A Relaxed Algorithm for Online Matrix Inversion

Arne Storjohann
astorjoh@uwaterloo.ca
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
N2L 3G1

Shiyun Yang
shiyyang@amazon.com
Amazon Canada Fullfillment Services Inc
1071 Mainland St, Vancouver, BC, Canada
V6B 5P9

ABSTRACT

We consider a variation of the well known problem of computing the unique solution to a nonsingular system $Ax = b$ of n linear equations over a field K. The variation assumes that A has generic rank profile and requires as output not only the single solution vector $A^{-1}b \in \mathsf{K}^{n\times 1}$, but rather the solution to all leading principle subsystems. Most importantly, the rows of the augmented system $\left[\begin{array}{c|c} A & b \end{array}\right]$ are given one at a time from first to last, and as soon as the next row is given the solution to the next leading principal subsystem should be produced. We call this problem OnlineSystem. The obvious iterative algorithm for OnlineSystem has a cost in terms of field operations that is cubic in the dimension of A. In this paper we introduce a relaxed representation for the inverse and show how to obtain an algorithm for OnlineSystem that allows us to incorporate matrix multiplication. As an application we show how to introduce fast matrix multiplication into the inherently iterative algorithm for row rank profile computation presented previously by the authors.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*algebraic algorithms, analysis of algorithms*; G.4 [**Mathematical Software**]: Algorithm Design and Analysis; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations in finite fields, computations on matrices*

General Terms

Algorithms

Keywords

Linear system; finite field; relaxed algorithm; rank profile

1. INTRODUCTION

Consider the well known problem of computing the solution to a nonsingular system of linear equations over a finite field K. The problem takes as input a nonsingular matrix $A \in \mathsf{K}^{n\times n}$ together with a right hand side vector $b \in \mathsf{K}^{n\times 1}$, and requires as output the solution $x := A^{-1}b \in \mathsf{K}^{n\times 1}$ to the linear system $Ax = b$. For an unstructured and dense input matrix A, the fastest known solutions (in terms of asymptotic complexity [1, 8] and in practice [2, 4, 5]) all reduce the problem to matrix multiplication by computing a decomposition of A or the inverse of A as the product of structured matrices, typically upper / lower triangular and permutation matrices (see [9] for a survey).

In this paper we consider an online version of the nonsingular linear system solving problem. Consider the augmented linear system $\left[\begin{array}{c|c} A & b \end{array}\right] \in \mathsf{K}^{n\times(n+1)}$. We can decompose this augmented system for $s = 1, 2, \ldots, n$ as

$$\left[\begin{array}{c|c} A & b \end{array}\right] = \left[\begin{array}{c|c|c} A_s & * & b_s \\ \hline * & * & * \end{array}\right] \in \mathsf{K}^{n\times(n+1)},$$

where $A_s \in \mathsf{K}^{s\times s}$ is the principal leading $s\times s$ submatrix of A and $b_s \in \mathsf{K}^{s\times 1}$ is the first s entries of b. Moreover, we assume that A has generic rank profile, that is, all of $A_1, A_2, \ldots, A_n$ are nonsingular. In this paper we show how to compute the sequence of solutions $A_1^{-1}b_1, A_2^{-1}b_2, \ldots, A_n^{-1}b_n$ in an *online* fashion in time $O(n^\omega)$, where $2 < \omega \leq 3$ is the exponent of matrix multiplication. By online we mean that the rows of the augmented system are given one at a time, from first to last: as soon as row s is given, the solution to $A_s^{-1}b_s$ should be produced. We call this problem OnlineSystem. Online algorithms for computer algebra were popularized by van der Hoeven with the introduction of online algorithms for formal power series multiplication [7,13]. Recently, an online algorithm for polynomial matrix order basis computation has been proposed [6].

Our interest in studying problem OnlineSystem is motivated by the following two problems that take as input an $A \in \mathsf{K}^{n\times m}$ of arbitrary rank.

- MaxIndependentRowSet: Compute the rank r and a list of r row indices such that these rows are linearly independent.
- RowRankProfile: Compute the rank r and the lexicographically minimal list $[i_1, i_2, \ldots, i_r]$ of row indices of A such that these rows are linearly independent.

In a surprising result, Cheung, Kwok and Lau [3] give a Monte Carlo algorithm for MaxIndependentRowSet with running time only $(r^\omega + n + m + |A|)^{1+o(1)}$ field operations in K. Here, $|A|$ denotes the number of nonzero entries of A and ω is the exponent for matrix multiplication. The following year, using an alternative technique [12], a Monte

ISSAC'15, July 6–9, 2015, Bath, United Kingdom.
Copyright © 2015 ACM 978-1-4503-3435-8/15/07 ...$15.00.
DOI: http://dx.doi.org/10.1145/2755996.2756672.

Carlo algorithm for RowRankProfile was presented that has running time $(r^3 + n + m + |A|)^{1+o(1)}$ operations in K. The algorithm for RowRankProfile is inherently iterative: the indices $i_1, i_2, \ldots, i_r$ are computed in succession, and each iteration requires the solution of the next principal subsystem of a linear system comprised of rank profile rows of A found so far. The main bottleneck to introduce matrix multiplication into the algorithm for RowRankProfile is to find a solution to problem OnlineSystem that allows the use of fast matrix multiplication; this paper presents such an algorithm.

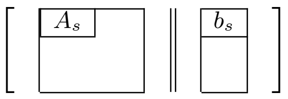
The rest of the paper is organized as follows. In Section 2 we show how problem OnlineSystem can be reduced to problem OnlineInverse: computing a representation of A_s^{-1} as the product of $2s$ structured matrices, with the decomposition of A_s^{-1} being produced as soon as row s of A is given, $s = 1, 2, \ldots, n$. Both problem OnlineSystem and OnlineInverse are defined precisely in Section 2. In Section 3 we recall the obvious iterative algorithm for OnlineInverse that has cost $O(n^3)$. In Section 4 we propose a relaxed representation of the inverse of a matrix. Instead of representing A_s^{-1} as the product of $2s$ structured matrices, we define a compressed representation of A_s^{-1} as the product of $O(\log s)$ structured matrices. Finally, Section 5 designs the algorithm for solving problem OnlineInverse in time $O(n^\omega)$. Our analysis assumes $\omega > 2$. Section 6 shows how to incorporate the fast algorithm for OnlineSystem to obtain a fast algorithm for rank profile computation. Section 7 concludes.

Following [12], throughout the paper we use the following notation. For a list $\mathcal{P} = [i_1, i_2, \ldots, i_k]$ of distinct row indices and $\mathcal{Q} = [j_1, j_2, \ldots, j_\ell]$ of distinct column indices, we write $A^{\mathcal{P}}$ to denote the submatrix of A consisting of rows $\mathcal{P}$, $A_{\mathcal{Q}}$ to denote the submatrix consisting of columns $\mathcal{Q}$, and $A_{\mathcal{Q}}^{\mathcal{P}}$ to denote the submatrix consisting of the intersection of rows $\mathcal{P}$ and columns $\mathcal{Q}$ of A.

2. INVERSE DECOMPOSITION

Let $A \in \mathsf{K}^{n\times n}$ and $b \in \mathsf{K}^{n\times 1}$ be given. Let A_s denote the leading $s \times s$ submatrix of A and $b_s \in \mathsf{K}^{s\times 1}$ be the vector containing the first s elements of b as shown in the following augmented system.

$$\left[\ A \parallel b\ \right] =$$



Assume A has generic rank profile, that is, A_s is nonsingular for $1 \le s \le n$. In this section we consider the following problem.

- OnlineSystem: Let $A \in \mathsf{K}^{n\times n}$ with generic rank profile and $b \in \mathsf{K}^{n\times 1}$ be given. Suppose the rows of the augmented system $\left[\ A \parallel b\ \right]$ are given one at a time, from first to last. As soon as rows $1, 2, \ldots, s$ of $\left[\ A \parallel b\ \right]$ are given, produce the subsystem solution $A_s^{-1} b_s$, for $s = 1, 2, \ldots, n$.

Because A is assumed to have generic rank profile, Gaussian elimination without pivoting produces a unique decomposition $A^{-1} = P_n \cdot P_{n-1} \cdots P_1$, where $P_s = R_s \cdot L_s$ can be represented as the product of the pair of structured matrices R_s and L_s, $1 \le s \le n$. The matrices R_s and L_s will be defined precisely in the next section. For now, consider the following example which shows the structure of matrices in the decomposition for $n = 6$.

$$A^{-1} = \underbrace{\overset{R_6}{\begin{bmatrix} 1 &&&&& * \\ & 1 &&&& * \\ && 1 &&& * \\ &&& 1 && * \\ &&&& 1 & * \\ &&&&& * \end{bmatrix}} \overset{L_6}{\begin{bmatrix} 1 &&&&& \\ & 1 &&&& \\ && 1 &&& \\ &&& 1 && \\ &&&& 1 & \\ * & * & * & * & * & 1 \end{bmatrix}}}_{P_6} \underbrace{\overset{R_5}{\begin{bmatrix} 1 &&&& * & \\ & 1 &&& * & \\ && 1 && * & \\ &&& 1 & * & \\ &&&& * & \\ &&&&& 1 \end{bmatrix}} \overset{L_5}{\begin{bmatrix} 1 &&&&& \\ & 1 &&&& \\ && 1 &&& \\ &&& 1 && \\ * & * & * & * & 1 & \\ &&&&& 1 \end{bmatrix}}}_{P_5} \cdots \tag{1}$$

Moreover, $P_s P_{s-1} \cdots P_1 = \operatorname{diag}(A_s^{-1}, I_{n-s})$ for $1 \le s \le n$. Thus, computing the sequence

$$A_1^{-1} b_1,\ A_2^{-1} b_2,\ A_3^{-1} b_3, \ldots, A_n^{-1} b_n \tag{2}$$

can be accomplished by computing the sequence

$$P_1 b,\ P_2 P_1 b,\ P_3 P_2 P_1 b, \ldots,\ P_n \cdots P_3 P_2 P_1 b. \tag{3}$$

This motivates the definition of the following problem.

- OnlineInverse: Suppose the rows of an $A \in \mathsf{K}^{n\times n}$ with generic rank profile are given one at a time, from first to last. As soon as rows $1, 2, \ldots, s$ of A are given, the matrix $P_s = R_s \cdot L_s$ should be produced, for $s = 1, 2, \ldots, n$. Note that P_s should be represented as the unevaluated product of the two structured matrices R_s and L_s.

On the one hand, if the representation of $P_1, P_2, \ldots, P_n$ as the pairs of multiplicands $R_1 \cdot L_1, R_2 \cdot L_2, \ldots, R_n \cdot L_n$ is known, the sequence shown in (3) can be computed in $O(n^2)$ field operations. On the other hand, a lower bound on the cost of solving OnlineInverse is $\Omega(n^2)$ since this a lower bound for the total size (number of field elements) to write down the sequence $P_1, P_2, \ldots, P_n$. It follows that any algorithm for OnlineInverse immediately gives an algorithm for OnlineSystem that supports the same running time bound. For the rest of this paper, we consider algorithms to solve OnlineInverse.

3. FULL DECOMPOSITION

For $1 \le s \le n$, let

$$A_s = \left[\begin{array}{c|c} A_{s-1} & u_s \\ \hline v_s & d_s \end{array}\right] \in \mathsf{K}^{s\times s},$$

where $u_s \in \mathsf{K}^{(s-1)\times 1}$, $v_s \in \mathsf{K}^{1\times(s-1)}$, and $d_s \in \mathsf{K}$. Suppose we have computed a decomposition of A_{s-1}^{-1} for some $s > 0$. Then Gaussian elimination produces a pair of $s \times s$ matrices $\bar{L}_s$ and $\bar{R}_s$ such that A_s^{-1} is equal to

$$\overset{\bar{R}_s}{\left[\begin{array}{c|c} I_{s-1} & -A_{s-1}^{-1} u_s \\ \hline & (d_s - v_s A_{s-1}^{-1} u_s)^{-1} \end{array}\right]} \overset{\bar{L}_s}{\left[\begin{array}{c|c} I_{s-1} & \\ \hline -v_s & 1 \end{array}\right]} \left[\begin{array}{c|c} A_{s-1}^{-1} & \\ \hline & 1 \end{array}\right]. \tag{4}$$

The exact formula for A_s^{-1} could be derived by multiplying together the expression for A_s^{-1} in (4). However, the algorithms we describe in this paper do not compute A_s^{-1} explicitly at each stage, but rather keep it as the product of structured matrices. For example, applying (4) for $s = 1, 2, \ldots, n$ gives the following full decomposition for the inverse of $A = A_n$.

$$\begin{aligned} A_n^{-1} &= P_n \cdot \left[\begin{array}{c|c} A_{n-1}^{-1} & \\ \hline & 1 \end{array}\right] \\ &= P_n \cdot P_{n-1} \cdot \left[\begin{array}{c|c} A_{n-2}^{-1} & \\ \hline & I_2 \end{array}\right] \\ &\vdots \\ &= P_n \cdot P_{n-1} \cdots P_1, \end{aligned}$$

where $P_s \in \mathsf{K}^{n\times n}$, $1 \le s \le n$, is the product of two structured matrices:

$$\begin{aligned} P_s &= R_s \cdot L_s \\ &= \left[\begin{array}{c|c} \bar{R}_s & \\ \hline & I_{n-s} \end{array}\right] \cdot \left[\begin{array}{c|c} \bar{L}_s & \\ \hline & I_{n-s} \end{array}\right] \\ &= \left[\begin{array}{ccc} I_{s-1} & \boxed{\phantom{\begin{matrix}x\\x\end{matrix}}} & \\ & & I_{n-s} \end{array}\right] \cdot \left[\begin{array}{ccc} I_{s-1} & & \\ \boxed{} & 1 & \\ & & I_{n-s} \end{array}\right], \end{aligned} \quad (5)$$

with $\bar{R}_s$ and $\bar{L}_s$ as in (4). Thus R_s and L_s are the identity matrices, except for possibly the column vector and row vector indicated by the rectangles in R_s and L_s, respectively.

Naturally, equation (4) gives an iterative approach to solve problem OnlineInverse. Once the matrix$\times$vector product $A_{s-1}^{-1}u_s$ has been computed, the representation $P_s = R_s \cdot L_s$ can be computed in a further $O(s)$ field operations. Computing $A_{s-1}^{-1}u_s$ is equivalent to premultiplying u_s by the leading principal $(s-1)\times(s-1)$ submatrices of $P_1, P_2, \cdots, P_{s-1}$ sequentially at a cost of $4s^2 + O(s)$ field operations. Overall, all the pairs of multiplicands $P_s = R_s \cdot L_s$, $1 \le s \le n$, can be computed sequentially in $2n^3 + O(n^2)$ field operations. In the next two sections we show how to incorporate matrix multiplication to solve problem OnlineInverse in overall time $O(n^\omega)$.

4. RELAXED DECOMPOSITION

To store each A_s^{-1} explicitly as a dense $s \times s$ matrix at each stage is too expensive, but the representation

$$\mathrm{diag}(A_s^{-1}, I_{n-s}) = P_s \cdot P_{s-1} \cdots P_1$$

as the product of s pairs of structured multiplicands is too lazy: both of these approaches lead to an algorithm with running time $\Omega(n^3)$. In this section, we present a relaxed inverse decomposition for A_s^{-1} such that, at stage s, after the first s rows of A are given, A_s^{-1} is represented as the product of HammingWeight$(s) \le \lceil \log s \rceil$ pairs of structured multiplicands, where HammingWeight(s) is the number of 1s in the binary representation of s.

Lemma 1. $(R_j \cdot L_j)\cdot(R_{j-1}\cdot L_{j-1})\cdots(R_i \cdot L_i)$ *can be expressed as the product* $R_{j\sim i}\cdot L_{j\sim i}$ *of two structured matrices with the shape*

$$R_{j\sim i} = \left[\begin{array}{ccc} I_{i-1} & \boxed{\begin{array}{c|c|c} & \cdots & \\ & & \end{array}} & \\ & & I_{n-j} \end{array}\right] \quad (6)$$

and

$$L_{j\sim i} = \left[\begin{array}{cccc} I_{i-1} & & & \\ \boxed{\begin{array}{c} \\ \hline \vdots \\ \hline \end{array}} & \begin{array}{ccc} 1 & & \\ & \ddots & \\ & & 1 \end{array} & \\ & & & I_{n-j} \end{array}\right], \quad (7)$$

where the column dimension of the submatrix indicated by the rectangle in $R_{j\sim i}$ *and the row dimension of the submatrix indicated by the rectangle in* $L_{j\sim i}$ *are both* $j - i + 1$.

Proof. Decompose

$$A_j = \left[\begin{array}{c|c} A_{i-1} & U \\ \hline V & D \end{array}\right]$$

where $U \in \mathsf{K}^{(i-1)\times(j-i+1)}$, $V \in \mathsf{K}^{(j-i+1)\times(i-1)}$, and $D \in \mathsf{K}^{(j-i+1)\times(j-i+1)}$, and consider the block case of (4). Block Gaussian elimination produces a pair of $j \times j$ matrices $\bar{R}_{j\sim i}$ and $\bar{L}_{j\sim i}$ such that

$$A_j^{-1} = \bar{R}_{j\sim i}\bar{L}_{j\sim i}\left[\begin{array}{c|c} A_{i-1}^{-1} & \\ \hline & I_{j-i+1} \end{array}\right],$$

where

$$\bar{R}_{j\sim i} = \left[\begin{array}{c|c} I_{i-1} & -A_{i-1}^{-1}U \\ \hline & (D - VA_{i-1}^{-1}U)^{-1} \end{array}\right]$$

and

$$\bar{L}_{j\sim i} = \left[\begin{array}{c|c} I_{i-1} & \\ \hline -V & I_{j-i+1} \end{array}\right].$$

The augmented matrices

$$R_{j\sim i} = \mathrm{diag}(\bar{R}_{j\sim i}, I_{n-j})$$

and

$$L_{j\sim i} = \mathrm{diag}(\bar{L}_{j\sim i}, I_{n-j})$$

have the shape shown in (6) and (7), respectively. Finally, note that both

$$(R_{j\sim i}\cdot L_{j\sim i})\left[\begin{array}{c|c} A_{i-1}^{-1} & \\ \hline & I_{n-i+1} \end{array}\right]$$

and

$$(R_j \cdot L_j)\cdot(R_{j-1}\cdot L_{j-1})\cdots(R_i\cdot L_i)\left[\begin{array}{c|c} A_{i-1}^{-1} & \\ \hline & I_{n-i+1} \end{array}\right]$$

are the inverse of

$$\left[\begin{array}{c|c} A_j & \\ \hline & I_{n-j} \end{array}\right].$$

The result follows by the uniqueness of the inverse. $\square$

Remark 2. Lemma 1 does not hold for general matrices $(R_* \cdot L_*)$ of the shape shown in (5). But it does hold in the case where the pairs of multiplicands $(R_* \cdot L_*)$ are arising from Gaussian elimination of the same matrix A.

Definition 3. $P_{j\sim i}$ *denotes the unevaluated product of the pair of matrices* $R_{j\sim i}\cdot L_{j\sim i}$ *with the shape as shown in (6) and (7). The* size *of* $P_{j\sim i}$ *is* $j - i + 1$*, the number of pairs of multiplicands in the product.*

Lemma 1 defines $P_{j\sim i}$ in terms of the input matrix A_j. But since $P_{j\sim i} = (R_j\cdot L_j)\cdot(R_{j-1}\cdot L_{j-1})\cdots(R_i\cdot L_i)$ it is possible to define $P_{j\sim i}$ in terms of the pairs of structured multiplicands $R_j \cdot L_j$, $R_{j-1} \cdot L_{j-1}, \ldots, R_i \cdot L_i$. We do not give such a formula for general $P_{j\sim i}$, but we will give the formula for a special case later. To have a basic idea we remark that the submatrix indicated by the rectangle in $L_{j\sim i}$ is equal to $-A^{[i,\ldots,j]}_{[1\ldots,i-1]}$. The following example gives the formula for a $P_{j\sim i}$ of size 2.

EXAMPLE 4. Let $\bar{P}_j$ and $\bar{P}_{j-1}$ be the leading $j \times j$ principal submatrix of P_j and P_{j-1}. Let

$$\bar{P}_j = \left[\begin{array}{cc|c|} \cline{3-3} I_{j-2} & & a \\ & 1 & b \\ & & c \\ \cline{3-3} \end{array}\right] \cdot \left[\begin{array}{ccc} I_{j-2} & & \\ & 1 & \\ \cline{1-2} \multicolumn{1}{|c}{d} & \multicolumn{1}{c|}{e} & 1 \\ \cline{1-2} \end{array}\right],$$

$$\bar{P}_{j-1} = \left[\begin{array}{c|c|c} \cline{2-2} I_{j-2} & f & \\ & g & \\ \cline{2-2} & & 1 \end{array}\right] \cdot \left[\begin{array}{ccc} I_{j-2} & & \\ \cline{1-1} \multicolumn{1}{|c|}{h} & 1 & \\ \cline{1-1} & & 1 \end{array}\right],$$

and

$$M = \begin{bmatrix} d & e \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix}.$$

Then $\bar{P}_{j\sim j-1}$ can be expressed as the product of two structured matrices as follows:

$$\bar{P}_{j\sim j-1} = \overset{\bar{R}_{j\sim j-1}}{\left[\begin{array}{c|c|c|} \cline{2-3} I_{j-2} & aM+f & a \\ & bM+g & b \\ & cM & c \\ \cline{2-3} \end{array}\right]} \cdot \overset{\bar{L}_{j\sim j-1}}{\left[\begin{array}{ccc} I_{j-2} & & \\ \cline{1-1} \multicolumn{1}{|c|}{h} & 1 & \\ \cline{1-1} \multicolumn{1}{|c|}{d} & & 1 \\ \cline{1-1} \end{array}\right]}.$$

As shown in Section 3, $\operatorname{diag}(A_s^{-1}, I_{n-s})$ can be expressed as the product of s pairs of structured multiplicands of size 1:

$$\operatorname{diag}(A_s^{-1}, I_{n-s}) = P_s \cdot P_{s-1} \cdots P_1.$$

Based on Lemma 1, we now introduce a relaxed/lazy representation for $\operatorname{diag}(A_s^{-1}, I_{n-s})$, denoted by $(A_s)_L^{-1}$, that expresses A_s^{-1} as the product of at most $\log s$ pairs of structured multiplicands. We first give a few examples of the relaxed representation before giving a precise definition.

EXAMPLE 5. The relaxed representation of A_s^{-1} for $1 \le s \le 8$.

s	$(A_s)_L^{-1}$
$1 = (1)_2$	$P_{1\sim 1}$
$2 = (10)_2$	$P_{2\sim 1}$
$3 = (11)_2$	$P_{3\sim 3} \cdot P_{2\sim 1}$
$4 = (100)_2$	$P_{4\sim 1}$
$5 = (101)_2$	$P_{5\sim 5} \cdot P_{4\sim 1}$
$6 = (110)_2$	$P_{6\sim 5} \cdot P_{4\sim 1}$
$7 = (111)_2$	$P_{7\sim 7} \cdot P_{6\sim 5} \cdot P_{4\sim 1}$
$8 = (1000)_2$	$P_{8\sim 1}$

The relaxed inverse decomposition for $s = 6, 7, 8$ are shown below. Note that for each of these examples we assume $n = s$, to avoid having to augment with I_{n-s}.

$$(A_6)_L^{-1} = \underbrace{\overset{R_{6\sim 5}}{\begin{bmatrix} 1 & & & & * & * \\ & 1 & & & * & * \\ & & 1 & & * & * \\ & & & 1 & * & * \\ & & & & * & * \\ & & & & * & * \end{bmatrix}} \cdot \overset{L_{6\sim 5}}{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ * & * & * & * & 1 & \\ * & * & * & * & & 1 \end{bmatrix}}}_{P_{6\sim 5}}$$

$$\times \underbrace{\overset{R_{4\sim 1}}{\begin{bmatrix} * & * & * & * & & \\ * & * & * & * & & \\ * & * & * & * & & \\ * & * & * & * & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}} \cdot \overset{L_{4\sim 1}}{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}}}_{P_{4\sim 1}}$$

$$(A_7)_L^{-1} = \underbrace{\overset{R_7}{\begin{bmatrix} 1 & & & & & & * \\ & 1 & & & & & * \\ & & 1 & & & & * \\ & & & 1 & & & * \\ & & & & 1 & & * \\ & & & & & 1 & * \\ & & & & & & * \end{bmatrix}} \cdot \overset{L_7}{\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ * & * & * & * & * & * & 1 \end{bmatrix}}}_{P_7}$$

$$\times \underbrace{\overset{R_{6\sim 5}}{\begin{bmatrix} 1 & & & & * & * & \\ & 1 & & & * & * & \\ & & 1 & & * & * & \\ & & & 1 & * & * & \\ & & & & * & * & \\ & & & & * & * & \\ & & & & & & 1 \end{bmatrix}} \cdot \overset{L_{6\sim 5}}{\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ * & * & * & * & 1 & & \\ * & * & * & * & & 1 & \\ & & & & & & 1 \end{bmatrix}}}_{P_{6\sim 5}}$$

$$\times \underbrace{\overset{R_{4\sim 1}}{\begin{bmatrix} * & * & * & * & & & \\ * & * & * & * & & & \\ * & * & * & * & & & \\ * & * & * & * & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}} \cdot \overset{L_{4\sim 1}}{\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}}}_{P_{4\sim 1}}$$

$$(A_8)_L^{-1} = \underbrace{\overset{R_{8\sim 1}}{\begin{bmatrix} * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \end{bmatrix}} \cdot \overset{L_{8\sim 1}}{\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}}}_{P_{8\sim 1}}$$

To understand the following formal definition of the relaxed inverse, consider the case $s = 7$ and $\operatorname{diag}(A_7^{-1}, I_{n-7})$. The full inverse decomposition is

$$\operatorname{diag}(A_7^{-1}, I_{n-7}) = P_7 \cdot P_6 \cdot P_5 \cdot P_4 \cdot P_3 \cdot P_2 \cdot P_1$$

while the relaxed/lazy representation is

$$\operatorname{diag}(A_7^{-1}, I_{n-7}) = P_{7\sim 7} \cdot P_{6\sim 5} \cdot P_{4\sim 1}.$$

The structure of the relaxed decomposition is determined as follows. The largest power of 2 that is less than or equal to 7 is 4, and thus in the relaxed representation the rightmost four pairs $P_4 \cdot P_3 \cdot P_2 \cdot P_1$ of multiplicands of size 1 are compressed into the single pair of multiplicands $P_{4\sim 1}$ of size 4. Continuing, the largest power of 2 that is less than or equal to $7 - 4$ is 2, so $P_6 \cdot P_5$ is compressed to $P_{6\sim 5}$. Finally, the largest power of 2 that is less than or equal to $7 - 4 - 2$ is 1, so $P_7 = P_{7\sim 7}$ is left alone.

DEFINITION 6. *Decompose a positive integer* $s = 2^{i_1} + 2^{i_2} + \cdots + 2^{i_\ell}$ *in binary representation with* $i_1 > i_2 >$

$\cdots > i_\ell \geq 0$. *The relaxed/lazy representation of the inverse* $(A_s)^{-1}$ *is defined as*

$$(A_s)_L^{-1} = \left(P_{s\sim 2^{i_1}+2^{i_2}+\cdots+2^{i_{\ell-1}}+1}\right)\cdots \\ \cdots\left(P_{2^{i_1}+2^{i_2}\sim 2^{i_1}+1}\right)\cdot\left(P_{2^{i_1}\sim 1}\right).$$

Note that when s is a power of two, we have $s = 2^{i_1}$ and

$$(A_s)_L^{-1} = P_{2^{i_1}\sim 1} = R_{2^i\sim 1}\cdot L_{2^i\sim 1} = R_{2^i\sim 1},$$

so A_s^{-1} is explicitly computed in this case. Otherwise, the relaxed representation $(A_s)_L^{-1}$ is comprised of the product of a sequence of pairs of structured multiplicands $P_{*\sim *}$. Observe that according to Definition 6 the sizes of the $P_{*\sim *}$ from right to left are $2^{i_1} > 2^{i_2} > \cdots > 2^\ell$.

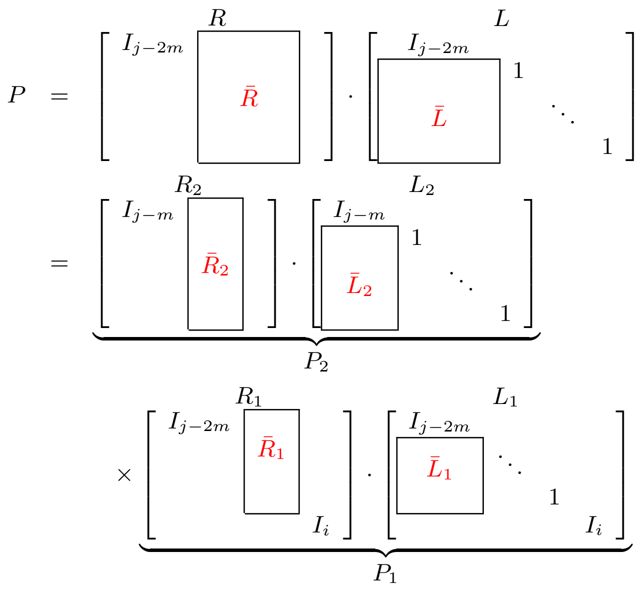
In the next section we will give algorithms for the online algorithm that starts with $\mathcal{R} = (A_0)_L^{-1}$ (which trivially consists of zero pairs of structured multiplicands) and updates $\mathcal{R}$ so that

$$\mathcal{R} = (A_1)_L^{-1}, \mathcal{R} = (A_2)_L^{-1}, \mathcal{R} = (A_3)_L^{-1}, \ldots.$$

It turns out — and may already be clear from Example 5 — that the construction of $(A_s)_L^{-1}$ from $(A_{s-1})_L^{-1}$ and $P_{s\sim s}$ may require the combinations of two adjacent $(P_{*\sim *})$ of equal size. In particular, we will need to compute the compression

$$P_{j\sim j-2m+1} = P_{j\sim j-m+1}\cdot P_{j-m\sim j-2m+1}, \qquad (8)$$

where both $P_{j\sim j-m+1}$ and $P_{j-m\sim j-2m+1}$ are of size m. Since this computation is important in deriving the cost of the relaxed approach, we give the formula and derive the cost for computing the left hand side of (8) given the two pairs of multiplicands on the right. For brevity, let $P_2 = R_2\cdot L_2$, $P_1 = R_1\cdot L_1$ and $P = R\cdot L$ denotes the leading $j\times j$ principal submatrix of $P_{j\sim j-m+1}$, $P_{j-m\sim j-2m+1}$ and $P_{j\sim j-2m+1}$ respectively. (Note that we are "overloading" the notation $P_2, R_2, L_2, \ldots$, but only temporarily in this example.) Then equation (8) is the block case of Example 4. We have



where

$$\bar{L} = \left[\begin{array}{c} \bar{L}_1 \\ (\bar{L}_2)_{[1,\ldots,j-2m]} \end{array}\right] \in \mathsf{K}^{2m\times(j-2m)} \qquad (9)$$

and $\bar{R}\in\mathsf{K}^{j\times 2m}$ can be computed by solving

$$R = (R_2L_2R_1L_1)L^{-1}$$

to obtain

$$\bar{R} = \left[\; \bar{R}_2(\bar{L}_2\bar{R}_1) + \left[\begin{array}{c}\bar{R}_1\\0\end{array}\right] \;\middle|\; \bar{R}_2 \;\right]. \qquad (10)$$

Theorem 7. *There exists an algorithm* `EqualSizeCompress` *that takes as input the two pairs of multiplicands* $P_{j\sim j-m+1}, P_{j-m\sim j-2m+1}\in\mathsf{K}^{n\times n}$*, both with size* m*, and returns as output the single pair of multiplicands* $P_{j\sim j-2m+1}\in\mathsf{K}^{n\times n}$ *of size* $2m$ *such that*

$$P_{j\sim j-2m+1} = P_{j\sim j-m+1}\cdot P_{j-m\sim j-2m+1}.$$

The cost of the algorithm is $O(nm^{\omega-1})$ *field operations in* K.

Proof. See (9) and (10) for the formula for computing the pair of multiplicands $P_{j\sim j-2m+1}$. Computing L is free since $\bar{L}$ can be read off from $\bar{L}_2$ and $\bar{L}_1$ directly. The dominating cost of computing R is to compute $\bar{R}_2(\bar{L}_2\bar{R}_1)$, where $\bar{R}_2\in\mathsf{K}^{j\times m}$, $\bar{L}_2\in\mathsf{K}^{m\times(j-m)}$ and $\bar{R}_1\in\mathsf{K}^{(j-m)\times m}$. The product of two $m\times m$ matrix can be computed in cm^ω field operations, for some fixed constant c. Dividing $\bar{R}_2$, $\bar{L}_2$, and $\bar{R}_1$ into at most $\lceil j/m\rceil$ blocks of dimension bounded by $m\times m$, $\bar{L}_2\bar{R}_1$ can be computed in $\lceil j/m\rceil cm^\omega + O(mj)$ field operations and $\bar{R}_2(\bar{L}_2\bar{R}_1)$ takes another $\lceil j/m\rceil cm^\omega + O(mj)$ field operations. The result now follows by noting that $j\leq n$. $\square$

5. RELAXED ONLINE INVERSION

The iterative approach to solve problem OnlineInverse has overall cost $2n^3 + O(n^2)$. In this section we show how to incorporate matrix multiplication to solve the problem OnlineInverse in $O(n^\omega)$ field operations in K. We adopt two ideas used in relaxed [7] and online [6] algorithms.

The first idea is to *relax*, that is, to use the relaxed representation $(A_s)_L^{-1}$ for A_s^{-1} as in Definition 6. The representation $(A_s)_L^{-1}$ for $s = 1,2,\ldots,n$ is constructed in an incremental fashion. Let $s > 0$ and suppose $(A_{s-1})_L^{-1}$ and P_s are known. Since $(A_s)^{-1} = P_s(A_{s-1})_L^{-1}$, the relaxed representation $(A_s)_L^{-1}$ is computed by compressing the pair of multiplicands P_s (of size one) and the first pair of multiplicands of $(A_{s-1})_L^{-1}$ into a single pair if they have equal size, repeating if required. Algorithm 1 outlines the algorithm that starts with $\mathcal{R} = (A_{s-1})_L^{-1}$ and P_s and updates $\mathcal{R}$ so that $\mathcal{R} = (A_s)_L^{-1}$.

Algorithm 1 `UpdateRelaxedInverse`$[\mathcal{R}](P_s)$

```
Require: $\mathcal{R} = (A_{s-1})_L^{-1}$, $P_s \in \mathsf{K}^{n\times n}$
Ensure: $\mathcal{R}$ is updated so that $\mathcal{R} = (A_s)_L^{-1}$
1: $P := P_s$;
2: $P_l :=$ first pair of multiplicands in $\mathcal{R}$;
3: while (size of $P$ = size of $P_l$) do
4:     $P :=$ EqualSizeCompress$(P, P_l)$;
5:     Remove $P_l$ from $\mathcal{R}$;
6:     $P_l :=$ first pair of multiplicands in $\mathcal{R}$;
7: end while
8: $\mathcal{R} := P\cdot\mathcal{R}$;
```

Note that when s is odd, then by Definition 6 the first pair of multiplicands in $(A_{s-1})_L^{-1}$ has size greater than one, so the while loop is never executed. For even values of s

the while loop is executed at least once, possibly as many as $\log s$ times.

Example 8. If $\mathcal{R} = (A_7)_L^{-1}$, the steps to update to $\mathcal{R} = (A_8)_L^{-1}$ are as follows.

$$\begin{aligned}(A_8)_L^{-1} &= P_8 \cdot (A_7)_L^{-1} \\ &= P_8 \cdot (P_7 \cdot P_{6\sim 5} \cdot P_{4\sim 1}) \\ &= P_{8\sim 7} \cdot P_{6\sim 5} \cdot P_{4\sim 1} &(11)\\ &= P_{8\sim 5} \cdot P_{4\sim 1} &(12)\\ &= P_{8\sim 1} &(13)\end{aligned}$$

Three compressions are required:

- $P_{8\sim 7} :=$ `EqualSizeCompress`$(P_{8\sim 8}, P_{7\sim 7})$ of size 1 to 2 in (11),
- $P_{8\sim 5} :=$ `EqualSizeCompress`$(P_{8\sim 7}, P_{6\sim 5})$ of size 2 to 4 in (12), and
- $P_{8\sim 1} :=$ `EqualSizeCompress`$(P_{8\sim 5}, P_{4\sim 1})$ of size 4 to 8 in (13).

The second idea is to *anticipate* computations. To make it clear that rows of A are given one at a time, we use a work matrix B for the anticipated computations. B is initialized to be the $n \times n$ zero matrix. At stage $s > 0$, row $A^{[s]}$ is copied to row $B^{[s]}$. At stage s the first s rows of the input matrix A are defined, and the (untransformed) input matrix can be decomposed as

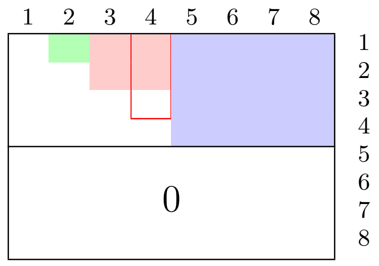
$$A = \left[\begin{array}{c|c|c} A_{s-1} & u_s & A^{[1\ldots s-1]}_{[s+1\ldots n]} \\ \hline v_s & d_s & A^{[s]}_{[s+1\ldots n]} \\ \hline \multicolumn{3}{c}{0} \end{array}\right] \in \mathsf{K}^{n\times n}. \qquad (14)$$

Recall that the dominant cost of computing the pair of multiplicands $P_s = R_s \cdot L_s$ arises from computing the matrix $\times$ vector product $A_{s-1}^{-1} u_s$ as shown in (5). At stage $s-1$, when $(A_{s-1})_L^{-1}$ has been computed, we do not apply A_{s-1}^{-1} to the single column u_s of A, which would be sufficient to compute the pair of multiplicands P_s at the next stage. Rather, we anticipate computations by applying the first element $P_{s\sim *}$ of the lazy representation $(A_{s-1})_L^{-1}$ to m columns of the work matrix B, where m is the size of $P_{s\sim *}$. This effectively incorporates matrix multiplication, and ensures that at the beginning of stage s we have $B^{[1,\ldots,s-1]}_{[s]} = (A_{s-1})^{-1} u_s$. Algorithm 2 computes P_s given row s of A using the work matrix B and $\mathcal{R} = (A_{s-1})_L^{-1}$. As a side effect, the algorithm updates the work matrix B as described above and updates $\mathcal{R}$ so that $\mathcal{R} = (A_s)_L^{-1}$. The algorithm first checks whether P_s exists, and reports "FAIL" if not. Detection is simple: if $d_s - v_s (A_{s-1})^{-1} u_s = 0$ then P_s does not exist.

Example 9. We consider the computations of the first four stages of the relaxed online inverse update for $A \in \mathsf{K}^{8\times 8}$. The following shows the work matrix B with certain submatrices highlighted.

Algorithm 2 `ComputeP`$[\mathcal{R}, B](A^{[s]})$

```
Require: $\mathcal{R} = (A_{s-1})_L^{-1}$ and $B \in \mathsf{K}^{n\times n}$ with $B^{[1,\ldots,s-1]}_{[s]} = (A_{s-1})^{-1} u_s$
Require: row $A^{[s]}$ of nonsingular $A \in \mathsf{K}^{n\times n}$
Ensure: $\mathcal{R}$ is updated so that $\mathcal{R} = (A_s)_L^{-1}$
Ensure: $B$ is updated by copying row $A^{[s]}$ to row $B^{[s]}$ and applying the first element $P_{s\sim *}$ of $(A_s)_L^{-1}$ to columns $s+1, s+2, \ldots, \min(s+m, n)$ of $B$, where $m$ is the size of $P_{s\sim *}$
Ensure: $P_s \in \mathsf{K}^{n\times n}$
 1: Copy $A^{[s]}$ to row $B^{[s]}$;
 2: if ($d_s - v_s B^{[1,\ldots,s-1]}_{[s]} = 0$) then
 3:     return "FAIL";
 4: end if
 5: Compute $P_s := R_s \cdot L_s$ using Equation (5);
 6: if ($s < n$) then
 7:     UpdateRelaxedInverse$[\mathcal{R}](P_s)$;
 8:     Let $P_{s\sim *}$ be the first element of $(A_s)_L^{-1}$;
 9:     Let $m$ be the size of $P_{s\sim *}$;
10:     $B_{[s+1,\ldots,\min(s+m,n)]} := P_{s\sim *} B_{[s+1,\ldots,\min(s+m,n)]}$;
11: end if;
12: return $P_s$;
```



At stage $s = 0$, B is initialized to be the 8×8 zero matrix. For $s = 1, 2, 3, 4$, the computations done at each stage are summarized below.

1. Copy row $A^{[1]}$ to row $B^{[1]}$ and compute $P_1 = R_1 \cdot L_1$. Apply P_1 to column 2 of B (🟩 area).
2. Copy row $A^{[2]}$ to row $B^{[2]}$ and compute $P_2 = R_2 \cdot L_2$. Compress $P_2 \cdot P_1 = P_{2\sim 1}$. Apply $P_{2\sim 1}$ to columns 3, 4 of B (🟥 area).
3. Copy row $A^{[3]}$ to row $B^{[3]}$ and compute $P_3 = R_3 \cdot L_3$. Apply P_3 to column 4 of B (□ area).
4. Copy row $A^{[4]}$ to row $B^{[4]}$ and compute $P_4 = R_4 \cdot L_4$. Compress $P_4 \cdot P_3 = P_{4\sim 3}$. Compress $P_{4\sim 3} \cdot P_{2\sim 1} = P_{4\sim 1}$. Apply $P_{4\sim 1}$ to columns 5,6,7,8 of B (🟦 area).

At stages $2^k = 1, 2, 4, \ldots$ the explicit inverse has been computed since $L_{2^k\sim 1} = I_n$ and $R_{2^k\sim 1} = \operatorname{diag}(A_{2^k}^{-1}, n - 2^k)$. At the end of stage 3, though, we only have the relaxed representation $\operatorname{diag}(A_3^{-1}, n-3) = P_3 \cdot P_{2\sim 1}$.

The relaxed algorithm to solve problem OnlineInverse is thus to initialize $\mathcal{R} = (A_0)_L^{-1}$, initialize B to be the $n \times n$

zero matrix, and call Algorithm 2 `ComputeP`$[\mathcal{R}, B](A^{[s]})$ for $s = 1, 2, \ldots, n$.

THEOREM 10. *There exists a relaxed algorithm to solve problem* ONLINEINVERSE *in $O(n^\omega)$ field operations in* K.

PROOF. For simplicity, assume n is a power of two; otherwise, we can augment A as $\mathrm{diag}(A, I_*)$. There are two parts of the cost: (1) computing P_s and updating B; (2) updating $(A_{s-1})_L^{-1}$ to $(A_s)_L^{-1}$ using P_s.

For part (1), the dominant cost is to perform the update to B according to step 10 of Algorithm 2. At each stage s, step 10 costs $O(s \cdot m^{w-1})$ field operations in K, where m is the size of the first element of $(A_s)_L^{-1}$. From Definition 6, m is the highest power of 2 that is less than or equal to s. Note that for $s = n$ no update to B is required. The sequence of sizes m for stages $s = 1, 2, \ldots, n-1$ stages are given by

$$(2^{\nu_2(s)})_{s=1}^{n-1} = 1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1, 16, \ldots, 2, 1,$$

where $\nu_2(s)$ is the highest power of two dividing s (sequence A00659 [11]). Note that the number 2^i in the sequence appears $n/2^{i+1}$ times. For instance, 1 appears every other number, 2 appears every four numbers, etc. For some absolute constant c, the total cost of all the updates to B is thus bounded by

$$\begin{aligned}
&\sum_{i=0}^{(\log n)-2} \frac{n}{2^{i+1}} \left(cn(2^i)^{\omega-1}\right) \\
&= \frac{cn^2}{2} \sum_{i=0}^{(\log n)-2} 2^{i(\omega-2)} \\
&= \frac{cn^2}{2} \left(1 + 2^{\omega-2} + 4^{(\omega-2)} + \cdots + (n/4)^{\omega-2}\right) \\
&= \frac{cn^2}{2} \left(\frac{2^{(\omega-2)(\log n-1)} - 1}{2^{\omega-2} - 1}\right) \\
&= \frac{cn^2}{2} \left(\frac{(n/2)^{\omega-2} - 1}{2^{\omega-2} - 1}\right) \\
&= \frac{cn^2}{2} O(n^{\omega-2}) \\
&= O(n^\omega).
\end{aligned}$$

Now consider part (2). The number of compressions done at stage s is equal to the maximal $t \in \mathbb{Z}$ such that $2^t \mid s$. Thus some stages are more costly than others. For example, when s is odd, $t = 0$ and no compressions are performed. When s is a power of 2, $t = \log_2 s$ compressions are required (see Figure 1, where compressions performed are highlighted in dashed lines).

We consider the overall cost for $1 \leq s \leq n$ as shown in Figure 2. Nodes in the inverse tree are computed in a bottom-up fashion. We label level $l = 0, 1, \ldots, \log n$ from leaf up to root level. For $l > 0$, nodes at level l can be computed using Theorem 7 with $m = 2^{l-1}$. The number of nodes at level l is $n/2^l$. Overall, the cost of constructing the whole inverse tree is

$$\sum_{l=1}^{\log n} \left(cn(2^{l-1})^{w-1}\right) \frac{n}{2^l} = O(n^\omega).$$

The derivation of the closed form for this summation are omitted since it is similar to the previous summation. The result follows by adding the cost of both parts. $\square$

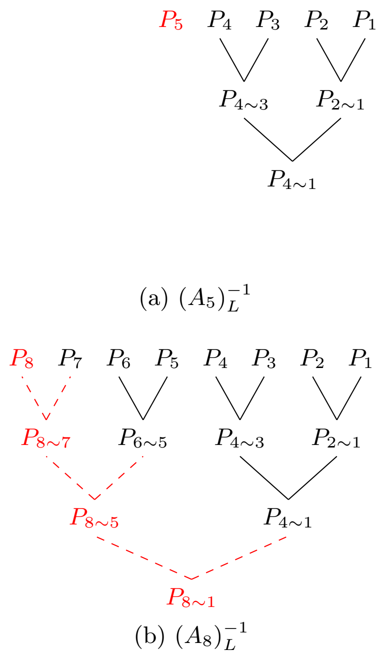


Figure 1: Examples of relaxed representation update

COROLLARY 11. *There exists an algorithm to solve problem* ONLINESYSTEM *in $O(n^\omega)$ field operations in* K.

6. APPLICATION TO RANK PROFILE

The algorithm for ONLINEINVERSE allows matrix multiplication to be incorporated in the algorithm for row rank profile presented in [12]. A complete exposition can be found in [14, Chapter 8] but we sketch the approach here.

- **Reduction to matrix of full column rank:** Use the Monte Carlo rank algorithm in [3, Theorem 2.11] to find a submatrix B of A that consists of $s \leq r$ linearly independent columns of A in time
$$(r^\omega + n + m + |A|)^{1+o(1)}.$$

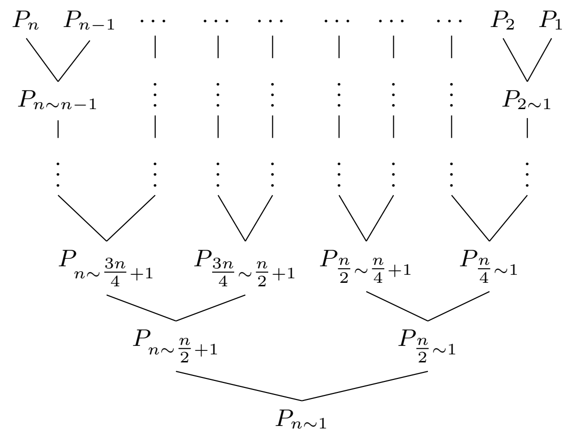


Figure 2: Online inverse tree

The algorithm should return $s = r$ with probability at least $3/4$. Provided $r = s$, the column space of the resulting matrix is equal to that of A so they must have the same row rank profile.

- **Reduction to matrix generic rank profile:** Let L be a lower triangular Toeplitz matrix L with entries chosen uniformly and randomly from K. Work with the unevaluated pair of matrices $B \cdot L$. With probability at least $1 - m(m+1)/(2\#\mathsf{K})$ the matrix obtained from the submatrix comprised of the rank profile columns of B postmultiplied by L will have generic rank profile [10, Theorem 2].

- Use a modification of the Monte Carlo rank profile algorithm supporting [12, Theorem 19] to compute the row rank profile of $B \cdot L$, where $B \cdot L$ is kept unevaluated. The key observation is that, if v is a linear combination of the rows of B, and we need to compute the dot product vLu for a column vector u, we can compute $v(Lu)$ instead of $(vL)u$: the latter expression implies we can construct a linear independence oracle from the rows of the (possibly sparse) B instead of the (probably dense) BL.

The above approach gives a Monte Carlo algorithm to compute the row rank profile of an arbitrary input matrix in time $(r^{\omega} + n + m + |A|)^{1+o(1)}$ field operations in K.

7. CONCLUSIONS

Our online algorithm for OnlineInverse assumed that at stage s the first s rows of A comprising the submatrix $A^{[1\cdots s]}$ be produced, $s = 1, 2, \ldots, n$. The requirement that the entire row be produced at each stage can be weakened somewhat. For example, at stage s, it suffices that the first $2 \cdot 2^{\lfloor \log_2 s \rfloor} \leq 2s$ entries of $A^{[1\cdots s]}$ be produced. In other words, at stage 1 the first 2 entries of the row is required, at stages 2–3 the first 4 entries of the rows are required, at stages 4–7 the first 8 entries of the rows are required, etc.

8. REFERENCES

[1] J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28:231–236, 1974.

[2] Z. Chen and A. Storjohann. A BLAS based C library for exact linear algebra on integer matrices. In M. Kauers, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'05*, pages 92–99. ACM Press, New York, 2005.

[3] H. Y. Cheung, T. C. Kwok, and L. C. Lau. Fast matrix rank algorithms and applications. *Journal of the ACM*, 60(5):733–751, 2013. Article No. 31.

[4] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. LinBox: A generic library for exact linear algebra. In A. J. Cohen and N. Gao, X.-S. andl Takayama, editors, *Proc. First Internat. Congress Math. Software ICMS 2002, Beijing, China*, pages 40–50, Singapore, 2002. World Scientific.

[5] J.-G. Dumas, T. Gautier, and C. Pernet. Finite field linear algebra subroutines. In T. Mora, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'02*, pages 63–74. ACM Press, New York, 2002.

[6] P. Giorgi and R. Lebreton. Online order basis algorithm and its impact on block Wiedemann algorithm. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'14*. ACM Press, New York, 2014.

[7] J. van der Hoeven. Relax, but don't be too lazy. *Journal of Symbolic Computation*, 36(6):479–542, 2002.

[8] O. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[9] C.-P. Jeannerod, C. Pernet, and A. Storjohann. Rank-profile revealing Gaussian elimination and the CUP matrix decomposition. *Journal of Symbolic Computation*, 56:56–58, 2013.

[10] E. Kaltofen and B. D. Saunders. On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9, Lecture Notes in Comput. Sci., vol. 539*, pages 29–38, 1991.

[11] N. J. A. Sloane. The on-line encyclopedia of integer sequences. *Notices of the American Mathematical Society*, 50(8):912–915, 2003.

[12] A. Storjohan and S. Yang. Linear independence oracles and applications to rectangular and low rank linear systems. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'14*. ACM Press, New York, 2014.

[13] J. van der Hoeven. Lazy multiplication of formal power series. In W. W. Küchlin, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'97*, pages 17–20. ACM Press, New York, 1997.

[14] S. Yang. Algorithms for fast linear system solving and rank profile computation. Master's thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2014.