A Generalization of the Fast LUP Matrix
Decomposition Algorithm and Applications

Oscar H. Ibarra* and Shlomo Moran*†

Department of Computer Science, University of Minnesota, Minneapolis, Minnesota

AND

Roger Hui‡

Department of Computer Science, University of Toronto, Toronto, Canada

Received May 27, 1981; revised July 25, 1981

We show that any \( m \times n \) matrix \( A \), over any field, can be written as a product, \( LSP \), of three matrices, where \( L \) is a lower triangular matrix with 1's on the main diagonal, \( S \) is an \( m \times n \) matrix which reduces to an upper triangular matrix with nonzero diagonal elements when the zero rows are deleted, and \( P \) is an \( n \times n \) permutation matrix. Moreover, \( L, S, \) and \( P \) can be found in \( O(m^{a-1}n) \) time, where the complexity of matrix multiplication is \( O(m^a) \). We use the \( LSP \) decomposition to construct fast algorithms for some important matrix problems. In particular, we develop \( O(m^{a-1}n) \) algorithms for the following problems, where \( A \) is any \( m \times n \) matrix: (1) Determine if the system of equations \( Ax = b \) (where \( b \) is a column vector) has a solution, and if so, find one such solution. (2) Find a generalized inverse, \( A^* \), of \( A \) (i.e., \( AA^*A = A \)). (3) Find simultaneously a maximal independent set of rows and a maximal independent set of columns of \( A \).

1. INTRODUCTION

In 1969, Strassen discovered a fast recursive algorithm for multiplying two \( n \times n \) matrices in \( O(n^{2.81}) \) time [8]. He also showed that any \( O(n^a) \) algorithm, where \( a > 2 \), can be used to obtain \( O(n^a) \) algorithms for matrix

*Research supported in part by NSF Grant MCS78-01736.
†Present Address: Department of Computer Science, Technion, Haifa 32000, Israel.
‡Research supported in part by the Natural Sciences and Engineering Research Council of Canada.

This bound has since been improved. The best bound known today is due to Coppersmith and Winograd, and is \( O(n^{2.495\ldots}) \).
inversion, computation of determinants, and solution of simultaneous linear equations, provided that each matrix encountered during the recursion process is nonsingular. This last result was later shown to hold for all matrices which are initially nonsingular by Bunch and Hopcroft (see [1, 4]). Thus, Gaussian elimination is not optimal for solving a system of simultaneous linear equations \( A \bar{x} = \bar{b} \) when \( A \) is nonsingular. The proof in [1] consisted of showing that any \( n \times n \) nonsingular matrix \( A \) can be decomposed into \( A = LUP \), where \( L \) is an \( n \times n \) lower triangular matrix with 1's on the main diagonal, \( U \) is an \( n \times n \) upper triangular matrix with nonzero diagonal elements, and \( P \) is an \( n \times n \) permutation matrix. Moreover, \( L \), \( U \), and \( P \) can be found in \( O(n^3) \) time. (Thus, to solve \( A \bar{x} = \bar{b} \), first solve \( L \bar{y} = \bar{b} \) for \( \bar{y} \) and then solve \( UP \bar{x} = \bar{y} \) for \( \bar{x} \). Hence, knowing \( L \), \( U \), and \( P \), the solution \( \bar{x} \) can be found in \( O(n^2) \) time.) When \( A \) is singular, an \( LUP \) decomposition may not exist, and even if it exists, it may not provide a fast algorithm for solving \( A \bar{x} = \bar{b} \) (see Section 2).

In this paper, we modify the algorithm of Bunch and Hopcroft [1] to show that any \( m \times n \) matrix \( A \) ( \( m \leq n \) ) can be decomposed into a product \( LSP \) of three matrices, where \( L \) is an \( m \times m \) lower triangular matrix with 1's on the main diagonal, \( S \) is an \( m \times n \) matrix which reduces to an upper triangular matrix with nonzero diagonal elements when the zero rows are deleted, and \( P \) is an \( n \times n \) permutation matrix. Moreover, \( L \), \( S \), and \( P \) can be found in \( O(m^{a-1}n) \) time. We then use this decomposition to develop \( O(m^{a-1}n) \) algorithms for the following problems, where \( A \) is any \( m \times n \) matrix:

1. Determine if the system \( A \bar{x} = \bar{b} \) (where \( \bar{b} \) is a column vector) has a solution, and if so find one such solution. This shows that Gaussian elimination is not optimal for solving any system of equations \( A \bar{x} = \bar{b} \).

2. Find a generalized inverse, \( A^* \), of \( A \) (i.e., \( AA^*A = A \)).

3. Find simultaneously (in \( A \)) a maximal independent set of rows and a maximal independent set of columns.

4. Diagonalize \( A \); i.e., find an \( m \times m \) nonsingular matrix \( X \) and an \( n \times n \) nonsingular matrix \( Y \) such that

\[
XAY = \begin{bmatrix}
I_r & 0 \\
0 & 0
\end{bmatrix},
\]

where \( I_r \) denotes the \( r \times r \) identity matrix.

5. Find the rank of \( A \).

6. Find the optimal (least square) solution to a (possibly inconsistent) system of equations \( A \bar{x} = \bar{b} \) over the complex field.
2. LSP Decomposition of Matrices

Let $A$ be an $m \times n$ matrix ($m \leq n$). An $LUP$ decomposition of $A$ (if it exists) is a triple $\langle L, U, P \rangle$, where $L$ is a lower triangular $m \times m$ matrix with 1's on the main diagonal, $U$ is an upper triangular $m \times n$ matrix, $P$ is an $n \times n$ permutation matrix, and $A = LUP$ [1]. If $A$ is nonsingular (i.e., $\text{rank}(A) = m$), then an $LUP$ decomposition of $A$ exists, and it can be used to obtain a solution to a system of simultaneous equations $A\vec{x} = \vec{b}$ in $O(m^2)$ arithmetic operations.

In [1], an algorithm which finds an $LUP$ decomposition of any $m \times n$ nonsingular matrix in $O(m^{a-1}n)$ time is given. This algorithm provides an $O(m^{a-1}n)$ algorithm to find a solution of $A\vec{x} = \vec{b}$. However, the algorithm may fail if $A$ is singular, and therefore it cannot be used to decide if a singular system of equations $A\vec{x} = \vec{b}$ has a solution (and to find one such solution if it exists). Moreover, we observe that if $A$ is singular, then an $LUP$ decomposition of $A$ may not really be useful in constructing a fast algorithm to solve $A\vec{x} = \vec{b}$. Consider, for example, the system $A\vec{x} = \vec{b}$, where

$$A_{2n \times 2n} = \begin{bmatrix} 0 & B \\ 0 & 0 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_{2n} \end{bmatrix}.$$

$B$ is any $n \times n$ matrix. $A$ is already upper triangular; hence, $A = U = LUP$ for $L = P = I_{2n}$. However, solving $U\vec{x} = \vec{b}$ is no easier than solving $B\vec{z} = \vec{c}$, where

$$\vec{z} = \begin{bmatrix} x_{n+1} \\ \vdots \\ x_{2n} \end{bmatrix} \quad \text{and} \quad \vec{c} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}.$$

**Remark.** An $LUP$ decomposition of a singular matrix may not always exist. For example, one can easily check that the matrix

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

does not have an $LUP$ decomposition.

Now define an $L'U'P$ decomposition of $A$, where now $L'$ is an $m \times m$ lower triangular matrix (possibly singular), $U'$ is an $m \times n$ upper triangular matrix with nonzero diagonal elements, and $P$ is a permutation matrix. It can be shown that any $m \times n$ matrix $A$ has an $L'U'P$ decomposition, and there is an algorithm to find $L'$, $U'$, and $P$ in $O(m^{a-1}n)$ time. However, we
again see that an $L'U'P$ decomposition may not be useful in solving systems of equations. For let

$$A_{2n \times 2n} = \begin{bmatrix} 0 & 0 \\ B & 0 \end{bmatrix} \quad \text{and} \quad \tilde{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_{2n} \end{bmatrix},$$

where $B$ is any $n \times n$ matrix. Then $A = L' = L'U'P$ for $U' = P = I_{2n}$. But then the complexity of solving $L'\bar{x} = \tilde{b}$ is equivalent to the complexity of solving $B\bar{z} = \bar{c}$, where

$$\bar{z} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \bar{c} = \begin{bmatrix} b_{n+1} \\ \vdots \\ b_{2n} \end{bmatrix}.$$

In this section, we introduce a decomposition called LSP. $L$ and $P$ are as in the $LUP$ decomposition, and $S$ is a matrix which reduces to an upper triangular matrix with nonzero diagonal elements when the zero rows are deleted. When $A$ is nonsingular, $LSP$ reduces to $LUP$. We shall define this concept formally, and we shall show that an $LSP$ decomposition of any $m \times n$ matrix can be found in $O(m^{a-1}n)$ time. We shall also show how an $LSP$ decomposition can be used to construct a fast algorithm to decide if $A\bar{x} = \tilde{b}$ has a solution, and to find one such solution if it exists.

**Definition.** An $m \times n$ matrix $S$ is semi-upper triangular (s.u.t.) if deleting the zero rows from $S$ results in an upper triangular matrix with nonzero diagonal elements. (By convention, the zero matrix $[0]_{m \times n}$ is s.u.t.)

**Example.** The following matrix is s.u.t.

$$S = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}.$$

**Definition.** Let $S_{m \times n}$ be s.u.t. of rank $r > 0$. (Clearly, the rank of $S$ is the number of its nonzero rows.) Let $S = [S_1 \mid S_2]$, where $S_1$ is an $m \times r$ matrix. Then an $m \times m$ matrix $S^{(-1)}$ is a left semi-inverse of $S$ if

$$S^{(-1)}S = \begin{bmatrix} I_r & \cdots \\ 0 & \cdots \\ 0 & \cdots \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad S^{(-1)}S_1 = \begin{bmatrix} I_r \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

**Lemma 2.1.** Let $S = [S_1 \mid S_2]$ be s.u.t. of rank $r > 0$, where $S_1$ is an $m \times r$ matrix. Then $S^{(-1)}$ can be computed in $O(m^a)$ time.
Proof. The following algorithm constructs $S^{(-1)}$:

1. Permute the rows of $S_1$ so that the resulting matrix is of the form

$$
\begin{bmatrix}
U \\
0
\end{bmatrix},
$$

where $U$ is an $r \times r$ upper triangular matrix with nonzero diagonal elements.

2. Compute $U^{-1}$ and set

$$
V_{m \times m} = \begin{bmatrix}
U^{-1} & 0 \\
0 & 0
\end{bmatrix}.
$$

3. Reverse the permutation done in step (1) on the columns of $V$. The resulting matrix is $S^{(-1)}$.

The correctness of the algorithm is easily verified. Clearly, the time complexity is $O(m + r^a + r^2) \leq O(m^a)$.

**Example.** Let

$$
S = \begin{bmatrix}
2 & 1 & 1 \\
0 & 0 & 0 \\
0 & 1 & 2
\end{bmatrix}.
$$

Then

$$
S_1 = \begin{bmatrix}
2 & 1 \\
0 & 0 \\
0 & 1
\end{bmatrix}, \quad S_2 = \begin{bmatrix}
1 \\
0 \\
2
\end{bmatrix},
$$

$$
U = \begin{bmatrix}
2 & 1 \\
0 & 1
\end{bmatrix}, \quad U^{-1} = \begin{bmatrix}
\frac{1}{2} & -\frac{1}{2} \\
0 & 1
\end{bmatrix},
$$

$$
V = \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}, \quad S^{(-1)} = \begin{bmatrix}
\frac{1}{2} & 0 & -\frac{1}{2} \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}.
$$

Indeed,

$$
S^{(-1)}S = \begin{bmatrix}
\frac{1}{2} & 0 & -\frac{1}{2} \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
2 & 1 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
-\frac{1}{2} & 2
\end{bmatrix} \begin{bmatrix}
0 & 1 \\
0 & 0 \\
0 & 0
\end{bmatrix} \quad \text{and} \quad S^{(-1)}S_1 = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & 0
\end{bmatrix}.
$$

\[\text{2}^\text{The inverse of an } r \times r \text{ triangular matrix can be found in } cr^a + O(r^2) \text{ steps, where } cn^a \text{ steps are required for matrix multiplication [1].}\]
We can now define an LSP decomposition formally.

**Definition.** An LSP decomposition of an $m \times n$ matrix $A$ is a triple $(L, S, P)$, where $L_{m \times m}$ is lower triangular with 1's on the main diagonal, $S_{m \times n}$ is s.t., and $P_{n \times n}$ is a permutation matrix.

We now generalize the algorithm in [1, 4] to find an LSP decomposition of an $m \times n$ matrix $A (m \leq n)$. Without loss of generality, we assume that $m$ and $n$ are powers of 2. If $m = 1$ a trivial algorithm of $O(n)$ steps is applied. (If $A$ is a zero vector $[0, \ldots, 0]$, then $L = [1]$, $S = A$, and $P = I_n$.) For a matrix with $2m$ rows, the following recursive procedure is applied:

Let

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where $A_1$ and $A_2$ are $m \times n$ matrices.

**Step 1.** Compute $L_1$, $S_1$, $P_1$—the LSP decomposition of $A_1$. Let $r_1 = \text{rank}(A_1)$. Compute $A'_2 = A_2P_1^{-1}$. Then

$$A = \begin{bmatrix} L_1 & 0 \\ 0 & I_m \end{bmatrix} \begin{bmatrix} S_1 \\ A'_2 \end{bmatrix}P_1.$$

If $S_1 = [0]_{m \times n}$, then let $C' = A_2$ and go to step 5. (In this case, $L_1 = I_m$ and $P_1 = I_n$.) Otherwise, let

$$\begin{bmatrix} S_1 \\ A'_2 \end{bmatrix} = \begin{bmatrix} S'_1 & B \\ F & C \end{bmatrix},$$

where $F$ and $S'_1$ are $m \times r_1$ matrices.

**Step 2.** Compute $S'_1^{(-1)}$.

**Step 3.** Let $F_{m \times m}$ be the matrix obtained by adding to $F$ $m - r_1$ zero columns, i.e., $F' = [F \mid 0]$. Compute $F'S'_1^{(-1)}$. By Lemma 2.1,

$$S'_1^{(-1)}S'_1 = \begin{bmatrix} I_{r_1} \\ 0 \end{bmatrix}.$$

Hence,

$$F'S'_1^{(-1)}S'_1 = [F \mid 0] \begin{bmatrix} I_{r_1} \\ 0 \end{bmatrix} = F.$$
Step 4. Compute $C' = C - F'S_1(-1)B$. Then

$$A = \begin{bmatrix} L_1 & 0 \\ F'S_1(-1) & I_m \end{bmatrix} \begin{bmatrix} S_1' & B \\ 0 & C' \end{bmatrix} P_1.$$  

Step 5. Compute $L_2, S_2, P_2$ — the LSP decomposition of $C'$. If $S_1 = [0]$, then

$$A = \begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix} \begin{bmatrix} 0 \\ S_2 \end{bmatrix} P_2,$$

where $L_1 = I_m$.

Else

$$A = \begin{bmatrix} L_1 & 0 \\ F'S_1(-1) & L_2 \end{bmatrix} \begin{bmatrix} S_1' & BP_2^{-1} \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} I_{r_1} & 0 \\ 0 & P_2 \end{bmatrix} P_1 = \text{LSP}.  

Time analysis. Let the complexity of $n \times n$ matrix multiplication be $cn^a$. (Note that this implies multiplication of $D_{m \times m}C_{m \times n}$, where $m \mid n$, takes $cm^{a-1}n$ steps.) Let $I(n)$ be the complexity of inverting an $n \times n$ triangular matrix. (As noted before, $I(n) = cn^a + O(n^2)$.) Then, for $m \leq n$ with $m$ and $n$ powers of 2, we have the following relation for the complexity $T(m, n)$ of the algorithm above (we neglect lower-order terms):

$$T(1, n) \leq bn$$

for some $b$ (without loss of generality assume $b \leq c$).

$$T(2m, n) \leq 2T(m, n) + I(m) + cm^a + cm^{a-1}n$$

$$\leq 2T(m, n) + 3cm^{a-1}n.$$  

This easily implies (by induction) that

$$T(m, n) \leq 3cm^{a-1}n / (2^{a-1} - 2).$$

From the discussion above, we have our first theorem:

**Theorem 2.1.** There is an $O(m^{a-1}n)$ algorithm to find an LSP decomposition of any $m \times n$ matrix ($m \leq n$).

**Corollary 2.1.** Let $m \leq n$. Each of the following problems can be solved in $O(m^{a-1}n)$ time:

1. Given an $m \times n$ matrix $A$, find its rank.

2. Given an $m \times n$ matrix $A$ and a column vector $\vec{b}$, determine if the system of equations $Ax = \vec{b}$ has a solution, and if so find one.
Proof. (1) To compute \( \text{rank}(A) \), decompose \( A \) into \( \text{LSP} \). Then \( \text{rank}(A) = \text{number of nonzero rows in } S \). (A different method for computing \( \text{rank}(A) \) in \( O(m^{a-1}n) \) time is given in [9].)

(2) To solve \( A\bar{x} = \bar{b} \), decompose \( A \) into \( \text{LSP} \). Next, solve \( L\tilde{y} = \tilde{b} \) for \( \tilde{y} \). Then solve \( SP\bar{x} = \tilde{y} \) for \( \bar{x} \). \( \square \)

A variation of the \( \text{LSP} \) decomposition which is sometimes more convenient to use is given by the following definition.

Definition. An \( \text{LQUP} \) decomposition of an \( m \times n \) matrix \( A \) is a quadruple, \( \langle L, Q, U, P \rangle \), where \( L_{m \times m} \) is lower triangular with 1's on the main diagonal, \( Q_{m \times m} \) and \( P_{n \times n} \) are permutation matrices, and

\[
U = \begin{bmatrix} U_1 \\ 0 \end{bmatrix},
\]

where \( U_1 \) is upper triangular with nonzero diagonal elements.

Theorem 2.2. There is an \( O(m^{a-1}n) \) algorithm to find an \( \text{LQUP} \) decomposition of any \( m \times n \) matrix \( (m \leq n) \).

Proof. Let \( A = \text{LSP} \). Let \( Q_1 \) be an \( m \times m \) permutation matrix such that

\[
Q_1S = \begin{bmatrix} U_1 \\ 0 \end{bmatrix},
\]

where \( U_1 \) is upper triangular with nonzero diagonal elements. (By the structure of \( S \), such a \( Q_1 \) exists and can be easily found in \( O(mr) \) steps.) Let \( Q = Q_1^{-1} \) and \( U = Q_1S \). Then \( \text{LQUP} = LQ^{-1}(Q_1S)P = \text{LSP} \). \( \square \)

3. Other Applications of \( \text{LSP/\text{LQUP}} \) Decomposition

In this section, we use the \( \text{LSP/\text{LQUP}} \) decomposition to provide fast algorithms for the following tasks, where \( A \) is an \( m \times n \) nonzero matrix:

1. Diagonalize \( A \); i.e., find nonsingular matrices \( X_{m \times m} \) and \( Y_{n \times n} \) such that

\[
XAY = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix},
\]

where \( r = \text{rank}(A) \).

2. Find a maximal nonsingular square submatrix of \( A \); i.e., find indices \( 1 \leq i_1, i_2, \ldots, i_r \leq m, 1 \leq j_1, j_2, \ldots, j_r \leq n \) such that the matrix
formed by the elements at the intersections of rows \(i_1, \ldots, i_r\) and columns \(j_1, \ldots, j_r\) is nonsingular, and \(r\) is maximal.

(3) Find a generalized inverse of \(A\), i.e., a matrix \(A^*_n\times_m\) such that \(AA^*A = A\) [2, 3, 6, 7].

Task 1 corresponds to the "classical" matrix diagonalization problem. Task 2 is a generalization of the problem of finding a maximal independent set of vectors (i.e., a "base") from a given set of vectors. Task 3 has many applications in applied mathematics [2, 3, 6, 7]. When \(A\) is nonsingular, fast algorithms for these tasks follow rather easily from the LUP decomposition [1].

First, we consider Task 1.

**Theorem 3.1.** There is an \(O(m^{a-1}n)\) algorithm to diagonalize an \(m \times n\) matrix.

**Proof.** Let \(A = LQUP\) be the decomposition described in Theorem 2.2, i.e.,

\[
U_{m \times n} = \begin{bmatrix} U_1 & F \\ 0 & 0 \end{bmatrix} = Q^{-1}L^{-1}AP^{-1},
\]

where \(U_1\) is an \(r \times r\) nonsingular, upper triangular matrix. Let \(X = Q^{-1}L^{-1}\) and \(Y = P^{-1}Y'\), where

\[
Y'_{n \times n} = \begin{bmatrix} U_1^{-1} & -U_1^{-1}F \\ 0 & I_{n-r} \end{bmatrix}.
\]

Then

\[
XAY = UY' = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}.
\]

The time to compute \(X\) is clearly \(O(m^a)\). To compute \(Y\), we first have to compute \(Y'\). Computing \(U_1^{-1}F\) requires \(O(r^{a-1}n)\) time. Now \(Y'\) has less than \(n(r + 1)\) nonzero entries. Thus, \(P^{-1}Y'\) can be computed in \(O(nr)\) additional steps. It follows that diagonalization can be carried out in \(O(m^{a-1}n)\) steps. \(\square\)

**Note.** The matrix

\[
N - P^{-1} \begin{bmatrix} -U_1^{-1}F \\ I_{n-r} \end{bmatrix}
\]
(\(U_i\) and \(F\) defined above) has rank \(n - r\). Moreover, \(AN = [0]_{m \times (n-r)}\). That is, the columns of \(N\) form a basis for the null space of \(A\). Thus, if \(A\bar{x} = \bar{b}\) has a solution, then we can find all solutions in the form \(\bar{x}_0 + N\bar{y}\), where \(\bar{x}_0\) is some solution (obtained, e.g., as in Corollary 2.1) and \(\bar{y}\) is any \(n-r\) element vector.

We now consider Task 2. We will need the following definition [5].

**Definition.** Let \(A_{m \times n}\) be a given matrix, and \(1 \leq i_1, i_2, \ldots, i_k \leq m\), \(1 \leq j_1, j_2, \ldots, j_r \leq n\). Then \(A[i_1, \ldots, i_k \mid j_1, \ldots, j_r]\) denotes the submatrix of \(A\) formed by the elements at the intersections of rows \(i_1, \ldots, i_k\) and columns \(j_1, \ldots, j_r\). If \(A\) is of rank \(r\) and for some \(i_1, \ldots, i_r, j_1, \ldots, j_r\), \(A' = A[i_1, \ldots, i_r \mid j_1, \ldots, j_r]\) is nonsingular, then \(A'\) is a maximal nonsingular square submatrix of \(A\).

It is known that for \(A\) of rank \(r\), \(A[i_1, \ldots, i_r \mid j_1, \ldots, j_r]\) is nonsingular if and only if rows \(A_{i_1}, \ldots, A_{i_r}\) are independent and columns \(A_{j_1}, \ldots, A_{j_r}\) are independent. The following lemma provides an easy way to find a nonsingular \(A[i_1, \ldots, i_r \mid j_1, \ldots, j_r]\) from the LSP decomposition of \(A\).

**Lemma 3.1.** Let \(A = \text{LSP}\) be of rank \(r\). Let \(i_1, \ldots, i_r\) be the nonzero rows of \(S\), and \(P(1) = j_1, \ldots, P(r) = j_r\). (\(P(i)\) is the image of \(i\) under permutation \(P\).)\(^3\) Then \(A[i_1, \ldots, i_r \mid j_1, \ldots, j_r]\) is nonsingular.

**Proof.** We need only show that rows \(A_{i_1}, \ldots, A_{i_r}\) are independent, and columns \(A_{j_1}, \ldots, A_{j_r}\) are independent. Clearly, rows \(S_{i_1}, \ldots, S_{i_r}\) are independent. One can easily verify that \(LS[i_1, \ldots, i_r \mid 1, 2, \ldots, n] = L[i_1, \ldots, i_r \mid i_1, \ldots, i_r]\). Since \(L\) is lower triangular and nonsingular, so is \(L[i_1, \ldots, i_r \mid i_1, \ldots, i_r]\). Hence, \(LS[i_1, \ldots, i_r \mid 1, 2, \ldots, n]\) is nonsingular, which means that rows \(i_1, \ldots, i_r\) in \(LS\) are linearly independent. Clearly, the multiplication of \(LS\) on the right by a permutation matrix \(P\) does not change this fact. It follows that \(A_{i_1}, \ldots, A_{i_r}\) are independent. Now columns \(S^1, \ldots, S^r\) are independent, and therefore so are the first \(r\) columns in \(LS\). It follows that in \(A = \text{LSP}\), columns \(P(1), \ldots, P(r)\) are independent. \(\Box\)

From Lemma 3.1, we have

**Theorem 3.2.** There is an \(\mathcal{O}(m^{r-1}n)\) algorithm to find a maximal nonsingular square submatrix of an \(m \times n\) matrix.

Finally, we consider Task 3. Recall that \(A^*_{n \times m}\) is a generalized inverse of \(A_{m \times n}\) if \(AA^*A = A\). A reflexive generalized inverse of \(A_{m \times n}\) is a matrix \(A'_{n \times m}\) satisfying \(AA'A = A\) and \(A'AA' = A'\) [3]. In general, a reflexive generalized

\(^3\)We can consider \(P\) as a permutation rather than a permutation matrix.
inverse is not unique. However, if \( A \) is square and nonsingular, then \( A^r = A^{-1} \) and is therefore unique.

**Theorem 3.3.** There is an \( O(m^{a-1}n) \) algorithm to find a reflexive generalized inverse of \( A \).

**Proof.** By Theorem 3.1, we can compute \( X \) and \( Y \) in time \( O(m^{a-1}n) \) such that

\[
XAY = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}_{m \times n}.
\]

Then \( A^* \) is a generalized inverse if and only if

\[
A^* = Y \begin{bmatrix} I_r & U \\ V & W \end{bmatrix} X
\]

for some \( U, V, W \). \( A^r \) is a reflexive generalized inverse if and only if

\[
A^r = Y \begin{bmatrix} I_r & U \\ V & VU \end{bmatrix} X
\]

for some \( U, V \) (see [2, 3]). \( \square \)

Let us restrict ourselves now to matrices over the complex field. The Moore–Penrose or pseudoinverse \( A^\dagger \) of \( A \) satisfies

\[
AA^\dagger A = A, \\
A^\dagger AA^\dagger = A^\dagger, \\
( AA^\dagger )^H = AA^\dagger, \\
( A^\dagger A )^H = A^\dagger A,
\]

where \( A^H \) is the conjugate transpose of \( A \). It can be shown that, for a complex \( A \), \( A^\dagger \) is unique [3].\(^4\) \( A^\dagger \) has the additional property that \( \bar{x}_0 = A^\dagger \bar{b} \) is the optimal solution to the (possibly inconsistent) system \( Ax = \bar{b} \), in the sense that for all vectors \( \bar{x}, \| Ax_0 - \bar{b} \| \leq \| Ax - \bar{b} \| \) and \( \| A\bar{x}_0 - \bar{b} \| = \| A\bar{x} - \bar{b} \| \) implies \( \| \bar{x}_0 \| \leq \| \bar{x} \| \).

**Theorem 3.4.** \( A^\dagger \) can be computed in time \( O(m^{a-1}n) \).

**Proof.** Consider \( A = LQUP \). Since the last \( m - r \) rows of \( UP \) are zeros, \( A = LQUP \), where \( LQ \) is the first \( r \) columns of \( LQ \) and \( UP \) is the first \( r \) rows of \( UP \). It is easily verified that \( A^\dagger = UP^H (UPUP^H)^{-1} (LQ^H LQ)^{-1} LQ^H \).

(\([0]_{m \times n}^\dagger = [0]_{n \times m} \). The total time is \( O(m^{a-1}n + r^a) = O(m^{a-1}n) \). \( \square \)

\(^4\) \( A^\dagger \) may not exist for matrices over fields with finite characteristic (where \( A^H \) is interpreted as the transpose of \( A \)).
ACKNOWLEDGMENTS

This paper was motivated by a remark made by an anonymous referee that the LUP decomposition technique in [1] can be used to find the rank of a matrix. (We note that a more direct way to find the rank using the LUP decomposition has been obtained in [9].)

REFERENCES