SIAM J. COMPUT.
Vol. 36, No. 1, pp. 184–206
© 2006 Society for Industrial and Applied Mathematics

FAST MONTE CARLO ALGORITHMS FOR MATRICES III: COMPUTING A COMPRESSED APPROXIMATE MATRIX DECOMPOSITION*

PETROS DRINEAS†, RAVI KANNAN‡, AND MICHAEL W. MAHONEY§

Abstract. In many applications, the data consist of (or may be naturally formulated as) an $m \times n$ matrix A which may be stored on disk but which is too large to be read into random access memory (RAM) or to practically perform superlinear polynomial time computations on it. Two algorithms are presented which, when given an $m \times n$ matrix A, compute approximations to A which are the product of three smaller matrices, C, U, and R, each of which may be computed rapidly. Let $A' = CUR$ be the computed approximate decomposition; both algorithms have provable bounds for the error matrix $A - A'$. In the first algorithm, c columns of A and r rows of A are randomly chosen. If the $m \times c$ matrix C consists of those c columns of A (after appropriate rescaling) and the $r \times n$ matrix R consists of those r rows of A (also after appropriate rescaling), then the $c \times r$ matrix U may be calculated from C and R. For any matrix X, let $\|X\|_F$ and $\|X\|_2$ denote its Frobenius norm and its spectral norm, respectively. It is proven that

$$\left\|A - A'\right\|_\xi \leq \min_{D:\mathrm{rank}(D)\leq k} \|A - D\|_\xi + poly(k, 1/c)\, \|A\|_F$$

holds in expectation and with high probability for both $\xi = 2, F$ and for all $k = 1, \ldots, \mathrm{rank}(A)$; thus by appropriate choice of k

$$\left\|A - A'\right\|_2 \leq \epsilon \|A\|_F$$

also holds in expectation and with high probability. This algorithm may be implemented without storing the matrix A in RAM, provided it can make two passes over the matrix stored in external memory and use $O(m + n)$ additional RAM (assuming that c and r are constants, independent of the size of the input). The second algorithm is similar except that it approximates the matrix C by randomly sampling a constant number of rows of C. Thus, it has additional error but it can be implemented in three passes over the matrix using only constant additional RAM. To achieve an additional error (beyond the best rank-k approximation) that is at most $\epsilon\|A\|_F$, both algorithms take time which is a low-degree polynomial in k, $1/\epsilon$, and $1/\delta$, where $\delta > 0$ is a failure probability; the first takes time linear in $\max(m, n)$ and the second takes time independent of m and n. The proofs for the error bounds make important use of matrix perturbation theory and previous work on approximating matrix multiplication and computing low-rank approximations to a matrix. The probability distribution over columns and rows and the rescaling are crucial features of the algorithms and must be chosen judiciously.

Key words. randomized algorithms, Monte Carlo methods, massive data sets, CUR matrix decomposition

AMS subject classification. 68W20

DOI. 10.1137/S0097539704442702

*Received by the editors April 5, 2004; accepted for publication (in revised form) November 17, 2005; published electronically May 26, 2006. The technical report version of this journal paper appeared as *Fast Monte Carlo Algorithms for Matrices* III: *Computing a Compressed Approximate Matrix Decomposition*, by P. Drineas, R. Kannan, and M. W. Mahoney [12]. A preliminary version of parts of this paper, in particular the main algorithm and main theorem of section 3, appeared as *Pass efficient algorithms for approximating large matrices*, by P. Drineas and R. Kannan [9].

http://www.siam.org/journals/sicomp/36-1/44270.html

†Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 (drinep@cs.rpi.edu).

‡Department of Computer Science, Yale University, New Haven, CT 06520 (kannan@cs.yale.edu).
This author was supported in part by a grant from the NSF.
§Department of Mathematics, Yale University, New Haven, CT 06520 (mahoney@cs.yale.edu).

1. Introduction. We are interested in developing and analyzing fast Monte Carlo algorithms for performing useful computations on large matrices. In this paper we consider a new method for computing a compressed approximate decomposition of a large matrix; in two related papers we consider matrix multiplication and the singular value decomposition (SVD) [10, 11]. Since such computations generally require time which is superlinear in the number of nonzero elements of the matrix, we expect our algorithms to be useful in many applications where data sets are modeled by matrices and are extremely large. In all of these cases, we assume that the input matrices are prohibitively large to store in random access memory (RAM) and thus that only external memory storage is possible. Thus, our algorithms will be allowed to read the matrices a few, e.g., one, two, or three, times and keep a small randomly chosen and rapidly computable "sketch" of the matrices in RAM; computations will then be performed on this "sketch." We will work within the framework of the pass-efficient computational model, in which the scarce computational resources are the number of passes over the data, the additional RAM space required, and the additional time required [10, 9].

In many applications, an $m \times n$ matrix A is stored on disk and is too large to be read into RAM or to practically perform superlinear polynomial time computations on it; thus, one may be interested in a succinctly described, easily computed $m \times n$ matrix A' that is an approximation to A. Let c and r be positive, usually constant, integers that we choose, and for any matrix X let $\|X\|_F$ and $\|X\|_2$ denote its Frobenius and spectral norms (as defined in section 2.1), respectively. We present two algorithms that compute an approximation A' to the matrix A that has the following properties:

(i) $A' = CUR$, where C is an $m \times c$ matrix consisting of c randomly picked columns of A, R is an $r \times n$ matrix consisting of r randomly picked rows of A, and U is a $c \times r$ matrix computed from C, R.

(ii) C, U, and R can be defined after making a small constant number of passes (2 or 3 for the two algorithms presented in this paper) through the whole matrix A from disk.

(iii) U can be constructed using additional RAM space and time that is $O(m + n)$ (for the LINEARTIMECUR algorithm) or is $O(1)$ (for the CONSTANTTIMECUR algorithm), assuming that c and r are constant.

(iv) For every $\epsilon > 0$ and every k such that $1 \leq k \leq \text{rank}(A)$ we can choose c and r (to be specified below) such that, with high probability, A' satisfies

$$\|A - A'\|_2 \leq \min_{D:\text{rank}(D)\leq k} \|A - D\|_2 + \epsilon \|A\|_F ,$$

and thus we can choose c and r such that $\|A - A'\|_2 \leq \epsilon \|A\|_F$.

(v) For every $\epsilon > 0$ and every k such that $1 \leq k \leq \text{rank}(A)$ we can choose c and r (to be specified below) such that, with high probability, A' satisfies

$$\|A - A'\|_F \leq \min_{D:\text{rank}(D)\leq k} \|A - D\|_F + \epsilon \|A\|_F .$$

In the first algorithm, the LINEARTIMECUR algorithm of section 3, c columns of A and r rows of A are randomly sampled. If the $m \times c$ matrix C consists of those c columns of A (after appropriate rescaling) and the $r \times n$ matrix R consists of those R rows of A (also after appropriate rescaling), then from C and R the $c \times r$ matrix U is computed. This algorithm may be implemented without storing the matrix A in RAM, provided it can make two passes over the matrix stored in external memory

TABLE 1
Summary of sampling complexity.

Additional error for:	LINEARTIMECUR	CONSTANTTIMECUR
$\|A - A'\|_2$	$c, r = \frac{\eta^2}{\epsilon^4}, \frac{k}{\delta^2\epsilon^2}$	$c, w, r = \frac{\eta^2}{\epsilon^8}, \frac{\eta^2}{\epsilon^8}, \frac{k}{\delta^2\epsilon^2}$
$\|A - A'\|_F$	$c, r = \frac{k\eta^2}{\epsilon^4}, \frac{k}{\delta^2\epsilon^2}$	$c, w, r = \frac{k^2\eta^2}{\epsilon^8}, \frac{k^2\eta^2}{\epsilon^8}, \frac{k}{\delta^2\epsilon^2}$

and use $O(m+n)$ additional RAM. The second algorithm, the CONSTANTTIMECUR algorithm of section 4, is similar except that it approximates the matrix C by randomly sampling a constant number w of rows of C. Thus, our second algorithm requires only constant additional RAM but uses a third pass over the data and has additional error. To achieve an additional error (beyond the best rank-k approximation) that is at most $\epsilon \|A\|_F$, both algorithms take time which is a low-degree polynomial in k, $1/\epsilon$, and $1/\delta$, where $\delta > 0$ is a failure probability; the first algorithm takes time linear in $\max(m, n)$ and the second takes time independent of m and n. See Table 1 for a summary of the dependence of the sampling complexity on k and ϵ, δ, and $\eta = 1 + \sqrt{8 \log(1/\delta)}$. (Note that the two algorithms we present in this paper are most interesting when the matrix A is well approximated by a rank-k matrix, where k is assumed to be constant with respect to m and n. In this case, c, r, and w are also constant with respect to m and n. This is assumed throughout the remainder of this paper.)

The proofs for the error bounds make important use of linear algebra and matrix perturbation theory; see [19, 22, 28, 6] for an overview of these topics. In particular, the proofs use the approximate matrix multiplication results of [10] and the approximate SVD results of [11]. As with those previous works, the probability distribution over the columns, the probability distribution over the rows, and the respective rescaling are crucial features of the algorithms which must be chosen judiciously. In addition, as a by-product of the CUR decomposition, we can estimate the top k singular values of A.

Our CUR approximations may be viewed as a "dimension reduction" technique. Two common techniques for dimension reduction are the SVD and multidimensional scaling; see [19, 25]. Another method that has attracted renewed interest recently is the traditional "random projection" method where one projects the problem into a randomly chosen low-dimensional subspace [24, 29, 23]. These methods do not share properties (i) and (ii) and thus are not suited for very large problems. Our algorithms achieve (i) and (ii) at the cost of the $\epsilon \|A\|_F$ error. In addition, our CUR approximations may be viewed as an approximate decomposition of a matrix $A \approx CUR$; as with other decompositions, the CUR decomposition both reveals information about the structure of the matrix and allows computations to be performed more efficiently. For example, in applications of the LINEARTIMECUR algorithm, A' could be stored in RAM since it can be stored in $O(m+n)$ space (instead of $O(mn)$ space); in addition, A' could then be operated upon by, e.g., applying $A' = CUR$ to a query vector $x \in \mathbb{R}^n$, which is an operation that can be performed in $O(m+n)$ space (instead of $O(mn)$ space if A is used).

Our CUR approximations have been used in applications such as the reconstruction of a matrix given a sample of the matrix in a recommendation systems context and for "similarity query" problems which are widely used in areas such as information retrieval [14]. In this application, after A has been preprocessed, one gets "query" vectors x and must find the similarity of x to each row of A. Here, the similarity of

two vectors is defined to be their dot product or their normalized dot product; our technique can handle both. See [7] for related discussion. Note that the measure $\|A\|_2$ is a worst-case measure and that this is more useful in many contexts than an average-case measure like $\|A\|_F$, since the relevant query x often comes from a small-dimensional subspace and is not random. See also [2, 1] for a nice discussion of these issues. Our CUR approximations have also been used in theoretical applications such as designing and analyzing approximation algorithms for the max-cut problem [13]. In these applications, the use of the constant additional space and time framework is essential.

In other related work, Achlioptas and McSherry have also computed succinctly described matrix approximations using somewhat different sampling techniques [2, 1]. Also included in [2, 1] is a comparison of their methods with those of [8, 9, 18] and thus with the results we present here. When compared with our LINEARTIMECUR algorithm, they achieve the same results for the Frobenius norm bound and slightly better results (with respect to $1/\epsilon$) for the spectral norm bound [1]; in their work, however, there is no analogue of our CONSTANTTIMECUR algorithm.

After this introduction, we provide in section 2 reviews of linear algebra, the pass-efficient model, and of several previous results from [10, 11] that are used in this paper. In section 3 we describe and analyze the LINEARTIMECUR algorithm which computes an approximate CUR decomposition of a matrix A using linear (in m and n) additional space and time, and in section 4 we describe and analyze the CONSTANTTIMECUR algorithm which computes a description of an approximate CUR decomposition of a matrix A using only constant additional space and time. Finally, in section 5 we provide a discussion and conclusion.

Finally, note that c and r enter into the asymptotic analysis; for improved clarity, however, we generally take them to be constants that do not vary.

2. Review of relevant background.

2.1. Review of linear algebra. This section contains a review of linear algebra that will be useful throughout the paper; for more details, see [19, 22, 28, 6] and the references therein.

For a vector $x \in \mathbb{R}^n$ we let $|x| = (\sum_{i=1}^n |x_i|^2)^{1/2}$ denote its Euclidean length. For a matrix $A \in \mathbb{R}^{m\times n}$ we let $A^{(j)}$, $j = 1, \ldots, n$, denote the jth column of A as a column vector and $A_{(i)}$, $i = 1, \ldots, m$, denote the ith row of A as a row vector. We denote matrix norms by $\|A\|_\xi$, using subscripts to distinguish between various norms. Of particular interest will be the Frobenius norm, the square of which is $\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2$, and the spectral norm, which is defined by $\|A\|_2 = \sup_{x\in\mathbb{R}^n,\ x\neq 0} \frac{|Ax|}{|x|}$. These norms are related to each other as $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\,\|A\|_2$.

If $A \in \mathbb{R}^{m\times n}$, then there exist orthogonal matrices $U = [u^1 u^2 \ldots u^m] \in \mathbb{R}^{m\times m}$ and $V = [v^1 v^2 \ldots v^n] \in \mathbb{R}^{n\times n}$, where $\{u^t\}_{t=1}^m \in \mathbb{R}^m$ and $\{v^t\}_{t=1}^n \in \mathbb{R}^n$ are such that

$$U^T A V = \Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_\rho),$$

where $\Sigma \in \mathbb{R}^{m\times n}$, $\rho = \min\{m, n\}$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho \geq 0$. Equivalently, $A = U\Sigma V^T$. The three matrices U, V, and Σ constitute the SVD of A. The σ_i are the singular values of A and the vectors u^i, v^i are the ith left and the ith right singular vectors, respectively. If $k \leq r = \mathrm{rank}(A)$ and we define $A_k = U_k \Sigma_k V_k^T = \sum_{t=1}^k \sigma_t u^t v^{t^T}$, then the distance (as measured by both $\|\cdot\|_2$ and $\|\cdot\|_F$) between A and

any rank-k approximation to A is minimized by A_k, i.e.,

$$\min_{D\in\mathbb{R}^{m\times n}:\mathrm{rank}(D)\leq k} \|A-D\|_2 = \|A-A_k\|_2 = \sigma_{k+1}(A), \tag{1}$$

$$\min_{D\in\mathbb{R}^{m\times n}:\mathrm{rank}(D)\leq k} \|A-D\|_F^2 = \|A-A_k\|_F^2 = \sum_{t=k+1}^{r} \sigma_t^2(A). \tag{2}$$

2.2. Review of the pass-efficient model. The pass-efficient model of data-streaming computation is a computational model that is motivated by the observation that in modern computers the amount of disk storage, i.e., sequential access memory, has increased very rapidly, while RAM and computing speeds have increased at a substantially slower pace [10, 9]. In the pass-efficient model the three scarce computational resources are number of passes over the data and the additional RAM space and additional time required by the algorithm. The data are assumed to be stored on a disk, to consist of elements whose sizes are bounded by a constant, and to be presented to an algorithm on a read-only tape. See [10] for more details.

2.3. Review of approximate matrix multiplication. The BasicMatrixMultiplication algorithm to approximate the product of two matrices is presented and analyzed in [10]. When this algorithm is given as input two matrices, $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{n\times p}$, a probability distribution $\{p_i\}_{i=1}^n$, and a number $c\leq n$, it returns as output two matrices, C and R, such that $CR\approx AB$; $C\in\mathbb{R}^{m\times c}$ is a matrix whose columns are c randomly chosen columns of A (suitably rescaled) and $R\in\mathbb{R}^{c\times p}$ is a matrix whose rows are the c corresponding rows of B (also suitably rescaled). An important aspect of this algorithm is the probability distribution $\{p_i\}_{i=1}^n$ used to choose column-row pairs. Although one could always use a uniform distribution, superior results are obtained if the probabilities are chosen judiciously. In particular, a set of sampling probabilities $\{p_i\}_{i=1}^n$ are the *optimal probabilities* (with respect to approximating the product AB) if they are of the form (3); for an explanation and discussion, see [10], where we prove the following.

Theorem 1. *Suppose $A\in\mathbb{R}^{m\times n}$, $B\in\mathbb{R}^{n\times p}$, $c\in\mathbb{Z}^+$ such that $1\leq c\leq n$, and $\{p_i\}_{i=1}^n$ are such that $p_i\geq 0$ and $\sum_{i=1}^n p_i = 1$. Construct C and R with the* BasicMatrixMultiplication *algorithm of* [10] *and let CR be an approximation to AB. If the probabilities $\{p_i\}_{i=1}^n$ are such that*

$$p_k = \frac{\left|A^{(k)}\right|\,\left|B_{(k)}\right|}{\sum_{k'=1}^n \left|A^{(k')}\right|\,\left|B_{(k')}\right|}, \tag{3}$$

then

$$\mathbf{E}\left[\,\|AB-CR\|_F\right] \leq \frac{1}{\sqrt{c}}\,\|A\|_F\,\|B\|_F\,. \tag{4}$$

If, in addition, we let $\delta\in(0,1)$ and $\eta = 1+\sqrt{8\log(1/\delta)}$, then with probability at least $1-\delta$,

$$\|AB-CR\|_F \leq \frac{\eta}{\sqrt{c}}\,\|A\|_F\,\|B\|_F\,. \tag{5}$$

Furthermore, if the probabilities $\{p_i\}_{i=1}^n$ are such that

$$p_k = \frac{\left|B_{(k)}\right|^2}{\|B\|_F^2}, \tag{6}$$

then

$$\mathbf{E}\left[\,\|AB-CR\|_F\right] \le \frac{1}{\sqrt{c}}\,\|A\|_F\,\|B\|_F\,. \tag{7}$$

Note that with probabilities of the form (6), we do not get a bound of the form $\|AB-CR\|_F \le \frac{1}{\sqrt{c}}\,\|A\|_F\,\|B\|_F$ by sampling $O(\log(1/\delta))$ columns without making additional, awkward assumptions on the input matrices; see [10]. Of course, by Markov's inequality we can (and will) obtain such a bound by sampling $O(1/\delta)$ columns.

2.4. Review of approximate SVD. The LINEARTIMESVD algorithm is presented in [11]. It is an algorithm which, when given a matrix $A \in \mathbb{R}^{m\times n}$, uses $O(m+n)$ additional space and time to compute an approximation to the top k singular values and the corresponding left singular vectors of A by randomly choosing c columns of A and rescaling each appropriately to construct a matrix $C \in \mathbb{R}^{m\times c}$, computing the top k singular values and corresponding right singular vectors of C, and using them to construct a matrix $H_k \in \mathbb{R}^{m\times k}$ consisting of approximations to the top k left singular vectors of A. In [11] we prove the following.

THEOREM 2. *Suppose $A \in \mathbb{R}^{m\times n}$ and let H_k be constructed from the* LINEARTIMESVD *algorithm of* [11]. *Then*

$$\left\|A - H_kH_k^TA\right\|_F^2 \le \|A-A_k\|_F^2 + 2\sqrt{k}\left\|AA^T-CC^T\right\|_F, \tag{8}$$

$$\left\|A - H_kH_k^TA\right\|_2^2 \le \|A-A_k\|_2^2 + 2\left\|AA^T-CC^T\right\|_2. \tag{9}$$

The CONSTANTTIMESVD algorithm is also presented in [11]. It is an algorithm which, when given a matrix $A \in \mathbb{R}^{m\times n}$, uses constant additional space and time to compute a description of an approximation to the top k singular values and the corresponding left singular vectors of A. It does so in a manner similar to that of the LINEARTIMESVD algorithm except that it performs a second level of sampling in order to estimate (rather than compute exactly) the top k singular values and corresponding singular vectors of C. The γ in the following theorem is a parameter of the CONSTANTTIMESVD algorithm of [11] that is related to the second level of sampling; it also appears in our CONSTANTTIMECUR algorithm, and is thus discussed in section 4. In [11] we prove the following.

THEOREM 3. *Suppose $A \in \mathbb{R}^{m\times n}$; let a description of $\tilde{H}_\ell$ be constructed from the* CONSTANTTIMESVD *algorithm of* [11] *by sampling c columns of A with probabilities $\{p_i\}_{i=1}^n$ and w rows of C with probabilities $\{q_j\}_{j=1}^m$ where $p_i = |A^{(i)}|^2/\|A\|_F^2$ and $q_j = |C_{(j)}|^2/\|C\|_F^2$. Let $\eta = 1+\sqrt{8\log(2/\delta)}$ and $\epsilon > 0$.*

If a Frobenius norm bound is desired, and hence the CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100k$, then by choosing $c = \Omega(k^2\eta^2/\epsilon^4)$ columns of A and $w = \Omega(k^2\eta^2/\epsilon^4)$ rows of C we have that with probability at least $1-\delta$,*

$$\left\|A - \tilde{H}_\ell\tilde{H}_\ell^TA\right\|_F^2 \le \|A-A_k\|_F^2 + \epsilon\,\|A\|_F^2. \tag{10}$$

If a spectral norm bound is desired, and hence the CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100$, then by choosing $c = \Omega(\eta^2/\epsilon^4)$ columns of A and $w = \Omega(\eta^2/\epsilon^4)$ rows of C we have that with probability at least $1-\delta$,*

$$\left\|A - \tilde{H}_\ell\tilde{H}_\ell^TA\right\|_2^2 \le \|A-A_k\|_2^2 + \epsilon\,\|A\|_F^2. \tag{11}$$

3. The linear time *CUR* decomposition. In this section we describe and analyze the LINEARTIMECUR algorithm, which computes an approximate CUR decomposition of a matrix $A \in \mathbb{R}^{m \times n}$ using linear (in m and n) additional space and time. In section 4 we describe and analyze the CONSTANTTIMECUR algorithm which computes a description of an approximate CUR decomposition of a matrix A using only constant additional space and time. Both algorithms will make extensive use of the corresponding results from [11] for approximating the SVD of a matrix as well as results from [10] on approximating the product of two matrices. As with the SVD algorithms, the CONSTANTTIMECUR algorithm has a similar flavor to the LINEARTIMECUR algorithm, but is technically more complex due to the second level of sampling required. Thus, in this section we provide an extensive description of the LINEARTIMECUR algorithm and the motivation and intuition behind it, and in section 4 we highlight the differences between the linear additional time framework and the constant additional time framework.

3.1. The algorithm. Given a matrix $A \in \mathbb{R}^{m \times n}$, we wish to compute a succinctly described, easily computed matrix A' that is decomposable as $A' = CUR \in \mathbb{R}^{m \times n}$ and that satisfies properties (i)–(v) of section 1. The LINEARTIMECUR algorithm, which is presented in Figure 1, accomplishes this by first forming a matrix $C \in \mathbb{R}^{m \times c}$ by rescaling a randomly chosen subset of c columns of A; the columns are chosen in c independent identical trials where in each trial the αth column of A is chosen with probability q_α, and if the αth column is chosen, it is rescaled by $1/\sqrt{cq_\alpha}$ before inclusion in C. The algorithm then forms a matrix $R \in \mathbb{R}^{r \times n}$ by rescaling a randomly chosen subset of r rows of A; the rows are chosen in r independent identical trials where in each trial the αth row of A is chosen with the probability p_α, and if the αth row is chosen, it is rescaled by $1/\sqrt{rp_\alpha}$ before inclusion in R. Using the same randomly chosen rows to construct R from A the algorithm also constructs a matrix Ψ from C in an identical manner. Thus, $\Psi \in \mathbb{R}^{r \times c}$ and $\Psi_{ij} = A_{i_{t_1} j_{t_2}} / \sqrt{crp_{i_{t_1}} q_{j_{t_2}}}$, where i_{t_1} is the element of $\{1, \ldots, m\}$ selected in the t_1th row sampling trial and j_{t_2} is the element of $\{1, \ldots, n\}$ selected in the t_2th column sampling trial.

The following sampling matrix formalism provides a convenient representation of our ideas, and will be used extensively in this section and the next. (See [10] for another use of this sampling matrix formalism.) Let us define the column sampling matrix $S_C \in \mathbb{R}^{n \times c}$ to be the zero-one matrix where $(S_C)_{ij} = 1$ if the ith column of A is chosen in the jth independent random trial, and $S_{ij} = 0$ otherwise; let us also define the associated rescaling matrix $D_C \in \mathbb{R}^{c \times c}$ to be the diagonal matrix with $(D_C)_{tt} = 1/\sqrt{cp_{i_t}}$, where i_t is the element of $\{1, \ldots, n\}$ chosen in the tth sampling trial. Let us similarly define $S_R \in \mathbb{R}^{r \times m}$ and $D_R \in \mathbb{R}^{r \times r}$ to be the row sampling matrix and associated diagonal rescaling matrix, respectively. In this notation,

$$C = AS_C D_C \qquad \text{and} \qquad R = D_R S_R A, \tag{12}$$

where $S_C D_C$ postmultiplies (and thus samples and rescales columns of) A to form C, and where $D_R S_R$ premultiplies (and thus samples and rescales rows of) A to form R. Thus, in this notation,

$$\Psi = D_R S_R C = D_R S_R A S_C D_C. \tag{13}$$

Given C, the LINEARTIMECUR algorithm computes the top k singular values, $\sigma_t^2(C)$, $t = 1, \ldots, k$, and the corresponding singular vectors, y^t, $t = 1, \ldots, k$, of $C^T C$. Note that these are also the squares of the singular values and the corresponding right

LINEARTIMECUR Algorithm.

Input: $A \in \mathbb{R}^{m \times n}$, $r, c, k \in \mathbb{Z}^+$ such that $1 \leq r \leq m$, $1 \leq c \leq n$, and $1 \leq k \leq \min(r, c)$, $\{p_i\}_{i=1}^m$ such that $p_i \geq 0$ and $\sum_{i=1}^m p_i = 1$, and $\{q_j\}_{j=1}^n$ such that $q_j \geq 0$ and $\sum_{j=1}^n q_j = 1$.

Output: $C \in \mathbb{R}^{m \times c}$, $U \in \mathbb{R}^{c \times r}$, and $R \in \mathbb{R}^{r \times n}$.

1. For $t = 1$ to c,
   (a) Pick $j_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}[j_t = \alpha] = q_\alpha$, $\alpha = 1, \ldots, n$.
   (b) Set $C^{(t)} = A^{(j_t)} / \sqrt{cq_{j_t}}$.
2. Compute $C^T C$ and its SVD; say $C^T C = \sum_{t=1}^c \sigma_t^2(C) y^t y^{t^T}$.
3. If $\sigma_k(C) = 0$, then let $k = \max\{k' : \sigma_{k'}(C) \neq 0\}$.
4. For $t = 1$ to r,
   (a) Pick $i_t \in \{1, \ldots, m\}$ with $\mathbf{Pr}[i_t = \alpha] = p_\alpha$, $\alpha = 1, \ldots, m$.
   (b) Set $R_{(t)} = A_{(i_t)} / \sqrt{rp_{i_t}}$.
   (c) Set $\Psi_{(t)} = C_{(i_t)} / \sqrt{rp_{i_t}}$.
5. Let $\Phi = \sum_{t=1}^k \frac{1}{\sigma_t^2(C)} y^t y^{t^T}$ and let $U = \Phi \Psi^T$.
6. Return C, U, and R.

FIG. 1. *The* LINEARTIMECUR *algorithm.*

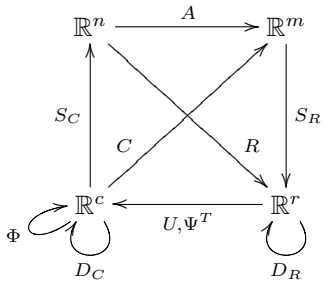

FIG. 2. *Diagram for the* LINEARTIMECUR *algorithm.*

singular vectors of C. Using these quantities, a matrix $\Phi \in \mathbb{R}^{c \times c}$ may be defined as

$$\Phi = \sum_{t=1}^{k} \frac{1}{\sigma_t^2(C)} y^t y^{t^T}, \tag{14}$$

from which $U \in \mathbb{R}^{c \times r}$ is constructed as $U = \Phi \Psi^T$. We could, of course, have defined a matrix Φ (and thus constructed a matrix U) from the singular vectors and singular values of RR^T in a manner analogous to that described above; in that case, the roles of the row sampling and column sampling would be reversed relative to the discussion below.

Figure 2 presents a diagram illustrating the action of the LINEARTIMECUR algorithm. The matrix A is shown as operating between the high-dimensional spaces $\mathbb{R}^n$ and $\mathbb{R}^m$. In addition, the matrix C is shown as operating between $\mathbb{R}^c$ and $\mathbb{R}^m$ and the matrix R is shown as operating between $\mathbb{R}^n$ and $\mathbb{R}^r$. Intuitively, one may think of $\mathbb{R}^c$ and $\mathbb{R}^r$ as being the most significant parts of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively, in terms of the action of A. Indeed, this will be the case when the sampling probabilities $\{p_i\}_{i=1}^m$

and $\{q_j\}_{j=1}^n$ satisfy certain conditions, as stated, e.g., in Theorem 4. The diagram also illustrates that $C = AS_CD_C$, $R = D_RS_RA$, and the matrix U which can be seen to be $U = \Phi\Psi^T = \Phi C^T(D_RS_R)^T$.

The matrix A is thus approximated by a matrix $A' = CUR$, where C is an $m \times c$ matrix consisting of c randomly chosen columns of A, R is an $r \times n$ matrix consisting of r randomly chosen rows of A, and $U = \Phi\Psi^T$ is a $c \times r$ matrix computed from C and R. As we shall see, if the column and row sampling probabilities are chosen judiciously, then c and r can be chosen to be constants (independent of m and n but depending on k and ϵ). Thus, (pictorially) we have that

$$\left(\begin{array}{c} \\ A \\ \\ \end{array}\right) \approx \left(\begin{array}{c} \\ C \\ \\ \end{array}\right) \left(\begin{array}{c} \\ U \\ \\ \end{array}\right) \left(\begin{array}{c} \\ R \\ \\ \end{array}\right). \tag{15}$$

The length of our succinct representation is $O(m+n)$. Note that if C and R are not explicitly needed, the length of the representation is a constant $O(1)$; this is because U is of constant size and only a constant number of bits are needed to specify which columns and rows of A are kept (along with their associated rescaling factors) in the construction of C and R, respectively.

Before proving the theorem, we would like to give some intuition as to why, if $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$ satisfy certain conditions, the product CUR, as computed from the LinearTimeCUR algorithm, then becomes a good approximation to A in the sense of requirements (iv) and (v) of section 1. Let $h^t = Cy^t/\sigma_t(C)$ be the left singular vectors of C. Thus, if $H_k = (h^1\ h^2 \dots h^k) \in \mathbb{R}^{m\times k}$, then $H_kH_k^TA$ is the projection of A onto the subspace spanned by the top k left singular vectors of C. If the column sampling probabilities $\{q_j\}_{j=1}^n$ satisfy certain conditions, then the top k of the h^t's are approximations to the top k left singular vectors of A in the sense that their projection can be shown to "capture" *almost* as much of A as the projection of A onto the space spanned by its own top k left singular vectors. Indeed, the content of the SVD results of [11] (which in turn depend on the matrix multiplication results of [10]) is that if the column sampling probabilities $\{q_j\}_{j=1}^n$ are chosen judiciously, then the error in $\|A - H_kH_k^TA\|_\xi$ beyond the error for the best rank-k approximation can be made arbitrarily small both in expectation and with high probability.

Although $H_kH_k^TA$ is an approximation to A, one might wonder whether the approximation $H_kH_k^TA$ can be approximated so as to satisfy properties (i)–(v). Indeed, this is exactly what the CUR decomposition does! Using our sampling matrix formalism, let us define

$$\widetilde{H_k^T} = H_k^T(D_RS_R)^T \qquad \text{and} \qquad \widetilde{A} = D_RS_RA \tag{16}$$

to be the column-sampled and rescaled version of H_k^T and row-sampled and rescaled version of A, respectively. (We will see that $\widetilde{H_k^T}\widetilde{A} \approx H_k^TA$ by Theorem 1.) Lemma 1 states that

$$\begin{aligned} CUR &= H_kH_k^T(D_RS_R)^TD_RS_RA \\ &= H_k\widetilde{H_k^T}\widetilde{A}. \end{aligned} \tag{17}$$

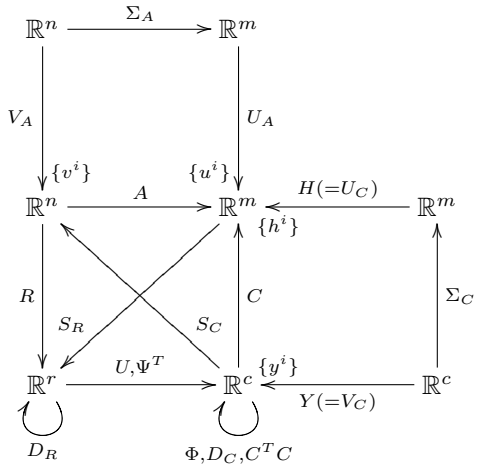

FIG. 3. *Another diagram for the* LINEARTIMECUR *algorithm.*

Thus, in order to provide a bound for $\|A - CUR\|_\xi$ for $\xi = 2, F$ we can first note that by the triangle inequality

$$\|A - CUR\|_\xi \leq \left\|A - H_k H_k^T A\right\|_\xi + \left\|H_k H_k^T A - CUR\right\|_\xi, \tag{18}$$

and then we can bound the two terms separately. The first term in (18) can be bounded using the SVD results of [11] if the column sampling probabilities satisfy certain conditions; in particular we will require that they be the optimal probabilities. Since $H_k^T H_k = I_k$, Lemma 2 states that

$$\begin{aligned} \left\|H_k H_k^T A - CUR\right\|_F &= \left\|H_k^T A - H_k^T (D_R S_R)^T D_R S_R A\right\|_F \\ &= \left\|H_k^T A - \widetilde{H_k^T} \widetilde{A}\right\|_F. \end{aligned} \tag{19}$$

Thus, the second term in (18) can be bounded by the matrix multiplication results of [10]; it will follow that if the sampling probabilities $\{p_i\}_{i=1}^m$ satisfy certain conditions, then $\widetilde{H_k^T} \widetilde{A} \approx H_k^T A$ in the sense that the error in $\|H_k^T A - \widetilde{H_k^T} \widetilde{A}\|_F$ can be bounded. Note that since the optimal probabilities depend on both H^T and A, and since we do not have access to H^T, our probabilities will not be optimal; nevertheless, although we will not obtain bounds with very high probability, as in [10] and [11], we will be able to apply Markov's inequality and thus achieve the bounds we desire.

A diagram illustrating the method (just described) that will be used to prove the correctness of the CUR algorithm is presented in Figure 3. In this figure, the locations of $\mathbb{R}^c$ and $\mathbb{R}^r$ and thus the directions of R, S_R, C, S_C, and U have been switched relative to their location in Figure 2. This presentation has several advantages: first, the SVD of C and the SVD of A can both be presented in the same figure as the CUR decomposition of A; second, one can see that bounding $\|A - H_k H_k^T A\|_\xi$ well in terms of $\|AA^T - CC^T\|_\xi$ depends on the probabilities used to sample the columns of A; and third, one can also see that bounding $\|H_k^T A - \widetilde{H_k^T} \widetilde{A}\|_F$ well depends on the probabilities used to sample the (columns of H_k^T and the corresponding) rows of A. See the corresponding figures in [10] and [11] for a comparison.

3.2. Analysis of the implementation and running time. In the LINEARTIMECUR algorithm the sampling probabilities $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$ (if they are chosen to be of the form used in Theorems 4 and 5) can be computed in one pass and $O(c+r)$ additional space and time using the SELECT algorithm of [10]. Given the elements to be sampled, the matrix C can then be constructed in one additional pass; this requires additional space and time that is $O(mc)$. Similarly, the matrix R can then be constructed in the same pass using additional space and time that is $O(nr)$. Given $C \in \mathbb{R}^{m\times c}$, computing C^TC requires $O(mc)$ additional space and $O(mc^2)$ additional time, and computing the SVD of C^TC requires $O(c^3)$ additional time. The matrix Ψ can be computed in the same second pass by sampling the same r rows of C that were used to construct R from A; this requires additional space and time that is $O(cr)$. The matrix Φ can be explicitly computed using $O(c^2k)$ additional time, and then the matrix $U = \Phi\Psi^T$ can be computed using $O(c^2r)$ additional time. Thus, since c, r, and k are assumed to be a constant, overall $O(m+n)$ additional space and time are required by the LINEARTIMECUR algorithm, and requirements (i)–(iii) of section 1 are satisfied. Note that the "description" of the solution that is computable in the allotted additional space and time is the explicit matrices C, U, and R.

3.3. Analysis of the sampling step. Before stating and proving the main theorem of this section, we will first prove two useful lemmas. Lemma 1 will establish (17) and Lemma 2 will establish (19).

LEMMA 1.

$$CUR = H_k\widetilde{H_k^T}\widetilde{A}.$$

Proof. Note that the SVD of C is $C = \sum_{t=1}^c \sigma_t(C)h^t y^{t^T}$, that the matrix $\Psi = D_RS_RC$, and that $U = \Phi\Psi^T$, where Φ is given by (14). Thus, we have that

$$\begin{aligned} CUR &= C\left(\sum_{t=1}^k \frac{1}{\sigma_t^2(C)} y^t y^{t^T}\right) C^T (D_RS_R)^T R \\ &= \left(\sum_{t_1} \sigma_{t_1}(C) h^{t_1} y^{t_1 T}\right)\left(\sum_{t_2=1}^k \frac{1}{\sigma_{t_2}^2(C)} y^{t_2} y^{t_2 T}\right)\left(\sum_{t_3} \sigma_{t_3}(C) y^{t_3} h^{t_3 T}\right)(D_RS_R)^T R \\ &= \left(\sum_{t=1}^k h^t h^{t^T}\right)(D_RS_R)^T R. \end{aligned}$$

The lemma follows since $\sum_{t=1}^k h^t h^{t^T} = H_kH_k^T$, since $R = D_RS_RA$, and from the definitions (16). ∎

LEMMA 2.

$$\left\|H_kH_k^TA - CUR\right\|_F = \left\|H_k^TA - \widetilde{H_k^T}\widetilde{A}\right\|_F.$$

Proof. From Lemma 1 we have that $CUR = H_k\widetilde{H_k^T}\widetilde{A}$. Let us define the matrix $\Omega \in \mathbb{R}^{k\times n}$ as

$$\Omega = H_k^TA - H_k^T(D_RS_R)^TD_RS_RA = H_k^TA - \widetilde{H_k^T}\widetilde{A}.$$

In addition, note that

$$\left\|H_kH_k^TA - H_k\widetilde{H_k^T}\widetilde{A}\right\|_F^2 = \|H_k\Omega\|_F^2 = \mathbf{Tr}\left(\Omega^TH_k^TH_k\Omega\right).$$

The lemma follows since $H_k^TH_k = I_k$ and since $\mathbf{Tr}\left(\Omega^T\Omega\right) = \|\Omega\|_F^2$. ∎

Here is our main theorem regarding the LINEARTIMECUR algorithm described in section 3.1. Note that in this theorem we restrict ourselves to sampling probabilities that are optimal in the sense of section 2.3.

THEOREM 4. *Suppose $A \in \mathbb{R}^{m\times n}$, and let C, U, and R be constructed from the* LINEARTIMECUR *algorithm by sampling c columns of A with probabilities $\{q_j\}_{j=1}^n$ and r rows of A with probabilities $\{p_i\}_{i=1}^m$. Assume that $p_i = |A_{(i)}|^2 / \|A\|_F^2$ and $q_j = |A^{(j)}|^2 / \|A\|_F^2$. Then*

$$\mathbf{E}\left[\,\|A - CUR\|_F\right] \le \|A - A_k\|_F + \left(\left(\frac{4k}{c}\right)^{1/4} + \left(\frac{k}{r}\right)^{1/2}\right)\|A\|_F\,, \tag{20}$$

$$\mathbf{E}\left[\,\|A - CUR\|_2\right] \le \|A - A_k\|_2 + \left(\left(\frac{4}{c}\right)^{1/4} + \left(\frac{k}{r}\right)^{1/2}\right)\|A\|_F\,. \tag{21}$$

In addition, if we let $\eta_c = 1 + \sqrt{8\log(1/\delta_c)}$ and let $\delta = \delta_r + \delta_c$, then with probability at least $1-\delta$,

$$\|A - CUR\|_F \le \|A - A_k\|_F + \left(\left(\frac{4k\eta_c^2}{c}\right)^{1/4} + \left(\frac{k}{\delta_r^2 r}\right)^{1/2}\right)\|A\|_F\,, \tag{22}$$

$$\|A - CUR\|_2 \le \|A - A_k\|_2 + \left(\left(\frac{4\eta_c^2}{c}\right)^{1/4} + \left(\frac{k}{\delta_r^2 r}\right)^{1/2}\right)\|A\|_F\,. \tag{23}$$

Proof. By the triangle inequality we have that

$$\|A - CUR\|_\xi \le \left\|A - H_k H_k^T A\right\|_\xi + \left\|H_k H_k^T A - CUR\right\|_\xi \tag{24}$$

for both $\xi = 2, F$; thus by Lemmas 1 and 2 we have that

$$\|A - CUR\|_\xi \le \left\|A - H_k H_k^T A\right\|_\xi + \left\|H_k^T A - H_k^T (D_R S_R)^T D_R S_R A\right\|_F \tag{25}$$

$$= \left\|A - H_k H_k^T A\right\|_\xi + \left\|H_k^T A - \widetilde{H_k^T}\,\widetilde{A}\right\|_F \tag{26}$$

for both $\xi = 2, F$, where (26) follows from the definitions (16). Then, note that the column sampling satisfies the requirements for the LINEARTIMESVD algorithm of [11]. Thus, by Theorem 2 it follows from (25) that

$$\|A - CUR\|_F \le \|A - A_k\|_F + (4k)^{1/4}\left\|AA^T - CC^T\right\|_F^{1/2} + \left\|H_k^T A - \widetilde{H_k^T}\,\widetilde{A}\right\|_F, \tag{27}$$

$$\|A - CUR\|_2 \le \|A - A_k\|_2 + \sqrt{2}\left\|AA^T - CC^T\right\|_F^{1/2} + \left\|H_k^T A - \widetilde{H_k^T}\,\widetilde{A}\right\|_F. \tag{28}$$

Note that from the LINEARTIMECUR algorithm the column sampling probabilities are of the form (3) with $B = A^T$; thus, they are optimal and $\mathbf{E}\left[\|AA^T - CC^T\|_F\right] \le \frac{1}{\sqrt{c}}\|A\|_F^2$. In addition, although the row sampling probabilities are not optimal, they are of the form (6); thus, since $\|H_k^T\|_F = \sqrt{k}$ we have that

$$\mathbf{E}\left[\left\|H_k^T A - \widetilde{H_k^T}\,\widetilde{A}\right\|_F\right] \le \sqrt{\frac{k}{r}}\,\|A\|_F\,. \tag{29}$$

Thus, by taking expectations of (27) and (28), by using Jensen's inequality and Theorem 1, (20) and (21) follow.

To establish (22) and (23) first let the events $\mathcal{E}_\xi, \xi = c, r$ be defined as follows:

$$\mathcal{E}_c : \left\|AA^T - CC^T\right\|_F \leq \frac{\eta_c}{\sqrt{c}} \|A\|_F^2 ,$$

$$\mathcal{E}_r : \left\|H_k^T A - \widetilde{H_k^T}\widetilde{A}\right\|_F \leq \frac{1}{\delta_r}\sqrt{\frac{k}{r}} \|A\|_F .$$

Thus, from Theorem 1 we have that $\mathbf{Pr}[\mathcal{E}_c] \geq 1-\delta_c$. By applying Markov's inequality to $\left\|H_k^T A - \widetilde{H_k^T}\widetilde{A}\right\|_F$ and using (29) we see that

$$\mathbf{Pr}\left[\left\|H_k^T A - \widetilde{H_k^T}\widetilde{A}\right\|_F \geq \frac{1}{\delta_r}\sqrt{\frac{k}{r}} \|A\|_F\right] \leq \delta_r$$

and thus that $\mathbf{Pr}[\mathcal{E}_r] \geq 1 - \delta_r$. The theorem then follows from (27) and (28) by considering the event $\mathcal{E}_c \bigcap \mathcal{E}_r$. □

Note that in the proof of Theorem 4 (and similarly in that of Theorem 6 in section 4) $\|A - CUR\|_\xi$ is bounded by bounding each of the terms $\|A - H_k H_k^T A\|_\xi$ and $\|H_k H_k^T A - CUR\|_\xi$ independently. In order to bound $\|A - H_k H_k^T A\|_\xi$, the SVD results for arbitrary probabilities from [11] are used, and then it is noted that the column sampling probabilities used in the LINEARTIMECUR algorithm are optimal for bounding $\|AA^T - CC^T\|_F$. Then, independently, $\|H_k H_k^T A - CUR\|_\xi$ is bounded by using the matrix multiplication results of [10]. Since the probabilities that are used for the row sampling are not optimal with respect to bounding $\|H_k^T A - \widetilde{H_k^T}\widetilde{A}\|_F$, we do not obtain that bound with high probability. Due to the use of Markov's inequality, we must sample a number of rows that is $O(1/\delta)$, whereas we only need to sample a number of columns that is $O(\log(1/\delta))$.

As a corollary of Theorem 4 we have the following theorem. In this theorem, in addition to using sampling probabilities that are optimal in the sense of section 2.3, we choose sufficiently many columns and rows to ensure that the additional error is less than $\epsilon' \|A\|_F$.

THEOREM 5. *Suppose $A \in \mathbb{R}^{m\times n}$, and let C, U, and R be constructed from the* LINEARTIMECUR *algorithm by sampling c columns of A with probabilities $\{q_j\}_{j=1}^n$ and r rows of A with probabilities $\{p_i\}_{i=1}^m$. Assume that $p_i = |A_{(i)}|^2/\|A\|_F^2$ and $q_j = |A^{(j)}|^2/\|A\|_F^2$ and let $\epsilon, \epsilon' > 0$ with $\epsilon = \epsilon'/2$.*

If $c \geq 4k/\epsilon^4$ and $r \geq k/\epsilon^2$, then

$$\mathbf{E}[\|A - CUR\|_F] \leq \|A - A_k\|_F + \epsilon' \|A\|_F , \tag{30}$$

and if $c \geq 4/\epsilon^4$ and $r \geq k/\epsilon^2$, then

$$\mathbf{E}[\|A - CUR\|_2] \leq \|A - A_k\|_2 + \epsilon' \|A\|_F . \tag{31}$$

In addition, if we let $\eta_c = 1 + \sqrt{8\log(1/\delta_c)}$ and let $\delta = \delta_r + \delta_c$, and if $c \geq 4k\eta_c^2/\epsilon^4$ and $r \geq k/\delta_r^2\epsilon^2$, then with probability at least $1 - \delta$,

$$\|A - CUR\|_F \leq \|A - A_k\|_F + \epsilon' \|A\|_F , \tag{32}$$

and if $c \geq 4\eta_c^2/\epsilon^4$ and $r \geq k/\delta_r^2\epsilon^2$, then with probability at least $1 - \delta$,

$$\|A - CUR\|_2 \leq \|A - A_k\|_2 + \epsilon' \|A\|_F . \tag{33}$$

The results of Theorems 4 and 5 for both the Frobenius norm and the spectral norm hold for all k and are of particular interest when A is well approximated by a matrix of low rank since then one may choose $k = O(1)$ and obtain a good approximation. In addition, since $\|A - A_t\|_2 \leq \|A\|_F / \sqrt{t}$ for all $t = 1, 2, \ldots, r$, the bounds with respect to the spectral norm have the following interesting property: from (31) we can see that

$$\mathbf{E}\left[\, \|A - CUR\|_2 \right] \leq \left(1/\sqrt{k} + \epsilon'\right) \|A\|_F ,$$

and similarly for (33). Thus, under the assumptions of Theorem 5 if we choose $k = 1/\epsilon'^2$ and let $\epsilon'' = 2\epsilon'$, then we have that

$$\mathbf{E}\left[\, \|A - CUR\|_2 \right] \leq \epsilon'' \|A\|_F \tag{34}$$

and that

$$\|A - CUR\|_2 \leq \epsilon'' \|A\|_F \tag{35}$$

holds with probability at least $1 - \delta$.

4. The constant time $CURD$ decomposition.

4.1. The algorithm. The ConstantTimeCUR algorithm is very similar in spirit to the LinearTimeCUR algorithm; thus, we only highlight its main features with an emphasis on similarities and differences between the two algorithms. Given a matrix $A \in \mathbb{R}^{m \times n}$, we wish to compute a description of a succinctly described, easily computed matrix A' that is decomposable as $A' = C\tilde{U}R \in \mathbb{R}^{m \times n}$ and that satisfies requirements (i)–(v) of section 1, where the additional RAM space and time to compute $\tilde{U}$ is $O(1)$. The ConstantTimeCUR algorithm, which is presented in Figure 4, accomplishes this by forming a matrix $C \in \mathbb{R}^{m \times c}$ by rescaling a randomly chosen subset of c columns of A, forming a matrix $R \in \mathbb{R}^{r \times n}$ by rescaling a randomly chosen subset of r rows of A, and forming a matrix $\Psi \in \mathbb{R}^{r \times c}$ from C by choosing the same randomly chosen rows used to construct R from A and rescaling appropriately. Given C, the ConstantTimeCUR algorithm randomly chooses and rescales w rows of C to form a matrix $W \in \mathbb{R}^{w \times c}$ and then computes the top ℓ singular values, $\sigma_t^2(W)$, $t = 1, \ldots, \ell$, and the corresponding singular vectors, z^t, $t = 1, \ldots, \ell$, of $W^T W$. Note that these are also approximations to the (squares of the) singular values and the corresponding right singular vectors of C. Using these quantities, a matrix $\tilde{\Phi} \in \mathbb{R}^{c \times c}$ may be defined as

$$\tilde{\Phi} = \sum_{t=1}^{\ell} \frac{1}{\sigma_t^2(W)} z^t z^{t^T}, \tag{36}$$

from which $\tilde{U} \in \mathbb{R}^{c \times r}$ is constructed as $\tilde{U} = \tilde{\Phi}\Psi^T$. Note that in the constant additional space and time framework the actual matrices C and R are not explicitly computed; instead the constant-sized matrix $\tilde{U}$ is computed and only a constant number of bits are stored to specify which columns and rows of A are kept (along with their associated rescaling factors) in the construction of C and R, respectively. Thus, the length of the succinct representation of A is a constant.

Figure 2 of section 3 provides a diagram illustrating the action of LinearTimeCUR algorithm, but the diagram and associated discussion are also relevant for the ConstantTimeCUR algorithm. Figure 5 also provides a diagram illustrating the

ConstantTimeCUR Algorithm.

Input: $A \in \mathbb{R}^{m \times n}$, $r, c, k \in \mathbb{Z}^+$ such that $1 \leq r \leq m$, $1 \leq c \leq n$, and $1 \leq k \leq \min(r, c)$, $\{p_i\}_{i=1}^m$ such that $p_i \geq 0$ and $\sum_{i=1}^m p_i = 1$, and $\{q_j\}_{j=1}^n$ such that $q_j \geq 0$ and $\sum_{j=1}^n q_j = 1$.

Output: $\tilde{U} \in \mathbb{R}^{c \times r}$ and a "description" of $C \in \mathbb{R}^{m \times c}$ and $R \in \mathbb{R}^{r \times n}$.

1. For $t = 1$ to c,
   (a) Pick $j_t \in \{1, \ldots, n\}$ with $\mathbf{Pr}[j_t = \alpha] = q_\alpha$, and save $\{(j_t, q_{j_t}) : t = 1, \ldots, c\}$.
   (b) Set $C^{(t)} = A^{(j_t)}/\sqrt{cq_{j_t}}$. (Note that C is not explicitly constructed in RAM.)
2. Choose $\{\pi_i\}_{i=1}^m$ such that $\pi_i = \left|C_{(i)}\right|^2 / \|C\|_F^2$.
3. For $t = 1$ to w,
   (a) Pick $i_t \in 1, \ldots, m$ with $\mathbf{Pr}[i_t = \alpha] = \pi_\alpha$, $\alpha = 1, \ldots, m$.
   (b) Set $W_{(t)} = C_{(i_t)}/\sqrt{w\pi_{i_t}}$.
4. Compute $W^T W$ and its SVD; say $W^T W = \sum_{t=1}^c \sigma_t^2(W) z^t z^{t^T}$.
5. If a $\|\cdot\|_F$ bound is desired, set $\gamma = \epsilon/100k$,
   Else if a $\|\cdot\|_2$ bound is desired, set $\gamma = \epsilon/100$.
6. Let $\ell = \min\{k, \max\{t : \sigma_t^2(W) \geq \gamma \|W\|_F^2\}\}$.
7. Keep singular values $\{\sigma_t(W)\}_{t=1}^\ell$ and their corresponding singular vectors $\{z^t\}_{t=1}^\ell$.
8. For $t = 1$ to r,
   (a) Pick $i_t \in \{1, \ldots, m\}$ with $\mathbf{Pr}[i_t = \alpha] = p_\alpha$, and save $\{(i_t, p_{i_t}) : t = 1, \ldots, r\}$.
   (b) Set $R_{(t)} = A_{(i_t)}/\sqrt{rp_{i_t}}$. (Note that R is not explicitly constructed in RAM.)
   (c) Set $\Psi_{(t)} = C_{(i_t)}/\sqrt{rp_{i_t}}$.
9. Let $\tilde{\Phi} = \sum_{t=1}^\ell \frac{1}{\sigma_t^2(W)} z^t z^{t^T}$ and let $\tilde{U} = \tilde{\Phi}\Psi^T$.
10. Return $\tilde{U}$, c column labels $\{(j_t, q_{j_t}) : t = 1, \ldots, c\}$, and r row labels $\{(i_t, p_{i_t}) : t = 1, \ldots, r\}$.

Fig. 4. *The* ConstantTimeCUR *algorithm.*

action of the ConstantTimeCUR algorithm, and is the analogue for the constant time $C\tilde{U}R$ of Figure 3; Figure 5 also illustrates that the matrix Y $(= V_C)$ (of Figure 3) consisting of the top k right singular vectors of C is not exactly computed, but is instead approximated by the matrix Z $(= V_W)$, where Z is a matrix whose columns $Z^{(t)} = z^t$ consist of the right singular vectors of W. Thus, the matrix H_k consisting of the left singular vectors of C is not exactly computed but is only approximated by $\tilde{H}_\ell$, where $\tilde{H}_\ell^{(t)} = \tilde{h}^t = Cz^t/\sigma_t(W)$ for $t = 1, \ldots, \ell$. Since by construction it is still the case that $R = D_R S_R A$ (and also that $C = AS_C D_C$ and $\Psi = D_R S_R C$), where the sampling matrices and the diagonal rescaling matrices are defined as in section 3.1, it follows from Lemma 3 that

$$C\tilde{U}R = \tilde{H}_\ell \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A. \tag{37}$$

In the linear time case we had that $H_k^T H_k = I_k$ since the columns of H_k were k of the left singular vectors of C. This allowed us to prove Lemma 2, as described

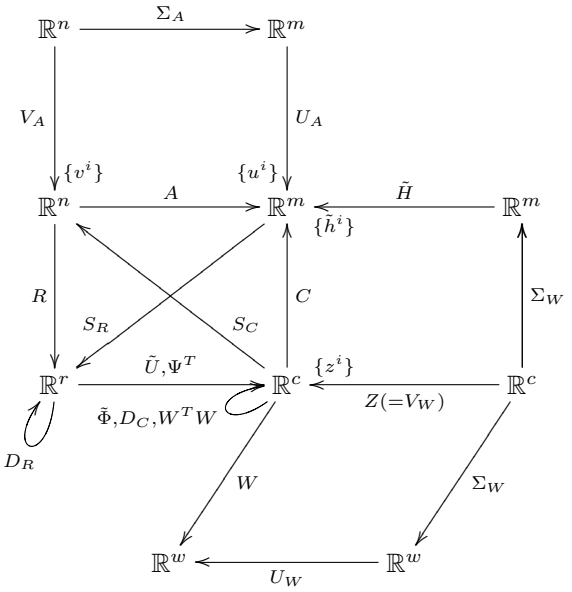

FIG. 5. *Diagram for the* CONSTANTTIMECUR *algorithm.*

in section 3.1. In the constant additional time setting, the columns of $\tilde{H}_\ell$ are only approximations to k of the left singular vectors of C, but we still have that $\tilde{H}_\ell^T \tilde{H}_\ell \approx I_\ell$. To quantify this, define $Z_{\alpha,\beta} \in \mathbb{R}^{c\times(\beta-\alpha+1)}$ to be the matrix whose columns are the αth through the βth singular vectors of W^TW and $T \in \mathbb{R}^{\ell\times\ell}$ to be the diagonal matrix with elements $T_{tt} = 1/\sigma_t(W)$. If we define the matrix $\Delta \in \mathbb{R}^{\ell\times\ell}$ to be

$$\Delta = TZ_{1,\ell}^T(C^TC - W^TW)Z_{1,\ell}T, \tag{38}$$

then Lemma 5 will establish that

$$\left\|\tilde{H}_\ell\tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \le (1 + \|\Delta\|_F^{1/2})\left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T(D_RS_R)^T D_RS_RA\right\|_F, \tag{39}$$

and Lemma 4 establishes that $\tilde{H}_\ell^T\tilde{H}_\ell = I_\ell + \Delta$. Thus, as in the linear time case, we can split

$$\left\|A - C\tilde{U}R\right\|_\xi \le \left\|A - \tilde{H}_\ell\tilde{H}_\ell^T A\right\|_\xi + \left\|\tilde{H}_\ell\tilde{H}_\ell^T A - C\tilde{U}R\right\|_\xi$$

for $\xi = 2, F$ and then bound the two terms separately. The first term can be bounded by the constant time SVD results of [11], the second term can be bounded by the matrix multiplication results of [10], and bounding the overall error depends on both results. Due to the two levels of sampling, the additional error will be larger than in the linear additional time framework, but it can be made arbitrarily small by choosing a constant number of rows and columns.

4.2. Analysis of the implementation and running time. In the CONSTANTTIMECUR algorithm the sampling probabilities $\{p_i\}_{i=1}^m$ and $\{q_j\}_{j=1}^n$ (if they are chosen to be of the form used in Theorem 6) can be computed in one pass and $O(c + r)$ additional space and time using the SELECT algorithm of [10]. Given the columns of A to be sampled, we do not explicitly construct the matrix C but instead

perform a second level of sampling and select w rows of C with probabilities $\{\pi_i\}_{i=1}^m$ in order to construct the matrix W; this requires a second pass and $O(w)$ additional space and time. Then, in a third pass we explicitly construct W; this requires additional space and time that is $O(cw)$. Similarly, a description of the matrix R can then be constructed in the same third pass using additional space and time that is $O(r)$. Then, given W, computing W^TW requires $O(c^2w)$ additional time and computing the SVD of W^TW requires $O(c^3)$ additional time. The matrix Ψ can be computed in the same third pass by sampling the same r rows of C that were used to construct R from A; this requires additional time that is $O(cr)$. The matrix $\tilde{\Phi}$ can be explicitly computed using $O(c^2k)$ additional time and then the matrix $U = \tilde{\Phi}\Psi^T$ can be computed using $O(c^2r)$ additional time. Thus, since c, r, and k are assumed to be constants, overall $O(1)$ additional space and time are required by the ConstantTimeCUR algorithm, and requirements (i)–(iii) of section 1 are satisfied. Note that the "description" of the solution that is computable in the allotted additional space and time is the matrix $\tilde{U}$, with the labels $i_1, \ldots, i_r$ and $j_1, \ldots, j_c$ indicating the rows chosen to construct C and R as well as the corresponding probabilities $\{p_{i_t}\}_{t=1}^r$ and $\{q_{j_t}\}_{t=1}^c$; we note that we need to know p_i only for the sampled rows i and q_j only for the sampled columns j.

4.3. Analysis of the sampling step. Before stating and proving the main theorem of this section, we will first prove several useful lemmas. First, in Lemma 3 we will establish (37).

Lemma 3.

$$C\tilde{U}R = \tilde{H}_\ell \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A.$$

Proof. Since $\tilde{h}^t = Cz^t/\sigma_t(W)$ for $t = 1, \ldots, \ell$, we have that $C = \sum_{t=1}^c \sigma_t(W)\tilde{h}^t z^{t^T}$. Thus, we have that

$$\begin{aligned} C\tilde{U}R &= C\left(\sum_{t=1}^{\ell} \frac{1}{\sigma_t^2(W)} z^t z^{t^T}\right) C^T (D_R S_R)^T R \\ &= \left(\sum_{t=1}^{\ell} \tilde{h}^t \tilde{h}^{t^T}\right) (D_R S_R)^T R. \end{aligned}$$

The lemma follows since $\sum_{t=1}^{\ell} \tilde{h}^t \tilde{h}^{t^T} = \tilde{H}_\ell \tilde{H}_\ell^T$ and since $R = D_R S_R A$. ∎

Next, Lemma 4 will characterize, in terms of Δ, the degree to which the columns of $\tilde{H}_\ell$ are not orthonormal. Note that it appeared in [11].

Lemma 4. *When written in the basis with respect to Z,*

$$\tilde{H}_\ell^T \tilde{H}_\ell = I_\ell + \Delta.$$

Furthermore, for $\xi = 2, F$,

$$\|\Delta\|_\xi \leq \frac{1}{\gamma \|W\|_F^2} \left\|C^T C - W^T W\right\|_\xi.$$

Proof. Recall that $\tilde{H}_\ell = CZ_{1,\ell}T$ and that $T^T Z_{1,\ell}^T W^T W Z_{1,\ell} T = I_\ell$, so that

$$\left\|\tilde{H}_\ell^T \tilde{H}_\ell - I_\ell\right\|_\xi = \left\|T^T Z_{1,\ell}^T C^T C Z_{1,\ell} T - T^T Z_{1,\ell}^T W^T W Z_{1,\ell} T\right\|_\xi \tag{40}$$

$$= \left\|T^T Z_{1,\ell}^T \left(C^T C - W^T W\right) Z_{1,\ell} T\right\|_\xi. \tag{41}$$

Using the submultiplicativity properties of the 2-norm, and in particular

$$\|AB\|_\xi \leq \|A\|_2 \|B\|_\xi , \tag{42}$$

$$\|AB\|_\xi \leq \|A\|_\xi \|B\|_2 , \tag{43}$$

for both $\xi = 2, F$, we get

$$\left\|\tilde{H}_\ell^T \tilde{H}_\ell - I_\ell\right\|_\xi \leq \left\|T^T Z_{1,\ell}^T\right\|_2 \left\|C^T C - W^T W\right\|_\xi \|Z_{1,\ell} T\|_2 \tag{44}$$

$$\leq \|T\|_2^2 \left\|C^T C - W^T W\right\|_\xi \tag{45}$$

$$\leq \max_{t=1,\ldots,\ell} \left(1/\sigma_t^2(W)\right) \left\|C^T C - W^T W\right\|_\xi , \tag{46}$$

since $\|Z_{1,\ell}\|_2 = 1$. The lemma follows since $\sigma_t^2(W) \geq \gamma \|W\|_F^2$ for all $t = 1, \ldots, \ell$ by the definition of ℓ. □

Next, in Lemma 5 we will establish (39).

LEMMA 5.

$$\left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \leq (1 + \|\Delta\|_F^{1/2}) \left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A\right\|_F .$$

Proof. From Lemma 3 we have that $C\tilde{U}R = \tilde{H}_\ell \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A$. Let us define the matrix $\Omega \in \mathbb{R}^{\ell \times n}$ as

$$\Omega = \tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A.$$

Thus, since $\mathbf{Tr}\left(XX^T\right) = \|X\|_F^2$ for a matrix X, we have

$$\begin{aligned} \left\|\tilde{H}_\ell \Omega\right\|_F^2 &= \mathbf{Tr}(\Omega^T \tilde{H}_\ell^T \tilde{H}_\ell \Omega) \\ &= \mathbf{Tr}(\Omega^T (I_\ell + \Delta)\Omega) \qquad (47) \\ &= \|\Omega\|_F^2 + \mathbf{Tr}(\Omega^T \Delta \Omega) \\ &\leq \|\Omega\|_F^2 + \|\Delta\|_2 \|\Omega\|_F^2 , \qquad (48) \end{aligned}$$

where (47) follows from Lemma 4 and (48) follows since $\left|\mathbf{Tr}(\Omega^T \Delta \Omega)\right| \leq \|\Delta\|_2 \mathbf{Tr}(\Omega^T \Omega)$. Thus,

$$\left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \leq (1 + \|\Delta\|_2)^{1/2} \left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A\right\|_F ,$$

and the lemma then follows. □

Finally, in Lemma 6 we show that $\|W\|_F = \|C\|_F = \|A\|_F$ when optimal probabilities are used. It also appeared in [11].

LEMMA 6. *Suppose $A \in \mathbb{R}^{m \times n}$, and run the* ConstantTimeCUR *algorithm by sampling c columns of A with probabilities $\{q_j\}_{j=1}^n$ (and then sampling w rows of C with probabilities $\{\pi_i\}_{i=1}^m$ to construct W) and r rows of A with probabilities $\{p_i\}_{i=1}^m$. Assume that $p_i = |A_{(i)}|^2 / \|A\|_F^2$ and $q_j = |A^{(j)}|^2 / \|A\|_F^2$. Then $\|W\|_F = \|C\|_F = \|A\|_F$.*

Proof. If $p_i = |A^{(i)}|^2 / \|A\|_F^2$, then we have that $\|C\|_F^2 = \sum_{t=1}^c |C^{(t)}|^2 = \sum_{t=1}^c \frac{|A^{(i_t)}|^2}{c p_{i_t}} = \|A\|_F^2$. Similarly, if $q_j = |C_{(j)}|^2 / \|C\|_F^2$, then we have that $\|W\|_F^2 = \sum_{t=1}^w \left|W_{(t)}\right|^2 = \sum_{t=1}^w \frac{|C_{(i_t)}|^2}{w q_{i_t}} = \|C\|_F^2$. The lemma follows. □

Here is our main theorem regarding the ConstantTimeCUR algorithm described in section 4.1. It is the constant time analogue of Theorem 5. Note that

in this theorem we restrict ourselves to sampling probabilities that are optimal in the sense of section 2.3, and to choosing sufficiently many columns and rows to ensure that the additional error is less than $\epsilon \|A\|_F$.

THEOREM 6. *Suppose $A \in \mathbb{R}^{m\times n}$, and let C, $\tilde{U}$, and R be constructed from the* CONSTANTTIMECUR *algorithm by sampling c columns of A with probabilities $\{q_j\}_{j=1}^n$ (and then sampling w rows of C with probabilities $\{\pi_i\}_{i=1}^m$ to construct W) and r rows of A with probabilities $\{p_i\}_{i=1}^m$. Assume that $p_i = |A_{(i)}|^2/\|A\|_F^2$ and $q_j = |A^{(j)}|^2/\|A\|_F^2$. Let $\eta = 1+\sqrt{8\log(3/\delta)}$ and $\epsilon > 0$.*

If a Frobenius norm bound is desired, and hence the CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100k$, then if we let $c = \Omega(k^2\eta^2/\epsilon^8)$, $w = \Omega(k^2\eta^2/\epsilon^8)$, and $r = \Omega(k/\delta^2\epsilon^2)$, then with probability at least $1-\delta$,*

$$\left\|A - C\tilde{U}R\right\|_F \leq \|A - A_k\|_F + \epsilon \|A\|_F . \tag{49}$$

If a spectral norm bound is desired, and hence the CONSTANTTIMESVD *algorithm is run with $\gamma = \epsilon/100$, then if we let $c = \Omega(\eta^2/\epsilon^8)$, $w = \Omega(\eta^2/\epsilon^8)$, and $r = \Omega(k/\delta^2\epsilon^2)$, then with probability at least $1-\delta$,*

$$\left\|A - C\tilde{U}R\right\|_2 \leq \|A - A_k\|_2 + \epsilon \|A\|_F . \tag{50}$$

Proof. Let us define the events:

$$\mathcal{E}_c : \left\|AA^T - CC^T\right\|_F \leq \frac{\eta}{\sqrt{c}} \|A\|_F^2 , \tag{51}$$

$$\mathcal{E}_w : \left\|C^TC - W^TW\right\|_F \leq \frac{\eta}{\sqrt{w}} \|A\|_F^2 , \tag{52}$$

$$\mathcal{E}_r : \left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A\right\|_F \leq \frac{3}{\delta\sqrt{r}} \left\|\tilde{H}_\ell\right\|_F \|A\|_F . \tag{53}$$

Under the assumptions of this theorem, event $\mathcal{E}_c$ holds with probability greater than $1-\delta/3$ by Theorem 1, and similarly for event $\mathcal{E}_w$. Next, we claim that $\mathcal{E}_r$ holds with probability greater than $1-\delta/3$. To prove this it suffices to prove that

$$\mathbf{E}\left[\left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A\right\|_F\right] \leq \frac{1}{\sqrt{r}} \left\|\tilde{H}_\ell^T\right\|_F \|A\|_F , \tag{54}$$

since the claim that $\mathbf{Pr}[\mathcal{E}_r] \geq 1-\delta/3$ follows immediately from (54) by Markov's inequality; but (54) follows from Theorem 1 since the probabilities $\{p_i\}_{i=1}^m$ used to sample the columns of $\tilde{H}_\ell$ and the corresponding rows of A are of the form (6). Thus, under the assumptions of the theorem,

$$\mathbf{Pr}[\mathcal{E}_\xi] \geq 1-\delta/3 \quad \text{for } \xi = c, w, r.$$

Next, from Lemma 4 and the Cauchy–Schwarz inequality, it follows that

$$\left\|\tilde{H}_\ell\right\|_F^2 = \sum_{t=1}^{\ell} \left|\tilde{h}^{t^T} \tilde{h}^t\right| = \sum_{t=1}^{\ell} 1 + \Delta_{tt} \leq k + \sqrt{k} \|\Delta\|_F . \tag{55}$$

Since $\sqrt{1+x} \leq 1 + \sqrt{x}$ for $x \geq 0$ it follows from (55) and (53) that under the event $\mathcal{E}_r$ we have

$$\left\|\tilde{H}_\ell^T A - \tilde{H}_\ell^T (D_R S_R)^T D_R S_R A\right\|_F \leq \frac{3}{\delta\sqrt{r}} \left(\sqrt{k} + k^{1/4} \|\Delta\|_F^{1/2}\right) \|A\|_F . \tag{56}$$

By combining (56) with Lemma 5 we have that

$$\begin{aligned}
\left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F &\leq \frac{3}{\delta\sqrt{r}} \left(1 + \|\Delta\|_F^{1/2}\right) \left(\sqrt{k} + k^{1/4} \|\Delta\|_F^{1/2}\right) \|A\|_F \\
&\leq \left(\frac{9k}{\delta^2 r}\right)^{1/2} \left(1 + \|\Delta\|_F^{1/2}\right)^2 \|A\|_F \qquad (57) \\
&\leq \left(\frac{9k}{\delta^2 r}\right)^{1/2} \left(1 + 3\|\Delta\|_F^{1/2}\right) \|A\|_F \qquad (58) \\
&\leq \left(\frac{9k}{\delta^2 r}\right)^{1/2} \left(1 + \frac{3\left\|C^T C - W^T W\right\|_F^{1/2}}{\sqrt{\gamma}\|W\|_F}\right) \|A\|_F , \qquad (59)
\end{aligned}$$

where (57) follows since $k^{1/4} \leq \sqrt{k}$, (58) follows by multiplying out terms and since $\|\Delta\|_F \leq 1$ under the assumptions of the theorem, and (59) follows from the bound on $\|\Delta\|_F$ in Lemma 4.

Let us first consider establishing the Frobenius norm bound of (49). Recall that in this case we have set $\gamma = \epsilon/100k$. We will use the triangle inequality to get

$$\left\|A - C\tilde{U}R\right\|_F \leq \left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_F + \left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \tag{60}$$

and will bound each term separately. First, using the probabilities $\{q_j\}_{j=1}^n$ and the values of $c, w = \Omega(k^2\eta^2/\epsilon^8)$, then under the event $\mathcal{E}_c \bigcap \mathcal{E}_w$ we have that

$$\left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_F \leq \|A - A_k\|_F + \frac{\epsilon}{2}\|A\|_F \tag{61}$$

by Theorem 3; see also [11]. In addition, using the sampling probabilities $\{p_i\}_{i=1}^m$ and values of $r = \Omega(k/\delta^2\epsilon^2)$ and $w = \Omega(k^2\eta^2/\epsilon^8)$, and noting Lemma 6, it follows from (59) that under the event $\mathcal{E}_r \bigcap \mathcal{E}_w$,

$$\left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \leq \frac{\epsilon}{2}\|A\|_F . \tag{62}$$

Thus, under the event $\mathcal{E}_c \bigcap \mathcal{E}_w \bigcap \mathcal{E}_r$, which has probability at least $1-\delta$, by combining (61) and (62) we see that (49) follows.

Let us next consider establishing the spectral norm bound of (50). Recall that in this case we have set $\gamma = \epsilon/100$ and that

$$\left\|A - C\tilde{U}R\right\|_2 \leq \left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_2 + \left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \tag{63}$$

by the submultiplicitivity of $\|\cdot\|_2$ and since $\|\cdot\|_2 \leq \|\cdot\|_F$. First, using the probabilities $\{q_j\}_{j=1}^n$ and the values of $c, w = \Omega(\eta^2/\epsilon^8)$, under the event $\mathcal{E}_c \bigcap \mathcal{E}_w$ we have that

$$\left\|A - \tilde{H}_\ell \tilde{H}_\ell^T A\right\|_2 \leq \|A - A_k\|_2 + \frac{\epsilon}{2}\|A\|_F \tag{64}$$

by Theorem 3; see also [11]. In addition, using the sampling probabilities $\{p_i\}_{i=1}^m$ and values of $r = \Omega(k/\delta^2\epsilon^2)$ and $w = \Omega(\eta^2/\epsilon^8)$, and noting Lemma 6, it follows from (59) that under the event $\mathcal{E}_r \bigcap \mathcal{E}_w$,

$$\left\|\tilde{H}_\ell \tilde{H}_\ell^T A - C\tilde{U}R\right\|_F \leq \frac{\epsilon}{2} \|A\|_F . \tag{65}$$

Thus, under the event $\mathcal{E}_c \bigcap \mathcal{E}_w \bigcap \mathcal{E}_r$, which has probability at least $1-\delta$, by combining (64) and (65) we see that (50) follows. □

As in the linear additional time framework, the results of Theorem 6 hold for all k and are of particular interest when A is well approximated by a matrix of low rank since then one may choose $k = O(1)$ and obtain a good approximation. In addition, it follows from (50) that

$$\left\|A - C\tilde{U}R\right\|_2 \leq (1/\sqrt{k} + \epsilon') \|A\|_F .$$

Thus, under the assumptions of Theorem 6 if we choose $k = 1/\epsilon'^2$ and let $\epsilon'' = 2\epsilon'$, then we have that

$$\|A - CUR\|_2 \leq \epsilon'' \|A\|_F \tag{66}$$

holds with probability at least $1 - \delta$.

We note the following lemma. This result is not needed in this paper, but is included for future reference [13].

LEMMA 7.

$$\left\|\tilde{U}\right\|_2 \leq \frac{O(1)}{\gamma \|A\|_F}.$$

Proof. First note that $\|\tilde{U}\|_2 = \|\tilde{\Phi}\Psi^T\|_2 \leq \|\tilde{\Phi}\|_2 \|\Psi^T\|_2$. Since $\tilde{\Phi} = \sum_{t=1}^{\ell} \frac{1}{\sigma_t^2(W)} z^t z^{tT}$, we see that $\|\tilde{\Phi}\|_2 = \frac{1}{\sigma_\ell^2(W)} \leq \frac{1}{\gamma \|W\|_F^2}$. The lemma follows since, when using the probabilities of Theorem 6, we have that $\|\Psi^T\|_F \leq O(1)\|A\|_F$ and that $1/\|W\|_F^2 \leq O(1)/\|A\|_F^2$. □

5. Discussion and conclusion. To put the CUR decomposition into context, it will be useful to contrast it with the SVD. The SVD of A expresses A as $A = \sum_{t=1}^{\rho} \sigma_t(A) u^t v^{tT}$. Keeping the first k terms of this expansion, i.e., keeping $A_k = \sum_{t=1}^{k} \sigma_t(A) u^t v^{tT} = U_k \Sigma_k V_k^T$, gives us the "optimal" rank-k approximation to A with respect to both the spectral norm and the Frobenius norm [19, 22]. Thus, computing the SVD gives us a good succinct approximation, since it only takes space $O(k(m+n))$ to write down U_k, Σ_k, V_k. However, the computational problem of finding the SVD cannot be carried out in a small constant number of passes. Our theorems say that weaker bounds, which are similar in spirit, may be achieved by CUR. Note, however, that although the SVD may be thought of as a rotation followed by a rescaling followed by a rotation, the CUR decomposition is quite different; both C and R perform an action like A and thus U must involve a pseudoinverse-like operation. Note also that the last upper bound, i.e., (v), is much smaller than $\|A\|_F$ when A has a good low-rank approximation. This is indeed the case for matrices occurring in many contexts, such as matrices for which principal component analysis is used.

Recent work has focused on developing new techniques for proving lower bounds on the number of queries a sampling algorithm is required to perform in order to

approximate a given function accurately with a low probability or error [3, 4]. In [4] these methods have been applied to the low-rank matrix approximation problem and to the matrix reconstruction problem. In the latter problem, the input is a matrix $A \in \mathbb{R}^{m \times n}$ and the goal is to find a matrix B that is close to A. In [4] it is shown that finding a B such that $\|A - B\|_F \leq \epsilon \|A\|_F$ requires $\Omega(mn)$ queries and that finding a B such that $\|A - B\|_2 \leq \epsilon \|A\|_F$ requires $\Omega(m + n)$ queries. Thus, our algorithm is optimal for constant ϵ.

During the time since this manuscript was submitted for journal publication we have become aware of other low-rank matrix decompositions of the form $A \approx CUR$, where C is a matrix consisting of a small number of columns of A, R is a matrix consisting of a small number of rows of A, and U is an appropriately defined low-dimensional matrix. Work by Stewart [26, 27, 5] and independently work by Goreinov, Tyrtyshnikov, and Zamarashkin [21, 20] have considered matrix decompositions with structural (but not algorithmic) properties quite similar to the CUR decompositions we have considered in this paper. Drineas and Mahoney have extended the CUR decompositions of this paper to kernel-based statistical learning [15, 16] and large tensor-based data [17]. These latter papers also contain a discussion of the relationship between the work presented in this paper and the related work of Stewart and Goreinov, Tyrtyshnikov, and Zamarashkin.

Acknowledgments. We would like to thank Dimitris Achlioptas for fruitful discussions and the National Science Foundation for partial support of this work. In addition, we would like to thank an anonymous reviewer for carefully reading the paper and making numerous useful suggestions; in particular, the reviewer provided an elegant, short proof for Lemma 4.

REFERENCES

[1] D. Achlioptas and F. McSherry, *Fast computation of low rank matrix approximations*, J. ACM, to appear.

[2] D. Achlioptas and F. McSherry, *Fast computation of low rank matrix approximations*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 611–618.

[3] Z. Bar-Yossef, *The Complexity of Massive Data Set Computations*, Ph.D. thesis, University of California, Berkeley, 2002.

[4] Z. Bar-Yossef, *Sampling lower bounds via information theory*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 335–344.

[5] M. W. Berry, S. A. Pulatova, and G. W. Stewart, *Computing Sparse Reduced-Rank Approximations to Sparse Matrices*, Tech. Report UMIACS TR-2004-32 CMSC TR-4589, University of Maryland, College Park, MD, 2004.

[6] R. Bhatia, *Matrix Analysis*, Springer-Verlag, New York, 1997.

[7] E. Cohen and D. D. Lewis, *Approximating matrix multiplication for pattern recognition tasks*, J. Algorithms, 30 (1999), pp. 211–252.

[8] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, *Clustering in large graphs and matrices*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 1999, pp. 291–299.

[9] P. Drineas and R. Kannan, *Pass efficient algorithms for approximating large matrices*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2003, pp. 223–232.

[10] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* I: *Approximating matrix multiplication*, SIAM J. Comput., 36 (2006), pp. 132–157.

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

[12] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo Algorithms for Matrices* III: *Computing a Compressed Approximate Matrix Decomposition*, Tech. Report

YALEU/DCS/TR-1271, Department of Computer Science, Yale University, New Haven, CT, 2004.

[13] P. Drineas, R. Kannan, and M. W. Mahoney, *Sampling Sub-problems of Heterogeneous Max-Cut Problems and Approximation Algorithms*, Tech. Report YALEU/DCS/TR-1283, Department of Computer Science, Yale University, New Haven, CT, 2004.

[14] P. Drineas, I. Kerenidis, and P. Raghavan, *Competitive recommendation systems*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 82–90.

[15] P. Drineas and M. W. Mahoney, *Approximating a Gram matrix for improved kernel-based learning*, in Proceedings of the 18th Annual Conference on Learning Theory, 2005, pp. 323–337.

[16] P. Drineas and M. W. Mahoney, *On the Nyström method for approximating a Gram matrix for improved kernel-based learning*, J. Machine Learning, 6 (2005), pp. 2153–2175.

[17] P. Drineas and M. W. Mahoney, *A Randomized Algorithm for a Tensor-Based Generalization of the Singular Value Decomposition*, Tech. Report YALEU/DCS/TR-1327, Department of Computer Science, Yale University, New Haven, CT, 2005.

[18] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 370–378.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.

[20] S. A. Goreinov and E. E. Tyrtyshnikov, *The maximum-volume concept in approximation by low-rank matrices*, Contemp. Math., 280 (2001), pp. 47–51.

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

[22] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.

[23] P. Indyk, *Stable distributions, pseudorandom generators, embeddings and data stream computation*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 189–197.

[24] J. Kleinberg, *Two algorithms for nearest-neighbor search in high dimensions*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 599–608.

[25] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate Analysis*, Academic Press, London, 1979.

[26] G. W. Stewart, *Four algorithms for the efficient computation of truncated QR approximations to a sparse matrix*, Numer. Math., 83 (1999), pp. 313–323.

[27] G. W. Stewart, *Error Analysis of the Quasi-Gram-Schmidt Algorithm*, Tech. Report UMIACS TR-2004-17 CMSC TR-4572, University of Maryland, College Park, MD, 2004.

[28] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990.

[29] S. Vempala, *Random projection: A new approach to VLSI layout*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 389–395.