Deterministic CUR for Improved Large-Scale Data Analysis: An Empirical Study

Christian Thurau, Kristian Kersting, and Christian Bauckhage
Fraunhofer IAIS, Germany
{firstname.lastname}@iais.fraunhofer.de

Abstract

Low-rank approximations which are computed from selected rows and columns of a given data matrix have attracted considerable attention lately. They have been proposed as an alternative to the SVD because they naturally lead to interpretable decompositions which was shown to be successful in application such as fraud detection, fMRI segmentation, and collaborative filtering. The CUR decomposition of large matrices, for example, samples rows and columns according to a probability distribution that depends on the Euclidean norm of rows or columns or on other measures of statistical leverage. At the same time, there are various deterministic approaches that do not resort to sampling and were found to often yield factorization of superior quality with respect to reconstruction accuracy. However, these are hardly applicable to large matrices as they typically suffer from high computational costs. Consequently, many practitioners in the field of data mining have abandon deterministic approaches in favor of randomized ones when dealing with today's large-scale data sets. In this paper, we empirically disprove this prejudice. We do so by introducing a novel, linear-time, deterministic CUR approach that adopts the recently introduced Simplex Volume Maximization approach for column selection. The latter has already been proven to be successful for NMF-like decompositions of matrices of billions of entries. Our exhaustive empirical study on more than 30 synthetic and real-world data sets demonstrates that it is also beneficial for CUR-like decompositions. Compared to other deterministic CUR-like methods, it provides comparable reconstruction quality but operates much faster so that it easily scales to matrices of billions of elements. Compared to sampling-based methods, it provides competitive reconstruction quality while staying in the same run-time complexity class.

1 Introduction

Low-rank approximations of data matrices are an important tool in data mining. They allow for embedding high dimensional data in lower dimensional spaces and can therefore mitigate effects due to noise, uncover latent relations, or facilitate further processing. These properties have been proven successful in applications areas such as bio-informatics, computer vision, text processing, recommender systems, social network analysis, among others. A well known low-rank approximation approach consists in truncating the Singular Value Decomposition (SVD), which expresses the data in terms of linear combinations of the top singular vectors. While these basis vectors are optimal in a statistical sense, the SVD has been criticized for it is less faithful to the nature of the data at hand. For instance, if the data are sparse the compressed representations are usually dense which leads to inefficient representations. Or, if the data consists entirely of non-negative vectors, there is no guarantee for an SVD-based low-dimensional embedding to maintain non-negativity. Nevertheless, data analysts often tend to assign a "physical" meaning to the resulting singular components. Such reification, however, must be based on an intimate knowledge of the application domain and cannot be justified from mathematics.

A way of circumventing these problems consists in computing low-rank approximations from selected rows and columns of a data matrix [15]. Corresponding approaches yield interpretable results because they embed the data in lower dimensional spaces whose basis vectors correspond to actual data points. They are guaranteed to preserve properties such as sparseness or non-negativity and enjoy increasing popularity in the data mining community [3, 11, 12, 17, 21, 22, 26, 28] where they have been applied to fraud detection, fMRI segmentation, collaborative filtering, and co-clustering.

The idea of selecting suitable rows and columns for low-rank matrix approximations has a venerable history in linear algebra [5]. Recent deterministic algorithms include a quasi-Gram-Schmidt method [1], the CGR decomposition [15], or a greedy, SVD-based method [7]. Unfortunately, current deterministic approaches quickly become infeasible if there are a very large number of data points. When dealing with today's large-scale data sets, many data mining practitioners therefore often abandon deterministic approaches and resort to randomized approaches. However, huge data such as collections of online books at Amazon$^{\text{TM}}$, image repositories at Flickr$^{\text{TM}}$or Google$^{\text{TM}}$, or personal health records [16, 19, 23, 24, 29] are becoming ever more common and thus pose a challenge to research on interpretable matrix factorization.

Informed sampling strategies mark a middle ground. Approaches that rely on probability distributions which depend on the statistical leverage, i.e. the top-k singular subspace of a matrix [22], or the Euclidean norm of rows and columns [9, 26] were introduced under the name CUR decomposition. As the CUR decomposition relies on informed row- and column-sampling strategies, it can scale to large data but only if statistical leverage is consider for k being rather small since the CUR algorithm involves computing the SVD of a matrix. In order to balance the tradeoff between computation time and approximation accuracy, the work in [4] recently proposed a hybrid approach where candidate columns are sampled from a data matrix and a deterministic algorithm is applied to finalize the selection of basis vectors for low-rank approximation. While this guarantees applicability to large matrices, the resulting factorizations often are of lower quality than those of recent deterministic methods [7].

1.1 Contribution Our main contribution in this paper is to demonstrate that deterministic CUR-like approaches can scale well, too, and therefore provide an alternative to sampling-based matrix factorization techniques. We introduce an algorithm for *large-scale deterministic CUR (LS-DCUR)* that is almost as efficient as norm-based sampling methods and as accurate as most recent deterministic techniques. To best of our knowledge, it is the first deterministic approach to row- or column selection that scales well to matrices of billions of entries.

Considerable speed up is due to incorporating a recent deterministic column selection approach called Simplex Volume Maximization [27, 28]. Similar to previous work [7, 15], Simplex Volume Maximization casts the optimization problem that governs row- and column selection as a volume maximization problem. In contrast to previous work, our approach does not require projections of the top-k singular vectors or other equally demanding intermediate computations. Instead, Simplex Volume Maximization applies techniques from the field of distance geometry [2] and relies on iterative distance computations only.

1.2 Properties of LS-DCUR Computing the LS-DCUR decomposition of a given data matrix requires efforts that are only linear in the number n of data; as the algorithm mainly consists of distance computations and linear passes over the data, the effort for computing k basis vectors is $O(kn)$ where $k \ll n$. As LS-DCUR is deterministic, it will always produce the same set of basis vectors for a fixed k and a given data matrix.

Other favorable properties of LS-DCUR become apparent from comparisons to related methods. For example, sampling based on the Euclidean-norm of column vectors is not applicable to data matrices whose columns are of unit norm as the corresponding probability distribution would be uniform. Methods based on the statistical leverage are applicable to unit-norm columns but do not scale well to matrices of many columns. LS-DCUR, however, applies to unit-norm columns and scales to very large matrices. Moreover, LS-DCUR is suitable for parallelization on distributed file systems such as the MapReduce framework [8]. As it only requires computing the distance to latest selected row or column, it is straightforward to implement the algorithm in a distributed manner such that it becomes possible to handle data matrices that exceed the storage capabilities of a single hard disk. While this is also true for methods relying on norm-based sampling, it becomes obviously more involved if these methods require computing the top-k singular vectors of a data matrix.

1.3 Empirical Evaluation In order to empirically validate the proposed approach to a deterministic CUR decomposition, we experiment with more than 30 synthetic and real-world matrices covering a wide range of mid- and large-scale as well as dense and sparse matrices. It shows that the proposed selection strategy achieves: **(a)** reconstruction accuracies similar to the most recent deterministic methods [7] when applied to small- or mid-size matrices, **(b)** significantly higher reconstruction accuracies than random selection strategies when applied to for large matrices, **(c)** run-time characteristics similar to the fastest possible random selection strategies. These benefits also become visible in an ongoing investigation of an approximation of co-clustering, i.e. the simultaneous clustering of columns and rows of a matrix. Initial results on the 20 Newsgroup data set, which consists of 100 words across $16,242$ documents in four categories, are promising. LS-DCUR achieved a document categorization accuracy of 0.58 ± 0.05 averaged over 10 reruns of the information-theoretic co-clustering method, while a randomized approach with 100 random restarts reached 0.45 ± 0.03.

Algorithm 1: Main steps of CGR/CUR

```
  Input: Matrix $\boldsymbol{X} \in \mathrm{R}^{m \times n}$, integer $c, r$
1 Select $c$ columns from $\boldsymbol{X}$ and construct
  $\boldsymbol{C} \in \mathbb{R}^{m \times c}$;
2 Select $r$ columns from $\boldsymbol{X}^T$ and construct
  $\boldsymbol{R} \in \mathbb{R}^{r \times n}$;
3 $\boldsymbol{U} = \boldsymbol{C}^+ \boldsymbol{V} \boldsymbol{R}^+$, where $\boldsymbol{C}^+, \boldsymbol{R}^+$ denote the
  Moore-Penrose generalized inverse of the
  matrices $\boldsymbol{C}$ and $\boldsymbol{R}$;
4 Return $\boldsymbol{C} \in \mathbb{R}^{m \times c}, \boldsymbol{U} \in \mathbb{R}^{c \times r}, \boldsymbol{R} \in \mathbb{R}^{r \times n}$
```

Algorithm 2: CUR with importance sampling based on the Euclidean-norm of columns (rows sampling works accordingly).

```
  Input: Matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, integer $k$
1 for $x = 1 \dots n$ do
2     $P(x) = \sum_i \boldsymbol{X}_{i,x}^2 / \sum_{i,j} \boldsymbol{X}_{i,j}^2$;
3 for $i = 1 \dots k$ do
4     Randomly select $\boldsymbol{W}_{*,i} \in \boldsymbol{X}$ based on $P(x)$
5 Return $\boldsymbol{W} \in \mathbb{R}^{m \times k}$
```

1.4 Overview We proceed as follows: in the next section we review the CUR decomposition and deterministic selection methods. Then, Sec. 3 introduces our novel approach to large-scale deterministic CUR and Sec. 4 discusses details as to its properties. In Sec. 5, we present our experimental evaluation. Before concluding, we discuss application scenarios and explain when and why the suggest algorithm is particularly useful and actually outperforms existing methods.

2 CUR Decomposition

The CUR decomposition attempts to select a number of rows and columns from a real valued $m \times n$ matrix $\boldsymbol{X}$ such that it can be accurately approximated as a weighted product of the rows and columns. That is, if $\boldsymbol{C}^{m \times c}$ is a matrix of c columns selected from $\boldsymbol{X}$, $\boldsymbol{R}^{r \times n}$ is a matrix of r rows selected from $\boldsymbol{X}$, and $\boldsymbol{U}^{c \times r}$ is $c \times r$ matrix of weights, then, $\boldsymbol{X}' = \boldsymbol{CUR}$ is a CUR approximation to $\boldsymbol{X}$. In order to determine suitable factor matrices $\boldsymbol{C}$ and $\boldsymbol{R}$, one commonly seeks to minimize the residual $\left\|\boldsymbol{X} - \boldsymbol{CUR}\right\|$. The weighting matrix $\boldsymbol{U}$ can be computed as $\boldsymbol{C}^+ \boldsymbol{X} \boldsymbol{R}^+$ where $\boldsymbol{C}^+$ and $\boldsymbol{R}^+$ denote the pseudo-inverse of $\boldsymbol{C}$ and $\boldsymbol{R}$, respectively. Algorithm 1 summarizes the main steps of traditional CUR; for this paper, the two selection steps are of primary interest.

Minimizing the residual $\left\|\boldsymbol{X} - \boldsymbol{CUR}\right\|$ is, alas, a non-trivial problem. It is a special case of the column subset selection problem (CSSP) where one seeks to select a set of columns from a matrix which capture as much information as possible and accurately approximate the original matrix [4]. Most deterministic selection methods applicable to CUR decompositions originate from considering the CSSP. A simple solution would be to consider all possible combinations of columns and choose the combination that leads to the most accurate approximation. However, this would lead to combinatorial explosion and it is obvious that brute force approaches like this do not apply to large data sets.

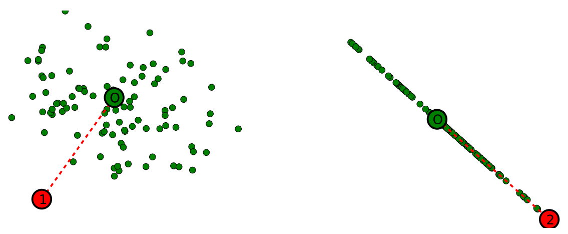
2.1 Importance Sampling (CUR-SL/CUR-L2)

One way of avoiding combinatorial explosion is to consider randomized selections of rows and columns. A corresponding, popular selection method for CUR decomposition applies importance sampling. That is, one computes an importance score for each column (or row) and samples columns (or rows) according to their distribution of scores. Two scores are commonly used, one is based on the Euclidean norm of row or column vectors (CUR-L2), the other is based on the statistical leverage (CUR-SL).

Norm-based scores, for instance for column $\boldsymbol{X}_t$ of matrix $\boldsymbol{X}$ are given by $P(t) = \sum_i X_{i,t}^2 / \sum_{i,j} X_{i,j}^2$; the corresponding algorithm is summarized in Alg. 2. Various extensions to this method have been proposed. For example, the approach in [26] further reduces computation times by avoiding to repeatedly sample the same row (or column).

Scores based on statistical leverage, say of columns, require to compute the top-k left singular vectors $\boldsymbol{V}^{k \times n}$ of $\boldsymbol{X}$. Then, the statistical leverage score π_t for a particular column $\boldsymbol{X}_t$ is computed by summing over the rows of the singular vectors, i.e. $\pi_t = \frac{1}{k} \sum_{i=1}^{k} V_{i,t}^2$. The π_t form a probability distribution over the set of columns where columns that capture more dominant parts of the spectrum of $\boldsymbol{X}$ are assigned a higher probabilities [22].

Randomized approaches are easy to implement and have been shown to work well in many practical applications. However, they also suffer from various shortcomings. On the one hand, norm-based importance scores cannot be applied to matrices of normalized rows or columns as score distributions would become uniform. In addition, they do not necessarily provide sufficient coverage of the data space because a corresponding criterion is not build into the selection process. On the other hand, methods based on the statistical leverage do not scale well to very large matrices because computing their top-k singular vectors scales quadratically with k and linearly with the dimensions of rows and columns and special care has to be taken e.g. when scaling them



(a) 1st selection (b) 2nd selection

Figure 1: Often, deterministic methods iteratively select data points or matrix columns in a Gram-Schmidt like manner. (a) the first selected sample has the highest norm; (b) the second selected sample is maximal among the data projected onto the subspace perpendicular to the already selected sample.

to massive graphs [18]. Most importantly, the resulting approximation accuracy often proves to be inferior to deterministic methods [7].

2.2 Deterministic Selection (DCUR) Recent years have seen an increased interest in deterministic selection strategies for the column subset selection problem (CSSP) and many different methods have been proposed [4]. They were shown to work well for smaller data sets where they often outperform randomized selection strategies [7]. Due to their computational complexity, however, they do not apply to large matrices.

At this point, it is worthwhile to point out certain geometric properties of the column subset selection problem. It was shown that an appropriate way of finding a *good* subset for the CSSP consists in selecting a number of columns such that their volume is maximal. This criterion is also referred to as maximum volume (Max-Vol) criterion [6, 13, 14]. Consequently, a *good* subset is one that maximizes the volume of the parallelepiped, i.e. the value of the determinant, spanned by the selected columns. In other words, given an $m \times n$ matrix $\boldsymbol{X}$, we may select k of its columns such that the volume $Vol(\boldsymbol{C}) = |\det \boldsymbol{C}|$ is maximized, where the $m \times k$ matrix $\boldsymbol{C}^{m \times k}$ contains the selected columns. Thus, instead of minimizing a norm to accomplish suitable CUR decompositions, we may just as well seek a selection of k columns or rows such that the corresponding volumes $|\det \boldsymbol{C}|$ and $|\det \boldsymbol{R}|$ are maximal.

The Max-Vol problem is provably NP-hard [6]. Therefore, approaches other than greedy (or deterministic) algorithms quickly become infeasible. Yet, as we shall see, considering the Max-Vol problem allows us to employ a highly efficient deterministic selection method that was recently derived from principles of distance geometry. In the following, we review this algorithm, show how to adapt it to the problem of CUR decompositions, and relate it to a recent deterministic method introduced in [7] to which we will refer to as *Greedy* in the following.

3 Large-scale Deterministic CUR (LS-DCUR)

The basic idea of many greedy deterministic selection strategies such as the one in [7] is to iteratively select rows or columns such that they minimizes the residual after projection onto subspaces orthogonal to already selected prototypes (see Fig. 1). In the following, we introduce an efficient approximation to deterministic, projective selection methods. The rather expensive projection step can be efficiently dealt with if the Max-Vol criterion is considered in the context of *distance geometry* [2]. To this end, we transfer the Max-Vol criterion from maximizing the volume of a parallelepiped to maximizing the volume of a simplex. This allows us to adapt a recent efficient algorithm for simplex volume maximization [28] to matrix factorization based on selections of rows and columns.

For a subset $\boldsymbol{C}$ of k columns from $\boldsymbol{X}$, let $\Delta(\boldsymbol{C})$ denote the k-dimensional simplex formed by the columns in $\boldsymbol{C}$, the volume of the simplex is given by $\mathrm{Vol}(\Delta(\boldsymbol{C})) = \frac{1}{n!} \det(\boldsymbol{c}_1 - \boldsymbol{c}_0, \ldots, \boldsymbol{c}_k - \boldsymbol{c}_0)$ where $\boldsymbol{c}_i$ denotes the i-th selected column. Let $\Delta(\boldsymbol{C}, \boldsymbol{0})$ denote the $k+1$-dimensional simplex including the origin, then $\mathrm{Vol}(\Delta(\boldsymbol{C}, \boldsymbol{0})) = \frac{1}{n!} \det(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$. Apparently, $\mathrm{Vol}(\Delta(\boldsymbol{C}, \boldsymbol{0}))$ is proportional to the volume $|\det \boldsymbol{C}|$ of the corresponding k-parallelepiped. Thus, transferring the maximum volume problem to simplices introduces an alternative objective function that is well-grounded in distance geometry. Seen from this point of view, the volume of the k-simplex is

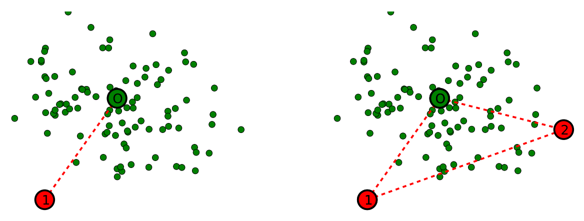
$$\mathrm{Vol}(\Delta(\boldsymbol{C}, \boldsymbol{0}))_k^2 = \theta \det \boldsymbol{A} \quad \text{where } \theta = \frac{-1^{k+1}}{2^k (k!)^2}. \tag{3.1}$$

Here, $\det \boldsymbol{A}$ is the the *Cayley-Menger* determinant [2] given by

$$\det \boldsymbol{A} = \begin{vmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & ||c_1||^2 & ||c_2||^2 & \cdots & ||c_{k+1}||^2 \\ 1 & ||c_1||^2 & 0 & \mathrm{d}_{2,1}^2 & \cdots & \mathrm{d}_{2,k+1}^2 \\ 1 & ||c_2||^2 & \mathrm{d}_{2,1}^2 & 0 & \cdots & \mathrm{d}_{3,k+1}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & ||c_{k+1}||^2 & \mathrm{d}_{2,k+1}^2 & \mathrm{d}_{3,k+1}^2 & \cdots & 0 \end{vmatrix}$$

where $\mathrm{d}_{i,j}^2$ is the squared distance between vertices or columns i and j of $\boldsymbol{C}$ and $||c_i||^2$ denotes the squared Euclidean norm of the i-th selected column.

Next, we will see that, since this distance geometric formulation is entirely based on vector norms and



(a) 1st selection (b) 2nd selection

Figure 2: LS-DCUR iteratively maximizes the volume of the simplex including the origin. It avoids the expensive projection step of other deterministic methods (Left). The first selected rows/columns has the maximal distance from the origin. (Right) The second selected row/column maximizes the volume given the already selected rows/columns.

edge lengths, it allows for an efficient, greedy selection algorithm.

Finding a globally optimal subset $\boldsymbol{C}$ that maximizes Eq. (3.1) requires the computation of all pairwise distances among the columns in $\boldsymbol{X}$. For large data sets, this is ill-advised as it requires efforts of $O(n^2)$ where n is the number of data samples. Eq. (3.1) allows us to devise an iterative, approximative procedure with efforts of $O(kn)$ which determines a set of k basis vectors that maximize the volume of the simplex. Given a simplex S consisting of $k-1$ vertices, we seek a new vertex $\boldsymbol{x}_\pi \in \boldsymbol{X}$ such that $\boldsymbol{x}_\pi = \arg\max_k \mathrm{Vol}(S \cup \boldsymbol{x}_k)^2$. If a number of vertices has already been acquired in a sequential manner, the following theorem can be proven assuming that the vertices are equidistant:

Theorem 1. ([28]) *Let S be an $(n-1)$-simplex. Suppose its vertices $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ are equidistant and that this distance is a. Also, suppose the distances between vertex $\boldsymbol{w}_{n+1}$ and the other vertices to be $\{d_{i,n+1}, \ldots, d_{n,n+1}\}$, then the volume of S is determined by*

$$Vol(S)_n^2 = \frac{a^{2n}}{2^n(n!)^2}\left(\frac{2}{a^4}\sum_{\substack{i=1\\j=i+1}}^{n} d_{i,n+1}^2 d_{j,n+1}^2 + \frac{2}{a^2}\sum_{i=1}^{n} d_{i,n+1}^2 - \frac{n-1}{a^4}\sum_{i=1}^{n} d_{i,n+1}^4 - n + 1\right).$$

From this, an iterative update procedure can be derived similar to the one in [28]. Note that, in contrast to [28], we are considering CUR decompositions, so that we have to maximize the matrix volume $\mathrm{Vol}(\Delta(\boldsymbol{C}, \boldsymbol{0}))_k^2$ instead of the simplex volume $\mathrm{Vol}(\Delta(\boldsymbol{C}))_k^2$. Furthermore, without loss of generality, we set $\mathrm{d}_{i,j}^2 \sim \log(\mathrm{d}_{i,j}^2)$, since we are only interested in finding the maximum rather than the exact value. This modification only impacts

Algorithm 3: Large-scale Deterministic CUR

```
   Input: Matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, integer $k$
1  for $j = 1 \ldots n$ do
2      $n_j \leftarrow \log(||\boldsymbol{X}_{*,j}||)$ ;
3      $\Phi_{0,j} \leftarrow n_j$ ;
4      $\Lambda_{0,j} \leftarrow n_j^2$ ;
5      $\Psi_{0,j} \leftarrow 0$ ;
6  select = $\arg\max_j(n_j)$;
7  a = $n_{\text{select}}$ ;
8  $\boldsymbol{w}_1 = \boldsymbol{X}_{*,\text{select}}$;
9  for $i = 2 \ldots k$ do
10     for $j = 1 \ldots n$ do
11         $p_j \leftarrow \log(d(\boldsymbol{w}_{i-1}, \boldsymbol{X}_{*,j}))$;
12         $\Phi_{i,j} \leftarrow \Phi_{i-1,j} + p_j$ ;
13         $\Lambda_{i,j} \leftarrow \Lambda_{i-1,j} + p_j^2$ ;
14         $\Psi_{i,j} \leftarrow \Psi_{i-1,j} + p_j * \Phi_{i-1}$ ;
15     select =
       $\arg\max_j\left(\mathrm{a} * \Phi_{i,j} + \Psi_{i,j} - \frac{(i-1)}{2}\Lambda_{i,j}\right)$;
16     $\boldsymbol{w}_i = \boldsymbol{X}_{*,\text{select}}$;
17     $\boldsymbol{W}_{*,i} = \boldsymbol{X}_{*,\text{select}}$;
18 Return $\boldsymbol{W} \in \mathbb{R}^{m \times k}$
```

changes the considered distances but does not alter the algorithm. We arrive at

(3.2)

$$\boldsymbol{v}_\pi = \arg\max_k \left(\log(a)\sum_{i=1}^{n}\log(\mathrm{d}_{i,k}) + \sum_{\substack{i=1\\j=i+1}}^{n}\log(\mathrm{d}_{i,k})\log(\mathrm{d}_{j,k}) - \frac{n-1}{2}\sum_{i=1}^{n}\log^2(\mathrm{d}_{i,k})\right).$$

Note that Theorem 1 assumes equidistant edge lengths a between the first k vertices. Generally, this is however not the case. The use of logarithmic scales considerably reduces the error introduced by this assumption as it effectively maps large distances to similar values and, in addition, provides numerically more stable implementations. Theorem 1, Eq. (3.2), and Eq (3.1) lead to Alg. 3. For the case of $n = 1$, i.e. for the first row or column to select, this basically yields the Euclidean norm $\boldsymbol{v}_\pi = \arg\max_k(\log(\mathrm{d}(\boldsymbol{0}, \boldsymbol{c}_k))) = \arg\max_k ||c_k||$ where $\mathrm{d}(\boldsymbol{0}, \boldsymbol{c}_k)$ denotes the distance of $\boldsymbol{c}_k$ to the origin. We note that the pairwise distances computed in earlier iterations can be reused in later steps. For retrieving k latent components, we need to compute the distance to all data samples exactly k times. The distances are computed with respect to the last selected basis vector. Large-scale Deterministic CUR (**LS-DCUR**) is more efficient than other deterministic methods as it supersedes the need for expensive projections of the data. Nevertheless, it aims for solutions that are similar to the Greedy algorithm as projections and orthogonality constraints are implicit parts of the distance geometric objective function. We also note that we introduced a small δ by assuming equidistant edge-length. As our experiments will show, this hardly impact the performance of our algorithm; on the contrary, it proves to be competitive in terms of approximation accuracy but is typically orders of magnitudes faster. Fig. 2 provides an illustrative example of how LS-DCUR determines a vertex $m+1$ that maximizes the simplex volume given that m vertices have already been found.

4 Properties of LS-DCUR

In the following, we examine the proposed deterministic CUR approach more closely and compare its behavior to other deterministic or importance sampling methods.

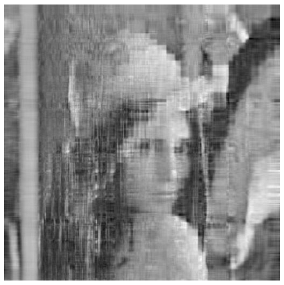
4.1 Computational Complexity Similar to other deterministic methods, the LS-DCUR algorithm is of linear time complexity. However, LS-DCUR avoids expensive projections or SVD computations and is therefore even faster than its competitors. For each selected column $\boldsymbol{W}_{*,i}$ we have to compute the distance $d(\boldsymbol{W}_{*,i-1}, \boldsymbol{v}_j), \boldsymbol{v}_j \in \boldsymbol{X}$ only once. If we assume k basis vectors, this translates to $\mathcal{O}(kn)$. For $k \ll n$, this results in linear time complexity $\mathcal{O}(n)$. The three simple additive operations in lines 12-14 of Alg. 3 do not majorize computation times for large n and conventional distance metrics. Moreover, computing distances of sparse vectors, for instance when decomposing sparse graphs, is very fast since this only scales with the number of nonzero entries rather than with the dimension of the data. Considering that CUR is often applied to sparse matrices, because, in contrast to the SVD, it preserves sparsity, this is an important aspect. For example, distance computations for the large graphs considered in our experiments did require only fractions of a second.

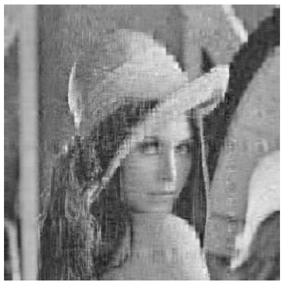
Traditional importance sampling procedures based on the Euclidean norm, too, can be efficiently implemented and only require $m+n$ computations of norms for a matrix $\boldsymbol{X}^{m \times n}$. However, it should be noted that in this case the arguably most expensive step of the CUR decomposition as outlined in Alg. 1 is the computation of the matrix $\boldsymbol{U} = \boldsymbol{C}^{+}\boldsymbol{X}\boldsymbol{R}^{+}$, since it involves two pseudo-inverses $\boldsymbol{C}^{+}$ and $\boldsymbol{R}^{+}$. If the SVD is applied in this step, one ends up with a complexity of $O(\min(nk^2, k^2n))$ for a matrix $\boldsymbol{C}^{m \times k}, \boldsymbol{R}^{k \times n}$. This cost can easily outweigh the costs for selecting rows and columns and also explains why LS-DCUR is in practice only slightly slower than random sampling. Compared to methods based on statistical leverage, it can be easily seen that these scale quadratically with k because computing statistical leverage score requires knowledge as to the top-k singular vectors. Especially for large data, where a possibly large number k of rows or columns are to be selected, the computational costs increase dramatically.

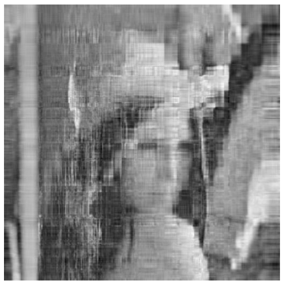
4.2 Selections Importance sampling yields rows and columns that exhibit a large vector norm or have a high statistical leverage. LS-DCUR yields rows and columns that have (1) a large norm, (2) large pairwise distances, and that (3) are (almost) orthogonal. Similar to other deterministic strategies, the selected rows and columns are ordered. The implicit selection criteria (large norm, large distance, orthogonality) have two interesting implications: first, the risk of multiple selections of the same sample is negligible. Thereby, similar to CMD [26], we reduce the size of the matrices $\boldsymbol{C}, \boldsymbol{R}$ and decrease computation times for $\boldsymbol{U}$ (note that most available implementations of CUR already incorporate this step as did we in our experiments). Second, selection methods that apply importance sampling based on Euclidean are not applicable to matrices that consist of unit-norm vectors such as, for example, normalized Histograms. For LS-DCUR, this is only an issue for the first selected row or column, as later selections do not depend on the norm but rather on pairwise distances.

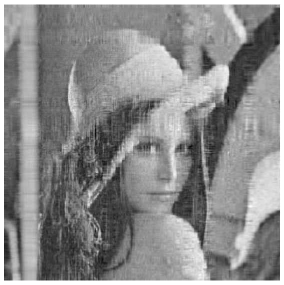
4.3 Interpretability and Data-space Coverage As selections made by random and deterministic approaches always correspond to actual data samples, they are easier to interpret as, for example, SVD decompositions [22]. Additionally, for the deterministic case, the resulting selections are often polar opposites. Unlike in importance sampling, where selected vectors may be rather similar, this further increases interpretability as it accommodates human cognition. By focusing on extreme opposites, we enlarge the margin of what we know and in turn our chance to separate things. Similar to deterministic methods that aim for approximations of the top-k singular vectors, we also obtain a potentially better coverage of the complete data space.

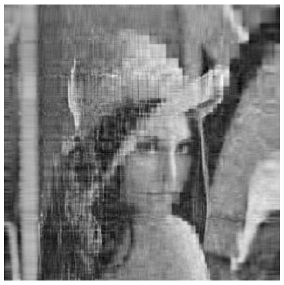
4.4 Parallelization The strictly sequential nature of LS-DCUR makes it well suited for very large data since it admits for parallelization. For example, using the MapReduce framework [8] on a distributed file system, the algorithm scales almost linearly with the number of nodes. Adapting the algorithm to a distributed file system is trivial as most computation time is spent on distance computations which are easy to parallelize. Note that in order to provide a fair comparison of methods, we did not use our own MapReduce implementation or any other form of parallelized processing in the experimental section.


(a) CUR-L2 (k=35)


(b) CUR-L2 (k=70)


(c) CUR-SL (k=35)


(d) CUR-SL (k=70)


(e) LS-DCUR (k=35)

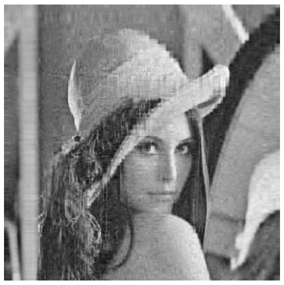

(f) LS-DCUR (k=70)

Figure 3: Qualitative comparison for the task of image reconstruction. An image is viewed as matrix and and subjected to CUR-SL, CUR-L2, and LS-DCUR with $k = 35, 70$ rows and columns (best viewed on screen).

(a) Lena

	CUR-L2	CUR-SL	Greedy	LS-DCUR
CUR-L2	0.0E–00○	1.3E–03○	−9.0E–03●	−5.3E–03●
CUR-SL	−1.3E–03○	0.0E–00○	−1.0E–02●	−6.6E–03●
Greedy	9.0E–03●	1.0E–02●	0.0E–00○	3.7E–03○
LS-DCUR	5.3E–03●	6.6E–03●	−3.7E–03○	0.0E–00○

(b) Standard normal

	CUR-L2	CUR-SL	Greedy	LS-DCUR
CUR-L2	0.0E–00○	−4.2E–05○	−5.5E–03●	−1.5E–03●
CUR-SL	4.2E–05○	0.0E–00○	−5.5E–03●	−1.5E–03●
Greedy	5.5E–03●	5.5E–03●	0.0E–00○	4.0E–03●
LS-DCUR	1.5E–03●	1.5E–03●	−4.0E–03●	0.0E–00○

(c) Scaled

	CUR-L2	CUR-SL	Greedy	LS-DCUR
CUR-L2	0.0E–00○	−5.9E–03●	−4.4E–02●	−1.6E–03○
CUR-SL	5.9E–03●	0.0E–00○	−3.8E–02●	4.3E–03○
Greedy	4.4E–02●	3.8E–02●	0.0E–00○	4.2E–02●
LS-DCUR	1.6E–03○	−4.3E–03○	−4.2E–02●	0.0E–00○

(d) Smooth

	CUR-L2	CUR-SL	Greedy	LS-DCUR
CUR-L2	0.0E–00○	3.4E–04○	−7.1E–06○	−4.8E–06○
CUR-SL	−3.4E–04○	0.0E–00○	−3.5E–04○	−3.4E–04○
Greedy	7.1E–06○	3.5E–04○	0.0E–00○	2.4E–06○
LS-DCUR	4.8E–06○	3.4E–04○	−2.4E–06○	0.0E–00○

Table 1: Average differences of the relative accuracy obtained for different methods/datasets. Positive values denote a better reconstruction of the query. Bullets ● denote significant differences (paired t-test at $p = 0.05$), circles ○ denote insignificant ones.

5 Experiments

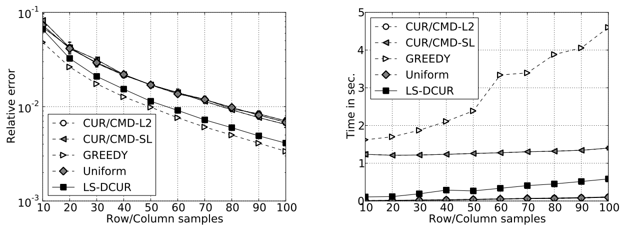
Our intention in this section is to investigate the following questions pertaining to LS-DCUR factorizations:

Q1 How does the presented large-scale deterministic CUR approach compare to the most recent deterministic methods for small/mid-size matrices?

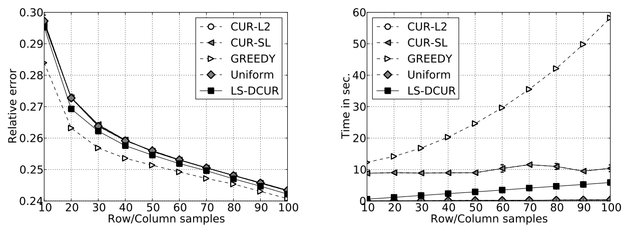
Q2 How does the presented selection scheme compare to importance sampling methods for large matrices?

Ideally, for small to mid-size matrices, LS-DCUR would show an accuracy comparable to recent deterministic methods and comparable run-time performance but higher accuracy than importance sampling methods.

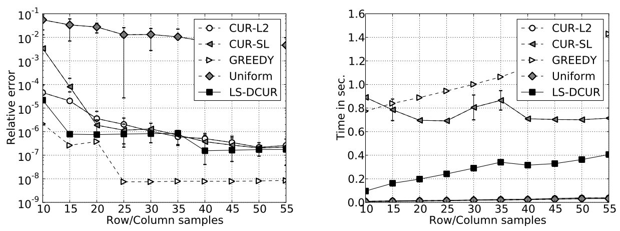
We compare LS-DCUR against *CUR-L2* (Euclidean-norm based selection), *CUR-SL* (statistical leverage based selection), and *Greedy* (a recent deterministic selection method that was shown to exceed various other methods [7]). To provide a fair comparison, we incorporate several extensions into the importance sampling based methods: both CUR-L2 and CUR-SL use the extensions proposed for CMD [26] and, in both cases, we sample exactly the same number of unique rows and columns as in the case of LS-DCUR and Greedy (double selections do not count as a selected row or column). For methods requiring computation of the top-k singular vectors (CUR-SL, Greedy), we specify a reasonable k. As setting it to the actual number of sampled rows and columns is not advisable, we follow the suggestion of [22] and over-sampled k; various experimental runs show that setting k to $\approx \frac{4}{5}$ of the number of row and column samples provides a convenient tradeoff between run-time performance and approximation accuracy; note that LS-DCUR does not require any additional parameters apart from the number of desired rows and columns. All tested methods were carefully implemented in Python/Numpy taking advantage of standard Lapack/Blas routines and the Arpack library as a sparse-eigenvalue solver.



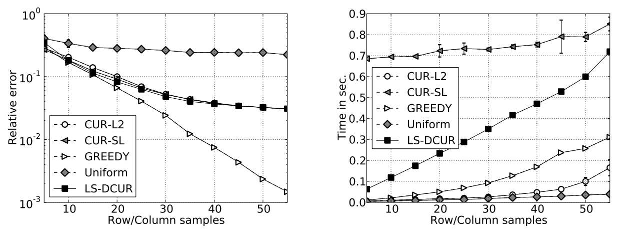
(a) Lena



(b) Standard normal



(c) Smooth



(d) Scaled

Figure 4: Results for four small/mid-size matrices. For each data set, the left plot shows the average accuracy for different numbers of selected rows and columns; the right plot shows the average processing time in seconds.

All experiments were carried out on a standard desktop computer using a single core and we refrained from using the aforementioned parallelized implementation of LS-DCUR for fair comparison[1].

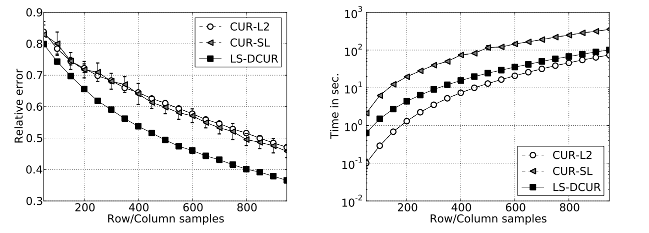
In every experiment, we compute the relative error and computation time in seconds. Specifically, let the sum of squared errors be defined as SSE $= \sum_{i,j}(\boldsymbol{X}_{i,j} - \boldsymbol{CUR}_{i,j})^2$. The relative SSE is then defined as relative SSE $=$ SSE$/\sum_{i,j}\boldsymbol{X}_{i,j}^2$. The computation time includes the selection process as well as the computation of the matrix $\boldsymbol{U}$ since it is an integral part of a CUR decompositions an in any case is needed for computing the error (this includes computation of the pseudo-inverse matrices $\boldsymbol{C}^+, \boldsymbol{R}^+$). Computation of $\boldsymbol{U}$ usually majorizes the computation time for all methods compared here. Note that, unlike e.g. in [26], in order to provide a fair comparison we do not compute the space requirements as we made sure that each method selects the same number of unique rows and columns. This actually favors randomized methods and has to be enforced for CUR-L2 and CUR-SL.

Accuracies and processing times for the randomized approaches are averaged over 3 runs for very large matrices (more than 50,000 rows or columns), and 10 runs for smaller matrices (up to 50,000 rows or columns). For very large matrices, we decided to skip Greedy and CUR-SL as the selection of only a few row or columns was already too time consuming. Note that the experimental design actually favors CUR-L2/CUR-SL. Real-world applications of importance sampling based CUR decompositions often require to sub-sample row and columns several times in order to achieve a sufficient performance. The methodology in [11] follows this strategy for co-clustering large-scale matrices, and [7] incorporates the increased runtime for sampling methods into their comparison of various methods. Thus, for any serious practical application, the computation time for CUR-L2/CUR-SL can be much higher than the numbers reported in the following.

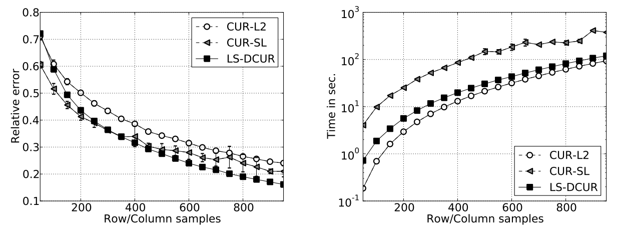
We considered various standard synthetic- and real-world data sets, consisting of both dense and sparse matrices of various sizes. Specifically, w.r.t question **Q1**, we considered the task of image compression using a standard benchmark image and three standard synthetic matrices (see e.g. [7, 25]). The first matrix is a 1000×1000 matrix with random values sampled from a standard normal distribution with zero mean and unit variance (labeled *standard normal*). The second matrix is a 200×200 matrix and presents a smooth function where each entry $\boldsymbol{x}_{i,j} = 1/(i^s + j^s)^{\frac{1}{s}}$ with $s = 2$ (labeled *smooth*). The third matrix is is a 500×500 scaled random matrix, with each element sampled from a random unit variance and zero mean distribution and scaled by $(20\epsilon)^{\frac{i}{n}}$ where ϵ is the machine precision (labeled *scaled*).

Addressing question **Q2** is the primary scope of this paper. Here, we consider several real-world data sets. The first collection consists of publicly available real-

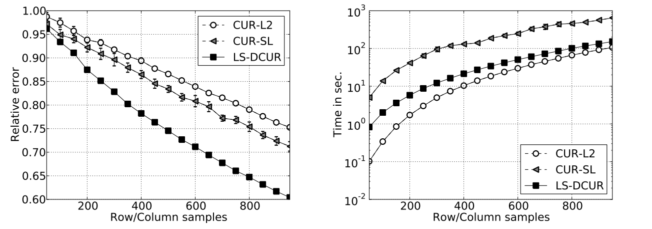
[1]The source code used in our experiments has been made available at http://code.google.com/p/pymf/.



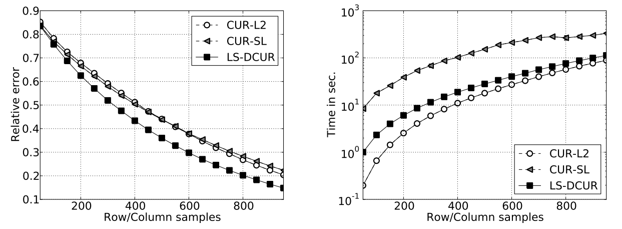
(a) Collab. net of Arxiv Gen. Rel. (5,242 nodes/28,980 edges)



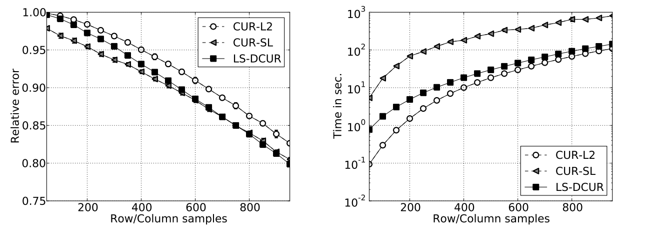
(b) Autonomous Systems graph (6,474 nodes/13,233 edges)



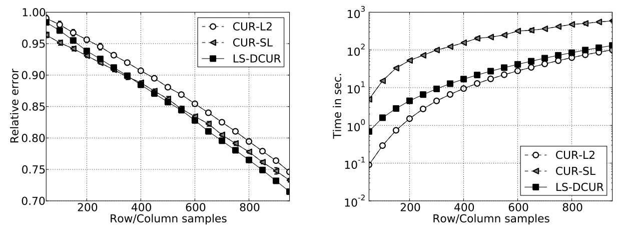
(c) Collab. net of Arxiv High Energy Phy. (9,877 nodes/51,971 edges)



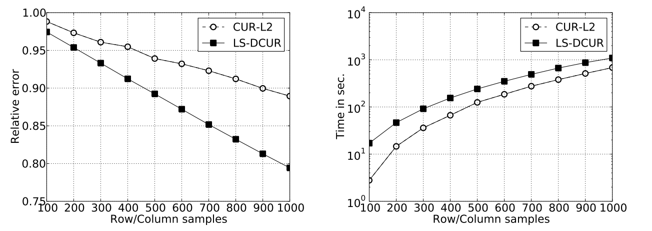
(d) Wikipedia who-votes-on-whom net (7,115 nodes/103,689 edges)



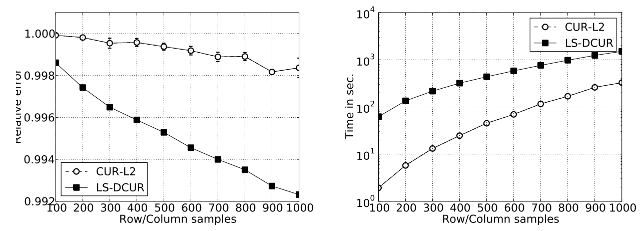
(e) Gnutella P2P net from August 4, 2002 (8,846 nodes/31,839 edges)



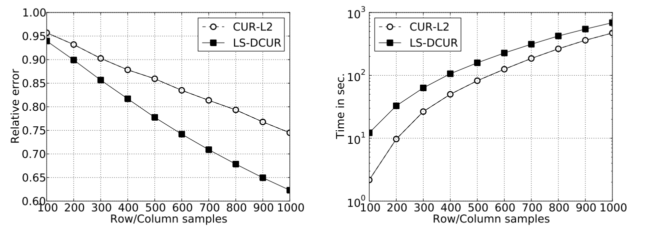
(f) Gnutella P2P net from August 6 2002 (8,717 nodes/31,525 edges)



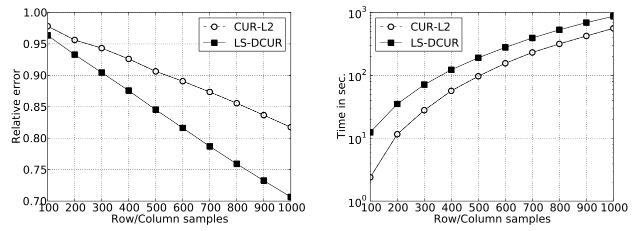
(g) Slashdot social network, November 2008 (77,360 nodes/905,468 edges)



(h) Amazon co-purchasing network, June 2003(403,394 nodes/3,387,388 edges)



(i) Who-trusts-whom network of Epinions.com (75,879 nodes/508,837 edges)



(j) Slashdot Zoo signed social network, Nov. 6 2008 (77,357 nodes/ 516,575 edges)

Figure 5: Results on sparse, real-world graphs. The goal is to achieve a high accuracy (good approximation), and low computation times (computation time uses a logarithmic-scale on the y-axis). Each plot varies the number of selected rows/columns from 100 to 1000, results for the randomized methods CUR-L2, and CUR-SL are averaged over 10 (3 for the larger graphs) runs. Overall, LS-DCUR compares favorably to both methods in terms of accuracy and runtime, especially considering that it is a deterministic approach that does not rely on sampling.

world large-scale graphs [2], represented as sparse adjacency matrices. The smallest graph (Collaboration net. of Arxiv General Relativity) has 5.242 nodes and 28.980 edges whereas the largest graph (Amazon product co-purchasing network from June 1) has 403, 394 nodes and 3, 387, 388 edges. To demonstrate the principal suitability of LS-DCUR for processing very large, dense matrices, we consider a data set of 80 million Google ^{TM} images [29]. Viewing each image (represented as a vector of gist descriptors) as a column, this results in a gigantic matrix containing $3072 \cdot 10^7$ elements.

5.1 Results Q1 (Small/Medium Matrices) Figure 3 shows a qualitative comparison of LS-DCUR and CUR-SL for image reconstruction using 35 and 70 selected rows and columns. Figure 4 presents further results for all four considered data sets for varying choices of parameters. It can be seen that deterministic methods outperform randomized approaches with respect to reconstruction accuracy. In terms of runtime performance, CUR-L2 is usually faster than the other methods. Due to the small size of the considered matrices, the mandatory computation of the top-k singular vectors for CUR-SL and Greedy only slightly lowers the (overall convenient) runtime performance. Nevertheless, most of the time, LS-DCUR outperforms Greedy, and CUR-SL with respect to computation time. To summarize, these results empirically validate that, with respect to accuracy, LS-DCUR comes close to the performance of Greedy when decomposing smaller, dense matrices. At the same time, it requires significantly lower computational costs.

5.2 Results Q2 (Large Matrices): Our experimental results are summarized in Fig. 5 and indicate that, with respect to accuracy and computation time, LS-DCUR outperforms CUR-L2 and CUR-SL. Specifically, for the smaller graphs, CUR-SL sometimes yields the best overall accuracy but suffers from high computational costs. To investigate this further, we performed a statistical significance analysis summarized in Tab. 2 and Tab. 3 where bold values indicate statistical significance. As one can see, LS-DCUR achieves a significantly higher accuracies than CUR-L2 for almost all data sets while being only slightly slower. Compared to CUR-SL, for some of the Gnutella graphs, it shows a lower accuracy but, at the same time, only needs a fraction of the processing time. Interestingly, for the larger graphs, CUR-L2 outperforms LS-DCUR on 3 graphs (Web-Notredame, Web-Stanford, Email-EU). These graphs are characterized by adjacency matrices of a very large number of extremely sparse rows and columns and there are only very few rows and columns with actual content. For example, for the Email-EU graph, about 1000 columns account for 75% of the total content of the 265,214 columns. Obviously, these kind of structures are extremely well suited to CUR-L2 as the selection probability for the vast majority of rows and columns is ≈ 0.

Dataset	Accuracy CUR-L2	Time CUR-L2
Amazon0302	-0.003 ± 0.0 ●	-223.06 ± 157.27●
Amazon0312	-0.004 ± 0.0 ●	-472.83 ± 331.72●
Amazon0505	-0.003 ± 0.0 ●	-518.2 ± 351.96●
Amazon0601	-0.004 ± 0.0 ●	-521.69 ± 358.42●
web-NotreDame	0.106 ± 0.03●	-544.44 ± 415.32●
web-Stanford	0.204 ± 0.02●	-604.68 ± 495.08●
sign-Slashdot106	-0.066 ± 0.03●	-131.04 ± 98.7 ●
sign-Slashdot216	-0.061 ± 0.03●	-138.32 ± 103.34●
sign-Slashdot221	-0.061 ± 0.03●	-138.42 ± 104.25●
Slashdot0811	-0.054 ± 0.03●	-174.11 ± 132.17●
Slashdot0902	-0.051 ± 0.03●	-180.48 ± 136.96●
Epinions1	-0.079 ± 0.04●	-98.85 ± 67.11●
Email-EuAll	0.138 ± 0.04●	-332.11 ± 243.38●

Table 3: Average differences of the relative accuracy and computation time in seconds measured between LS-DCUR and CUR-L2 (due to the size of the matrices computation of CUR-SL was infeasible). Negative values for the accuracy indicate a better reconstruction for LS-DCUR, positives values for runtime indicate a faster processing for LS-DCUR. Significant differences (paired t-test at $p = 0.05$) are indicated using a bullet ●; insignificant ones are shown using a circle ○.

Finally, we investigated the scalability of LS-DCUR to the gigantic matrix of $384 \times 80.000.000$ elements containing 80 million images. Here, we only explored the runtime performance of the selection of LS-DCUR. On a single core standard desktop computer, it took about 4 hours to extract a single row or column sample. On a 32 node Hadoop cluster, however, it took only 6 minutes to select one particular row or column.

5.3 Discussion Summarizing our results, an extensive empirical evaluation showed that LS-DCUR can outperform standard importance sampling methods such as CUR-L2 or CUR-SL on large, real-world matrices. Gain in performance was observed for both reconstruction accuracy and runtime. Note that due to the anticipated excessive computational costs and long runtime for the large matrices considered to answer question **Q2**, we did not compare to other deterministic methods. Nevertheless, we should point out that, for

[2] `http://snap.stanford.edu/data/index.html`

Dataset	Accuracy		Time	
	CUR-L2	CUR-SL	CUR-L2	CUR-SL
CA-GrQc	−0.095±0.03●	−0.089±0.02●	−12.19± 8.35●	96.1 ± 77.77●
as20000102	−0.063±0.02●	−0.009±0.04○	−10.77± 7.9 ●	107.11± 82.44●
CA-HepTh	−0.107±0.04●	−0.078±0.03●	−18.4 ±13.89●	192.6 ±150.82●
Wiki-Vote	−0.064±0.02●	−0.063±0.02●	−11.13± 7.17●	121.83± 75.46●
p2p-Gnutella04	−0.019±0.01●	0.008±0.01●	−13.69±10.15●	273.19±202.87●
p2p-Gnutella05	−0.002±0.0 ●	0.024±0.01●	−12.89± 9.67●	204.82±147.42●
p2p-Gnutella06	−0.023±0.01●	−0.002±0.01○	−12.24± 9.14●	210.08±147.48●
p2p-Gnutella08	0.04 ±0.02●	0.071±0.03●	−12.6 ± 9.89●	134.44±100.76●
p2p-Gnutella09	0.038±0.01●	0.071±0.02●	−13.6 ±10.74●	183.31±135.34●
p2p-Gnutella24	−0.017±0.01●	−0.001±0.01○	−22.79±16.54●	812.68±614.35●
p2p-Gnutella25	−0.018±0.01●	0.002±0.0 ●	−20.33±15.04●	720.56±563.4 ●
p2p-Gnutella30	−0.008±0.0 ●	0.018±0.0 ●	−28.77±21.07●	1047.7 ±862.2 ●
Email-Enron	−0.058±0.02●	−0.022±0.02●	−60.57±38.46●	822.47±559.3 ●
Cit-HepTh	−0.059±0.02●	−0.032±0.01●	−35.24±21.73●	655.57±468.06●
CA-CondMat	−0.083±0.02●	−0.047±0.02●	−38.06±27.45●	447.83±351.11●

Table 2: Average differences of the relative accuracy and computation time in seconds measured between LS-DCUR and CUR-L2, CUR-SL. Negative values for the accuracy indicate a better reconstruction for LS-DCUR, positives values for runtime indicate a faster processing for LS-DCUR. Significant differences (paired t-test at $p = 0.05$) are indicated using a bullet ●; insignificant ones are shown using a circle ○.

the small-scale matrices considered in our experiments, the deterministic Greedy algorithm we considered in answering question **Q1** provides a slightly higher accuracy than LS-DCUR. However, this is well in line with the goals of the introduced LS-DCUR algorithm which primarily targets very large matrices. Compared to sampling approaches, the rapid selection and high reconstruction accuracy of the LS-DCUR algorithm make it a good candidate for smaller matrices, too.

Our experiments also demonstrated that deterministic and sampling-based selections of rows and columns might be useful for different situations. For example, when selecting only a small amount of rows and columns from a very large sparse matrix, importance sampling shows a similar or sometimes higher accuracy than LS-DCUR. We attribute this to fact that the probability for picking orthogonal vectors increases with the size of the matrix and the degree of sparsity. In situation like these, one therefore does not explicitly need to enforce orthogonality or, for the case of LS-DCUR, large pairwise distances, since they will result implicitly.

6 Conclusion

As very large data matrices have become a commodity in scientific and economic applications alike, there is an increasing demand for low-rank approximations techniques that cope with massive data sets. Recent interpretable approaches that select rows and columns are either deterministic and apply only to smaller matrices or follow a randomized selection procedure and scale to larger matrices. In this paper, we showed that deterministic approaches, too, can scale to huge data matrices. We adopted a selection method that is based on principles from distance geometry and introduced the first deterministic selection method that scales well to matrices with billions of entries. Our extensive empirical evaluation revealed that the resulting *Large-scale Deterministic CUR* approach essentially combines the benefits of existing deterministic and randomized CUR approaches: it yields deterministic selections of rows and columns and provides high reconstruction accuracies at low computational costs. Thus, LS-DCUR should be of considerable interest to the data mining practitioner.

There are several interesting directions for future work. A straight forward speed-up of the proposed algorithm would result from using approximate distances which would no longer require access to all matrix entries. With respect to practical applications, we are currently adopting LS-DCUR to co-clustering problems [11]. Employing LS-DCUR within challenging applications such as segmentation of fMRI data or topic models for text collections with billions of data points are other promising scenarios. Already, we recently applied deterministic column selection to the problem of identifying relevant patterns in massive, hyper-spectral, high-dimensional sensor readings for plant phenotyping [10, 20]. Here a good selection after only a single run is crucial because reconstruction can take several days; the data matrix considered in [20] consists of more than 12 billion entries and is still growing. Therefore, a purely random selection is not an option. Finally, it is interesting to adopt LS-CUR to multi-view data and to data streams as well as to develop hybrid CUR approaches based on LS-DCUR.

References

[1] M. Berry, S. Pulatova, and G. Stewart. Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices. *ACM Trans. Math. Softw.*, 31:252–269, 2005.

[2] L. M. Blumenthal. *Theory and Applications of Distance Geometry.* Oxford University Press, 1953.

[3] C. Boutsidis, M. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proc. ACM SIGKDD*, 2008.

[4] C. Boutsidis, M. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *Proc. ACM SODA*, 2009.

[5] P. Businger and G. H. Golub. Linear least squares solutions by householder transformations. *Numer. Math.*, 7(3):269–276, 1965.

[6] A. Civril and M. Magdon-Ismail. On Selecting A Maximum Volume Sub-matrix of a Matrix and Related Problems. *Theoret. Comput. Sci.*, 410(47–49):4801–4811, 2009.

[7] A. Civril and M. Magdon-Ismail. Column subset selection via sparse approximation of svd. *Theoret. Comput. Sci.*, 2011. in press, DOI:10.1016/j.tcs.2011.11.019.

[8] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. of the ACM*, 51(1):107–113, 2008.

[9] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo Algorithms III: Computing a Compressed Approixmate Matrix Decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006.

[10] Editorial. How to Feed a Hungry World. *Nature*, 7306(466):531–532, 2010.

[11] P. Feng, Z. Xiang, and W. Wei. CRD: fast coclustering on large datasets utilizing sampling based matrix decomposition. In *Proc. ACM SIGMOD*, 2008.

[12] A. Frieze, R. Kannan, and S. Vempala. Fast montecarlo algorithms for finding lowrank approximations. *J. of the ACM*, 51(6):1025–1041, 2004.

[13] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin. How to find a good submatrix. In V. Olshevsky and E. Tyrtyshnikov, editors, *Matrix Methods: Theory, Algorithms, Applications.* World Scientific, Hackensack, NY, 2010.

[14] S. Goreinov and E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. In V. Olshevsky, editor, *Structured Matrices in Mathematics, Computer Science, and Engineering I*, volume 280 of *Contemp. Math.* AMS, Providence, RI, 2001.

[15] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra Appl.*, 261(1-3):1 – 21, 1997.

[16] A. Halevy, P. Norvig, and F. Pereira. The Unreasonable Effectiveness of Data. *IEEE Intell. Syst.*, 24(2):8–12, 2009.

[17] S. Hyvönen, P. Miettinen, and E. Terzi. Interpretable nonnegative matrix decompositions. In *Proc. ACM SIGKDD*, 2008.

[18] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *Proc. PAKDD*, 2011.

[19] U. Kang, C. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system. In *Proc. IEEE ICDM*, 2009.

[20] K. Kersting, M. Wahabzada, C. Roemer, C. Thurau, A. Ballvora, U. Rascher, J. Leon, C. Bauckhage, and L. Pluemer. Simplex distributions for embedding data matrices over time. In *Proc. SIAM SDM*, 2012.

[21] M. Mahoney, M. Maggioni, and P. Drineas. Tensor-cur decompositions for tensor-based data. In *Proc. ACM SIGKDD*, 2006.

[22] M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proc. Natl. Acad. Sci. USA*, 106(3):697–702, 2009.

[23] T. Mitchell. Mining Our Reality. *Science*, 326:1644–1645, 2009.

[24] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *J. Mach. Learn. Res.*, 10:1801–1828, 2009.

[25] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J. Matrix Anal. Appl.*, 30:939–956, 2008.

[26] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is More: Compact Matrix Decomposition for Large Sparse Graphs. In *Proc. SIAM SDM*, 2007.

[27] C. Thurau, K. Kersting, and C. Bauckhage. Yes We Can – Simplex Volume Maximization for Descriptive Web–Scale Matrix Factorization. In *Proc. ACM CIKM*, 2010.

[28] C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage. Descriptive Matrix Factorization for Sustainability: Adopting the Principle of Opposites. *Data Min. Knowl. Discov.*, 24(2):325–354, 2012.

[29] A. Torralba, R. Fergus, and W. T. Freeman. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.