SAMPLING AND MULTILEVEL COARSENING ALGORITHMS FOR FAST MATRIX APPROXIMATIONS[*]

SHASHANKA UBARU AND YOUSEF SAAD[†]

Abstract. This paper addresses matrix approximation problems for matrices that are large, sparse and/or that are representations of large graphs. To tackle these problems, we consider algorithms that are based primarily on coarsening techniques, possibly combined with random sampling. A multilevel coarsening technique is proposed which utilizes a hypergraph associated with the data matrix and a graph coarsening strategy based on column matching. Theoretical results are established that characterize the quality of the dimension reduction achieved by a coarsening step, when a proper column matching strategy is employed. We consider a number of standard applications of this technique as well as a few new ones. Among the standard applications we first consider the problem of computing the *partial SVD* for which a combination of sampling and coarsening yields significantly improved SVD results relative to sampling alone. We also consider the *Column subset selection* problem, a popular low rank approximation method used in data related applications, and show how multilevel coarsening can be adapted for this problem. Similarly, we consider the problem of *graph sparsification* and show how coarsening techniques can be employed to solve it. Numerical experiments illustrate the performances of the methods in various applications.

Key words. Singular values, SVD, randomization, subspace iteration, coarsening, multilevel methods.

AMS subject classifications. 15A69, 15A18

1. Introduction.

Many modern applications related to data often involve very large datasets, but their relevant information lie on a low dimensional subspace. In many of these applications, the data matrices are often sparse and/or are representations of large graphs. In recent years, there has been a surge of interest in approximating large matrices in a variety of different ways, such as by low rank approximations [16, 25, 37], graph sparsification [55, 27], and compression [32]. Low rank approximations include the partial singular value decomposition (SVD) [25] and Column Subset Selection (the CSS Problem) [7]. A variety of methods have been developed to efficiently compute partial SVDs of matrices [49, 23], a problem that has been studied for a few decades. However, traditional methods for partial SVD computations cannot cope with very large data matrices. Such datasets prohibit even the use of rather ubiquitous methods such as the Lanczos or subspace iteration algorithms [49, 50], since these algorithms require consecutive accesses to the whole matrix multiple times. Computing such matrix approximations is even harder in the scenarios where the matrix under consideration receives frequent updates in the form of new columns or rows.

Much recent attention has been devoted to a class of 'random sampling' techniques [15, 16, 25] whereby an approximate partial SVD is obtained from a small subset of the matrix only, or possibly a few subsets. Random sampling is well-established (theoretically) and is proven to give good results in some situations, see [37] for a review. In this paper we will consider random sampling methods as one of the tools to down sample very large datasets. However, because randomized methods assume no prior information on the data and are independent of the input matrix they are often termed "data-oblivious" [1]. Because of this feature, they can be suboptimal in many situations since they do not exploit any available information or structures in

[*]This work was supported by NSF under the grant NSF/CCF-1318597.

[†]Department of Computer Science and Engineering, University of Minnesota at Twin Cities, MN 55455. Email: ubaru001@umn.edu, saad@umn.edu.

the matrix. One of the goals of this work is to show that multilevel graph coarsening techniques [26] can be good alternatives to randomized sampling.

Coarsening a graph (or a hypergraph) $G = (V, E)$ means finding a 'coarse' approximation $\bar{G} = (\bar{V}, \bar{E})$ to G with $|\bar{V}| < |V|$, which is a reduced representation of the original graph G, that retains as much of the structure of the original graph as possible. Multilevel coarsening refers to the technique of recursively coarsening the original graph to obtain a succession of smaller graphs that approximate the original graph G. Several methods exist in the literature for coarsening graphs and hypergraphs [26, 28, 9, 29]. These techniques are relatively more expensive than down-sampling with column norm probabilities [16] but they are more accurate. Moreover, coarsening methods will be inexpensive compared to the popular leverage scores based sampling [17] which is more accurate than norm sampling. For very large matrices, a typical algorithm would first perform randomized sampling to reduce the size of the problem and then utilize a multilevel coarsening technique for computing an approximate partial SVD of the reduced matrix.

Our Contribution. In this paper, we present a multilevel coarsening technique that utilizes a hypergraph associated with the data matrix and a coarsening strategy that is based on column matching, and discuss various applications for this technique. We begin by discussing different approaches to find partial SVD of large matrices, starting with random sampling methods. We also consider incremental sampling, where we start with small samples and then increase the size until a certain criterion is satisfied. The second approach is to replace random sampling, with a form of multilevel coarsening technique. A middle ground solution is to start with random coarsening and then utilize multilevel coarsening on the resulting sampled subset. The coarsening techniques exploit inherent redundancies and structures in the matrix and perform better than randomized sampling in many cases as is confirmed by the experiments. We establish theoretical error analysis for a class of coarsening techniques. We also show how the SVD update approach, see [65] or subspace iteration can be used after the sampling or coarsening step to improve the SVD results. This approach is useful when an accurate SVD of a large matrix is desired.

The second low rank approximation problem considered in this paper is that of *column subset selection problem* [7, 66] (CSSP) or CUR decomposition [36, 17]. Popular methods for CSSP use leverage score sampling method for sampling/selecting the columns. Computing the leverage scores requires a partial SVD of the matrix and this may be expensive, particularly for large matrices and when the (numerical) rank is not small. In this work, we show how the graph coarsening techniques can be adapted for column subset selection (CSSP). The coarsening approach is an inexpensive alternative for this problem and performs well in many situations.

The third problem we consider is that of *graph sparsification* [31, 55, 27]. Here, given a large (possibly dense) graph G, we wish to obtain a sparsified graph $\tilde{G}$ that has significantly fewer edges than G but still maintains important properties of the original graph. Graph sparsification allows one to operate on large (dense) graphs G with a reduced space and time complexity. In particular, we are interested in spectral sparsifier, where the Laplacian of $\tilde{G}$ spectrally approximates the Laplacian of G [56, 27, 67]. That is, the spectral norm of the Laplacian of the sparsified graph is close to the spectral norm of the Laplacian of G, within a certain additive or multiplicative factor. Such spectral sparsifiers can help approximately solve linear systems with the Laplacian of G and to approximate effective resistances, spectral clusterings, random walk properties, and a variety of other computations. We again show how the graph coarsening techniques can be adapted to achieve graph sparsifications. We also present a few new applications for coarsening methods, see section 2.

Outline. The outline of this paper is as follows. Section 2, discusses a few applications of graph coarsening. Section 3 describes existing popular algorithms that are used for low rank approximation. The graph coarsening techniques and the multilevel algorithms are described in sec. 4. In particular, we present a hypergraph coarsening technique based on column matching. We also discuss methods to improve the SVD obtained from randomized and coarsening methods. In section 5, we establish a theoretical error analysis for the coarsening method. We also discuss the existing theory for randomized sampling and subspace iteration. Numerical experiments illustrating the performances of these methods in a variety of applications are presented in section 6.

2. Applications.

We present a few applications of (multilevel) coarsening methods. In these applications, we typically encounter large matrices, and these are often sparse and/or representations of graphs.

i. Latent Semantic Indexing. Latent semantic indexing (LSI) is a popular text mining technique for analyzing a collection of documents that are similar [13, 33, 5, 30]. Given a user's query, the method is used to retrieve a set of documents from a given collection that are relevant to the query. Truncated SVD [5] and related methods [30] are popular tools used in the LSI applications. The argument is that a low rank approximation preserves the important underlying structure associated with terms and documents, and removes the noise or variability in word usage [16]. Multilevel coarsening for LSI was considered in [51]. In this work, we revisit this idea and show how hypergraph coarsening can be employed in this application.

ii. Projective clustering. Several projective clustering methods such as Isomap [58], Local Linear Embedding (LLE) [47], spectral clustering [40], subspace clustering [43, 18], Laplacian eigenmaps [4] and others involve partial eigen-decomposition and SVD computation of a graph Laplacian. Various kernel based learning methods [39] also involve SVD computation of large graph Laplacians. In most applications today, the number of data-points are large and computing the singular vectors (eigenvectors) will be expensive. Graph coarsening is a handy tool to reduce the number of data-points in these applications, see [20, 41] for results.

iii. Eigengene analysis. Analysis of gene expression DNA microarray data has become an important tool when studying a variety of biological processes [2, 46, 44]. In a microarray dataset, we have m genes (from m individuals possibly from different populations) and a series of n arrays probe genome-wide expression levels in n different samples, possibly under n different experimental conditions. The data is large with several individuals and gene expressions, but is known to be of low rank. Hence, it has been shown that a small number of eigengenes and eigenarrays (few singular vectors) are sufficient to capture most of the gene expression information [2]. Article [44] showed how column subset selection (CSSP) can be used for selecting a subset of gene expressions that describe the population well in terms of spectral information captured by the reduction. In this work, we show how hypergraph coarsening can be adapted to choose a good (small) subset of genes in this application.

iv. Multilabel Classification. The last application we consider is that of multilabel classification in machine learning applications [60, 61]. In the multilabel classification problem, we are given a set of labeled training data $\{(x_i, y_i)\}_{i=1}^{n}$, where each $x_i \in \mathbb{R}^p$ is an input feature for a data instance which belongs to one or more classes, and $y_i \in \{0, 1\}^d$ are vectors indicating the corresponding labels (classes) to which the data instances belong. A vector y_i has a one at the jth coordinate if the instance belongs to j-th class. We wish to learn a mapping (prediction rule) between the features and the labels, in order to be able to predict a class label vector y of a new data point x. Such multilabel classification problems occur in many domains such as computer vision, text mining, and bioinformatics [59, 57], and modern applications involve a large number of labels.

A popular approach to handle classification problems with many classes is to begin by reducing the effective number of labels by means of so-called embedding-based approaches. The label dimension is reduced by projecting label vectors onto a low dimensional space, based on the assumption that the label matrix $Y = [y_1, \ldots, y_n]$ has a low-rank. The reduction is achieved in different ways, for example, by using SVD in [57] and column subset selection in [6]. In this work, we demonstrate how hypergraph coarsening can be employed to reduce the number of classes, and yet achieve accurate learning and prediction.

Article [54] discusses a number of methods that rely on clustering the data first in order to built a reduced dimension representation. It can be viewed as a top-down approach whereas coarsening is a bottom-up method.

3. Background.

In this section, we review three popular classes of methods used for calculating the partial SVD of matrices. The first class is based on randomized sampling. We also consider the column subset selection (CSSP) and graph sparsification problems using randomized sampling, in particular leverage score sampling. The second class is the set of methods based on subspace iteration, and the third is the set of SVD-updating algorithms [68, 65]. We consider the latter two classes of methods as tools to improve the results obtained by sampling and coarsening methods. Hence, we are particularly interested in the situation where the matrix A under consideration receives updates in the form of new columns. In fact when coupling with the multilevel algorithms (which we will discuss in sec. 4), these updates are not small since the number of columns can double.

3.1. Random sampling.

Randomized algorithms have become popular in recent years due to their broad applications and the related theoretical analysis developed which give results that are independent of the matrix spectrum. Several 'randomized embedding' and 'sketching' methods have been proposed for low rank approximation and for computing the partial SVD [38, 35, 25, 62] starting with the seminal work of Frieze et al. [21]. Drineas et al. [15, 16] presented the randomized subsampling algorithms, where a submatrix (certain columns of the matrix) is randomly selected based on a certain probability distribution. Their method samples the columns based on column norms. Given a matrix $A \in \mathbb{R}^{m\times n}$, they sample its columns such that the i-th column is sampled with the probability p_i given by

$$p_i = \frac{\beta\|A^{(i)}\|_2^2}{\|A\|_F^2},$$

where $\beta < 1$ is a positive constant and $A^{(i)}$ is the i-th column of A. Using the above distribution, c columns are selected and the subsampled matrix C is formed by scaling the columns by $1/\sqrt{cp_i}$. Then, the SVD of C is computed. The approximations obtained by this randomization method will yield reasonable results only when there is a sharp decay in the singular value spectrum.

3.2. Column Subset Selection.

Another popular dimensionality reduction method which we consider in this paper is the column subset selection (CSSP) [7]. If a subset of the rows is also selected, then the method leads to the CUR decomposition [36]. These methods can be viewed as extensions of the randomized sampling based algorithms. Let $A \in \mathbb{R}^{m\times n}$ be a large data matrix whose columns we wish to select and suppose V_k is a matrix whose columns are the top k right singular vectors of A. Then, the leverage score of the i-th column of A is given by

$$\ell_i = \frac{1}{k}\|V_k(i,:)\|_2^2,$$

the scaled square norm of the i-th row of V_k. Then, in leverage scores sampling, the columns of A are sampled using the probability distribution $p_i = \min\{1, \ell_i\}$. The most popular methods for CSSP involve the use of this leverage scores as the probability distribution for columns selection [17, 7, 36, 8]. Greedy subset selection algorithms have been also proposed based on the right singular vectors of the matrix [44, 3]. However, these methods may be expensive since one needs to compute the top k singular vectors. In this work, we see how the coarsened graph, i.e., the columns obtained by graph coarsening perform in CSSP.

3.3. Graph Sparsification. Sparsification of large graphs has several computational (cost and space) advantages and has hence found many applications [31, 34, 53, 55, 56]. Given a large graph $G = (V, E)$ with n vertices, we wish to find a sparse approximation to this graph that preserves certain information of the original graph such as the spectral information [56, 27], structures like clusters within in the graph [31, 34], etc. Let $B \in \mathbb{R}^{\binom{n}{2}\times n}$ be the vertex edge incidence matrix of the graph G, where eth row b_e of B for edge $e = (u, v)$ of the graph has a value $\sqrt{w_e}$ in columns u and v, and zero elsewhere. The corresponding Laplacian of the graph is then given by $K = B^TB$.

The spectral sparsification problem involves computing a weighted subgraph $\tilde{G}$ of G such that if $\tilde{K}$ is the Laplacian of $\tilde{G}$, then $x^T\tilde{K}x$ is close to x^TKx for any $x \in \mathbb{R}^n$. Many methods have been proposed for the spectral sparsification of graphs, see e.g., [55, 56, 27, 67]. A popular approach is to perform row sampling of the matrix B using the leverage score sampling [27]. Considering the SVD of $B = U\Sigma V^T$, the leverage scores ℓ_i for a row b_i of B can be computed as $\ell_i = \|u_i\|_2^2 \leq 1$ using the rows of U. This leverage score is related to the effective resistance of edge i [55]. By sampling the rows of B according to their leverage scores it is possible to obtain a matrix $\tilde{B}$, such that $\tilde{K} = \tilde{B}^T\tilde{B}$ and $x^T\tilde{K}x$ is close to x^TKx for any $x \in \mathbb{R}^n$. In section 4, we show how the rows of B can we selected via coarsening.

3.4. Subspace iteration. Subspace iteration is a well-established method used for solving eigenvalue and singular value problems [23, 49]. We review this algorithm as it will be exploited later as a tool to improve SVD results obtained by sampling and coarsening methods. A known advantage of the subspace iteration algorithm is that it is very robust and that it tolerates changes in the matrix [50]. This is important in our context. Let us consider a general matrix $A \in \mathbb{R}^{m\times n}$, not necessarily associated with a graph. The subspace iteration algorithm can easily be adapted to the situation where a previous SVD is available for a smaller version of A with fewer rows or columns, obtained by subsampling or coarsening for example. Indeed, let A_s be a column-sampled version of A. In matlab notation we represent this as $A_s = A(:, J_s)$ where J_s is a subset of the column index $[1 : n]$. Let A_t be another subsample of A, where we assume that $J_s \subset J_t$. Then if $A_s = U_s\Sigma_sV_s^T$, we can perform a few steps of subspace iteration updates as shown in Algorithm 1.

3.5. SVD updates from subspaces. A well known algorithm for updating the SVD is the 'updating algorithm' of Zha and Simon [68]. Given a matrix $A \in \mathbb{R}^{m\times n}$

Algorithm 1 Incremental Subspace Iteration

```
Start: $U = U_s$
for $i = 1 : \text{iter}$ do
    $V = A_t^T U$
    $U = A_t V$
    $U := qr(U, 0)$;    $V := qr(V, 0)$;
    $S = U^T A_t V$
    if condition then
        // Diagonalize $S$ to obtain current estimate of singular vectors and values
        $[R_U, \Sigma, R_V] = \text{svd}(S)$; $U := U_{i+1} R_U$; $V := V_{i+1} R_V$
    end if
end for
```

and its partial SVD $[U_k, \Sigma_k, V_k]$, the matrix A is updated by adding columns D to it, resulting in a new matrix $A_D = [A, D]$. The algorithm then first computes

$$(1) \qquad (I - U_k U_k^T)D = \hat{U}_p R,$$

the truncated QR decomposition of $(I - U_k U_k^T)D$, where $\hat{U}_p \in \mathbb{R}^{m \times p}$ has orthonormal columns and $R \in \mathbb{R}^{p \times p}$ is upper triangular. Given (1), one can observe that

$$(2) \qquad A_D = [U_k,\ \hat{U}_p] H_D \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}^T,\ H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & R \end{bmatrix},$$

where I_p denotes the p-by-p identity matrix. Thus, if Θ_k, F_k, and G_k are the matrices corresponding to the k dominant singular values of $H_D \in \mathbb{R}^{(k+p)\times(k+p)}$ and their left and right singular vectors, respectively, then the desired updates $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ are given by

$$(3) \qquad \tilde{\Sigma}_k = \Theta_k,\ \tilde{U}_k = [U_k,\ \hat{U}_p]F_k,\ \text{and } \tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k\ .$$

The QR decomposition in the first step eq. (2) can be expensive when the updates are large so an improved version of this algorithm was proposed in [65] where this factorization is replaced by a low rank approximation of the same matrix. That is, for a rank l, we compute a rank-l approximation, $(I - U_k U_k^T)D = X_l S_l Y_l^T$. Then, the matrix H_D is the update equation (3) will be

$$H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & S_l Y_l^T \end{bmatrix}$$

with $\bar{U} = [U_k, X_l]$. The idea is that the update D will likely be low rank outside the previous top k singular vector space. Hence a low rank approximation of $(I - U_k U_k^T)D$ suffices, thus reducing the cost.

In the low rank approximation applications, the rank k will be typically much smaller than n, and it can be computed inexpensively using the recently proposed numerical rank estimation methods [63, 64].

4. Coarsening.

The previous section discussed randomization methods, which work well in certain situations, for example, when there is a good gap in the spectrum or there is a sharp spectral decay. An alternative method to reduce the matrix dimension, particularly when the matrices are associated with graphs, is to coarsen the data with

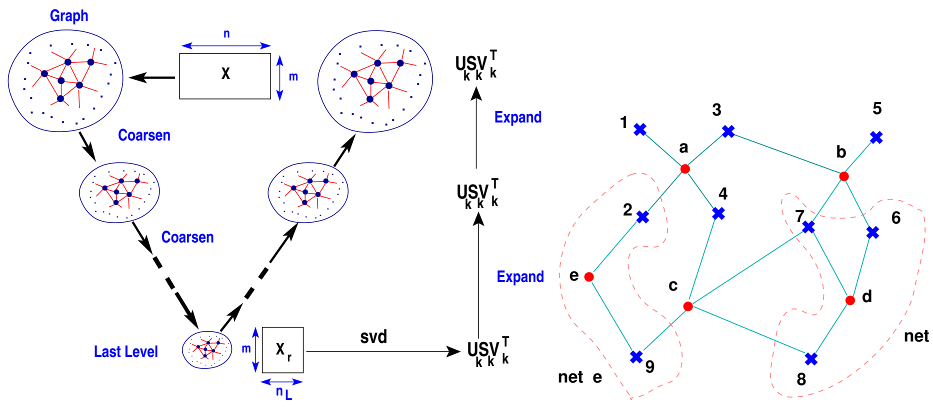

FIG. 1. *Left: Coarsening / uncoarsening procedure; Right : A sample hypergraph*

the help of graph coarsening, perform all computations on the resulting reduced size matrix, and then project back to the original space. Similarly to the idea of sampling columns and computing the SVD of the smaller sampled matrix, in the coarsening methods, we compute the SVD from the matrix corresponding to the coarser data. It is also possible to then wind back up and correct the SVD gradually, in a way similar to V-cycle techniques in multigrid [51], this is illustrated in Figure 1(left). See, for example [70, 51, 20, 45] for a few illustrations where coarsening is used in data-related applications.

Before coarsening, we first need to build a graph representing the data. This first step may be expensive in some cases but for data represented by sparse matrices, the graph is available from the data itself in the form of a standard graph or a hypergraph. For dense data, we need to set-up a similarity graph, see [10] for a fast algorithm to achieve this. This paper will focus on sparse data such as the data sets available in text mining, gene expressions and multilabel classification, to mention a few examples. In such cases, the data is represented by a (rectangular) sparse matrix and it is most convenient to use hypergraph models [70] for coarsening.

4.1. Hypergraph Coarsening. Hypergraphs extend the classical notion of graphs. A hypergraph $H = (V, E)$ consists of a set of vertices V and a set of hyperedges E [9, 70]. In a standard graph an edge connects two vertices, whereas a hyperedge may connect an arbitrary subset of vertices. A hypergraph $H = (V, E)$ can be canonically represented by a boolean matrix A, where the vertices in V and hyperedges (nets) in E are represented by the columns and rows of A, respectively. This is called the *row-net model*. Each hyperedge, a row of A, connects the vertices whose corresponding entries in that row are non-zero. An illustration is provided in Figure 1(Right), where $V = \{1, \ldots, 9\}$ and $E = \{a, \ldots, e\}$ with $a = \{1, 2, 3, 4\}$, $b = \{3, 5, 6, 7\}$, $c = \{4, 7, 8, 9\}$, $d = \{6, 7, 8\}$, and $e = \{2, 9\}$.

Given a (sparse) data set of n entries in $\mathbb{R}^m$ represented by a matrix $A \in \mathbb{R}^{m\times n}$, we can consider a corresponding hypergraph $H = (V, E)$ with vertex set V corresponding to the columns of A. Several methods exist for coarsening hypergraphs, see, e.g., [9, 29]. In this work, we consider a hypergraph coarsening based on column matching, which is a modified version of the *maximum-weight matching* method, e.g., [9, 14]. The modified approach follows the maximum-weight matching method and computes the non-zero inner product $\langle a^{(i)}, a^{(j)}\rangle$ between two vertices i and j. Note that, the inner product between vectors is related to the angle between the vectors, i.e.,

Algorithm 2 Hypergraph coarsening by column matching.

```
**Input:** $A \in \mathbb{R}^{m\times n}$, $\epsilon \in (0,1)$.
**Output:** Coarse matrix $C \in \mathbb{R}^{m\times c}$.
repeat
    Randomly pick $i \in S$; $S := S - \{i\}$.
    Set ip[$k$] := 0 for $k = 1,\dots,n$, and $p = 1$.
    for all $j$ with $a_{ij} \neq 0$ do
        for all $k$ with $a_{jk} \neq 0$ do
            ip[$k$] := ip[$k$] + $a_{ij}a_{jk}$.   // (*)
        end for
    end for
    $j :=$ argmax{ip[$k$] : $k \in S$}
    $csq\theta = \frac{\text{ip}[j]^2}{\|a^{(i)}\|^2\|a^{(j)}\|^2}$.
    if [ ($csq\theta \geq \frac{1}{1+\epsilon^2}$)] then
        $c^{(p)} := \sqrt{1+csq\theta}a^{(i)}$. // The denser of columns $a^{(i)}$ and $a^{(j)}$
        $S := S - \{j\}; p = p + 1$.
    else
        $c^{(p)} := a^{(i)}$.
        $p = p + 1$.
    end if
until $S = \emptyset$
```

$\langle a^{(i)}, a^{(j)}\rangle = \|a^{(i)}\|\|a^{(j)}\|\cos\theta_{ij}$. Next, we match two vertices only if the angle between the vertices (columns) is such that, $\tan\theta_{ij} \leq \epsilon$, for a constant $0 < \epsilon < 1$. Another feature of the algorithm is that it applies a scaling to the coarsened columns in order to reduce the error. In summary, we combine two columns $a^{(i)}$ and $a^{(j)}$ if the angle between them is such that, $\tan\theta_{ij} \leq \epsilon$. We replace the two columns $a^{(i)}$ and $a^{(j)}$ by

$$c^{(p)} = \left(\sqrt{1+\cos^2\theta_{ij}}\right) a^{(i)}$$

or $a^{(j)}$, the one that has more nonzeros. This minor modification provides some control over the coarsening procedure using the parameter ϵ and, more importantly, it helps establish theoretical results for the method, see section 5.

The vertices can be visited in a random order, or in the 'natural' order in which they are listed. For each unmatched vertex i, all the unmatched neighbor vertices j are explored and the inner product between i and each j is computed. This typically requires the data structures of A and its transpose, in that a fast access to rows and columns is required. The vertex j with the highest non-zero inner product $\langle a^{(i)}, a^{(j)}\rangle$ is considered and if the angle between them is such that $\tan\theta_{ij} \leq \epsilon$ (or $\cos^2\theta_{ij} \geq \frac{1}{1+\epsilon^2}$)), then i is matched with j and the procedure is repeated until all vertices have been matched. Algorithm 2 gives details on the procedure.

Note that the loop (*) computes the inner product (ip[k]) of columns i and k of A. The pairing used by the algorithm relies only on the sparsity pattern. It is clear that these entries can also be used to obtain a pairing based on the cosine of the angles between columns i and k. The coarse column $c^{(p)}$ is defined as the 'denser of columns $a^{(i)}$ and $a^{(j)}$'. In other models the sum is sometimes used.

Computing the cosine angle between column i and all other columns is equivalent to computing the i-th row of A^TA, in fact only the upper triangular part of the row. For sparse matrices, the computation of the inner product (cosine angle) between the columns can be achieved inexpensively by modifying the cosine algorithm in [48] developed for matrix blocks detection.

Computational Cost. The cost of computing all inner products of column i with columns of A is the sum of number of nonzeros of each columns involved:

$$\sum_{j=1}^{|a^{(i)}|} |a^{(j)}|,$$

where $a^{(i)}$ is the i-th column and $|\cdot|$ denotes cardinality. If ν_c (resp. ν_r) is the maximum number of nonzeros in each column (resp. row), then an upper bound for the above cost is $n\nu_r\nu_c$. This basic cost is equivalent to computing the upper triangular part of A^TA. Several simplifications and improvements can be added to reduce the cost. First, we can skip the columns that are already matched. In this way, fewer inner products are computed as the algorithm progresses. In addition, since we only need the angle to be such that $\tan\theta_{ij} \le \epsilon$, we can reduce the computation cost significantly by stopping as soon as we encounter a column with which the angle is smaller than the threshold. Article [11] uses the angle based column matching idea for dense subgraph detection in graphs, and describes efficient methods to compute the inner products.

4.2. Multilevel SVD computations. Given a sparse matrix A, we can use Algorithm 2 to recursively coarsen the corresponding hypergraph in the row-net model level by level, and obtain a sequence of sparse matrices $A_1, A_2, \ldots, A_r$ with $A_0 = A$, where A_i corresponds to the coarse graph H_i of level i for $i = 1, \ldots, r$, and A_r represents the lowest level graph H_r. This provides a reduced size matrix which likely is a good representation of the original data. Note that, recursive coarsening will be inexpensive since the inner products required in the further levels are already computed in the first level of coarsening.

In the multilevel framework of hypergraph coarsening we apply the matrix approximation method, say using SVD, to the coarsened data matrix $A_r \in \mathbb{R}^{m\times n_r}$ at the lowest level, where n_r is the number of data items at coarse level r ($n_r < n$). A low-rank matrix approximation can be viewed as a linear projection of the columns into a lower dimensional space. In other words we have a matrix $\hat{A}_r \in \mathbb{R}^{d\times n_r}$ ($d < m$). Applying the same linear projection to $A \in \mathbb{R}^{m\times n}$ produces $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$), and one can expect that $\hat{A}$ preserves certain features of A. This linear projection is then applied to the original data $A \in \mathbb{R}^{m\times n}$ to obtain a reduced representation $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$) of the original data. The procedure is illustrated in Figure 1 (left). The multilevel idea is used in the ConstantTimeSVD algorithm proposed in [16].

Another strategy for reducing the matrix dimension is to mix the two techniques: Coarsening may still be exceedingly expensive for some type of data where there is no immediate graph available to exploit for coarsening. In this case, a good strategy would be to downsample first using the randomized methods, then construct a graph and coarsen it. In section 6, we compare the SVDs obtained from pure randomization methods against those obtained from coarsening and also randomization + coarsening.

4.3. CSSP and graph sparsification. The multilevel coarsening technique presented can be applied for the column subset selection problem (CSSP) as well as for the graph sparsification problem. We can use Algorithm 2 to coarsen the matrix, which is equivalent to selecting columns of the matrix. The only modification in the algorithm required is that the columns selected are not scaled. The coarse matrix C contains few columns of the original matrix A and yet preserves the structure of A.

Algorithm 3 Incremental SVD

```
Start: select $k$ columns of $A$ by random sampling or coarsening, define $A_s$ as this sample of columns.
repeat
    Update (compute if started) SVD of $A_s$ via SVD-update or subspace iteration.
    Add columns of $A$ to $A_s$
until converged
```

For graph sparsification, we can apply the coarsening procedure on the matrix B^T, i.e., coarsen the rows of the vertex edge incidence B, which yields us fewer edges, $\tilde{B}$ with fewer rows. The analysis in section 5 shows how this coarsening strategy is indeed a spectral sparsifier, shows $x^T\tilde{B}^T\tilde{B}x$ is close to x^TB^TBx. Since we achieve sparsification via matching, the structures such as clusters within the original graph are also preserved.

4.4. Incremental SVD.

Next, we explore some combined algorithms that improve the randomized sampling and coarsening SVD results significantly. The typical overall algorithm which we call Incremental SVD algorithm is sketched in Algorithm 3.

A version of this Incremental algorithm has been briefly discussed in [24], where the basic randomized algorithm is combined with subspace iteration, see Algorithm 8.1 in the reference. For subspace iteration, we know that each iteration takes the computed subspace closer to the subspace spanned by the target singular vectors. If the initial subspace is close to the span of the actual top k singular vectors, fewer iterations will be needed to get accurate results. The theoretical results established in the following section, give us an idea how close the subspace obtained from the coarsening technique will be to the span of the top k singular vectors of the matrix. In such cases, a few steps of the subspace iteration will then yield very accurate results.

For the SVD-RR update method, it is known that the method performs well when the updates are of low rank and do not affect the dominant subspace, the subspace spanned by the top k singular vectors which of interest, too much [65]. Since the random sampling and the coarsening methods return a good approximation to the dominant subspace, we can assume that the updates in the incremental SVD are of low rank, and these updates likely effect the dominant subspace only slightly. Hence, the SVD-RR update gives accurate results.

5. Analysis.

In this section, we establish theoretical results for the coarsening technique based on column matching. Suppose in the coarsening strategy, we combine two columns $a^{(i)}$ and $a^{(\hat{i})}$ if the angle between them is such that, $\tan\theta_i \leq \epsilon$. We replace the two columns $a^{(i)}$ and $a^{(\hat{i})}$ by $c^{(p)} = (\sqrt{1+\cos^2\theta_i})a^{(i)}$ (or $a^{(\hat{i})}$, the one with more nonzeros). We then have the following key result.

LEMMA 5.1. *Given $A \in \mathbb{R}^{m\times n}$, let $C \in \mathbb{R}^{m\times c}$ be the coarsened matrix of A obtained by one level of coarsening of A with columns $a^{(i)}$ and $a^{(\hat{i})}$ matched if $\tan\theta_i \leq \epsilon$, for $0 < \epsilon < 1$. Then,*

$$|x^TAA^Tx - x^TCC^Tx| \leq 3\epsilon\|A\|_F^2, \tag{4}$$

for any $x \in \mathbb{R}^n : \|x\| = 1$.

Proof. Let $(i,\hat{i})$ be a pair of matched column indices with i being the index of the column that is retained after scaling. We denote by I the set of all indices of the retained columns and $\hat{I}$ the set of the remaining columns.

We know that $\sigma_i^2(A) = \sigma_i(AA^T) = \max_{\|x\|=1} x^TAA^Tx$, and also $x^TAA^Tx = \|A^Tx\|_2^2 = \sum_{i=1}^n \langle a^{(i)}, x\rangle^2$. Similarly, consider $x^TCC^Tx = \|C^Tx\|_2^2 = \sum_{i\in I}\langle c_i, x\rangle^2 =$

$\sum_{i\in I}(1+c_i^2)\langle a^{(i)},x\rangle^2$, where indices $c_i=\cos\theta_i$. Next, we have,

$$\begin{aligned}|x^TAA^Tx-x^TCC^Tx| &= |\sum_{i\in I\cup\hat{I}}\langle a^{(i)},x\rangle^2-\sum_{i\in I}(1+c_i^2)\langle a^{(i)},x\rangle^2|\\ &\le |\sum_{\hat{i}\in\hat{I}}\langle a^{(\hat{i})},x\rangle^2-\sum_{i\in I}c_i^2\langle a^{(i)},x\rangle^2|\\ &= \sum_{(i,\hat{i})\in I\times\hat{I}}\left[\langle a^{(\hat{i})},x\rangle^2-c_i^2\langle a^{(i)},x\rangle^2\right]\end{aligned}$$

where the set $I\times\hat{I}$ consists of pairs of indices $(i,\hat{i})$ that are matched. Next, we consider the inner term in the summation. Let the column $a^{(\hat{i})}$ be decomposed as follows:

$$a^{(\hat{i})}=c_ia^{(i)}+s_iw,$$

where $s_i=\sin\theta_i$ and $w=\|a^{(i)}\|\bar{w}$ with $\bar{w}$ a unit vector that is orthogonal to $a^{(i)}$ (hence, $w\perp a^{(i)}$ and has the same length). Then,

$$\begin{aligned}|\langle a^{(\hat{i})},x\rangle^2-c_i^2\langle a^{(i)},x\rangle^2| &= \left|\langle c_ia^{(i)}+s_iw,x\rangle^2-c_i^2\langle a^{(i)},x\rangle^2\right|\\ &= \left|c_i^2\langle a^{(i)},x\rangle^2+2c_is_i\langle a^{(i)},x\rangle\langle w,x\rangle+s_i^2\langle w,x\rangle^2-c_i^2\langle a^{(i)},x\rangle^2\right|\\ &= |\sin 2\theta_i\langle a^{(i)},x\rangle\langle w,x\rangle+\sin\theta_i^2\langle w,x\rangle^2|\end{aligned}$$

Let $t_i=\tan\theta_i$, then we have $\sin 2\theta_i=\frac{2t_i}{1+t_i^2}$ and using the fact that $|\langle w,x\rangle|\le\|a^{(i)}\|\equiv\eta$ and $\langle a^{(i)},x\rangle\le\eta$, we get

$$\begin{aligned}|\sin 2\theta_i\langle a^{(i)},x\rangle\langle w,x\rangle+\sin\theta_i^2\langle w,x\rangle^2| &\le \eta^2\sin 2\theta_i\left[1+\frac{\sin^2\theta_i}{2\sin\theta_i\cos\theta_i}\right]\\ &= \eta^2\sin 2\theta_i\left[1+\frac{\tan\theta_i}{2}\right]\\ &\le \frac{2\eta^2t_i+(\eta t_i)^2}{1+t_i^2}\\ &\le 2\eta^2t_i+(\eta t_i)^2.\end{aligned}$$

Now, since our algorithm combines two columns only if $\tan(\theta_i)\le\epsilon$ (or $\cos^2\theta\ge 1/(1+\epsilon^2)$), we have

$$|\langle a^{(\hat{i})},x\rangle^2-c_i^2\langle a^{(i)},x\rangle^2|\le 2\eta^2\epsilon+\eta^2\epsilon^2\le 3\epsilon\eta^2$$

as $\epsilon<1$ and $\eta>1$. We can further improve the bound to $2\eta\epsilon+(\eta\epsilon)^2\le 2.5\eta\epsilon$, provided $(\eta\epsilon)\le 0.5$. Thus, we have

$$|x^TAA^Tx-x^TCC^Tx|\le 3\epsilon\sum_{i\in I}\|a^{(i)}\|^2\le 3\epsilon\|A\|_F^2.$$

∎

The above lemma gives us bounds on the Rayleigh Quotients of the coarsened matrix C. This result helps to establish the following error bounds.

THEOREM 5.2. *Given $A \in \mathbb{R}^{m\times n}$, let $C \in \mathbb{R}^{m\times c}$ be the coarsened matrix of A obtained by one level coarsening of A with columns $a^{(i)}$ and $a^{(\hat{i})}$ combined if $\tan\theta_i \le \epsilon$, for $0 < \epsilon < 1$. Let H_k be the matrix consisting of the top k left singular vectors of C as columns. Then, we have*

$$\|A - H_kH_k^TA\|_F^2 \le \|A - A_k\|_F^2 + 6k\epsilon\|A\|_F^2 \tag{5}$$
$$\|A - H_kH_k^TA\|_2^2 \le \|A - A_k\|_2^2 + 6\epsilon\|A\|_F^2, \tag{6}$$

where A_k is the best rank k approximation of A.

Proof. Frobenius norm error: First, we prove the Frobenius norm error bound. We can express $\|A - H_kH_k^TA\|_F^2$:

$$\begin{aligned}\|A - H_kH_k^TA\|_F^2 &= Tr((A - H_kH_k^TA)^T(A - H_kH_k^TA)) \\ &= Tr(A^TA - 2A^TH_kH_k^TA + A^TH_kH_k^TH_kH_k^TA) \\ &= Tr(A^TA) - Tr(A^TH_kH_k^TA) \\ &= \|A\|_F^2 - \|A^TH_k\|_F^2.\end{aligned} \tag{7}$$

We get the above simplifications using the equalities: $\|X\|_F^2 = Tr(X^TX)$ and $H_k^TH_k = I$. Let $h^{(i)}$ for $i = 1, \dots, k$ be the columns of H_k. Then, the second term in the above equation is $\|A^TH_k\|_F^2 = \sum_{i=1}^k \|A^Th^{(i)}\|^2$.

From Lemma 5.1, we have for each i,

$$|\|A^Th^{(i)}\|^2 - \|C^Th^{(i)}\|^2| = |\|A^Th^{(i)}\|^2 - \sigma_i^2(C)| \le 3\epsilon\|A\|_F^2,$$

since $h^{(i)}$'s are the singular vectors of C. Summing over k singular vectors, we get

$$|\|A^TH_k\|_F^2 - \sum_{i=1}^k \sigma_i^2(C)| \le 3\epsilon k\|A\|_F^2. \tag{8}$$

From the perturbation theory [23, Thm. 8.1.4], we have

$$|\sigma_i^2(C) - \sigma_i^2(A)| \le \|AA^T - CC^T\|_2,$$

for $i = 1, \dots, n$. Next, we have

$$\|AA^T - CC^T\|_2 = \max_{x\in\mathbb{R}^n:\|x\|=1} |x^T(AA^T - CC^T)x| \le 3\epsilon\|A\|_F^2,$$

from Lemma 5.1. Hence, summing over k singular values,

$$\left|\sum_{i=1}^k \sigma_i^2(C) - \sum_{i=1}^k \sigma_i^2(A)\right| \le 3\epsilon k\|A\|_F^2. \tag{9}$$

Combining (8) and (9), we get

$$\left|\|A^TH_k\|_F^2 - \sum_{i=1}^k \sigma_i^2(A)\right| \le 6\epsilon k\|A\|_F^2.$$

Combining this relation with (7), gives us the Frobenius norm error bound (since $\|A\|_F^2 - \sum_{i=1}^k \sigma_i^2(A) = \|A - A_k\|_F^2$).

Spectral norm error: Next, we prove the spectral norm error bound. Let $\mathcal{H}_k = range(H_k) = span(h^{(1)}, \ldots, h^{(k)})$ and let $\mathcal{H}_{n-k}$ be the orthogonal complement of $\mathcal{H}_k$. For $x \in \mathbb{R}^n$, let $x = \alpha y + \beta z$, where $y \in \mathcal{H}_k, z \in \mathcal{H}_{n-k}$ and $\alpha^2 + \beta^2 = 1$. Then,

$$\begin{aligned}
\|A - H_k H_k^T A\|_2^2 &= \max_{x \in \mathbb{R}^n : \|x\|=1} \|x^T(A - H_k H_k^T A)\|^2 \\
&= \max_{y,z} \|(\alpha y^T + \beta z^T)(A - H_k H_k^T A)\|^2 \\
&\leq \max_{y \in \mathcal{H}_k : \|y\|=1} \|y^T(A - H_k H_k^T A)\|^2 + \max_{z \in \mathcal{H}_{n-k} : \|z\|=1} \|z^T(A - H_k H_k^T A)\|^2 \\
&= \max_{z \in \mathcal{H}_{n-k} : \|z\|=1} \|z^T A\|^2,
\end{aligned}$$

since $\alpha, \beta \leq 1$ and for any $y \in \mathcal{H}_k, y^T H_k H_k^T = y^T$, so the first term is zero and for any $z \in \mathcal{H}_{n-k}, z^T H_k H_k^T = 0$. Next,

$$\begin{aligned}
\|z^T A\|^2 &= \|z^T C\|^2 + [\|z^T A\|^2 - \|z^T C\|^2] \\
&\leq \sigma_{k+1}^2(C) + 3\epsilon\|A\|_F^2 \\
&\leq \sigma_{k+1}^2(A) + 6\epsilon\|A\|_F^2 \\
&= \|A - A_k\|_2^2 + 6\epsilon\|A\|_F^2.
\end{aligned}$$

Since $|\|z^T A\|^2 - \|z^T C\|^2| \leq 3\epsilon\|A\|_F^2$ from Lemma 5.1, $\max_{z \in \mathcal{H}_{n-k} : \|z\|=1} \|z^T C\|^2 = \sigma_{k+1}^2(C)$, and $|\sigma_i^2(C) - \sigma_i^2(A)| \leq \|AA^T - CC^T\|_2 \leq 3\epsilon\|A\|_F^2$. ∎

We observe that our main Theorem (Theorem 5.2) is similar to the results developed for randomized sampling, see [15, 16]. For randomized sampling, the error reduces as the number of columns c that are sampled increases. For coarsening, the error is smaller if the angles between the columns that are combined are smaller. The number of columns is related to these angles which in turn depends on the structure of the matrix. Existing theoretical results for subspace iteration are discussed in the Appendix.

6. Numerical Experiments.

This section describes a number of experiments to illustrate the performances of the different methods discussed. The latter part of the section focuses on the performance of the coarsening method in the applications discussed in section 2.

6.1. SVD Comparisons.

In the first set of experiments, we use three term-by-document datasets and compare the sampling, coarsening and combined methods to compute the SVD. The tests are with unweighted versions of the CRANFIELD dataset (1398 documents, 5204 terms), MEDLINE dataset (1033 documents, 8322 terms) and TIME dataset (425 documents, 13057 terms). We will use these three datasets in the experiments for column subset selection and in the latent semantic indexing application examples, which will give us an extensive evaluation of the performances of the methods compared.

Figure 2 illustrates the following experiment with the three datasets. Results from four different methods are plotted. The first solid curve (labeled ‘exact’) shows the singular values of matrix A from 20 to 50 computed using the svds function in Matlab (the results obtained by the four methods for top twenty singular values were similar). The diamond curve labeled ‘coarsen’, shows the singular values obtained by one level of coarsening using Algorithm 2. The star curve (labeled ‘rand’) shows the singular values obtained by random sampling using column norms, with a sample size equal to the size obtained with one level of coarsening. We note that the result obtained

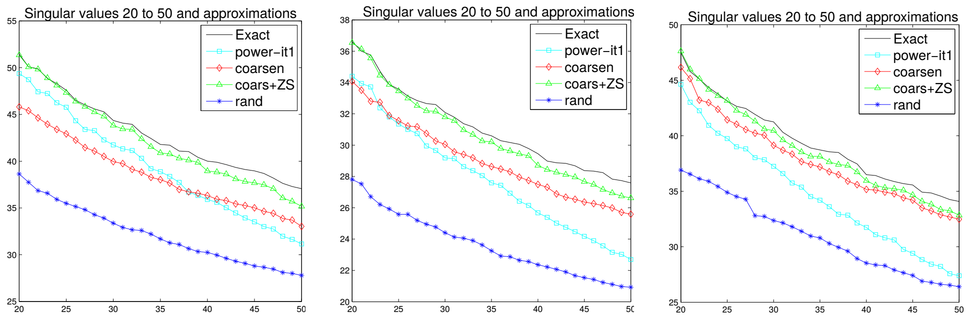

FIG. 2. *Results for the datasets CRANFIELD (left), MEDLINE (middle), and TIME (right).*

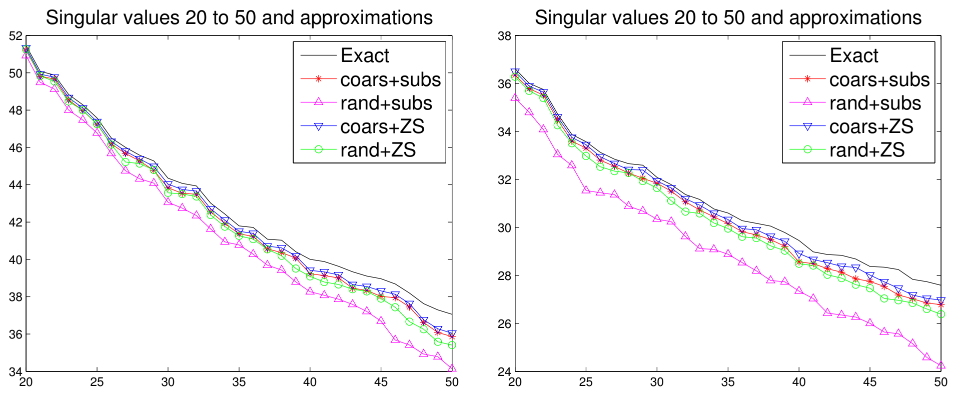

FIG. 3. *Second set of results for the CRANFIELD (left) and the MEDLINE datasets (right).*

by coarsening is much better than that obtained by random sampling. However, we know that the approximations obtained by either sampling or coarsening cannot be highly accurate. In order to get improved results, we can invoke incremental SVD algorithms, Algorithm 3. The curve with triangles labeled 'coars+ZS' shows the singular values obtained when Zha Simon algorithm was used to improve the results obtained by the coarsening algorithm. Here, we consider the singular vectors of the coarse matrix and use the remaining part of the matrix to update these singular vectors and singular values. We have also included the results obtained by one iteration of power method [25], i.e., from the SVD of the matrix $Y = (AA^T)A\Omega$, where Ω is a random Gaussian matrix of same size as the coarse matrix. We see that the smaller singular values obtained from the coarsening algorithms are better than those obtained by the one-step power method.

As discussed in section 4, a possible way of improving the SVD results obtained by a coarsening or random sampling step is to resort to subspace iteration or use the SVD update algorithms as in the first experiment. Figure 3 illustrates such results with incremental SVD algorithms for the CRANFIELD (left) and the MEDLINE (right) datasets. We have not reported the results for the TIME dataset since it is hard to distinguish the results obtained by different algorithms for this case. First, subspace iteration is performed using the matrix A and the singular vectors obtained from coarsening or random sampling. The curve 'coars+subs' (star) corresponds to the singular values obtained when subspace iteration was used to improve the SVD obtained by coarsening. Similarly, for the curve labeled 'rand+subs' (triangle up), subspace

TABLE 1
Low rank approximation: Coarsening, random sampling, and rand+coarsening. Error1 = $\|A - H_k H_k^T A\|_F$; Error2= $\frac{1}{k}\sum_k \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$

Dataset	n	k	c	Coarsen		Rand Sampl		Rand+Coars	
				Err1	Err2	Err1	Err2	Err1	Err2
Kohonen	4470	50	1256	86.26	0.366	93.07	0.434	93.47	0.566
aft01	8205	50	1040	913.3	0.299	1006.2	0.614	985.3	0.598
FA	10617	30	1504	27.79	0.131	28.63	0.410	28.38	0.288
chipcool0	20082	30	2533	6.091	0.313	6.199	0.360	6.183	0.301
brainpc2	27607	30	865	2357.5	0.579	2825.0	0.603	2555.8	0.585
scfxm1-2b	33047	25	2567	2326.1	–	2328.8	–	2327.5	–
thermomechTC	102158	30	6286	2063.2	–	2079.7	–	2076.9	–
Webbase-1M	1000005	25	15625	–	–	3564.5	–	3551.7	–

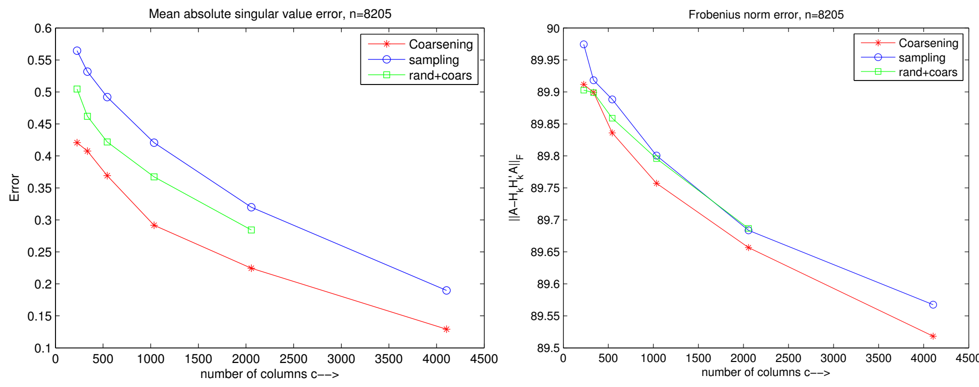

FIG. 4. *Mean absolute singular value errors $\frac{1}{k}\sum_k \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$ (Left) and Frobenius norm errors $\|A - H_k H_k^T A\|_F$ (right) for the three methods for* `aft01` *dataset ($k = 30$).*

iteration was used with the singular vectors obtained from randomized sampling. We have included the results when the SVD update algorithm was used to improve the SVD obtained by coarsening ('coars+ZS') and random sampling ('rand+ZS'), respectively. These plots show that both the SVD update algorithm and subspace iteration improve the accuracy of the SVD significantly.

Next, we compare the performances of coarsening and random sampling for computing the low rank approximation of matrices. We also consider the combined method of sampling followed by coarsening discussed in the introduction and in section 4. Table 1 shows comparison results between the three methods, namely, Coarsening, random sampling, and random sampling+coarsening for low rank approximation of matrices from various applications. All matrices were obtained from the SuiteSparse matrix collection: https://sparse.tamu.edu/ [12] and are sparse. The errors reported are the Frobenius norm error = $\|A - H_k H_k^T A\|_F$ in computing the rank k approximation and the average absolute normalized error in the singular values = $\frac{1}{k}\sum_k \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$ for rank k as listed in third column. The size of the input matrix and the number of columns in the coarsened/subsampled matrix are listed in the second and fourth columns, respectively. For very large matrices, the exact singular values cannot be computed, hence we were unable to report Error2 for the last 3 matrices. For Webbase-1M (size 10^6), it is impractical to do full coarsening. Hence, we only report errors for random sampling, and random sampling+coarsening.

Figure 4 plots the two errors $\|A - H_k H_k^T A\|_F$ and $\frac{1}{k}\sum_k \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$ with $k = 30$ for the three methods for `aft01` dataset when different levels of coarsening were used,

Table 2
CSSP: Coarsening versus leverage score sampling.

Dataset	Size	Rank k	c	Coarsening		levSamp
				levels	error	error
CRAN	1398	25	88	4	496.96	501.32
		50	88	4	467.49	477.25
		150	175	3	375.40	383.23
MED	1033	50	65	4	384.91	376.23
		100	130	4	341.51	339.01
TIME	425	25	107	2	411.71	412.77
		50	107	2	371.35	372.66
		50	54	3	389.69	391.91
Kohonen	4470	25	981	2	31.89	36.36
Erdos992	6100	50	924	3	100.9	99.29
FA	10617	50	2051	3	26.33	28.37
chipcool0	20082	100	1405	4	6.05	6.14

i.e., the number of columns sampled/coarsened were increased. Here for 'rand+coars' we proceed as follows. First, half of the columns are randomly sampled and then a multilevel coarsening is performed with one level less than the pure coarsening method reported in the previous column. Hence, we do not have errors for $c = n/2$. Coarsening clearly yields better results (lower errors) than the randomized sampling method. The combined method of random sampling+coarsening works well and performs better than randomized sampling in most cases. For a smaller number of columns, i.e., more levels in coarsening, the Frobenius norm error for rand+coarsen approaches that of full coarsening. However, note that the coarsening procedure is expensive compared to column norm sampling.

In all the above experiments, we have used maximum matching for coarsening. The choice of ϵ, the parameter that decides the angle for matching does not seem to affect the errors directly. If we choose smaller ϵ, we will have a larger coarse matrix C (fewer columns are combined) and the error will be small. If we choose a larger ϵ, more columns are combined and the results are typically equivalent to just simply using maximum matching ignoring the angle constraint. Thus, in general, the performance of the coarsening technique depends on the structure of the considered matrix. If we have more columns that are close to each other, i.e., make smaller angle between each other, the coarsening technique will combine more columns, we can choose a smaller ϵ and yet obtain good results. If the matrix is very sparse or if the columns make large angles between each other, coarsening might not yield a coarse matrix since it will not be able to match many columns. Therefore, selecting the smallest ϵ that will yield a small coarse matrix and yet lead to good approximations will depend on the structure of the input matrix.

6.2. Column Subset Selection.

In the following experiment, we compare the performance of the coarsening method against the leverage score sampling method for column subset selection. We report results for the same three term-by-document datasets used in the first set of experiments. We also include results obtained for a few sparse matrices from the SuiteSparse matrix collection.

Table 2 presents a few comparisons. The errors reported are the Frobenius norm errors $\|A - P_C A\|_F$, where P_C is the projector onto $span(C)$, and C is the coarsened/sampled matrix which is computed by the multilevel coarsening method or using leverage score sampling of A with the top k singular vectors as reported in

TABLE 3
Graph Sparsification: Coarsening versus leverage score sampling. Error$= \frac{1}{r}\sum_r \frac{|\sigma_i(\tilde{K})-\sigma_i(K)|}{\sigma_i(K)}$

Dataset	m	r	$\frac{nnz(\tilde{K})}{nnz(K)}$	Coarsening		levSamp
				levels	error	error
sprand	1290	332	0.29	2	0.541	0.575
	1951	499	0.28	2	0.542	0.579
	2676	679	0.27	2	0.537	0.580
Maragal4	6005	460	0.11	4	0.416	0.569
rosen1	12599	1738	0.18	3	0.482	0.304
G1	19176	2486	0.14	3	0.549	0.635
bibd13-6	25428	1619	0.08	4	0.901	0.920

the second column. The number of columns c in each test is reported in the third column which is the same for both methods. Recall that for CSSP, the coarsening and sampling algorithms do not perform a post-scaling of the columns that are selected. We see that the multilevel coarsening method performs very well and is comparable with leverage score sampling in most cases. Note that the standard leverage score sampling requires the computation of the r top singular vectors and this can substantially more expensive than coarsening especially when r is large.

6.3. Graph Sparsification. The next experiment illustrates how the coarsening method can be used for graph sparsification. We again compare the performance of the coarsening approach to the leverage score sampling method [27] for graph spectral sparsification. Recall that spectral sparsification accounts to computing a sparse graph $\tilde{G}$ that approximates the original graph G such that the singular values of the graph Laplacian $\tilde{K}$ of $\tilde{G}$ are close to those of K, Laplacian of G.

Table 3 lists the errors obtained when the coarsening and the leverage score sampling approaches were used to compute a sparse graph $\tilde{G}$ for different sparse random graphs and few matrices related to graphs from the SuiteSparse database. Given a graph G, we can form a vertex edge incidence matrix B, such that the Laplacian $K = B^TB$. Then, sampling/coarsening the rows of B to get $\tilde{B}$ gives us a sparse graph with Laplacian $\tilde{K} = \tilde{B}^T\tilde{B}$. The type of graph or the names are given in the first column of the table and the number of rows m in corresponding vertex edge incidence matrix B is given in the second column. The number of rows r in the coarse matrix $\tilde{B}$ is listed in the third column. The ratios of sparsity in $\tilde{K}$ and K are also given. This ratio indicates the amount of sparsity achieved by sampling/coarsening. Since, we have same number of rows in the coarsened and sampled matrix $\tilde{B}$, this ratio will be the same for both methods. The error reported is the normalized mean absolute error in the singular values of K and $\tilde{K}$, Error$= \frac{1}{r}\sum_r \frac{|\sigma_i(\tilde{K})-\sigma_i(K)|}{\sigma_i(K)}$, which tells us how close the sparser matrix $\tilde{K}$ is to K spectrally. We see that in most cases, the coarsening approach performs similarly to or better than leverage score sampling.

6.4. Applications. In this section, we illustrate the performance of the coarsening technique in the various applications introduced in section 2.

6.4.1. Latent Semantic Indexing. The first application we consider is Latent Semantic Indexing (LSI) [13, 33]. In LSI, we have a term-document matrix $A \in \mathbb{R}^{m\times n}$, representing m documents and n terms that frequently occur in the documents, where A_{ij} is the frequency of the jth term in the i-th document. A query is an n-vector $q \in \mathbb{R}^n$, normalized to 1, where the jth component of a query vector is interpreted as

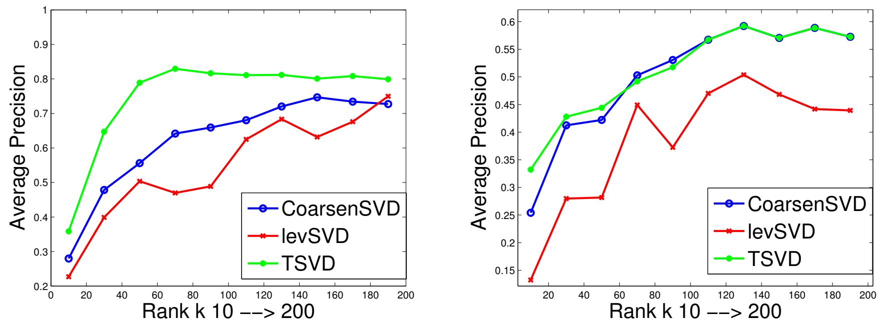

FIG. 5. *LSI results for the MEDLINE dataset on left and TIME dataset on the right.*

the frequency with which the jth term occurs in a topic. Typically, the number of topics to which the documents are related is smaller than the number of unique terms n. Hence, finding a set of k topics that best describe the collection of documents for a given k, corresponds to keeping only the top k singular vectors of A, and obtaining a rank k approximation. The truncated SVD and related methods are often used in LSI applications. The argument is that a low rank approximation captures the important underlying intrinsic semantic associated with terms and documents, and removes the noise or variability in word usage [33]. In this experiment, we employ the Coarsen SVD and leverage score sampling SVD algorithms to perform information retrieval techniques by Latent Semantic Indexing (LSI) [51].

Given a term-by-document data $A \in \mathbb{R}^{m\times n}$, we normalize the data using TF-IDF (term frequency-inverse document frequency) scaling. We also normalize the columns to unit vectors. Query matching is the process of finding the documents most relevant to a given query $q \in \mathbb{R}^m$.

Figure 5 plots the average precision against the dimension/rank k for MEDLINE and TIME datasets. When the term-document matrix A is large, the computation of the SVD factorization can be expensive for large ranks k. The multi-level techniques will find a smaller set of document vectors, denoted by$A_r \in \mathbb{R}^{m\times n_r}$, to represent A ($n_r < n$). For leverage score sampling, we sample A_r using leverage scores with k equal to the rank shown on the x axis. Just like in the standard LSI, we compute the truncated SVD of $A_r = U_d\Sigma_d V_d^T$, where d is the rank. Now the reduced representation of A is $\hat{A} = \Sigma_d^{-1}U_d^T A$. Each query q is transformed to a reduced representation $\hat{q} = \Sigma_d^{-1}U_d^T q$. The similarity of q and a_i are measured by the cosine distance between $\hat{q}$ and $\hat{a}$ for $i = 1,\ldots,n$. This example clearly illustrates the advantage of the coarsening method over randomized sampling and leverage scores. The multilevel coarsening method performs better than the sampling method in this application and in some cases it performs as well as the truncated SVD method. Multilevel coarsening algorithms for LSI applications, have been discussed in [51] where additional details can be found.

6.4.2. Projective clustering. The next application we consider is a set of nonlinear projection based clustering techniques. We illustrate how the multilevel coarsening methods can be used for data reduction in this application. We consider three types of nonlinear projection methods, namely, Isomap [58], Local Linear Embedding (LLE) [47] and Laplacian Eigenmaps [4]. Multilevel algorithm have been used in the clustering application, for example, article [41] uses a multilevel algorithm, based on MinMaxCut, for document clustering, and Fang et. al. [20] applied the mutlilevel algorithms for spectral clustering and manifold learning.

Given n data-points, most of the projective clustering methods start by con-

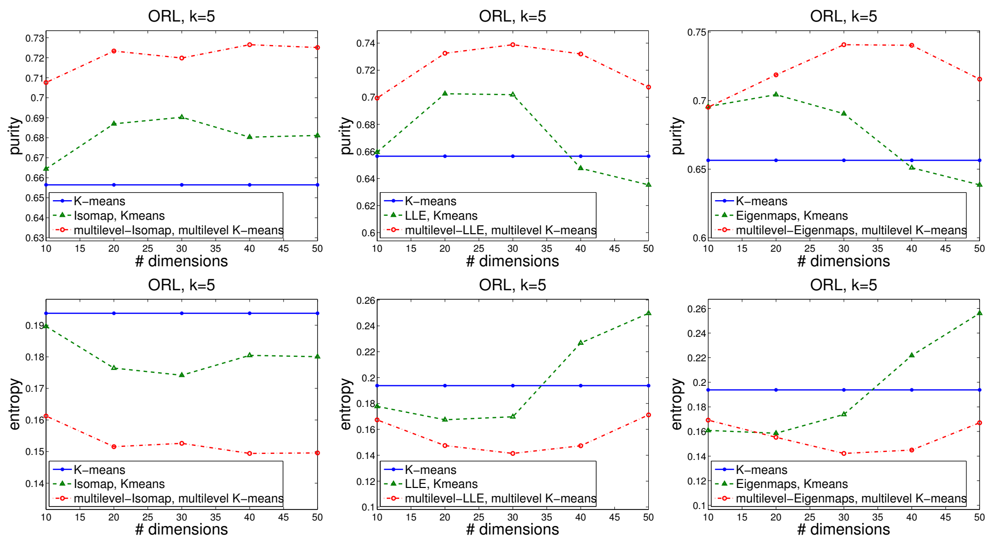

FIG. 6. *Purity and entropy values versus dimensions for three types of clustering for ORL dataset.*

structing a graph with edges defined based on certain criteria such as new distance metrics or manifolds, nearest neighbors, points on a same subspace, etc. The graph Laplacian corresponding to the graph is considered, and for a given k, the top k eigenvectors of a shifted Laplacian matrix, whose top eigenvectors correspond to the bottom eigenvectors of the original graph, are used to cluster the points. We use the following two evaluation metrics to analyze the quality of the clusters obtained, namely *purity* and *entropy* [69] given by:

$$\text{purity} = \sum_{i=1}^{K} \frac{n_i}{n}\text{purity}(i); \qquad \text{purity}(i) = \frac{1}{n_i}\max_j(n_i^j), \qquad \text{and}$$

$$\text{entropy} = \sum_{i=1}^{K} \frac{n_i}{n}\text{entropy}(i); \text{entropy}(i) = -\sum_{j=1}^{K} \frac{n_i^j}{n_i}\log_K \frac{n_i^j}{n_i},$$

where K is the number of clusters, n_i^j is the number of entries of class j in cluster i, and n_i is the number of data in cluster i. Here, we assume that the labels indicating the class to which data belong are available.

In figure 6 we present results for three types of projective clustering methods, viz., Isomap, LLE and eigenmaps when coarsening was used before dimensionality reduction. The dataset used is the popular ORL face dataset [52], which contains 40 subjects and 10 grayscale images each of size 112×92 with various facial expressions (matrix size is 10304×400). For the projective methods, we first construct a k-nearest neighbor graph with $k = 5$, and use embedding dimensions $p = 10, \ldots, 50$. Note that even though the data is dense, the kNN graph is sparse. The figure presents the purity and entropy values obtained for the three projective clustering methods for these different dimensions p with (circle) and without (triangle) coarsening the graph. The solid lines indicate the results when kmeans was directly used on the data without dimensionality reduction. We see that the projective methods give improved clustering quality in terms of both purity and entropy, and coarsening further improves their results in many cases by reducing redundancy. This method was also discussed in [19] where additional results and illustrations with other applications can be found.

TABLE 4
TaggingSNP: Coarsening, Leverage Score sampling and Greedy selection

Data	Size	c	Coarsen	Lev. Samp.	Greedy
Yaledataset/SORCS3	1966×53	14	0.0893	0.1057	0.0494
Yaledataset/PAH	1979×32	9	0.1210	0.2210	0.0966
Yaledataset/HOXB	1953×96	24	0.1083	0.1624	0.0595
Yaledataset/17q25	1962×63	16	0.2239	0.2544	0.1595
HapMap/SORCS3	268×307	39	0.0325	0.0447	0.0104
HapMap/PAH	266×88	22	0.0643	0.0777	0.0311
HapMap/HOXB	269×571	72	0.0258	0.0428	0.0111
HapMap/17q25	265×370	47	0.0821	0.1190	0.0533

6.4.3. Genomics - Tagging SNPs. The third application we consider is that of DNA microarray gene analysis. The data from microarray experiments is represented as a matrix $A \in \mathbb{R}^{m\times n}$, where A_{ij} indicates whether the jth expression level exists for gene i. Typically, the matrix could have entries $\{-1, 0, 1\}$ indicating whether the expression exists (± 1) or not (0) and the sign indicating the order of the sequence. Article [44] used CSSP with a greedy selection algorithm to select a subset of gene expressions or single nucleotide polymorphisms (SNPs) from a table of SNPs for different populations that capture the spectral information (variations) of population. The subset of SNPs are called *tagging SNPs* (tSNPs). Here we show how the coarsening method can be applied in this application to select columns (and thus tSNPs) from the table of SNPs, which characterize the extent to which major patterns of variation of the intrapopulation data are captured by a small number of tSNPs.

We use the same two datasets as in [44], namely the Yale dataset and the Hapmap datset. The Yale dataset[1] [42] contains a total of 248 SNPs for around 2000 unrelated individuals from 38 populations each from around the world. We consider four genomic regions (*SORCS3,PAH,HOXB,* and *17q25*). The HapMap project[2] [22] (phase I) released a public database of 1,000,000 SNP typed in different populations. From this database, we consider the data for the same four regions. Using the SNP table, an encoding matrix A is formed with entries $\{1, 0, 1\}$ indicating whether the expression exists (± 1) or not (0) and the sign indicating the order of the sequence, see supplementary material of [44] for details on this encoding. We obtained such encoded matrices, made available online by the authors of [44], from http://www.asifj.org/.

Table 4 lists the errors obtained from the three different methods, namely, Coarsening, Leverage Score sampling and Greedy selection [44] for different populations. The error reported is given by $nnz(\hat{A} - A)/nnz(A)$, where A is the input encoding matrix, C is the sampled/coarsened matrix, $\hat{A} = CC^{\dagger}A$, is the projection of A onto C and $nnz(A)$ is the number of elements in A. The greedy algorithm considers each column of the matrix sequentially, projects the remaining columns onto the considered column and chooses the column that gives least error as defined above. The algorithm then repeats the procedure to select the next column and so on. This algorithm is very expensive but performs very well in practice. We observe that the coarsening algorithm performs better than leverage score sampling and the performance is comparable to that of the greedy algorithm in some cases. The coarsening algorithm is inexpensive compared to leverage score sampling and is significantly cheaper than the greedy algorithm.

[1] http://alfred.med.yale.edu/
[2] https://www.ncbi.nlm.nih.gov/variation/news/NCBI_retiring_HapMap/

TABLE 5
Multilabel Classification using CSSP (leverage score sampling) and coarsening: Average training and test errors and Precison@k, k =sparsity.

Data	Method	c	Train Err	Train P@k	Test Err	Test P@k
Mediamill, $d = 101, n =$	Coars	51	**10.487**	0.766	**8.707**	**0.713**
$10000, nt = 2001, p = 120$.	CSSP	51	10.520	**0.782**	12.17	0.377
Bibtex, $d = 159, n =$	Coars	80	**1.440**	**0.705**	4.533	**0.383**
$6000, nt = 1501, p = 1836$.	CSSP	80	1.575	0.618	**4.293**	0.380
Delicious, $d = 983, n =$	Coars	246	**50.943**	0.639	**74.852**	0.455
$5000, nt = 1000, p = 500$.	CSSP	246	53.222	**0.655**	77.937	**0.468**
Eurlex, $d = 3993, n =$	Coars	500	2.554	**0.591**	**73.577**	0.3485
$5000, nt = 1000, p = 5000$.	CSSP	500	**2.246**	0.504	81.989	**0.370**

6.4.4. Multilabel Classification. The last application we consider is that of multilabel classification (MLC). As seen in section 2, the most common approach to handle large number of labels in this problem is to perform a label dimension reduction assuming a low rank property of labels, i.e., only few labels are important. In this section, we propose to reduce the label dimension based on hypergraph coarsening. Article [6] presented a method for MLC based on CSSP using leverage score sampling. The idea is to replace sampling by hypergraph coarsening in this method.

Table 5 list the results obtained for MLC when coarsening and leverage score sampling (CSSP) were used for label reduction in the algorithm of [6] on different popular multilabel datasets. All datasets were obtained from https://manikvarma.github.io/downloads/XC/XMLRepository.html. The gist of the ML-CSSP algorithm is as follows: Given data with a large number of labels $Y \in \mathbb{B}^{n\times d}$, where $\mathbb{B}$ is a binary field with entries $\{0, 1\}$, we reduce the label dimension by subsampling or coarsening the label matrix, i.e., we reduce the d labels to $c < d$ labels. We then train c binary classifiers for these reduced c labels. For a new data point, we can predict whether the data-point belongs to the c reduced labels using the c binary classifiers, by getting a c dimensional predicted label vector. We then project the predicted vector onto d dimension and then use rounding to get the final d dimensional predicted vector.

All prediction errors reported (training and test) are Hamming loss errors, number of classes the predicted label vector differs from the exact label vector. The second metric used is $Precison@k$, which is a popular metric used in MLC literature [61]. This measures the precision of predicting the first k coordinates $|supp(\hat{y}_{1:k}) \cap supp(y)|/k$, where $supp(x) = \{i|x_i \neq 0\}$. In the above results, we chose k =the actual sparsity of the predicted label vector. This is equivalent to checking whether or not the proposed method predicted all the labels the data belongs to correctly. Other values of k such as Precision@k for $k = 1, 3, 5$ are used, where one is checking whether the top 1,3 or 5 labels respectively are predicted correctly, ignoring other and false labels. The better of the two results is highlighted. In this application too, we see that the coarsening method performs well and in many cases does better than the CSSP method which is more expensive.

7. Conclusion.
This paper advocated the use of coarsening techniques for three matrix approximation problems, namely, partial SVD, column subset selection and graph sparsification, and illustrated how the coarsening methods, and a combination of sampling and coarsening methods can be applied to solve these problems. We presented a few (new) applications for the coarsening technique, and demonstrated via several experiments that the coarsening technique performs very well in practice, better than the randomized methods in many cases. This is due to the fact that the coarsening technique exploits the structure of the input matrix. Coarsening is also

inexpensive compared to leverage score sampling, and yields comparable results. We also developed theoretical error bounds for the coarsening method. Interesting future work includes modifying the proposed coarsening technique for online and streaming settings.

REFERENCES

[1] Nir Ailon and Bernard Chazelle, *Faster dimension reduction*, Communications of the ACM, 53 (2010), pp. 97–104.

[2] Orly Alter, Patrick O Brown, and David Botstein, *Singular value decomposition for genome-wide expression data processing and modeling*, Proceedings of the National Academy of Sciences, 97 (2000), pp. 10101–10106.

[3] Haim Avron and Christos Boutsidis, *Faster subset selection for matrices and applications*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 1464–1499.

[4] Mikhail Belkin and Partha Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural computation, 15 (2003), pp. 1373–1396.

[5] Michael W Berry, Susan T Dumais, and Gavin W OBrien, *Using linear algebra for intelligent information retrieval*, SIAM review, 37 (1995), pp. 573–595.

[6] Wei Bi and James Tin Yau Kwok, *Efficient multi-label classification with many labels*, in 30th International Conference on Machine Learning, ICML 2013, 2013, pp. 405–413.

[7] Christos Boutsidis, Michael W Mahoney, and Petros Drineas, *An improved approximation algorithm for the column subset selection problem*, in Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 968–977.

[8] Christos Boutsidis and David P Woodruff, *Optimal cur matrix decompositions*, SIAM Journal on Computing, 46 (2017), pp. 543–589.

[9] U. V. Catalyurek and C. Aykanat, *Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication*, IEEE Transaction on Parallel and Distributed Systems, 10 (1999), pp. 673–693.

[10] J. Chen, H. R. Fang, and Y. Saad, *Fast approximate knn graph construction for high dimensional data via recursive Lanczos bisection*, Journal of Machine Learning Research, 10 (2009), pp. 1989–2012.

[11] Jie Chen and Yousef Saad, *Dense subgraph extraction with application to community detection*, IEEE Transactions on Knowledge and Data Engineering, 24 (2012), pp. 1216–1230.

[12] Timothy A Davis and Yifan Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), p. 1.

[13] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman, *Indexing by latent semantic analysis*, Journal of the American society for information science, 41 (1990), p. 391.

[14] K. Devine, E. G. Boman, R. Heaphy, R. Bisseling, and U. V. Catalyurek, *Parallel hypergraph partitioning for scientific computing*, in 20th International Parallel and Distributed Processing Symposium (IPDPS), 2006, p. 10.

[15] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V Vinay, *Clustering large graphs via the singular value decomposition*, Machine learning, 56 (2004), pp. 9–33.

[16] Petros Drineas, Ravi Kannan, and Michael 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.

[17] Petros Drineas, Michael W Mahoney, and S Muthukrishnan, *Relative-error cur matrix decompositions*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 844–881.

[18] Ehsan Elhamifar and René Vidal, *Sparse subspace clustering*, in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2790–2797.

[19] Haw-ren Fang, Sophia Sakellaridi, and Yousef Saad, *Multilevel nonlinear dimensionality reduction for manifold learning*, tech. report, Minnesota Supercomputer Institute, University of Minnesota, 2009.

[20] ———, *Multilevel manifold learning with application to spectral clustering*, in Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, 2010, pp. 419–428.

[21] Alan Frieze, Ravi Kannan, and Santosh Vempala, *Fast monte-carlo algorithms for finding low-rank approximations*, Journal of the ACM (JACM), 51 (2004), pp. 1025–1041. Preliminary version in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998.

[22] Richard A Gibbs, John W Belmont, Paul Hardenbol, Thomas D Willis, Fuli Yu, Huanming Yang, Lan-Yang Ch'ang, Wei Huang, Bin Liu, Yan Shen, et al., *The international hapmap project*, Nature, 426 (2003), pp. 789–796.

[23] Gene H. Golub and Charles F Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.

[24] Ming Gu, *Subspace iteration randomization and singular value problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1139–A1173.

[25] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.

[26] Bruce Hendrickson and Robert Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings of the 1995 ACM/IEEE conference on Supercomputing, ACM, 1995, p. 28.

[27] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford, *Single pass spectral sparsification in dynamic streams*, in Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on, IEEE, 2014, pp. 561–570.

[28] George Karypis and Vipin Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on scientific Computing, 20 (1998), pp. 359–392.

[29] G. Karypis and V. Kumar, *Multilevel k-way hypergraph partitioning*, VLSI Design, 11 (2000), pp. 285–300.

[30] Effrosini Kokiopoulou and Yousef Saad, *Polynomial filtering in latent semantic indexing for information retrieval*, in Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2004, pp. 104–111.

[31] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J-H Cui, and A. G. Percus, *Reducing large internet topologies for faster simulations*, in NETWORKING 2005, LNCS 3462, 2005, p. 328341.

[32] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar, *Sampling methods for the nyström method*, Journal of Machine Learning Research, 13 (2012), pp. 981–1006.

[33] Thomas K Landauer, Peter W Foltz, and Darrell Laham, *An introduction to latent semantic analysis*, Discourse processes, 25 (1998), pp. 259–284.

[34] Jure Leskovec and Christos Faloutsos, *Sampling from large graphs*, in Proceedings of KDD'06, Philadelphia, PA, USA, August 20-23, 2006.

[35] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 20167–20172.

[36] Michael W Mahoney and Petros Drineas, *Cur matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 697–702.

[37] Michael W Mahoney et al., *Randomized algorithms for matrices and data*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 123–224.

[38] Per-Gunnar Martinsson, Vladimir Rockhlin, and Mark Tygert, *A randomized algorithm for the approximation of matrices*, tech. report, DTIC Document, 2006.

[39] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf, *An introduction to kernel-based learning algorithms*, IEEE TRANSACTIONS ON NEURAL NETWORKS, 12 (2001), p. 181.

[40] Andrew Y Ng, Michael I Jordan, and Yair Weiss, *On spectral clustering: Analysis and an algorithm*, in Advances in Neural Information Processing Systems, 2001, pp. 849–856.

[41] Suely Oliveira and Sang-Cheol Seok, *A multi-level approach for document clustering*, Computational Science–ICCS 2005, (2005), pp. 23–32.

[42] Michael V Osier, Kei-Hoi Cheung, Judith R Kidd, Andrew J Pakstis, Perry L Miller, and Kenneth K Kidd, *Alfred: an allele frequency database for diverse populations and dna polymorphismsan update*, Nucleic acids research, 29 (2001), pp. 317–319.

[43] Lance Parsons, Ehtesham Haque, and Huan Liu, *Subspace clustering for high dimensional data: a review*, Acm Sigkdd Explorations Newsletter, 6 (2004), pp. 90–105.

[44] Peristera Paschou, Michael W Mahoney, Asif Javed, Judith R Kidd, Andrew J Pakstis, Sheng Gu, Kenneth K Kidd, and Petros Drineas, *Intra-and interpopulation genotype reconstruction from tagging snps*, Genome Research, 17 (2007), pp. 96–107.

[45] H. r. Fang and Y. Saad, *Hypergraph-based multilevel matrix approximation for text information retrieval*, in Proceedings of the ACM International Conference on Information and Knowledge Management, 2010, Jimmy Huang et al., ed., 2010, pp. 1597–1600.

[46] Soumya Raychaudhuri, Joshua M Stuart, and Russ B Altman, *Principal components analysis to summarize microarray experiments: application to sporulation time series*, in Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, NIH Public Access, 2000, p. 455.

[47] Sam T Roweis and Lawrence K Saul, *Nonlinear dimensionality reduction by locally linear*

embedding, science, 290 (2000), pp. 2323–2326.

[48] Yousef Saad, *Finding exact and approximate block structures for ilu preconditioning*, SIAM Journal on Scientific Computing, 24 (2003), pp. 1107–1123.

[49] Yousef Saad, *Numerical Methods for Large Eigenvalue Problems- classics edition*, SIAM, Philadelpha, PA, 2011.

[50] Yousef Saad, *Analysis of subspace iteration for eigenvalue problems with evolving matrices*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 103–122.

[51] S. Sakellaridi, H. R. Fang, and Y. Saad, *Graph-based multilevel dimensionality reduction with applications to eigenfaces and latent semantic indexing*, in Proceedings of Int. Conf. Mach. Learn. Appls. (ICMLA), 2008, M. Arif Wani, ed., IEEE comp. Soc., 2008, pp. 194–200.

[52] Ferdinando S Samaria and Andy C Harter, *Parameterisation of a stochastic model for human face identification*, in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, IEEE, 1994, pp. 138–142.

[53] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan, *Local graph sparsification for scalable clustering*, in Proceedings of SGMOD'11, Athens, Greece, June 11-16, 2011.

[54] Berkant Savas and Inderjit S Dhillon, *Clustered low rank approximation of graphs in information science applications*, in Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, 2011, pp. 164–175.

[55] Daniel A Spielman and Nikhil Srivastava, *Graph sparsification by effective resistances*, SIAM Journal on Computing, 40 (2011), pp. 1913–1926.

[56] Daniel A Spielman and Shang-Hua Teng, *Spectral sparsification of graphs*, SIAM Journal on Computing, 40 (2011), pp. 981–1025.

[57] Farbound Tai and Hsuan-Tien Lin, *Multilabel classification with principal label space transformation*, Neural Computation, 24 (2012), pp. 2508–2542.

[58] Joshua B Tenenbaum, Vin De Silva, and John C Langford, *A global geometric framework for nonlinear dimensionality reduction*, science, 290 (2000), pp. 2319–2323.

[59] Konstantinos Trohidis, *Multi-label classification of music into emotions.*, in 9th International Con- ference on Music Information Retrieval, 2008, pp. 325– 330.

[60] Grigorios Tsoumakas and Ioannis Katakis, *Multi-label classification: An overview*, in Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications, IGI Global, 2008, pp. 64–74.

[61] Shashanka Ubaru and Arya Mazumdar, *Multilabel classification with group testing and codes*, in Proceedings of the 34th International Conference on Machine Learning, Doina Precup and Yee Whye Teh, eds., vol. 70 of Proceedings of Machine Learning Research, International Convention Centre, Sydney, Australia, 06–11 Aug 2017, PMLR, pp. 3492–3501.

[62] Shashanka Ubaru, Arya Mazumdar, and Yousef Saad, *Low rank approximation and decomposition of large matrices using error correcting codes*, IEEE Transactions on Information Theory, 63 (2017), pp. 5544–5558.

[63] Shashanka Ubaru and Yousef Saad, *Fast methods for estimating the numerical rank of large matrices*, in Proceedings of The 33rd International Conference on Machine Learning, 2016, pp. 468–477.

[64] Shashanka Ubaru, Yousef Saad, and Abd-Krim Seghouane, *Fast estimation of approximate matrix ranks using spectral densities*, Neural Computation, 29 (2017), pp. 1317–1351.

[65] Eugene Vecharynski and Yousef Saad, *Fast updating algorithms for latent semantic indexing*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 1105–1131.

[66] Yining Wang and Aarti Singh, *An empirical comparison of sampling techniques for matrix column subset selection*, in Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on, IEEE, 2015, pp. 1069–1074.

[67] David P Woodruff, *Sketching as a tool for numerical linear algebra*, Foundations and Trends® in Theoretical Computer Science, 10 (2014), pp. 1–157.

[68] Hongyuan Zha and Horst D Simon, *On updating problems in latent semantic indexing*, SIAM Journal on Scientific Computing, 21 (1999), pp. 782–791.

[69] Ying Zhao and George Karypis, *Empirical and theoretical comparisons of selected criterion functions for document clustering*, Machine Learning, 55 (2004), pp. 311–331.

[70] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf, *Learning with hypergraphs: clustering, classification, and embedding*, in Proceedings of the 19th International Conference on Neural Information Processing Systems, MIT Press, 2006, pp. 1601–1608.

Appendix A. Existing Theory - Subspace Iteration.

Here we discuss the theoretical results for the subspace iteration algorithm established in the literature.

The subspace iteration algorithm has been employed and analyzed in the literature since a long time. The most recent analyses of subspace iteration appeared in [25, 24] and [50]. We present the following theorem which combines the results from [25, 24, 50].

Theorem A.3 (Deterministic bounds). *Given $A \in \mathbb{R}^{m\times n}$ with SVD $A = U\Sigma V^T$ and an initial subspace $\Omega \in \mathbb{R}^{n\times k}$. Let V_k be the top k right singular vectors with $\Omega_1 = V_k^T\Omega$, and V_{n-k} the bottom $n-k$ right singular vectors with $\Omega_2 = V_{n-k}^T\Omega$. Let Q be the subspace obtained after q steps of subspace iteration. Then, if Ω_1 is full rank, we have*

$$\|A - QQ^TA\| \leq (1 + \|\Omega_2\|\|\Omega_1^\dagger\|)^{1/(4q+2)}\sigma_{k+1}. \tag{10}$$

If $\tilde{\sigma}_j$ for $j = 1,\dots,k$ are the singular values obtained after q steps of subspace iteration. Then, we have

$$\sigma_j \geq \tilde{\sigma}_j \geq \frac{\sigma_j}{\sqrt{1 + \|\Omega_2\|^2\|\Omega_1^\dagger\|^2\left(\frac{\sigma_{k+1}}{\sigma_j}\right)^{(4q+2)}}}. \tag{11}$$

In addition we have,

$$\|q_j - u_j\| \leq \left(\frac{\sigma_{k+1}}{\sigma_j}\right)^q \|\Omega_{(j)} - u_j\|.$$

Thus, we need Ω_1 to be full rank and the error depends on its pseudoinverse, i.e., its smallest singular value. If the initial subspace is close to the top k singular vectors V_1, then Ω_1 will be well conditioned and the subspace iteration will converge rapidly. We know that the randomized subsampling as well as the coarsening algorithms give good approximation to the top k subspace. Hence, the incremental SVD algorithm presented in section 4 should converge rapidly.