cdots, 500\}$. 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 seams to stagnate. For example, there is only slight improvement in Algorithm 3 (right panel) when increasing the clustering from 20×20 to 30×30. Also in this case one may benefit by using different

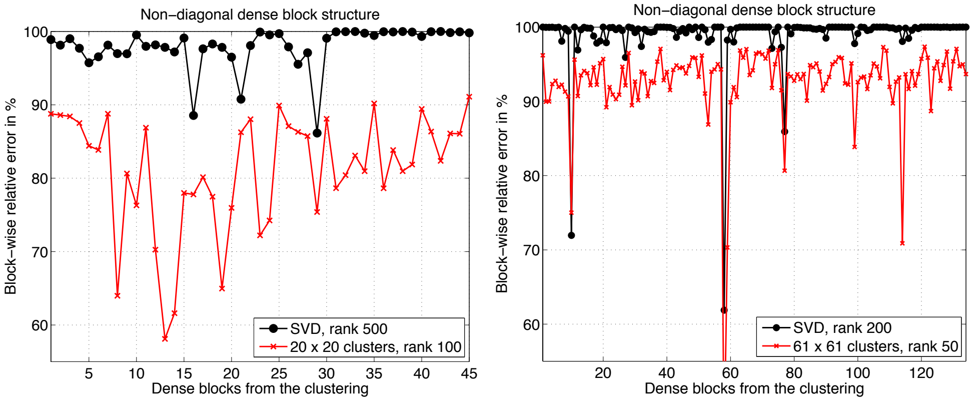

FIG. 4.4. *Information captured from the dense blocks in a regular and clustered matrix approximation. The block-wise relative error is* $\|A_{ij} - \hat{A}_{ij}\|_F / \|A_{ij}\|_F$*, for* $(i,j) \in \mathcal{S}$*. Left panel is for the Notre Dame matrix and right panel is for the LiveJournal matrix.*

ranks for different dense blocks in order to balance approximation quality, memory consumption, and structure preservation.

4.4. Block-wise 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 4.4 we present two experiments that validate this claim. Let A be partitioned as in (3.6). We will investigate and compare the block-wise 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 $8.8 * 10^7$ floating point numbers (this last entry is off the chart in Figure 4.3). Algorithm 3 uses the 20 × 20 clustering from Table 4.1 giving 45 dense blocks. Each dense block is approximated with a rank-100 truncated SVD achieving about 87% in overall relative error with $6 * 10^7$ (about 30% less memory) floating point numbers. We see in Figure 4.4 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 $1.5 * 10^9$ floating point numbers. In Algorithm 3 we use the 61 × 61 clustering from Table 4.1 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 $0.9 * 10^9$ (about 40% less memory) floating point numbers. The right panel of Figure 4.4 shows a similar result. The regular truncated 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%.

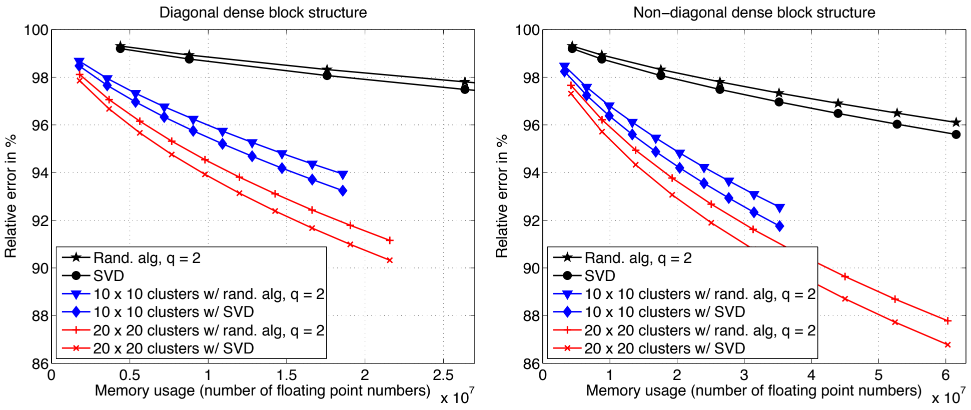

FIG. 4.5. *Relative errors for the Notre Dame matrix approximations. The left panel shows results for the clustered matrix approximation using diagonal dense block structure, while the right panel uses a non-diagonal dense block structure.*

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 non-clusterd methods. We will experiment with the Notre Dame matrix using the 10×10 and 20×20 clusterings from Table 4.1. In the left panel of Figure 4.5 we use Algorithm 2 and a probabilistic version of it, in which the block-wise SVD approximations are replaced with probabilistic approximations. In the right panel we use Algorithm 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 [25]. In both panels we also present non-clustered 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 non-clustered 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 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 4.6 we present two plots with timing results using the *cputime*-function in MATLAB for the experiments in Figure 4.5. 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 are considerably faster than the SVD based methods; (3) There are very small timing differences in the probabilistic methods when considering different number of clusters; (4) We would like to 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

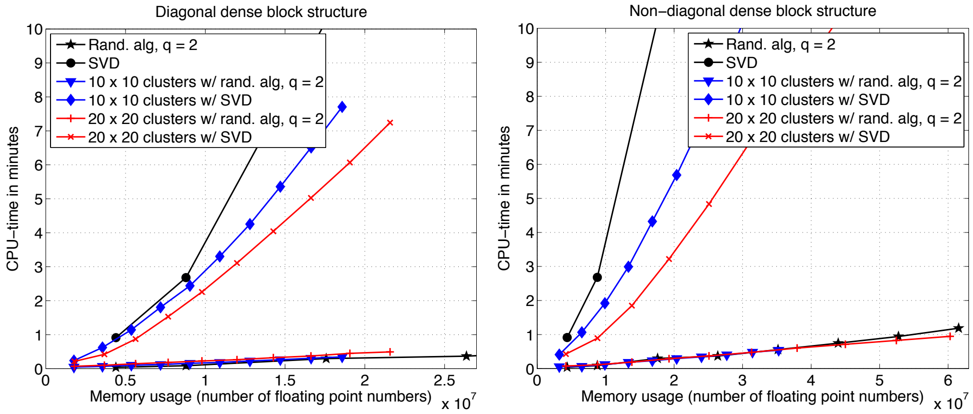

FIG. 4.6. *Timing results corresponding to the experiments in Figure 4.5.*

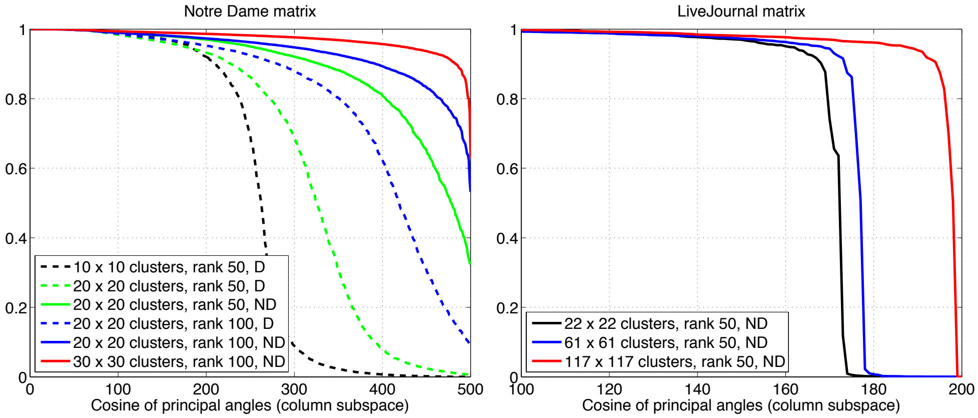

FIG. 4.7. *Left panel: cosine of the principal angles for the Notre Dame matrix for six different experiments. Right panel: cosine of the principal angles for the LiveJournal matrix. "D" in the description indicates diagonal dense structure, and "ND" indicates and non-diagonal dense block structure. Also note that the x-axis for the right panel starts at 100 as all three curves are very close to one in the omitted part.*

answered by examining the singular values σ_i of $U^{\mathsf{T}}\bar{U}$, since $\sigma_i = \cos(\theta_i)$ are cosines of the principal angles between the two subspaces [5]. In Figure 4.7 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 σ_i are close to one. We use the same clusterings 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: (1) increasing the number of clusters; (2) increasing the rank in the block-wise approximations; and (3) using non-diagonal block structure instead of diagonal blocks only; brings range($\bar{U}$) closer to the range(U) in significant jumps. For example, to verify the third claim, compare the two green curves, or the two blue curves.

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 (non-diagonal block structure). We compute one approximation for each clustering setting from Table 4.1, and in each approximation we use $k_{ij} = 50$ in the block-wise approximations. From Figure 4.2 we see that all three clustered approximations use about the same memory usage as a rank-100 truncated SVD approximation. Thus, $\bar{U}$ uses about half the memory rquired 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 approaches range(U) with significant steps.

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

5. Conclusions.

In this paper we have developed a framework for matrix approximations that preserve 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 contain 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 block-wise 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 blockdiagonal $\bar{U}$ 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 the rank-wise optimal truncated SVD approximations; 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, where as 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 KDD '06: Proceedings of the 12th ACM SIGKDD, New York, NY, USA, 2006, pp. 44–54.

[3] Albert-Laszlo Barabási and Réka 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, September 2003.

[5] Å. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Mathematics of Computation, 27 (1973), pp. 579–594.

[6] C. Boutsidis, M. W. Mahoney, and P. Drineas, *An improved approximation algorithm for the column subset selection problem*, in SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, 2009, pp. 968–977.

[7] N. Cristianini, J. Shawe-Taylor, and J. S. Kandola, *Spectral kernel methods for clustering*, in NIPS, 2001, pp. 649–655.

[8] S. Deerwester, *Improving information retrieval with latent semantic indexing*, in Proceedings of the 51st ASIS Annual Meeting (ASIS '88), Christine L. Borgman and Edward Y. H. Pai, eds., vol. 25, American Society for Information Science, 1988, pp. 36–40.

[9] 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, New York, NY, USA, 2001, ACM, pp. 269–274.

[10] 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.

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

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

[13] E. Estrada and D. J. Higham, *Network properties revealed through matrix functions*, SIAM Revew, 52 (2010), pp. 696–714.

[14] 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.

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

[16] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.

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

[18] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graph evolution: Densification and shrinking diameters*, ACM Trans. Knowl. Discov. Data, 1 (2007).

[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 '08: Proceeding of the 17th international conference on World Wide Web, New York, NY, USA, 2008, ACM, pp. 695–704.

[20] ———, *Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters*, Internet Mathematics, 6 (2009), pp. 29–123.

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

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

[23] 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.

[24] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 1100–1124.

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

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

[27] 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), 2012.

[28] 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, Inc., 2014, pp. 2798–2806.

[29] 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, 2012, pp. 331–342.

[30] 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, Hironori Kasahara and Keiji Kimura, eds., vol. 7760 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 76–95.

[31] V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. S. Dhillon, *Scalable affiliation recommendation using auxiliary networks*, ACM Transactions on Intelligent Systems and Technology, 3 (2011), pp. 3:1–3:20.

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

[33] J. Whang, I. S. Dhillon, and D. Gleich, *Non-exhaustive, overlapping k-means*, in Proceedings of the 2015 SIAM International Conference on Data Mining (SDM), 2015.

[34] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993), pp. 1101–1113.

[35] W. W. Zachary, *An information flow model for conflict and fission in small groups*, Journal of Anthropological Research, 33 (1977), pp. 452–473.

[36] Hongyuan Zha, Chris Ding, Ming Gu, Xiaofeng He, and Horst Simon., *Spectral relaxation for k-means clustering*, in Neural Information Processing Systems, vol. 14, 2001, pp. 1057–1064.