Received: 9 March 2018 | Revised: 1 October 2018 | Accepted: 30 January 2019

DOI: 10.1002/nla.2234

RESEARCH ARTICLE

WILEY

Sampling and multilevel coarsening algorithms for fast matrix approximations

Shashanka Ubaru[1] | Yousef Saad[2]

[1]IBM Thomas J. Watson Research Center, Yorktown Heights, New York

[2]Department of Computer Science and Engineering, University of Minnesota at Twin Cities, Minneapolis, Minnesota

Correspondence
Shashanka Ubaru, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.
Email: ubaru001@umn.edu

Funding information
National Science Foundation, Grant/Award Number: NSF/CCF-1318597

Summary

This paper addresses matrix approximation problems for matrices that are large, sparse, and/or 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. We consider a number of standard applications of this technique as well as a few new ones. Among standard applications, we first consider the problem of computing *partial singular value decomposition*, for which a combination of sampling and coarsening yields significantly improved singular value decomposition 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. We also establish theoretical results that characterize the approximation error obtained and the quality of the dimension reduction achieved by a coarsening step, when a proper column matching strategy is employed. Numerical experiments illustrate the performances of the methods in a few applications.

KEYWORDS

coarsening, multilevel methods, randomization, singular values, subspace iteration, SVD

1 | INTRODUCTION

Modern applications involving data often rely on very large data sets, but in most situations, the relevant information lies in a low-dimensional subspace. In many of these applications, the data matrices are 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,[1–3] graph sparsification,[4,5] and compression.[6] Methods to obtain low-rank approximations include partial singular value decomposition (SVD)[2] and column subset selection (CSS).[7] Efficient methods have been developed to compute partial SVD,[8,9] 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 data sets prohibit even the use of rather ubiquitous methods such as the Lanczos or subspace iteration algorithms,[8,10] since these algorithms

This work was undertaken when the author was a student at University of Minnesota.

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 "randomization" techniques,[1,2,11] whereby an approximate partial SVD is obtained from a small randomly sampled subset of the matrix, or possibly a few subsets. Several randomized embedding and sketching methods have also been proposed.[2,12] These randomization techniques are well established (theoretically) and are proven to give good results in some situations (see the work of Mahoney[3] for a review). In this paper, we will consider random sampling methods as a first step to downsample very large data sets when necessary. However, randomized methods by themselves can be suboptimal in many situations since they do not exploit available information or the redundancies in the matrix. For example, many sampling methods only consider column norms, and embedding and sketching methods are usually independent of the input matrix. For this reason, such schemes are often termed "data oblivious."[13] One of the goals of this work is to show that multilevel graph coarsening, a technique that is often used in the different context of graph partitioning,[14] can provide superior alternatives to randomized sampling, at a moderate cost.

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. The problem of graph and hypergraph coarsening has been extensively studied in the literature.[14–17]

These techniques are more accurate than downsampling with column-norm probabilities[1] and yield comparable results to the popular leverage score–based sampling,[18] which can be expensive but more accurate than column norm–based sampling. Moreover, coarsening will be inexpensive compared to these sampling methods for large (size $n > 10^5$) and sparse ($\text{nz}(A) = O(n)$) matrices. For really large ($n > 10^6$) matrices, a typical algorithm would first perform uniform downsampling of the matrix to reduce the size of the problem and then utilize a multilevel coarsening technique for computing an approximate partial SVD of the reduced matrix.

The second low-rank approximation problem considered in this paper is the *CSS problem* (CSSP)[7,19] or CUR decomposition.[18,20] Here, the goal is to select a subset of columns that best represent the given matrix spectrally, that is, with respect to spectral and Frobenius norms. Popular methods for the CSSP use the leverage score sampling measure 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. We will see how graph coarsening techniques can be adapted for CSS. The coarsening approach is an inexpensive alternative for this problem, and it performs well in many situations as the experiments will show.

The third problem we consider is that of *graph sparsification*.[4,5,21] 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 reduced space and time complexity. In particular, we are interested in spectral sparsifiers, where the Laplacian of $\tilde{G}$ spectrally approximates the Laplacian of G.[5,12,22] 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 will again show how graph coarsening can be adapted for the task of graph sparsification.

The outline of this paper is as follows. Section 2 describes a few existing algorithms to compute (low-rank) approximations of matrices. Graph coarsening and multilevel algorithms are presented in Section 3. In particular, we present a hypergraph coarsening technique based on column matching and discuss methods to improve the SVD obtained from randomization and coarsening methods. In Section 4, we establish theoretical error bounds for the coarsening method. Section 5 discusses a few data-related applications of graph coarsening. Numerical experiments illustrating the performances of these methods in a variety of applications are presented in Section 6.

2 | BACKGROUND

In this section, we review three well-known classes of methods to calculate the partial SVD of matrices. The first class is based on randomized sampling. We also consider solving the CSSP and the graph sparsification problem using randomized sampling, particularly based on 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.[23,24] 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 combined with the multilevel algorithms to be discussed in Section 3, these updates are not small since the number of columns can double.

2.1 | Random sampling

Randomized algorithms have attracted much attention in recent years due to their broad applications as well as to the related theoretical results that have been shown, which 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.[2,25–28] Drineas et al.[1,11] presented the randomized subsampling algorithms, where a submatrix consisting of a few columns of the original matrix is randomly selected based on a certain probability distribution. Their method samples the columns based on column norms. Specifically, column i of a given matrix $A \in \mathbb{R}^{m\times n}$ is selected with 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 ith column of A. Using the above distribution, c columns are selected, and the subsampled matrix C is formed by scaling the columns by $1/\sqrt{cp_i}$. The SVD of C is then computed. The approximations obtained by this randomization method will yield reasonable results only when there is a sharp decay in the singular value spectrum and when computing all the column norms and then sampling can become expensive for really large matrices.

2.2 | Column subset selection

Another well-known dimensionality reduction method considered in this paper is the CSS (CSSP)[7] technique. If a subset of the rows is also selected, then the method leads to CUR decomposition.[20] 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 ith column of A is given by

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

the scaled square norm of the ith row of V_k. In leverage score sampling, the columns of A are then sampled using the probability distribution $p_i = \min\{1, \ell_i\}$. The most popular methods for the CSSP involve the use of this leverage scores as the probability distribution for column selection.[7,18,20,29] Greedy subset selection algorithms have also been proposed based on the right singular vectors of the matrix.[30,31] Note that these methods may be expensive when rank k is not small since they require the top-k singular vectors. An alternative solution to the CSSP is to use the coarsened graph consisting of the columns obtained by graph coarsening. We will see how this technique compares with standard ones.

2.3 | Graph sparsification

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 spectral information,[5,22] structures like clusters within the graph,[21,32] etc. Sparsifying large graphs has several computational advantages and has hence found many applications.[4,21,22,32,33]

Let $B \in \mathbb{R}^{\binom{n}{2}\times n}$ be the vertex edge incidence matrix of graph G, where the 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, in which w_e is the weight of the edge. The corresponding Laplacian of the graph is then given by $K = B^T B$. 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 $x \in \mathbb{R}^n$. A number of methods have been proposed for sparsifying graphs.[4,5,12,22] One such approach performs row sampling of the matrix B using the leverage score sampling criterion.[5] 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.[4] 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 3, we show how the rows of B can be selected via coarsening.

2.4 | Subspace iteration

Subspace iteration is a well-established method used for solving eigenvalue and singular value problems.[8,9] 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.[10] 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 $A_s = U_s\Sigma_s V_s^T$, we can perform a few steps of subspace iteration updates as shown in Algorithm 1. Note that the last two steps in the algorithm, which diagonalize S, are needed only if we require the current estimate of singular vectors and values.

Algorithm 1 Incremental subspace iteration

Input: U_s, from $A_s = U_s\Sigma_s V_s^T$, coarsened matrix of A, A_t (larger coarsened matrix), and no. of iterations *iter*.
Output: Updated SVD $[U, \Sigma, V]$ of A.
Start: $U = U_s$
for $i = 1 : iter$ **do**
  $V = A_t^T U$
  $U = A_t V$
  $U := qr(U, 0); \quad V := qr(V, 0);$
  $S = U^T A_t V$
  $[R_U, \Sigma, R_V] = \text{svd}(S); U := UR_U; V := VR_V$
  // Above two steps used only if we require current estimate of singular vectors and values
end for

2.5 | SVD updates from subspaces

A well-known algorithm for updating the SVD is the "updating algorithm" of Zha and Simon.[23] Given a matrix $A \in \mathbb{R}^{m\times n}$ 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 $A_D = [A, D]$, where $D \in \mathbb{R}^{m\times p}$ represents the added columns. The algorithm then first computes

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

the truncated (thin) 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

$$A_D = \begin{bmatrix} U_k, & \hat{U}_p \end{bmatrix} H_D \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}^T, \quad H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & R \end{bmatrix}, \tag{2}$$

where I_p denotes the $p \times p$ 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

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

The QR decomposition in the first step given by Equation (1) can be expensive when the updates are large; hence, an improved version of this algorithm was proposed in the work of Vecharynski and Saad,[24] 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 in 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 applications involving low-rank approximations, rank k will be typically much smaller than n, and it can be inexpensively estimated using recently proposed methods.[34,35]

3 | COARSENING

As discussed in the previous section, randomized sampling methods can be an effective approach for computing the partial SVD in certain situations, for example, when there is a good gap in the spectrum or if there is a sharp spectral decay. An alternative approach to reduce the matrix dimension, particularly when the matrices are associated with graphs, is to coarsen the data with the help of graph coarsening, perform all computations on the resulting reduced-size matrix, and then project back to the original space. Similar 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[36]; this is illustrated in Figure 1 (left). See, for example, other works[36–39] 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 the work of Chen et al.[40] 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 (MLC), to mention a few examples, and advocate the use of the coarsening technique for dimension reduction. In such cases, the data are represented by a (rectangular) sparse matrix, and it is most convenient to use hypergraph models[37] for coarsening.

3.1 | 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.[16,37] 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 sparse 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, that is, the columns, whose corresponding entries in that row are nonzero. 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 representing the columns of A. Several methods exist for coarsening hypergraphs.[16,17] Here, we consider a hypergraph coarsening based on column matching, which is a modified version of

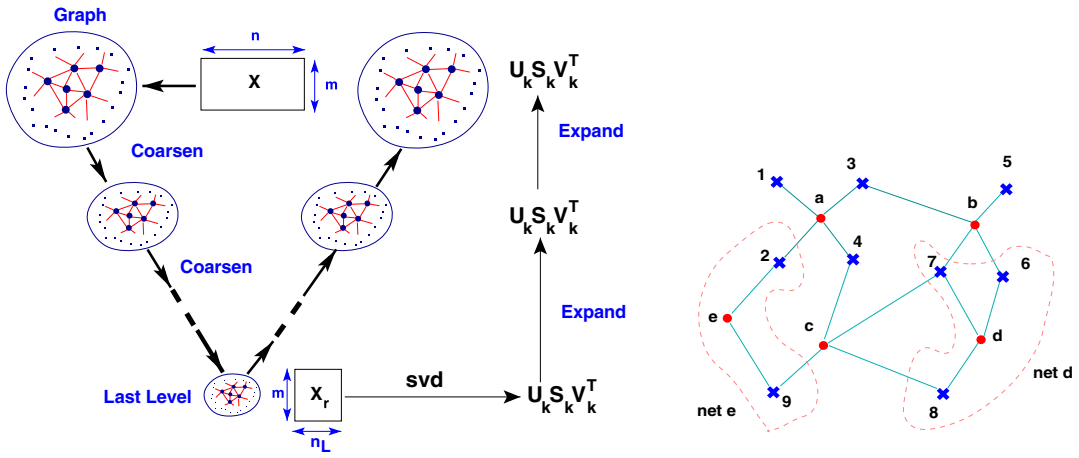

FIGURE 1 (Left) Coarsening/uncoarsening procedure. (Right) A sample hypergraph

the *maximum-weight matching* method.[16,41] The modified approach follows the maximum-weight matching method and computes the nonzero inner product $\langle a^{(i)}, a^{(j)} \rangle$ between two vertices i and j, that is, the ith and jth columns of A. Note that the inner product between vectors is related to the angle between the vectors, that is, $\langle a^{(i)}, a^{(j)} \rangle = \|a^{(i)}\| \|a^{(j)}\| \cos\theta_{ij}$. The proposed coarsening strategy is to match two vertices, that is, columns, only if the angle between the vertices is such that $\tan\theta_{ij} \leq \epsilon$ for a constant $0 < \epsilon < 1$. Another feature of the proposed 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

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

where $a^{(i)}$ is replaced by $a^{(j)}$ if the latter 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 4).

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 fast access to rows and columns is required. The vertex j with the highest nonzero 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 provides details on the procedure.

Algorithm 2 Hypergraph coarsening by column matching

Input: $A \in \mathbb{R}^{m \times n}$, $\epsilon \in (0, 1)$.
Output: Coarse matrix $C \in \mathbb{R}^{m \times c}$.
$Idx := \{1, \ldots, n\}$ // (Set of unmatched vertices)
Set $\text{ip}[k] := 0$ for $k = 1, \ldots, n$, and $\ell = 1$. // (Initialization)
repeat
  Randomly pick $i \in Idx$; $Idx := Idx - \{i\}$.
  for all j with $a_{ij} \neq 0$ **do** // (*)
    for all k with $a_{jk} \neq 0$ **do**
      $\text{ip}[k] := \text{ip}[k] + a_{ij} a_{jk}$.
    end for
  end for
  $j := \text{argmax}\{\text{ip}[k] : k \in Idx\}$
  $csq\theta = \frac{\text{ip}[j]^2}{\|a^{(i)}\|^2 \|a^{(j)}\|^2}$.
  if $\left[\left(csq\theta \geq \frac{1}{1+\epsilon^2} \right) \right]$ **then** // Match only if the angle satisfies the condition
    $c^{(\ell)} := \sqrt{1 + csq\theta}\, a^{(i)}$. // The denser of columns $a^{(i)}$ and $a^{(j)}$
    $Idx := Idx - \{j\}$; $\ell = \ell + 1$.
  else
    $c^{(\ell)} := a^{(i)}$.
    $\ell = \ell + 1$.
  end if
  Reset nonzero values of ip to zero// (A sparse operation)
until $Idx = \emptyset$

Computing the cosine of the angle between column i and all other columns is equivalent to computing the ith row of $A^T A$. In fact, we only need to compute the upper triangular part of $A^T A$ since it is symmetric. For sparse matrices, the inexpensive computation of the inner product between the columns used in the algorithm is achieved by modifying the cosine algorithm in the work of Saad[42] developed for matrix block detection. Thus, loop (*) computes all inner products of column i with all other columns and accumulates these in the sparse "row" ip[.]. This amounts, in essence, to computing the ith row of $A^T A$ as the combination of rows: $\sum_{a_{ij} \neq 0} a_{ij} a_{j,:}$. As indicated in the line just before the end, resetting ip[.] to zero is a sparse operation that does not require zeroing out the whole vector but only those entries that are nonzero.

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. Note that the number of vertices (columns) in C will depend on the redundancy among the data and the ϵ value chosen (see further discussion in Section 4).

Computational cost

We saw earlier that the inner products of a given column i with all other columns amount to computing the nonzero values of the ith row of the upper triangular part of A^TA. If we call $Adj_A(i)$ the set of nonzero indices of the ith column of A, then according to what was said above, the cost of computing ip[.] by the algorithm is

$$\sum_{j \in Adj_A(i),\ j>i} |Adj_{A^T}(j)|,$$

where $|\cdot|$ is the cardinality of the set. If ν_c (respectively, ν_r) is the maximum number of nonzeros in each column (respectively, row), then an upper bound for the above cost is $n\nu_r\nu_c$, which is the same upper bound as that of 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 $\tan\theta_{ij} \leq \epsilon$, 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. Hence, this coarsening approach can be quite inexpensive for very large sparse matrices (see Section 6 for numerical results). The work of Chen and Saad[43] uses the angle-based column matching idea for dense subgraph detection in graphs and describes efficient methods to compute the inner products.

3.2 | Multilevel SVD computations

Given a sparse matrix A, we can use Algorithm 2 repeatedly with different (increasing) ϵ values, to recursively coarsen the corresponding hypergraph and obtain a sequence of sparse matrices $A_1, A_2, \ldots, A_s$ with $A_0 = A$, where A_i corresponds to the coarse graph H_i of level i for $i = 1, \ldots, s$, and A_s represents the lowest-level graph H_s. This provides a reduced-size matrix, which will likely be 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, for instance, low-rank approximation using the SVD, to the coarsened data matrix $A_s \in \mathbb{R}^{m\times n_s}$ at the lowest level, where n_s is the number of columns at the coarse level s ($n_s < n$). A low-rank matrix approximation can be viewed as a linear projection of the rows into a lower-dimensional space. In other words, we have a projected matrix $\hat{A}_s \in \mathbb{R}^{d\times n_s}$ ($d < m$) from the coarse matrix A_s that preserves certain features of A_s (for instance, spans the top-k singular vectors of A_s). Suppose we apply the same linear projection to $A \in \mathbb{R}^{m\times n}$ (the original matrix), resulting in $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$), we can expect that $\hat{A}$ preserves certain features of A (approximately spans the top-k singular vectors of A). Hence, we can obtain a reduced representation $\hat{A} \in \mathbb{R}^{d\times n}$ ($d < m$) of the original data A using this linear projection. The procedure is illustrated in Figure 1 (left). A multilevel randomized sampling technique is discussed in the work of Drineas et al.[1] Another strategy for reducing the matrix dimension is to mix the two techniques: Coarsening may still be exceedingly expensive for some types of large 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 (for example, uniform sampling) and then construct a graph and coarsen it. In Section 6, we compare the SVDs obtained from the pure-randomization method (column-norm sampling) against those obtained from coarsening and a combination of randomization (uniform sampling) and coarsening.

3.3 | CSSP and graph sparsification

The multilevel coarsening technique just presented can be applied for the CSSP as well as for the graph sparsification problem. We can use Algorithm 2 to coarsen the matrix, and this is equivalent to selecting columns of the matrix. The only required modification in the algorithm is that the columns selected are no longer scaled. The coarse matrix C contains a subset of the columns of the original matrix A, and the analysis will show that it is a faithful representation of the original A.

For graph sparsification, we can apply the coarsening procedure on the vertex edge incidence matrix B corresponding to graph G. That is, we coarsen the rows of B (by applying Algorithm 2 to B^T) and obtain a matrix $\tilde{B}$ with fewer rows, yielding us a graph $\tilde{G}$ with fewer edges. From the analysis in Section 4, we can show that this coarsening strategy will indeed result in a spectral sparsifier, that is, we can show that $x^T\tilde{B}^T\tilde{B}x$ will be close to x^TB^TBx for any vector $x : \|x\| = 1$. Since we achieve sparsification via matching, the structures such as clusters within the original graph are also preserved.

3.4 | Incremental SVD

Next, we explore combined algorithms with the goal of improving the randomized sampling and coarsening SVD results significantly. The typical overall algorithm, which we call the incremental SVD algorithm, will start with a sampled/coarsened matrix A_s using one of the randomized methods or multilevel coarsening discussed above and compute its partial SVD. We then add columns of A to A_s and use Algorithm 1 or the SVD-update algorithm to update the SVD results. A version of this incremental algorithm has been briefly discussed in the work of Gu,[44] where the basic randomized algorithm is combined with subspace iteration.

Roughly speaking, we know that each iteration of the subspace iteration algorithm will make 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 an idea on how the top-k singular vectors for $k < c$ of the coarsened matrix relate to the span of the top-k singular vectors of A. In such cases, a few steps of the subspace iteration will then yield very accurate results.

For the SVD-updating method discussed in Section 2.5, it is known that the method performs well when the updates are of low rank and when they do not significantly affect the dominant subspace, the subspace spanned by the top-k singular vectors of interest.[24] 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 affect the dominant subspace only slightly. This is the reason we expect that the SVD-updating method will yield improved results.

4 | ANALYSIS

In this section, we establish initial theoretical results for the coarsening technique based on column matching. In the coarsening strategy of Algorithm 2, we combine two columns $a^{(i)}$ and $a^{(j)}$ if the angle between them is such that $\tan\theta_i \le \epsilon$. That is, we set $c^{(\ell)} = (\sqrt{1+\cos^2\theta_i})a^{(i)}$ (or $a^{(j)}$, if it has more nonzeros) in place of the two columns in the coarsened matrix C. We then have the following result, which relates the Rayleigh quotients of the coarsened matrix C to that of A (indicates $AA^T \approx CC^T$, spectrally).

Lemma 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 using Algorithm 2 with columns $a^{(i)}$ and $a^{(j)}$ matched if $\tan\theta_i \le \epsilon$, for $0 < \epsilon < 1$. Then, we have*

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

for any $x \in \mathbb{R}^m$ with $\|x\|_2 = 1$.

Proof. Let (i,j) 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 by J the set of the remaining columns.

We know that $x^TAA^Tx = \|A^Tx\|_2^2 = \sum_{i=1}^n \langle a^{(i)},x\rangle^2$. 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 $c_i = \cos\theta_i$. We have

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

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

$$a^{(j)} = c_i a^{(i)} + s_i w,$$

where $s_i = \sin\theta_i$ and $w = \|a^{(i)}\|\bar{w}$, with $\bar{w}$ as a unit vector that is orthogonal to $a^{(i)}$. Then, we have

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

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

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

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

$$\left|\langle a^{(j)}, x\rangle^2 - c_i^2\langle a^{(i)}, x\rangle^2\right| \le 2\eta_i^2\epsilon + \eta_i^2\epsilon^2 \le 3\epsilon\eta_i^2$$

as $\epsilon < 1$. Thus, we have

$$\left|x^T A A^T x - x^T C C^T x\right| \le 3\epsilon \sum_{i\in I} \|a^{(i)}\|^2 \le 3\epsilon\|A\|_F^2.$$

□

Note that the above bound will be better in practice because the last inequality may not be tight. In fact, improving this result could be an interesting question to investigate. We also observe that the above result will hold even if we consider combining multiple columns that are within the angle $\theta = \tan^{-1}(\epsilon)$ from each other into one in Algorithm 2. The number of columns c in the coarsened matrix C will depend on the given data (the number of pairs of columns that are within the desired angle). The above lemma is similar to the results established in the work of Liberty[45] for deterministic matrix sketching. It can be used to develop the following error bounds.

Theorem 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 $a^{(j)}$ 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, for $k \le c$. Then, we have*

$$\left\|A - H_k H_k^T A\right\|_F^2 \le \|A - A_k\|_F^2 + 6k\epsilon\|A\|_F^2 \tag{5}$$

$$\left\|A - H_k H_k^T A\right\|_2^2 \le \|A - A_k\|_2^2 + 6\epsilon\|A\|_F^2, \tag{6}$$

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 $\|A - H_k H_k^T A\|_F^2$ as

$$\begin{aligned}\left\|A - H_k H_k^T A\right\|_F^2 &= \mathrm{Tr}\left(\left(A - H_k H_k^T A\right)^T\left(A - H_k H_k^T A\right)\right) \\ &= \mathrm{Tr}\left(A^T A - 2A^T H_k H_k^T A + A^T H_k H_k^T H_k H_k^T A\right) \\ &= \mathrm{Tr}\left(A^T A\right) - \mathrm{Tr}\left(A^T H_k H_k^T A\right) \\ &= \|A\|_F^2 - \left\|A^T H_k\right\|_F^2.\end{aligned} \tag{7}$$

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

From Lemma 1, for each i, we have

$$\left| \left\|A^T h^{(i)}\right\|^2 - \left\|C^T h^{(i)}\right\|^2 \right| = \left| \left\|A^T h^{(i)}\right\|^2 - \sigma_i^2(C) \right| \leq 3\epsilon \|A\|_F^2,$$

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

$$\left| \|A^T H_k\|_F^2 - \sum_{i=1}^k \sigma_i^2(C) \right| \leq 3\epsilon k \|A\|_F^2. \tag{8}$$

From the perturbation theory (see theorem 8.1.4 in the work of Golub and Van Loan[9]), for $i = 1, \ldots, c$, we have

$$\left| \sigma_i^2(C) - \sigma_i^2(A) \right| \leq \left\|AA^T - CC^T\right\|_2.$$

Next, Lemma 1 implies

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

Hence, summing up over k singular values, we obtain

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

Combining (8) and (9), we get

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

Along with (7), this relation 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: Let $\mathcal{H}_k = \mathrm{range}(H_k) = \mathrm{span}(h^{(1)}, \ldots, h^{(k)})$, 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, we have

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

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,

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

Since $\left| \|z^T A\|^2 - \|z^T C\|^2 \right| \leq 3\epsilon \|A\|_F^2$ from Lemma 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 Theorem 1 is similar to the results developed for randomized sampling methods (see the works of Drineas et al.[1,11]). One notable difference is that to achieve the above result for a given rank k, the method based on column norms for randomized sampling requires $c = \Theta(k/\epsilon^2)$ columns to be sampled to form C. For a small ϵ, the number of columns c will be quite large. The error for a given rank k diminishes as c increases, but when k is large, a large number of columns will be required to get a good approximation. For the coarsening method, the above error bounds hold for any $k \leq c$, and the number of columns c will depend primarily on the given data. The error will be smaller if the angles between the columns that are combined are smaller. The number of columns is related to these angles, and this, in turn, depends on the redundancy among columns of the given matrix. As future work, it would be interesting to say how many

distinct columns will be needed to ensure that the subspace spanned by the columns of C is a good rank-k approximation to the range of A.

The subspace iteration algorithm has been extensively studied, and the most recent analyses of subspace iteration appeared in the works of Halko et al.,[2] Gu,[44] and Saad.[10] The convergence of subspace iteration depends on the quality of the initial subspace as an approximation to the dominant subspace, that is, the subspace associated with the top singular values. The above theorem shows that the coarsening algorithm gives a reasonable approximation to the subspace spanned by the top-k singular vectors. This approximation is therefore a good starting point for the incremental SVD algorithm presented in Section 3.

5 | APPLICATIONS

Here, we present a few applications where the (multilevel) coarsening method discussed in the previous section can be employed. The matrices encountered in these applications are typically large and sparse, often representing graphs.

1. Latent semantic indexing - Latent semantic indexing (LSI) is a well-established text mining technique for processing queries in a collection of documents.[46–49] Given a user's query, the method is used to retrieve a set of documents from a given collection that are most relevant to the query. The truncated SVD[48] and related techniques[49] are common tools used in LSI. The argument exploited 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.[1] Multilevel coarsening for LSI was considered in the work of Sakellaridi et al.[36] Here, we revisit and expand this idea and show how hypergraph coarsening can be employed in this application.
2. Projective clustering - Several projective clustering methods, such as isomap,[50] local linear embedding (LLE),[51] spectral clustering,[52] subspace clustering,[53,54] Laplacian eigenmaps,[55] and others, involve a partial eigendecomposition or a partial SVD of graph Laplacians. Various kernel-based learning methods[56] also require the (partial) SVD of large-graph Laplacians. In most applications today, the number of data points is large, and computing singular vectors (eigenvectors) is expensive in most cases. Graph coarsening is an effective strategy to reduce the number of data points in these applications (see the works of Fang et al.[38] and Oliveira and Seok[57] for a few illustrations).
3. Eigengene analysis - Analyzing gene expression DNA microarray data has become an important tool in the study of a variety of biological processes.[30,58,59] In a microarray data set, we have m genes (from m individuals possibly from different populations) and a series of n-array probe genome-wide expression levels in n different samples, possibly under n different experimental conditions. The data are large with several individuals and gene expressions but are 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.[58] The work of Paschou et al.[30] showed how CSS (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 the next section, we will show how hypergraph coarsening can be adapted to choose a good (small) subset of genes in this application.
4. Multilabel classification - The last application we consider is that of MLC in machine learning.[60,61] In the MLC 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 that 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 the jth 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 MLC problems occur in many domains, such as computer vision, text mining, and bioinformatics,[62,63] and modern applications involve a large number of labels.

A common 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 the work of Tai and Lin[63] and using CSS in the work of Bi and Kwok.[64] In this work, we demonstrate how hypergraph coarsening can be employed to reduce the number of classes, and yet achieve accurate learning and prediction.

The work of Savas and Dhillon[65] discusses a number of methods that rely on clustering the data first in order to build a reduced-dimension representation. It can be viewed as a top-down approach, whereas coarsening is a bottom-up method.

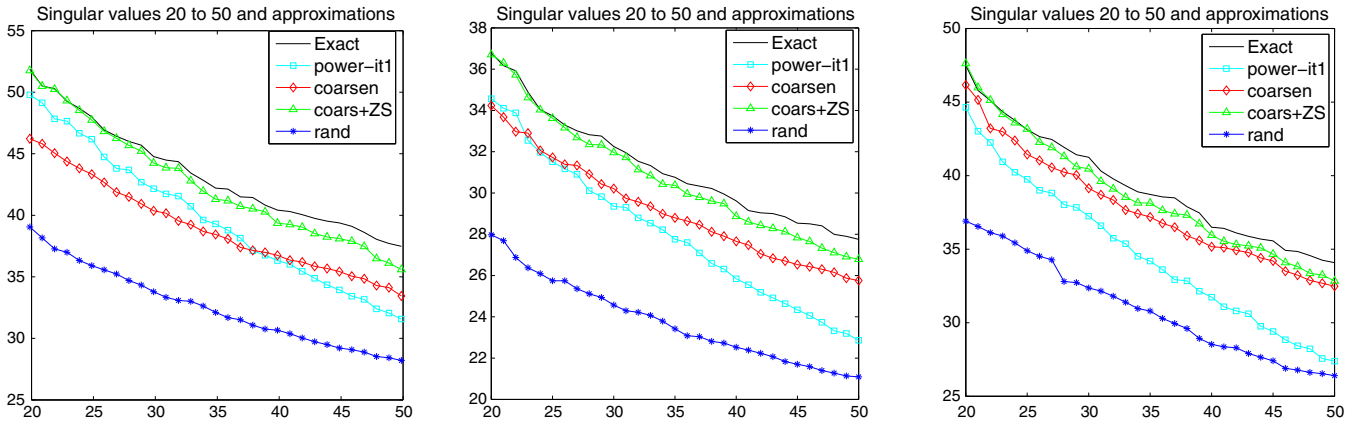

FIGURE 2 Results for the data sets CRANFIELD (left), MEDLINE (middle), and TIME (right)

6 | NUMERICAL EXPERIMENTS

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

6.1 | SVD comparisons

In the first set of experiments, we start with three small term-by-document data sets and compare the sampling, coarsening, and combined methods to compute the SVD. The tests are with unweighted versions of the CRANFIELD data set (1,398 documents, 5,204 terms), the MEDLINE data set (1,033 documents, 8,322 terms), and the TIME data set (425 documents, 13,057 terms). These data sets are used in the LSI application examples.

Figure 2 illustrates the following experiment with the three data sets. Results from four different methods are plotted. The first solid curve (labeled "exact") shows the singular values numbered 20 to 50 of a matrix A computed using the `svds` function in MATLAB (the results obtained by the four methods for the top-20 singular values were similar). The diamond curve labeled "coarsen" shows the singular values obtained by one level of coarsening using Algorithm 2 (details will be given later). The star curve (labeled "rand") shows the singular values obtained by the random column-norm sampling method (columns sampled with probability $p_i = \|a_i\|_2^2/\|A\|_F^2$), with a sample size c equal to the size of coarsened matrix C obtained with one level of coarsening. We note that the result obtained by coarsening is much better than that obtained by random sampling method, particularly for large k. This behavior was also predicted by our theoretical results. 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 the incremental SVD algorithms. The curve with triangles labeled "coars+ZS" shows the singular values obtained when the Zha–Simon algorithm was employed 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 the power method,[2] that is, from the SVD of the matrix $Y = (AA^T)A\Omega$, where Ω is a random Gaussian matrix with the same number of columns 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 3, a possible way to improve 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 MEDLINE (right) data sets. We have not reported the results for the TIME data set 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 column-norm 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 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 column-norm sampling ("rand+ZS"), respectively. These plots show that both the SVD-update algorithm and subspace iteration improve the accuracy of the SVD significantly.

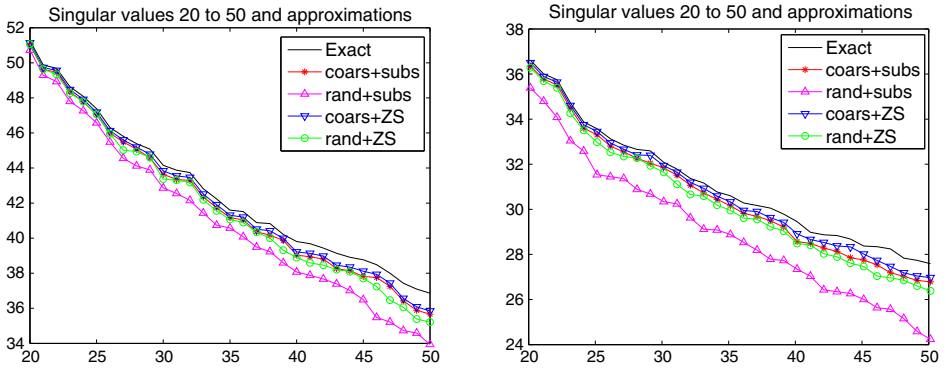

FIGURE 3 Second set of results for the CRANFIELD (left) and MEDLINE (right) data sets

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

Data set	n	k	c	Coarsening			Col. Norm Sampling			Rand+Coarsening		
				Err1	Err2	Time	Err1	Err2	Time	Err1	Err2	Time
Kohonen	4,470	50	1,256	86.26	0.366	1.52 s	93.07	0.434	0.30 s	93.47	0.566	0.43 s
aft01 (n)	8,205	50	1,040	913.3	0.299	2.26 s	1006.2	0.614	0.81 s	985.3	0.598	0.63 s
FA	10,617	30	1,504	27.79	0.131	2.06 s	28.63	0.410	1.83 s	28.38	0.288	0.43 s
chipcool0	20,082	30	2,533	6.091	0.313	5.69 s	6.199	0.360	5.92 s	6.183	0.301	1.27 s
scfxm1-2b	33,047	25	2,066	2256.8	0.202	2.8 s	2328.8	0.263	5.38 s	2327.5	0.153	0.57 s
thermechTC	102,158	30	6,286	2.063	0.214	7.8 s	2.079	0.279	164 s	2.076	0.269	2.51 s
HTC9129(n)	226,340	25	14,147	212.8	0.074	247.6 s	226.34	0.102	481.7 s	234.08	0.1512	57.8 s
ASIC-680(n)	682,712	25	21,335	2913	0.239	57.4 s	2925	0.304	17.3 m	2931	0.448	13.2 s
Webbase1M	1,000,005	25	15,625	3502.2	–	15.3 m	3564.5	–	17.8 m	3551.7	–	5.1 m
Netherland	2,216,688	25	34,636	1.53e4	–	51.8 s	2.05e4	–	584.3 s	1.58e4	–	15.3 s
adaptive	6,815,744	25	106,496	2850.2	–	297.4 s	3256.8	–	52.6 m	2984.7	–	98.3 s
delaunay-23	8,388,608	25	131,072	3145.4	–	13.2 m	3904.2	–	38.2 m	3446.1	–	3.8 m

Next, we compare the computational times and performances of our coarsening method and the column-norm sampling method for computing the low-rank approximation of matrices. We also consider the combined method of uniform downsampling followed by coarsening discussed in the Introduction and in Section 3. Table 1 shows comparison results between the three methods, namely, coarsening, column-norm sampling, and uniform 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/)[66] 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 (k listed in the third column). For the smaller matrices, we also report the average absolute normalized error in the singular values $= \frac{1}{k}\sum_{i \le k} \frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i}$ for rank k, which is related to our result in Lemma 1 and spectral approximation that is of interest. We also report the computational time in seconds (s) or minutes (m) for the three methods (using the `cputime` function in MATLAB). All experiments were conducted on an Intel i5-4590 CPU @3.30GHz machine. 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. Some of the matrices (indicated by (n)), which had large Frobenius norms, were normalized by $\sqrt{\|X\|_F}$. For smaller matrices, the exact singular values σ_i's are available in the database. For matrices with $n > 10^5$, these were computed using the "svds" function. For larger matrices with $n > 10^6$, the exact singular values cannot be computed; hence, we were unable to report Error2 for these matrices.

For the combined method (random sampling+coarsening), we proceed as follows: First, half of the columns ($n/2$) are randomly sampled (uniform downsampling), and then, multilevel coarsening is performed with one level less than the pure-coarsening method reported in the previous column. We note that coarsening clearly yields better results (lower errors) than the column-norm sampling method. The combined method of random sampling+coarsening works well and performs better than norm sampling in most cases. Coarsening is relatively expensive for smaller matrices. However, we note that for larger (and sparse) matrices, coarsening becomes less expensive than column-norm sampling, which requires computing the norms of n columns and then sampling based on the ratio of the norms. For really large matrices,

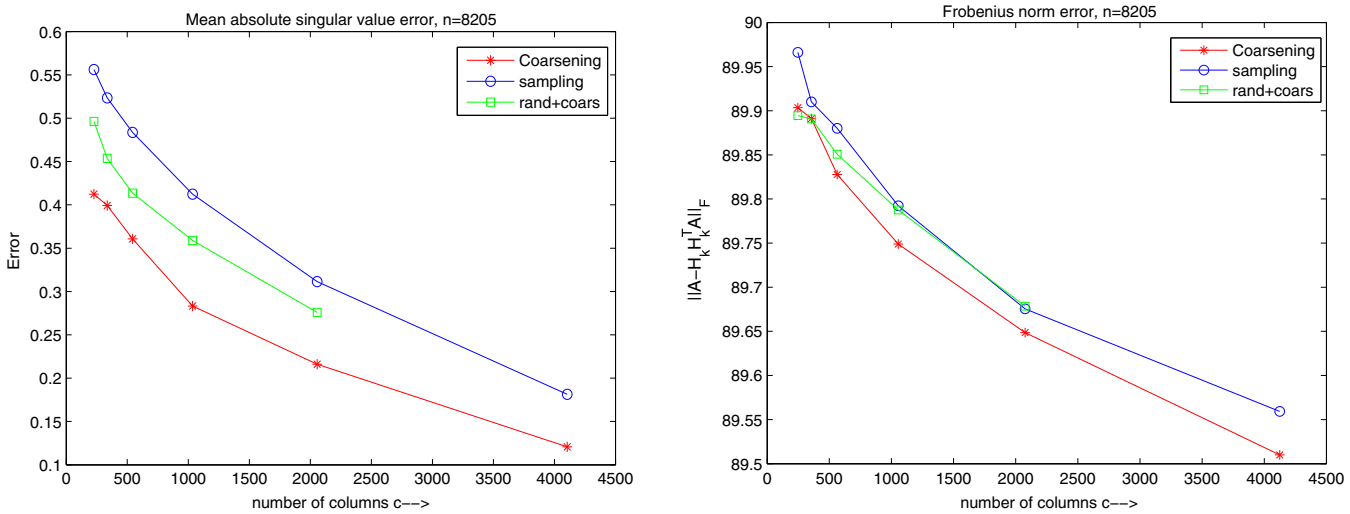

FIGURE 4 Mean absolute singular value errors $\frac{1}{k}\sum_{i=1}^{k}\frac{|\hat{\sigma}_i-\sigma_i|}{\sigma_i}$ (left) and Frobenius-norm errors $\|A - H_k H_k^T A\|_F$ (right) for the three methods for the `aft01` data set ($k = 30$)

the "rand+coarse" method seems to be a good choice due to its computational cost. The coarsening methods scale easily to even larger matrices. However, estimating the error $\|A - H_k H_k^T A\|_F$ becomes impractical.

For further illustration, Figure 4 plots the two errors $\|A - H_k H_k^T A\|_F$ and $\frac{1}{k}\sum_{i\leq k}\frac{|\hat{\sigma}_i-\sigma_i|}{\sigma_i}$ with $k = 30$ for the three methods for the aft01 data set when different levels of coarsening were used, that is, the number of columns sampled/coarsened were increased. For "rand+coars," we do not have errors for $c = n/2$ since this was obtained with uniform sampling. For a smaller number of columns, that is, more levels in coarsening, the Frobenius-norm error for rand+coarsen approaches that of pure coarsening.

In all the above experiments, we have used maximum matching for coarsening. That is, in Algorithm 2, we match the current column with the column with which it has maximum inner product. The choice of ϵ, the parameter that decides the angle for matching, does not seem to affect the errors directly. If we choose a 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 intrinsic properties of the matrix under consideration. If we have more columns that are close to each other, that is, that make a small angle between each other, the coarsening technique will combine more columns, and 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 yield a coarse matrix that has nearly as many columns as the original matrix since it will not be able to match many columns.

6.2 | Column subset selection

In the following experiment, we compare the performance of the coarsening method against the leverage score sampling method for CSS (CSSP). We report the results for the term-by-document data sets (used in the first set of experiments) and for a few sparse matrices from the SuiteSparse matrix collection.

Table 2 presents the performances and computational times of the two methods. 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 that is computed by the multilevel coarsening method or using leverage score sampling of A with the top-k singular vectors as reported in the second column. Note that $P_C = H_k H_k^T$ from Theorem 1, with $k = c$. For larger matrices, we computed the projector as $P_C = H_{2k} H_{2k}^T$ since it was not practical to compute the exact projector for such large matrices (even after coarsening). The number of columns c in each test is reported in the third column, which is the same for both methods. Recall that, for the CSSP, the coarsening and sampling algorithms do not perform a post-scaling of the columns that are selected. The multilevel coarsening method performs quite well, yielding results that are comparable with those of leverage score sampling. Recall that the standard leverage score sampling requires the computation of the top-k singular vectors, which is inexpensive for small sparse matrices but can be substantially more expensive than coarsening for

TABLE 2 Column subset selection problem: coarsening versus leverage score sampling

Data set	Size	Rank k	c	Coarsening		levSamp	
				Levels	Error	Time	Error	Time
TIME	425	25	107	2	411.71	6.4 s	412.77	0.52 s
		50	107	2	371.35	6.4 s	372.66	1.08 s
		50	54	3	389.69	6.8 s	391.91	1.08 s
MED	1,033	50	65	4	384.91	5.3 s	376.23	1.1 s
		100	130	3	341.51	5.3 s	339.01	2.2 s
Kohonen	4,470	25	981	3	31.89	2.4 s	36.36	1.4 s
Erdos992	6,100	50	924	3	100.9	0.85 s	99.29	3.21 s
FA	10,617	50	2,051	3	26.33	2.53 s	28.37	3.72 s
chipcool0	20,082	100	1,405	4	6.05	7.02 s	6.14	35.69 s
scfxm1-2b	33,047	100	2,066	4	2195.4	4.67 s	2149.4	14.46 s
thermomechTC	102,158	100	6,385	4	2.03	12.7 s	2.01	166.1 s
ASIC-680 (n)	682,712	100	42,670	4	304.9	57.3 s	322.1	179.8 s

large sparse matrices, especially when k is large. Hence, we note that for the large matrices, coarsening is less expensive than leverage score sampling. Again, the coarsening method easily scales to even larger matrices. However, computing the errors (and leverage scores) become inviable.

6.3 | Graph sparsification

The next experiment illustrates how coarsening can be used for (spectral) graph sparsification. We again compare the performance of the coarsening approach to the leverage score sampling method[5] for graph spectral sparsification. Recall that spectral sparsification amounts 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 and the computational costs when the coarsening and leverage score sampling approaches were used to compute a sparse graph $\tilde{G}$ for different sparse random graphs and a few matrices related to graphs from the SuiteSparse collection. Given a graph G, we can form a vertex edge incidence matrix B, such that the graph Laplacian of G is $K = B^T B$. 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 the 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 the 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 $= \frac{1}{r}\sum_{i=1}^{r} \frac{|\sigma_i(\tilde{K}) - \sigma_i(K)|}{\sigma_i(K)}$, which tells us how close the sparser matrix $\tilde{K}$ is to K spectrally (related to the result in Lemma 1). We see that, in most cases, the coarsening approach yields a smaller error than with leverage score sampling. This is particularly true when the input graph is relatively dense and its pattern has certain structure.

TABLE 3 Graph sparsification: coarsening versus leverage score sampling. Error $= \frac{1}{r}\sum_{i=1}^{r} \frac{|\sigma_i(\tilde{K}) - \sigma_i(K)|}{\sigma_i(K)}$

Data set	m	r	$\frac{nnz(\tilde{K})}{nnz(K)}$	Coarsening			levSamp	
				Levels	Error	Time	Error	Time
sprand	1,954	489	0.27	2	0.545	0.84 s	0.578	0.23 s
	17,670	4,418	0.257	2	0.283	9.05 s	0.371	8.27 s
	49,478	12,370	0.251	2	0.102	41.9 s	0.243	33.6 s
	110,660	27,665	0.253	2	0.089	118.9 s	0.117	244.3 s
	110,660	13,888	0.129	3	0.144	130.7 s	0.308	242.1 s
	249,041	31,131	0.128	4	0.104	426.1 s	0.165	731.0 s
G1	19,176	4,794	0.265	2	0.151	4.7 s	0.221	17.1 s
comsol	49,585	6,337	0.281	2	0.157	5.4 s	0.154	21.3 s
heart2	171,254	21,407	0.126	3	0.265	121.5 s	0.241	126.3 s
heart1	347,853	21,335	0.126	3	0.166	409.7 s	0.208	515.8 s

6.4 | Applications

In this section, we illustrate the performance of the coarsening technique in the various applications introduced in Section 5.

6.4.1 | Latent semantic indexing

The first application we consider is LSI.[46,47] In LSI, we have a term–document matrix $A \in \mathbb{R}^{m\times n}$, with m representing documents and n representing terms that frequently occur in the documents, where A_{ij} is the frequency of the jth term in the ith 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 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.[47] In this experiment, we employ the coarsen SVD and leverage score sampling SVD algorithms to perform information retrieval techniques by LSI.[36]

Given a term-by-document data $A \in \mathbb{R}^{m\times n}$, we normalize the data using 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 data sets. When the term–document matrix A is large, the computation of the SVD factorization can be expensive for large ranks of k. The multilevel 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 (dimension) chosen. Now, the reduced representation of A is $\hat{A} = \Sigma_d^{-1}U_d^T A$. Each query q is transformed into a reduced representation $\hat{q} = \Sigma_d^{-1}U_d^T q$. The similarity of q and a_i is measured by the cosine distance between $\hat{q}$ and â 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 the work of Sakellaridi et al.[36] 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,[50] LLE,[51] and Laplacian eigenmaps.[55] Multilevel algorithms have been used in the clustering application, for example, the work of Oliveira and Seok[57] uses a multilevel algorithm, based on MinMaxCut, for

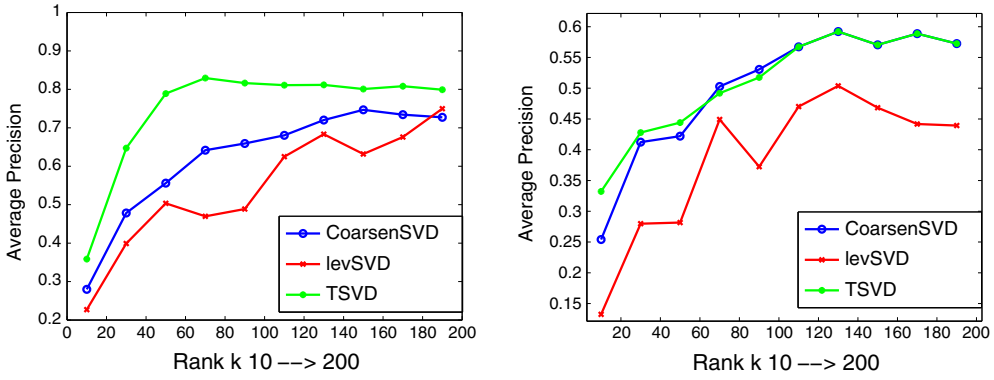

FIGURE 5 Latent semantic indexing results for the MEDLINE data set (left) and the TIME data set (right). SVD, singular value decomposition

document clustering, and the work of Fang et al.[38] applied the multilevel algorithms for spectral clustering and manifold learning.

Given n data points, most of the projective clustering methods start by constructing a graph with edges defined based on certain criteria, such as new distance metrics or manifolds, nearest neighbors, points on the 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[67], given by

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

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

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 the data belong are available. The statistical meaning of these metrics can be found in the work of Zhao and Karypis,[67] along with empirical and theoretical comparisons of these metrics with other standard metrics.

In Figure 6, we present the results for the three types of projective clustering methods, namely, isomap, LLE, and eigenmaps, when coarsening was used before dimensionality reduction. The data set used is the popular ORL face data set,[68] which contains 40 subjects and 10 grayscale images, each of size 112×92 with various facial expressions (the matrix size is $10,304 \times 400$). 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 are dense, the k-nearest neighbor 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 k-means 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 the work of Fang et al.,[69] where additional results and illustrations with other applications can be found. We refer to the works of Fang et al.[38,69] for additional details of this application and methods.

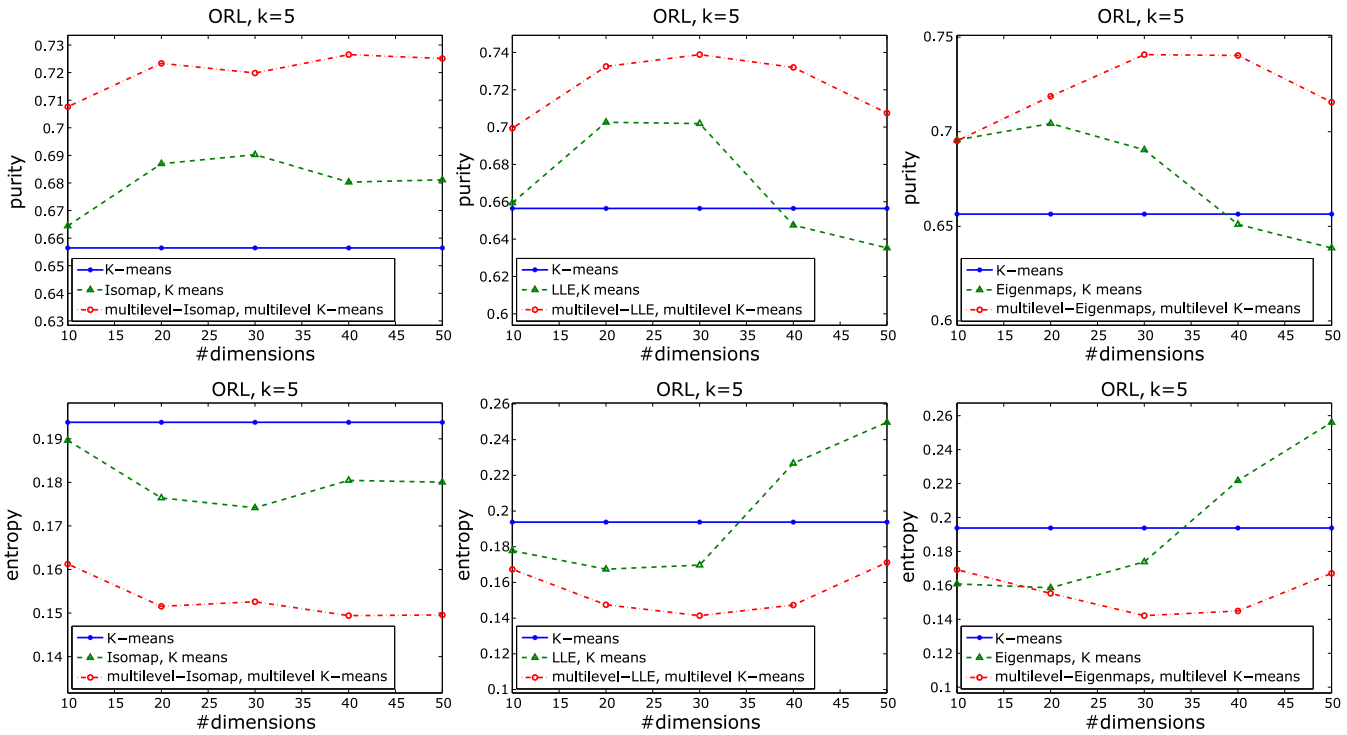

FIGURE 6 Purity and entropy values versus dimensions for the three types of clustering for the ORL data set. LLE, local linear embedding

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 are 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 (see the supplementary material in the work of Paschou et al.[30] for details on this encoding). Paschou et al.[30] used the 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 is 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 data sets as in the work of Paschou et al.,[30] namely, the Yale data set and the HapMap data set. The Yale data set*[70] contains a total of 248 SNPs for around 2,000 unrelated individuals from 38 populations from around the world. We consider four genomic regions (SORCS3, PAH, HOXB, and 17q25). The HapMap project†[71] (phase I) released a public database of 1,000,000 SNPs 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 $\{-1, 0, 1\}$, with the same meaning of the three possible values as discussed above. We obtained these encoded matrices, made available online by Paschou et al.,[30] from http://www.asifj.org/.

Table 4 lists the errors obtained from the three different methods, namely, coarsening, leverage score sampling, and greedy selection[30] for different populations. If $nnz(X)$ is the number of nonzero elements in a matrix X, the error that is reported is given by $nnz(\hat{A} - A)/nnz(A)$, where A is the input encoding matrix, and $\hat{A} = CC^{\dagger}A$ is the projection of A onto the sampled/coarsened C. The greedy algorithm considers each column of the matrix sequentially, projects the remaining columns onto the considered column, and chooses the column that gives the 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 it performs rather well in practice. Observe that the coarsening algorithm performs better than leverage score sampling, and the performance is comparable with that of the greedy algorithm in some cases. The coarsening approach is less expensive than leverage score sampling, which, in turn, is much less expensive than the greedy algorithm.

6.4.4 | Multilabel classification

The last application we consider is that of MLC. As seen in Section 5, the most common approach to handle a large number of labels in this problem is to perform label dimension reduction assuming a low-rank property of labels, that is, assuming that only a few labels are important. In this section, we propose to reduce the label dimension based on hypergraph coarsening. The work of Bi and Kwok[64] presented a method for MLC based on the CSSP using leverage score sampling. The idea is to replace sampling by hypergraph coarsening in this method.

*http://alfred.med.yale.edu/
†https://www.ncbi.nlm.nih.gov/variation/news/NCBI_retiring_HapMap/

TABLE 5 Multilabel classification using the column subset selection problem (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, $d = 101, n = 10,000, nt = 2001, p = 120$.	Coars	51	**10.487**	0.766	**8.707**	**0.713**
	CSSP	51	10.520	**0.782**	12.17	0.377
Bibtex, $d = 159, n = 6,000, nt = 1501, p = 1836$.	Coars	80	**1.440**	**0.705**	4.533	**0.383**
	CSSP	80	1.575	0.618	**4.293**	0.380
Delicious, $d = 983, n = 5,000, nt = 1000, p = 500$.	Coars	246	**50.943**	0.639	**74.852**	0.455
	CSSP	246	53.222	**0.655**	77.937	**0.468**
Eurlex, $d = 3993, n = 5,000, nt = 1000, p = 5000$.	Coars	500	2.554	**0.591**	**73.577**	0.3485
	CSSP	500	**2.246**	0.504	81.989	**0.370**

Table 5 lists the results obtained for MLC when coarsening and leverage score sampling (CSSP) were used for label reduction in the algorithm of Bi and Kwok[64] on different popular multilabel data sets. All data sets 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 $\{0, 1\}$, the label dimension is reduced by subsampling or coarsening the label matrix, leading to $c < d$ labels. The next step is to 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, that is, the number of classes for which the predicted label vector differs from the exact label vector. The second metric used is *Precison@k*, which is a common metric used in the MLC literature.[61] It measures the precision of predicting the first-k coordinates $|\text{supp}(\hat{y}_{1:k}) \cap \text{supp}(y)|/k$, where $\text{supp}(x) = \{i | x_i \neq 0\}$. 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 belong 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 also, we see that the coarsening method performs well, outperforming the more costly CSSP method in a few cases.

7 | CONCLUSION

This paper advocated the use of coarsening techniques for a number of matrix approximation problems, including the computation of partial SVD, the CSSP, and graph sparsification. We illustrated how coarsening methods and a combination of sampling and coarsening can be applied to solve these problems and presented a few (new) applications for the coarsening technique.

The experiments showed that the techniques based on coarsening perform quite well in practice, better than the randomized methods in many cases. Coarsening is also inexpensive compared to the sampling methods for large sparse matrices. While coarsening has traditionally been exploited in a completely different context to devise multilevel schemes for sparse systems as well as for graph partitioning, it appears that the same principles offer a tremendous potential for solving problems related to data.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under Grant NSF/CCF-1318597. Shashanka Ubaru was supported by the IBM Goldstine Postdoctoral Fellowship.

CONFLICT OF INTEREST STATEMENT

The authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements) or nonfinancial interest (such as personal or professional relationships, affiliations, knowledge, or beliefs) in the subject matter or materials discussed in this manuscript.

ORCID

Shashanka Ubaru https://orcid.org/0000-0001-6942-7158

REFERENCES

1. Drineas P, Kannan R, Mahoney MW. Fast Monte Carlo algorithms for matrices II: computing a low-rank approximation to a matrix. SIAM J Comput. 2006;36(1):158–183.
2. Halko N, Martinsson P-G, Tropp JA. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review. 2011;53(2):217–288.
3. Mahoney MW. Randomized algorithms for matrices and data. Found Trends Mach Learn. 2011;3(2):123–224.
4. Spielman DA, Srivastava N. Graph sparsification by effective resistances. SIAM J Comput. 2011;40(6):1913–1926.
5. Kapralov M, Lee YT, Musco CN, Musco CP, Sidford A. Single pass spectral sparsification in dynamic streams. Paper presented at: IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS 2014); 2014 Oct 18–21; Philadelphia, PA. Piscataway, NJ: IEEE; 2014.
6. Kumar S, Mohri M, Talwalkar A. Sampling methods for the Nyström method. J Mach Learn Res. 2012;13(Apr):981–1006.
7. Boutsidis C, Mahoney MW, Drineas P. An improved approximation algorithm for the column subset selection problem. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms; 2009 Jan 4–6; New York, NY. Philadelphia, PA: SIAM; 2009.
8. Saad Y. Numerical methods for large eigenvalue problems: Classics edition. Philadelphia, PA: SIAM; 2011.
9. Golub GH, Van Loan CF. Matrix computations. Vol. 3. Baltimore, MD: Johns Hopkins University Press; 2012.
10. Saad Y. Analysis of subspace iteration for eigenvalue problems with evolving matrices. SIAM J Matrix Anal Appl. 2016;37(1):103–122.
11. Drineas P, Frieze A, Kannan R, Vempala S, Vinay V. Clustering large graphs via the singular value decomposition. Machine Learning. 2004;56(1-3):9–33.
12. Woodruff DP. Sketching as a tool for numerical linear algebra. Found Trends Theor Comput Sci. 2014;10(1-2):1–157.
13. Ailon N, Chazelle B. Faster dimension reduction. Commun ACM. 2010;53(2):97–104.
14. Hendrickson B, Leland R. A multilevel algorithm for partitioning graphs. In: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing; 1995 Dec 3–6; San Diego, CA. Piscataway, NJ: IEEE; 1995.
15. Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput. 1998;20(1):359–392.
16. Catalyurek UV, Aykanat C. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. IEEE Trans Parallel Distributed Syst. 1999;10(7):673–693.
17. Karypis G, Kumar V. Multilevel *k*-way hypergraph partitioning. VLSI Design. 2000;11(3):285–300.
18. Drineas P, Mahoney MW, Muthukrishnan S. Relative-error CUR matrix decompositions. SIAM J Matrix Anal Appl. 2008;30(2):844–881.
19. Wang Y, Singh A. An empirical comparison of sampling techniques for matrix column subset selection. Paper presented at: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton); 2015 Sept 29–Oct 2; Monticello, IL. Piscataway, NJ: IEEE; 2015.
20. Mahoney MW, Drineas P. CUR matrix decompositions for improved data analysis. Proc Natl Acad Sci. 2009;106:697–702.
21. Krishnamurthy V, Faloutsos M, Chrobak M, Lao L, Cui J-H, Percus AG. Reducing large internet topologies for faster simulations. In: NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2005. p. 328–341. Lecture notes in computer science, No. 3462.
22. Spielman DA, Teng S-H. Spectral sparsification of graphs. SIAM J Comput. 2011;40(4):981–1025.
23. Zha H, Simon HD. On updating problems in latent semantic indexing. SIAM J Sci Comput. 1999;21(2):782–791.
24. Vecharynski E, Saad Y. Fast updating algorithms for latent semantic indexing. SIAM J Matrix Anal Appl. 2014;35(3):1105–1131.
25. Frieze A, Kannan R, Vempala S. Fast Monte-Carlo algorithms for finding low-rank approximations. J ACM. 2004;51(6):1025–1041. Preliminary version in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998.
26. Martinsson P-G, Rockhlin V, Tygert M. A randomized algorithm for the approximation of matrices. New Haven, CT: Department of Computer Science, Yale University; 2006. Technical report YALEU/DCS/TR-1361.
27. Liberty E, Woolfe F, Martinsson P-G, Rokhlin V, Tygert M. Randomized algorithms for the low-rank approximation of matrices. Proc Natl Acad Sci. 2007;104:20167–20172.
28. Ubaru S, Mazumdar A, Saad Y. Low rank approximation and decomposition of large matrices using error correcting codes. IEEE Trans Inf Theory. 2017;63(9):5544–5558.
29. Boutsidis C, Woodruff DP. Optimal CUR matrix decompositions. SIAM J Comput. 2017;46(2):543–589.
30. Paschou P, Mahoney MW, Javed A, et al. Intra-and interpopulation genotype reconstruction from tagging SNPs. Genome Research. 2007;17(1):96–107.
31. Avron H, Boutsidis C. Faster subset selection for matrices and applications. SIAM J Matrix Anal Appl. 2013;34(4):1464–1499.
32. Leskovec J, Faloutsos C. Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06); 2006 Aug 20–23; Philadelphia, PA. New York, NY: ACM; 2006.
33. Satuluri V, Parthasarathy S, Ruan Y. Local graph sparsification for scalable clustering. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SGMOD'11); 2011 Jun 12–16; Athens, Greece. New York, NY: ACM; 2011.

34. Ubaru S, Saad Y. Fast methods for estimating the numerical rank of large matrices. In: Proceedings of the 33rd International Conference on Machine Learning; 2016 Jun 19–24; New York, NY. JMLR; 2016 Jun 19–24.
35. Ubaru S, Saad Y, Seghouane A-K. Fast estimation of approximate matrix ranks using spectral densities. Neural Computation. 2017;29(5):1317–1351.
36. Sakellaridi S, Fang HR, Saad Y. Graph-based multilevel dimensionality reduction with applications to eigenfaces and latent semantic indexing. Paper presented at: 2008 Seventh International Conference on Machine Learning and Applications; 2008 Dec 11–13; San Diego, CA. Piscataway, NJ: IEEE; 2008. p. 194–200.
37. Zhou D, Huang J, Schölkopf B. Learning with hypergraphs: clustering, classification, and embedding. In: Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06); 2006; Canada. Cambridge, MA: MIT Press; 2006.
38. Fang H, Sakellaridi S, Saad Y. Multilevel manifold learning with application to spectral clustering. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management; 2010 Oct 26–30; Toronto, ON. New York, NY: ACM; 2010.
39. Fang H, Saad Y. Hypergraph-based multilevel matrix approximation for text information retrieval. In: Proceedings of the ACM International Conference on Information and Knowledge Management; 2010; Toronto, ON. New York, NY: ACM; 2010.
40. Chen J, Fang HR, Saad Y. Fast approximate *k*NN graph construction for high dimensional data via recursive Lanczos bisection. J Mach Learn Res. 2009;10:1989–2012.
41. Devine K, Boman EG, Heaphy RT, Bisseling RH, Catalyurek UV. Parallel hypergraph partitioning for scientific computing. In: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium; 2006 Apr 25–29; Rhodes Island, Greece. Piscataway, NJ: IEEE; 2006.
42. Saad Y. Finding exact and approximate block structures for ILU preconditioning. SIAM J Sci Comput. 2003;24(4):1107–1123.
43. Chen J, Saad Y. Dense subgraph extraction with application to community detection. IEEE Trans Knowl Data Eng. 2012;24(7):1216–1230.
44. Gu M. Subspace iteration randomization and singular value problems. SIAM J Sci Comput. 2015;37(3):A1139–A1173.
45. Liberty E. Simple and deterministic matrix sketching. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2013 Aug 11–14; Chicago, IL. New York, NY: ACM; 2013.
46. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. J Am Soc Inf Sci. 1990;41(6):391–407.
47. Landauer TK, Foltz PW, Laham D. An introduction to latent semantic analysis. Discourse Processes. 1998;25(2-3):259–284.
48. Berry MW, Dumais ST, O'Brien GW. Using linear algebra for intelligent information retrieval. SIAM Review. 1995;37(4):573–595.
49. Kokiopoulou E, Saad Y. 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; 2004 Jul 25–29; Sheffield, UK. New York, NY: ACM; 2004.
50. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290(5500):2319–2323.
51. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290(5500):2323–2326.
52. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. Paper presented at: Advances in Neural Information Processing Systems; 2001 Dec 3–8; Vancouver, Canada. p. 849–856.
53. Parsons L, Haque E, Liu H. Subspace clustering for high dimensional data: a review. ACM SIGKDD Explor Newsl. 2004;6(1):90–105.
54. Elhamifar E, Vidal R. Sparse subspace clustering. Paper presented at: 2009 IEEE Conference on Computer Vision and Pattern Recognition; 2009 Jun 20–25; Miami, FL. Piscataway, NJ: IEEE; 2009.
55. Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation. 2003;15(6):1373–1396.
56. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B. An introduction to kernel-based learning algorithms. IEEE Trans Neural Netw. 2001;12(2):181–201.
57. Oliveira S, Seok S-C. A multi-level approach for document clustering. Computational science - ICCS 2005 5th International Conference, Atlanta, GA, USA, May 22-25, 2005. Proceedings, Part I. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2005. p. 23–32.
58. Alter O, Brown PO, Botstein D. Singular value decomposition for genome-wide expression data processing and modeling. Proc Natl Acad Sci. 2000;97:10101–10106.
59. Raychaudhuri S, Stuart JM, Altman RB. Principal components analysis to summarize microarray experiments: application to sporulation time series. Paper presented at: Pacific Symposium on Biocomputing; 2000 Jan 4–9; Honolulu, Hawaii. Hackensack, NJ: World Scientific Publishing; 2000.
60. Tsoumakas G, Katakis I. Multi-label classification: An overview. In: Data warehousing and mining: Concepts, methodologies, tools, and applications. Hershey, PA: IGI Global; 2008.
61. Ubaru S, Mazumdar A. Multilabel classification with group testing and codes. In: Proceedings of the 34th International Conference on Machine Learning; 2017 Aug 6–11; Sydney, Australia. New York, NY: ACM; 2017.
62. Trohidis K. Multi-label classification of music into emotions. Paper presented at: 9th International Conference on Music Information Retrieval; 2008 Sep 14–18; Philadelphia, PA.
63. Tai F, Lin H-T. Multilabel classification with principal label space transformation. Neural Computation. 2012;24(9):2508–2542.
64. Bi W, Kwok JT. Efficient multi-label classification with many labels. In: Proceedings of the 30th International Conference on Machine Learning; 2013 Jun 16–21; Atlanta, GA. JMLR; 2013.
65. Savas B, Dhillon IS. Clustered low rank approximation of graphs in information science applications. In: Proceedings of the 2011 SIAM International Conference on Data Mining; 2011 Apr 28–30; Mesa, AZ. Philadelphia, PA: SIAM; 2011.

66. Davis TA, Hu Y. The University of Florida sparse matrix collection. ACM Trans Math Softw. 2011;38(1). Article No. 1.
67. Zhao Y, Karypis G. Empirical and theoretical comparisons of selected criterion functions for document clustering. Machine Learning. 2004;55(3):311–331.
68. Samaria FS, Harter AC. Parameterisation of a stochastic model for human face identification. In: Proceedings of 1994 IEEE Workshop on Applications of Computer Vision; 1994 Dec 5–7; Sarasota, FL. Piscataway, NJ: IEEE; 1994.
69. Fang HR, Sakellaridi S, Saad Y. Multilevel nonlinear dimensionality reduction for manifold learning. Minneapolis, MN: Minnesota Supercomputer Institute, University of Minnesota; 2009. Technical report.
70. Osier MV, Cheung K-H, Kidd JR, Pakstis AJ, Miller PL, Kidd KK. Alfred: an allele frequency database for diverse populations and DNA polymorphisms—an update. Nucleic Acids Res. 2001;29(1):317–319.
71. Gibbs RA, Belmont JW, Hardenbol P, et al. The international HapMap project. Nature. 2003;426(6968):789–796.

How to cite this article: Ubaru S, Saad Y. Sampling and multilevel coarsening algorithms for fast matrix approximations. *Numer Linear Algebra Appl*. 2019;e2234. https://doi.org/10.1002/nla.2234