REPRINTED FROM
NAVAL RESEARCH LOGISTICS QUARTERLY

DECEMBER 1970
VOL. 17, NO. 4

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited

OFFICE OF NAVAL RESEARCH

NAVSO P-1278
Best Available Copy
A GENERALIZED UPPER BOUNDING METHOD FOR DOUBLY COUPLED LINEAR PROGRAMS

James K. Harriman

Naval Postgraduate School
Monterey, California

and

Leon S. Lasdon

Case Western Reserve University
Cleveland, Ohio

1. INTRODUCTION

The constraints of large linear programs can often be partitioned into independent subsets, except for relatively few coupling rows and coupling columns. The individual subsets may, for example, arise from constraints on the activity levels of subdivisions of a large corporation. Alternatively, such blocks may arise from activities in different time periods. The coupling rows may arise from limitations on shared resources or from combining the outputs of subdivisions to meet overall demands. The coupling columns arise from activities which involve different time periods (e.g., storage), or which involve different subdivisions (e.g., transportation or assembly). The case with only coupling rows or coupling columns, but not both has received much attention [8], [9]. A smaller amount of work has been done on the problem with both coupling rows and columns. Ritter has proposed a dual method [2], [7]. Except for the preliminary work of Wehner and White [10] and Hersteterman [11], there is no primal algorithm which exploits the structure of this problem. There is a need for such an algorithm, since such problems occur often in practice. A primal method is desirable since in large problems slow convergence may force termination of the algorithm prior to optimality.

The algorithm proposed here is an extension of the generalized upper bounding method for problems without coupling columns proposed in [5] and [6]. It produces the same sequence of extreme point solutions as the primal simplex method, and hence has the desirable convergence properties of that algorithm. However, the operations within each simplex iteration are organized to take maximal advantage of problem structure. Because of this structure it is possible to perform the computations while maintaining a compact representation of the basis inverse. In particular it is sufficient to maintain and update at each cycle a working basis inverse, inverses from each block, and possibly another matrix, I, in (4). The dimension of the working basis need never be more than the number of coupling rows in the problem plus the number of coupling columns in the current basis. Hence its dimension may change from cycle to cycle. At most one of the block inverses need be updated at any iteration. Given these quantities, all information needed to carry out a simplex iteration can easily be obtained. Further, efficient relations are given for updating the working basis and block inverses and I for the next cycle.

A significant amount of computational work has been done on a special class of production and inventory problems. Problems as large as 362 rows by 3225 columns were solved. Computation times
are encouraging, although no comparisons with other methods are available. The results indicate that the algorithm is sensitive to the dimension of the working basis and that effective measures can be taken to minimize this dimension.

When there are only coupling rows, the algorithm simplifies considerably and reduces to the procedure described in [5]. When each diagonal block contains only a single row, it reduces further to the generalized upper bounding method of Dantzig and Van Slyke [1].

2. BASIS STRUCTURE

The problem considered here is

\[ \text{minimize } \lambda_0 \]

subject to

\[ \bar{B}_s \lambda_s + \sum_{i=1}^{p} \bar{A}_i \lambda_i = b_0 \]

\[ \bar{B}_s \lambda_s + \bar{B}_s \lambda_s = b_i \quad i = 1, \ldots, p, \]

where \( \lambda \) is a vector with \( n \) components, all of which must be nonnegative except for \( \lambda_0 \), the first component of \( \lambda \). The column corresponding to \( \lambda_0 \) is all zero except for a 1 in the first row. This first row of the constraint matrix then defines the objective function. In matrix form the constraints can be represented:

\[
\begin{array}{ccccccc}
\bar{B}_s & \bar{A}_1 & \bar{A}_2 & \cdots & \bar{A}_{p-1} & \bar{A}_p \\
\bar{B}_1 & & & & & \\
\bar{B}_2 & & & & & \\
\vdots & & \phi & & & \\
\bar{B}_p & & & & & \\
S_0 & S_1 & S_2 & \cdots & S_{p-1} & S_p \\
\end{array}
\]

\[ = b_0 \quad m_s \]

\[ = b_1 \quad m_1 \]

\[ = b_2 \quad m_2 \]

\[ \vdots \quad \vdots \]

\[ \vdots \quad \vdots \]

\[ = b_p \quad m_p \]

\[ \text{total} = M \text{ rows} \]

\[ \text{No. of columns} \]

\[ n_0 \quad n_1 \quad n_2 \quad \cdots \quad n_{p-1} \quad n_p \quad \text{total} = N \text{ columns} \]

The problem has \( p \) diagonal blocks of dimension \( m_s \times n \), and \( m \)-shaped border consisting of \( m_s \) coupling constraints and \( n_s \) coupling variables. The set \( S \) denotes either the variables in the vector \( \lambda \) or the columns of (1) associated with these variables. The total constraint matrix has dimension \( M \times N \). We assume throughout that this matrix has rank \( M \) so that any basis matrix for the system will contain \( M \) columns and will be nonsingular.
Consider the structure of any basis, \( B \), for the system of constraints (4). Arranging the columns of \( B \) in the same order as in (4), yields the basis matrix in Figure 1 where the shaded areas contain nonzero entries. The column for \( v_0 \) is always in the basis and will always be the first basic column.

In this example the problem has seven blocks. There are no columns from block 3 in this particular basis. The basis matrix consists of \( q \) rectangular blocks, \( q \leq p \), roughly along the diagonal with borders from the coupling rows and coupling columns. The rectangular blocks may be either "tall," "square," or "wide."

The algorithm to be developed depends on having the lower right hand partition of \( B \) consist of square nonsingular blocks along the diagonal. Our strategy for handling the nonsquare blocks will be to rearrange the rows and columns of the basis matrix as follows:

a) For blocks with a column excess, move the extra columns over beside the coupling columns leaving a square block.

b) For blocks with a row excess, move the extra rows up with the coupling constraint rows leaving a square block.

c) Assure that the resulting square diagonal blocks are nonsingular.

Performing these operations on the basis \( B \) in Figure 1 will give the matrix in Figure 2, where we have
labeled the resulting nine submatrices for reference in Theorem 1. The submatrix $\gamma_5$ contains square nonsingular blocks and hence it is nonsingular. For this example block 5 was still singular after it was made square, so extra row-column pairs were removed from it until the remaining block became nonsingular. This introduced more excess rows and columns and created nonzero elements in the submatrix $\beta_5$.

For computational purposes the block diagonal section, $\gamma_5$ should be as large as possible, but still nonsingular. A lower bound on its size is now derived.

**THEOREM 1.** Let $S$ be the dimension of the block diagonal section $\gamma_5$ of Figure 2, and $U$ the dimension of the entire basis matrix. Suppose that the partitioning has been done so all diagonal blocks are nonsingular, and that there is no alternate partitioning which would give larger nonsingular blocks. Then $S \leq U - l_n - m_n$.

**PROOF:** Let any basis matrix for (1) be partitioned as illustrated in Figure 2:

$$B = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{bmatrix}.$$  

The block diagonal submatrix $\gamma_5$ is nonsingular, and hence has rank $S$. Consider the submatrix $[\beta_1, \gamma_1]$. It also has rank $S$ since it has $S$ linearly independent columns (those of $\gamma_5$) and only $S$ rows. Suppose there is a row $[\beta, \gamma]$ of the submatrix $[\beta_2, \gamma_2]$ which is not a linear combination of the rows of $[\beta_1, \gamma_1]$ and, say, this row is an excess row from block $j$. By the special structure of this row ($\gamma$ is zero except in block $j$) there must be at least one excess column in $[\beta_1, \gamma_1]$ from the same block $j$, otherwise the row $[\beta, \gamma]$ could not be independent. Since the row $[\beta, \gamma]$ is independent of the rows in the $j$th block of $[\beta_1, \gamma_1]$, this row and one of the excess columns can be adjoined to the $j$th block to give a larger square nonsingular block than before. But this contradicts the hypothesis of the theorem. Hence every row of $[\beta_2, \gamma_2]$ is a linear combination of rows of $[\beta_1, \gamma_1]$. This proves that the submatrix

$$K = \begin{bmatrix} \beta_1 & \gamma_1 \\ \beta_3 & \gamma_3 \end{bmatrix}$$

has rank $S$. Adjoin the first $l - 1$ columns to $K$ giving the submatrix $L = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_3 & \gamma_3 \end{bmatrix}$. $L$ has rank $S + l_n$ since only $l_n + 1$ columns were added and the first of these is all zero in these rows. Finally adjoin the first $m_n$ rows to $L$ giving the entire basis matrix

$$B = \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{bmatrix}.$$  

Then $B$ has rank $S + l_n + m_n$ since only $m_n$ rows have been added to $L$ and hence there can be at most $m_n$ more independent rows in $B$ than in $L$. But rank $B \leq U$, so that $S + l_n + m_n \leq U - l_n - m_n$, completing the proof of Theorem 1.
3. THE ALGORITHM

Our approach to the solution of doubly coupled problems will be to apply the revised primal simplex method to the problem. The special structure of the basis matrix will be exploited to simplify the computational and storage problems which occur for large problems. The revised simplex method involves the following steps at each iteration:

a) Find the simplex multipliers \( \pi = c_B B^{-1} \)

b) Price out the nonbasic columns \( P_p \)

\[ c_j = c_j - \pi P_j \]

and choose a column \( P^* \) with negative \( c^* \) to enter the basis if the current solution is not optimal.

c) Transform the entering column in terms of the current basis.

\[ \hat{P}_* = B^{-1} P_* \]

d) Determine the column to leave the basis,

\[ \hat{a} \]

where \( \hat{a}_i \) is component \( i \) of \( \hat{P}_* \).

e) Pivot to update \( B^{-1} \) and the current solution to account for the basis change.

Steps a) and e) involve multiplication by the basis inverse matrix \( B^{-1} \). To maintain \( B^{-1} \) for large problems requires extensive storage and computation, since even though \( B \) has special structure, \( B^{-1} \) is essentially dense with nonzero elements. Instead of performing the computations in a and e directly, we will solve the equations

\[
\pi B = c_B \text{ (solve for } \pi) \)

and

\[
B P_* = P_* \text{ (solve for } \hat{P}_*)
\]

by making a transformation of the basis matrix \( B \rightarrow R \) where \( R \) is block triangular. The resulting equations in terms of \( R \) will make it possible to exploit the special structure of \( B \).

Consider a nonsingular matrix \( T \) defined so that \( BT = R \) is upper block triangular.

\[
\begin{bmatrix}
G & H \\
J & B_i
\end{bmatrix}
\begin{bmatrix}
T \\
I
\end{bmatrix}
= \begin{bmatrix}
R \\
O
\end{bmatrix}
\]

where \( R_i \) is the block diagonal section \( y_i, G, H, \) and \( f \) are the remaining partitions of \( B \)

\[
\begin{pmatrix}
m & B_i \\
0 & B_j
\end{pmatrix}
\]

and
416 J. K. HARTMAN AND L. S. FASDON

(5) \[ F = -B_i f_j. \]

The matrix \( B \) is central to the procedure and will be called the working basis. From (4) and (5) we see that

(6) \[ R = G + HV = G - HB_i f_j. \]

By Theorem 1, \( B \) can always be partitioned so that the dimension of \( B \) is at most \( m_o + l_o \).

THEOREM 2: The working basis \( B \) is nonsingular.

PROOF: \( R \) is nonsingular since both \( B \) and \( T \) are. Since all elements of \( R \) below \( B \) are zero, \( B \) must be nonsingular.

To find the simplex multipliers, consider Eq. (2). Multiply through by \( T \) giving

(7) \[ \pi R = \pi BT = \pi H T = (1, 0, \ldots, 0) T = (1, 0, \ldots, 0). \]

The last equality holds since only \( v_{10} \) appears in the objective function, and by convention \( v_{10} \) is always the first basic variable. Since \( T \) is nonsingular, this multiplication will not change the solution \( \pi \).

Now partition \( \pi \) as \( \pi = (\pi_n, \pi_s, \ldots, \pi_{n_0}) \), where

\( \pi_n \) has \( M - S \) components and is the multiplier for rows in the working basis \( B \).

\( \pi_s \) has \( S \) components and is the multiplier for rows in the \( j \)th diagonal block \( R_j \) which is \( S \times S \) and nonsingular, \( j = 1, \ldots, q \). From the structure of \( R \) in (4), Eq. (7) becomes

(8) \[ \pi B = (1, 0, \ldots, 0) \quad (M - S \text{ components}), \]

and

(9) \[ \pi H_j \cdot \pi R_j = 0 \quad (S \text{ components}) \quad j = 1, \ldots, q, \]

where \( H_j \) is the submatrix of \( H \) containing the \( S \) columns in the \( j \)th block. Solving (8) gives

(10) \[ \pi_n = (1, 0, \ldots, 0) B^{-1} \quad \text{first row of } B^{-1}, \]

which can be substituted into (9) to give

(11) \[ \pi_s = -\pi_n H_j R_j^{-1} \quad j = 1, \ldots, q. \]

Hence to obtain the simplex multipliers it suffices to know the inverse of the working basis and the inverse of each of the diagonal blocks.

Next consider Eq. (12) for transforming the entering column in terms of the current basis. Define a nonsingular change of variables by

(12) \[ z = T \cdot \theta. \]

Then \( P' = Tz \), so Eq. (12) becomes

(13) \[ R : BT : B P' = P'. \]

which is a block triangular system and hence easier to solve than (12). Partition \( z \) and \( P' \) as \( (z_n, z_s, \ldots, z_q) \) and \( (P_{n0}, P_{s0}, \ldots, P_{q0}) \), respectively. Then (13) can be written as
\begin{align*}
\text{B}_{2n} + \sum_{j=1}^{q} H_{ij} \cdot P_{m} \\
\text{B}_{j} \cdot z_j = P_{m} \quad j = 1, \ldots, q.
\end{align*}

Solving (15) gives
\begin{equation}
z_j = H_{ji} \cdot P_{m},
\end{equation}
which when substituted into (14) gives
\begin{equation}
z_n = B \cdot \left( P_m - \sum_{j=1}^{q} H_{ij} \right).
\end{equation}

If \( P_i \) is not a coupling column, these formulas simplify since \( P_{m} = 0 \) for all but one \( j = 1, \ldots, q \). Hence by (16) \( z_j = 0 \) for all but one \( j \) and the summation in (17) has only one term.

It is now a simple matter to obtain \( \mathbf{\hat{P}} \) from \( z \) using \( \mathbf{\hat{P}} = \mathbf{T} \mathbf{z} \). With \( \mathbf{\hat{P}} \), partitioned as \( \mathbf{\hat{P}}_m, \mathbf{\hat{P}}_a, \ldots, \mathbf{\hat{P}}_q \) the structure of \( \mathbf{T} \) in (4) gives
\begin{align*}
\mathbf{\hat{P}}_m &= I_z = z_m \quad \text{and} \\
\mathbf{\hat{P}}_{aq} &= V_{2n} + I_z = V_{2n} + z, \quad j = 1, \ldots, q.
\end{align*}
where \( V_j \) is the submatrix of \( V \) containing the rows in the \( j \)th block. This completes the transformation of the column entering the basis.

4. CHANGING THE BASIS

Pricing out columns to select the entering column and choosing the column to leave the basis are done as in the ordinary revised simplex method, so it only remains to describe the updating procedures. Since the entire basis inverse \( \mathbf{B}^{-1} \) is not needed, the updating requirements will be somewhat different from those of the ordinary revised simplex algorithm. In particular, reviewing the solution of Eqs. (2) and (3) shows that it is sufficient to update \( \mathbf{H}^{-1}, \mathbf{H}^{-1}_j (j = 1, \ldots, q) \), and \( V \) at each iteration.

Before describing the various cases which can occur, we derive a general result for updating the working basis inverse.

THEOREM 3: If a basis change can be described by \( \mathbf{^*\hat{B}}^{-1} = \mathbf{EB}^{-1} \), where \( \mathbf{^*B}^{-1} \) is the basis inverse after the change, and \( \mathbf{E} \) is any transformation matrix, then the working basis inverse can be updated by
\begin{equation}
\mathbf{^*\hat{B}}^{-1} = (E_1 + E_3F)B^{-1},
\end{equation}
where \( E_1 \) and \( E_3 \) are partitions of \( \mathbf{E} \) to be described in the proof.

PROOF: From the definition and nonsingularity of \( \mathbf{R} \) and \( \mathbf{T} \), in (41, 51)
\begin{equation}
\mathbf{^*R}^{-1} = \mathbf{^*T}^{-1}\mathbf{^*B}^{-1} = \mathbf{^*T}^{-1}\mathbf{^*E} \mathbf{B}^{-1} = \mathbf{^*T}^{-1}\mathbf{^*ETR}^{-1}.
\end{equation}

Here all \(^*\)ed symbols relate to the system after the basis change is made. Writing (21) in partitioned form and using partitioned inverse theorems gives
To update the working basis inverse, note that it appears in the upper left hand partition of \( R^{-1} \). Hence deleting all but that submatrix gives the following specialization of (22):

\[
\begin{pmatrix}
I & 0 \\
-\mathbf{v} & \mathbf{I}
\end{pmatrix}
\begin{pmatrix}
E_1 \\
E_3
\end{pmatrix}
\begin{pmatrix}
V \\
1
\end{pmatrix}
\begin{pmatrix}
I & 0 \\
-\mathbf{v} & \mathbf{I}
\end{pmatrix}
\begin{pmatrix}
E_1 \\
E_3
\end{pmatrix}
\begin{pmatrix}
V \\
1
\end{pmatrix}
= 
\begin{pmatrix}
E_1 + E_2 \mathbf{v}' \\
V \mathbf{B}^{-1}
\end{pmatrix}
\]

Thus \( E_1 + E_2 \mathbf{v}' \) is a transformation matrix for the working basis inverse.

Let the columns from the block diagonal section of \( B \) be called "key" columns. The remaining columns of \( B \) are called "non-key." There are several cases to consider in the updating procedure:

CASE 1: The column leaving the basis is non-key. Then the entering column can be brought into the basis as a non-key column, and the standard updating formula for \( B^{-1} \) is \( \mathbf{E} \mathbf{B}^{-1} \). Here \( \mathbf{E} \) is an elementary matrix, equal to the identity except in the uth column which is given by \( [\eta_1, \eta_2, \ldots, \eta_M]' \), where

\[
\eta_i = -\frac{\partial a_{ie}}{\partial a_{ir}} \quad i = 1, \ldots, M \\
\eta_e = \frac{1}{\partial a_{re}}
\]

recalling that the uth basic column is leaving the basis. Since the leaving column is non-key, partitioning \( \mathbf{E} \) as in Theorem 3 gives \( \mathbf{E}_3 = 0 \), and \( \mathbf{E}_1 \) is an \( M-S \times M-S \) elementary matrix equal to the identity except in the uth column which is \( (\eta_1, \ldots, \eta_M)' \).

To update the working basis inverse for Case 1, substitute these into (20):

\[
\mathbf{B}^{-1} = (\mathbf{E}_1 + \mathbf{E}_2 \mathbf{v}') \mathbf{B}^{-1} = (\mathbf{E}_1 + \mathbf{0F}) \mathbf{B}^{-1} = \mathbf{E}_3 \mathbf{B}^{-1}.
\]

Hence \( \mathbf{B}^{-1} \) is updated by a single pivot operation.

None of the key columns in \( \mathbf{B} \) change in Case 1, and hence none of the diagonal blocks \( B_{ij} \) in \( B_i \) will change. Thus the block inverses, \( B_{ij}^{-1} \), require no updating. To update \( \mathbf{F} \) recall that \( \mathbf{F} = -\mathbf{B}_1^{-1} \mathbf{v} \). Now \( \mathbf{B}_1^{-1} = B_i^{-1} \) as shown above, and \( \mathbf{v} \mathbf{F} \) is different from \( \mathbf{F} \) only in the uth column which is replaced by the last \( S \) components of the entering column \( (\eta_1, \ldots, \eta_M)' \). Thus only the uth column of \( \mathbf{F} \) will change, and this will be replaced by
\[(26)\]

\[-B_1 (P_{d1}, \ldots, P_{d9}) = -(z_1, \ldots, z_9)\]

which has already been calculated in transforming the entering column. This completes the updating calculations for Case 1.

**CASE 2:** The column leaving the basis is a key column (from the block diagonal section).

**CASE 2a:** Both the leaving column and the entering column are from the same block \(B_d\). Then if the entering column will leave \(B_d\) nonsingular, a direct pivot can be performed and the basis change can be described by \(^*B = EB \cdot\), where \(E\) is an elementary column matrix. The \(E\) matrix differs from that in Case 1 only because the \((\eta_l, \ldots, \eta_M)\) column is now in the key section of the matrix. Thus \(E_1 = I_{B_d + M - S}\), and \(E_2\) has only one nonzero column \((\eta_l, \ldots, \eta_M)\), which is the \(r = (M-S)\)th.

To update the working basis inverse apply formula (20):

\[
^*B = (E_1 + E_2^*) B^{-1}
\]

\[
= \left( I + \begin{bmatrix} y_l & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & y_M \end{bmatrix} \right) B^{-1}
\]

(27)

\[
= \left( I + \begin{bmatrix} y_l & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \ddots & y_M \end{bmatrix} \right) B^{-1} + \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}^* B^{-1},
\]

where \(r\) is the \((r = (M-S))\)th row of \(V\).

In the block diagonal submatrix \(B_d\), only the \((r = (M-S))\)th column will change. Hence only one diagonal block inverse, say \(B_{j-1}\), will change; \(B_{j-1}\) will be updated by a single pivot, or equivalently.

\[
^*B_{j-1} = \tilde{E} B_{j-1}
\]

where \(\tilde{E}\) is an elementary column matrix formed from the elements of the new column. This change will leave \(B_d\) nonsingular if and only if the pivot element in this new column is nonzero. This condition must be checked to see if Case 2a can be applied.

The \(j\) submatrix will not change, and only the \(j\)th block of \(R_i^{-1}\) changes, so in \(V = -R_i^{-1} J\) only the \(j\)th partition will change. The updated version is given by

\[
^*V_j = -({^*B_{j-1}) J_j} = -\tilde{E} B_{j-1} J_j = \tilde{E} V_j,
\]

so the same elementary matrix is used to update \(V\).

**CASE 2b:** The column leaving the basis is a key column from block \(B_d\). Case 2a cannot be used either because the entering column is not from the same block (it may be a coupling column) or because the nonsingularity test in Case 2a failed. Then we can avoid making \(B_d\) smaller only if there is an excess column (see Figure 2) from block \(j\) which can be exchanged with the leaving column. If there is such a column, say the \(k\)th basic column, then after the exchange \(^*B = BE\), where \(E\) is a permutation matrix, an identity matrix with the \(r\)th and \(k\)th columns exchanged. Since \(E = E\), \(^*B = EB\).
Then (20) yields \( *R^{-1} = (E_1 + E_2V)R^{-1} \), where \( E_1 \) is an \( M \times S \) identity except for a zero in the \( k \)th diagonal position, and \( E_2 \) is zero except for a one in the \( k \)th row and \( r = (M-S) \)th column. Hence

\[
*R^{-1} = (E_1 + E_2V)R^{-1} = \begin{bmatrix}
1 & 1 & 1 \\
r & 1 \\
1 & 1
\end{bmatrix}
\]

where \( r \) is the \( r = (M-S) \)th row of \( I \). This is a valid interchange if and only if \( E_1 + E_2V \) is nonsingular, which is true if and only if the \( r \)th element of the row \( r \) is nonzero. If all elements of \( r \) in the excess columns are zero, then no interchange is possible and we proceed to Case 2c.

If an exchange occurs, one column of the block \( B_{ij} \) will change. Hence the inverse can be updated by a simple pivot

\[
*R^{-1} = \hat{E}B_{ij}^{-1},
\]

where \( \hat{E} \) is an elementary matrix. The elements needed to form the eta column in \( \hat{E} \) are found in the \( r \)th column of \( B_{ij}^{-1} \).

To update \( I \), note that by definition \( *V = -*R_1^{-1}*J_1 \). Only the \( j \)th block of \( B_1^{-1} \) changes, and a single column of \( I \) changes, but this column is zero outside the \( r \)th block \( J_1 \). Hence only the submatrix \( V_j \) needs to be updated.

\[
*V_j = -*R_1^{-1}*J_1 = -\hat{E}B_{ij}^{-1}*J_1.
\]

Now \( *J_1 = J_1 \) except in the \( k \)th column which is replaced by the leaving column which we will suppose to be the \( k \)th column in \( B_{ij} \). Thus \( B_1^{-1}*J_1 = -V_j \) except in the \( k \)th column which becomes the \( k \)th unit vector, so that

\[
*V_j = \hat{E}V_j
\]

except in the \( k \)th column which is the negative of the eta column of \( \hat{E} \).

This completes the updating required for an exchange of key and non-key basic columns. After the exchange the leaving column is non-key, so Case 1 can be used to bring the new column into the basis. It should be noted that Case 1 uses the elements \( \hat{a}_n \) of the transformed entering column (see (21)) and the vector \((z_1, \ldots, z_k)\) (see (26)). The interchange will affect these quantities. To update them interchange \( \hat{a}_n \) and \( \hat{a}_k \), and replace \( z_k \) by \( \hat{E}z_k \).

CASE 2c: The column leaving the basis is a key column from block \( B_{ij} \), and neither Case 2a nor 2b apply. When the column leaves, the new \( *H_{ij} \) will have one less column, and hence to remain square and nonsingular it must also lose a row. The process is most easily described by a repartitioning step followed by an application of Case 1. In the repartitioning step, the leaving column is shifted to the excess column partition, and some row of \( B_0 \) is made an excess row.

Without loss of generality assume that the pair being shifted is the first column and row of \( B_{ij} \).
The basis matrix before and after the change are given below. All elements of the matrices are identical, only the partitioning changes.

\[
\begin{align*}
\text{Original Partitioning: } & G & H \\
B = & B_{11} & \\
& J & B_i \\
\end{align*}
\]

This is the original partitioning. The row and column indicated by dashed lines will be added to the non-key section resulting in the new partitioning scheme given below.

\[
\begin{align*}
\text{New Partitioning: } & *G & *H \\
*B = & *B_{11} & \\
& *J & *B_i \\
\end{align*}
\]

In this new partitioning of the basis, \( *G \) has become larger by one row and one column, \( *B_{11} \) has become smaller, and the other partitions change in corresponding ways.

Now \( *B = B \), but because of the changes in partition sizes, \( *T \neq T \) and \( *R \neq R \). Hence we must compute \( *B_{11} \), \( *B \), and \( *V \).

\( *B_{11} \) is computed via standard formulas. If

\[
B_{11} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}
\]

where \( Z \) contains all but the first row and column, then

\[
*B_{11} = Z - \frac{1}{W} X
\]

This is possible only if \( W \neq 0 \) which provides a criterion for choosing the row to be shifted.\(^\dagger\)

To obtain \( *R \) recall that \( B = *B \) so

\[
*R = *T *B^{-1} = *T *B^{-1} = *T *TR^{-1}.
\]

\(^\dagger\) In general, if the \( k \)th column of \( R_k \) is being shifted, then the \( k \)th row can be shifted if the \( k \)th element of row \( k \) in \( R_k \) is non-zero. There must always be at least one non-zero element in row \( k \) since \( R_k \) is non-singular.
In partitioned form this is

\[
\begin{bmatrix}
  -R^{-1} & -R^{-1}H & B_1
  \\
  0 & *B_1 & 1
\end{bmatrix}
  =
\begin{bmatrix}
  I & O & 1 & O
  \\
  -V & 1 & V & I
\end{bmatrix}
  +
\begin{bmatrix}
  B^{-1} & -R^{-1}HB_1
  \\
  0 & B_1
\end{bmatrix}
\]

To isolate \(R^{-1}\) delete all but the upper left hand corner of this expression giving:

\[
\begin{bmatrix}
  *B_1
  \\
  0 & 1
\end{bmatrix}
  =
\begin{bmatrix}
  I & O & 1 & O
  \\
  -V & 1 & V & I
\end{bmatrix}
  +
\begin{bmatrix}
  B^{-1} & h
  \\
  W & Y
\end{bmatrix}
\]

where \(h\) is the first column of \(-R^{-1}HB_1\) and \([W, Y]\) is the first column of \(B_1\)

\[
\begin{bmatrix}
  I & O & 1 & O
  \\
  -V & 1 & V & I
\end{bmatrix}
  =
\begin{bmatrix}
  B^{-1} & h
  \\
  W & Y
\end{bmatrix}
\]

where \(v\) is the first row of \(V\).

\[
\begin{bmatrix}
  B^{-1} & h
  \\
  vB^{-1} & vh + W
\end{bmatrix}
\]

The vector \(h\) is computed as

\[
\begin{align*}
h & = \text{first column of } -R^{-1}HB_1 \\
& = R^{-1}H \text{ (first column of } B_1) \\
& = R^{-1}H(vW + Y) \\
& = R^{-1}H(vW + Y).
\end{align*}
\]

where \(H_1\) is the first partition of \(H\).

We note that the repartitioning here changes all partitions of \(B\). Hence in the expression \(R = G - HH_F\) for the working basis, all four factors change. Nevertheless, we need only add a "border" to \(R^{-1}\) to get \(R^{-1}\). Starting from the equation \(T^{-1} = R^{-1}RT^{-1}\) and arguing as above, a formula for updating \(I\) is obtained.
where \( \bar{v} \) is the first row of \( F \), \( \bar{v} \) is the rest of the first block of \( I \), and \( \bar{v} \) is all other blocks of \( F \). Note that \( \bar{v} \) has one less row and one more column than \( I \) and that the only nontrivial change occurs in the first block.

Performing these operations repartitions the basis matrix so that the leaving column becomes non-key. Applying Case 1 will then complete the basis change for Case 2c.

CASE 3: Using the above cases the simplex method can be performed, but the working basis will increase in dimension by one each time Case 2c occurs. Hence it may be desirable to periodically repartition the basis by moving excess rows and columns into the block diagonal section, essentially the reverse of Case 2c.

Suppose we have an excess row and an excess column from block \( j \). Consider the following submatrices

\[
\begin{array}{c|c}
\alpha & \beta \\
\hline
\gamma & B_{i,j} \\
\end{array}
\]

Bringing the row and column into the key section would mean adding them to \( B_{i,j} \) resulting in a larger \( j \)th diagonal block.

\[
\bar{B}_{i,j} = \begin{bmatrix}
\alpha & \beta \\
\gamma & B_{i,j}
\end{bmatrix}
\]

We already know \( B_{i,j} \) and we want to compute the inverse of the new block. If this inverse exists suppose that it is partitioned as

\[
\bar{B}_{i,j}^{-1} = \begin{bmatrix}
\Lambda & X \\
Y & Z
\end{bmatrix}
\]

Then [3] the blocks are given by

\[
\begin{align*}
\Phi &= 1/(\alpha - \beta B_{i,j} \gamma) \\
\Lambda &= \Phi \beta B_{i,j} \\
Y &= B_{i,j} \gamma \Phi \\
Z &= B_{i,j} \gamma - B_{i,j} \gamma \Lambda
\end{align*}
\]

and the inverse exists if and only if

\[
\gamma = B_{i,j} \gamma \Lambda = 0
\]
For computational purposes, note that \(-R_{\beta, \gamma}\) is just the \(\beta\)th partition of the column in \(F\) corresponding to the column being moved. Hence to test for possible pairs to be moved requires computation of an inner product of \(\beta\) with a column of \(F\).

For convenience in expression, and without loss of generality, we again assume that the last row and column of the non-key section will become the first row and column of the key section. The elements of \(\beta B\) will be exactly the same as the elements of \(B\)—only the partitions will change. The new working basis \(\beta B\) will become smaller. To find an expression for its inverse consider the expression

\[
\beta R = T^* \beta B = T^* T = TR^T.
\]

Writing this in partitioned form will give the following matrix equation:

\[
\begin{pmatrix}
\beta R & \beta H \beta R \\
0 & \beta R
\end{pmatrix}
= \begin{pmatrix}
I & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix}
\beta R \\
\beta R
\end{pmatrix}
= \begin{pmatrix} \beta R \\
\beta R
\end{pmatrix}.
\]

Taking only the first rows and columns of this equation gives:

\[
\begin{pmatrix}
\beta R \\
\beta R
\end{pmatrix}
= \begin{pmatrix}
I & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix}
\beta R \\
\beta R
\end{pmatrix}
= \begin{pmatrix} \beta R \\
\beta R
\end{pmatrix}.
\]

where \(\beta R\) is just \(\beta R\) with the last row and column deleted. Thus to get the new working basis inverse, merely delete the last row and the last column in the current working basis inverse.

To calculate \(\beta I\) use the equation

\[
\beta R = T^* \beta B = T^* R = TR T
\]

Then writing this in partitioned form and proceeding as above yields the result

\[
\begin{pmatrix}
\beta R \\
\beta R
\end{pmatrix}
= \begin{pmatrix}
I & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix}
\beta R \\
\beta R
\end{pmatrix}
= \begin{pmatrix} \beta R \\
\beta R
\end{pmatrix}.
\]

where \(\beta R\) is the first partition of \(\beta R\) less the last column, and \(\beta R\) is the remaining partitions of \(\beta R\) less the last column. For this to be computed we must find \(\phi\) which is the bottom row of the working basis less its last component \(c\). The working basis is not carried along explicitly, so we must compute \(\phi\).

\[
R - G - H \text{ from } \phi.
\]
where $G$ and $H$ are partitions of the basis matrix $B$ and $V$ is updated at each iteration of the process.

\[(53)\quad [\phi, \zeta] = \text{last row of } B \]
\[= \text{last row of } G + (\text{last row of } H) \cdot V \]
\[= g + h \cdot V, \quad (g = \text{last row of } G) \]
\[h = \text{last row of } H) \]
\[= g + [\beta, 0] \cdot V \]
\[= g + \beta \cdot V. \]

and dropping the last element of this gives $\phi$.

In the above calculations the most complicated formulas were those for updating $V$ in the two repartitioning procedures, Eqs. (52) and (53). An alternative procedure in these cases is to compute $V$ from its definition $V = -B_i' f$. In each case, only one partition of $I$ changes, say partition $I_1$, given by

\[(54)\quad V_i = -B_{i1}' f_i, \]

with $B_{i1}$ of dimension $s_j$. To compute this requires $s_j$ multiplications for each nonzero column of $f_i$, a total of at most $(M - S) \times s_j$ multiplications. The updating calculations require on the order of $(M - S) \times s_i$ multiplications. Hence they are computationally superior, but require more extensive program logic.

After performing the appropriate updating procedure, we are ready to start the next simplex iteration. Thus the description of the basic algorithm is complete.

5. SPECIALIZATION TO PROBLEMS WITH ONLY COUPLING ROWS

For problems which have no coupling columns, the algorithm simplifies considerably. The major simplification is that no repartitioning will ever be necessary and there will never by any excess rows. Hence Cases 2c and 3 of the updating procedure, which are the most complicated, are never needed. The working basis $B$ in (6) will always have exactly as many rows as there are coupling constraints, and each of the blocks $B_{ij}$ has dimension $m_i$ (see (11). Relations (10), (11) for the simplex multipliers remain the same as do relations (16), (19) for the transformed entering column. In updating, only Cases 1, 2a, and 2b can occur. These remain essentially the same. The computations involving $I$ simplify because each column of $I$ except the first has only one nonzero partition. This specialization of the algorithm is very similar to the method proposed by Kaul [5] and is described by Lasdon in [6].

6. APPLICATIONS TO PRODUCTION AND INVENTORY PROBLEMS

Consider a corporation which wishes to schedule the production of $K$ products at $T$ plants for $T$ time periods into the future. At each plant, and for each period the demand for each product is considered known and must be met in that period. The operation of the $i$th plant in the $t$th time period is hence limited by these $K$ demand constraints and also by $s$ constraints on locally available resources (e.g., plant capacity, labor) giving rise to a constraint block with $K \times s$ rows. There are $Kt$ such diagonal blocks. These blocks are coupled by constraints on scarce corporate resources which are allocated across the various plants and budgeted over time (e.g., corporate capital, scarce raw materials, highly
skilled labor. In addition to producing for immediate demand, any plant may
a) Produce a product and place it in inventory for future use.
b) Produce a product and ship it to some other plant which has a shortage of that product.

Each of the inventory and transportation activities gives rise to a column which couples two of the
diagonal blocks. Thus we have a doubly coupled linear program to solve. The number of rows is
on the order of $(K + r)LT$, so truly large problems may result. There are $KL(T(2L + T - 3)/2$ coupling
columns arising from the inventory and transportation activities, but these have very special structure.
Each column has only a cost coefficient, a single $+1$ in the $d$th demand equation for one block, and a
single $-1$ in the $d$th demand equation for another block. Hence they may be stored implicitly and
priced with minimal effort. Since these coupling activities incur a cost in addition to the production
cost, we anticipate that even though there are many such activities, relatively few will be profitable.
Hence in any basis there should be relatively few coupling columns so that the algorithm described
can be applied.

A number of computational simplifications appear. In computing the transformed entering
column, at most two of the $z_i$ in (16) will be nonzero, so (17) simplifies considerably. The special form
of the coupling columns implies that in $J$ any coupling column has only 2 nonzero elements $+1$ and
$-1$, while an excess column has only one partition nonzero. Hence in $F = -B_I^T J$ a coupling column
has at most two nonzero partitions and these are columns of the block inverses $B^{-1}_{ij}$. An excess column
has one nonzero partition in $F$. Hence it is probably best to compute $F$ at each iteration instead of
updating it. This is desirable since the most complicated update formulas are those involving $F$.

7. COMPUTATIONAL RESULTS

The algorithm described above has been coded and used to solve a number of test problems of
the type described in section 6. The program was written in FORTRAN V for the Univac 1108 computer
at Case Western Reserve University. The special structure of these problems made it possible
to solve reasonably large programs all in core. All problems were solved in single precision arithmetic.
Whenever a block inverse or the working basis inverse had been updated 50 times, it was re-inverted
using a standard Gaussian elimination routine. Good numerical accuracy was obtained in that different
runs on the same problem yielded solutions which were the same to seven significant figures. The
code was not written to be competitive with commercial routines, but rather to investigate the effects
of various pricing and repartitioning strategies. Nevertheless the solution times recorded are encouraging.

Data describing the test problems is given in Table 1. The notation used is as in section 6. For each
problem size, two problems were formulated. The lower numbered problem of each pair was constructed
to be relatively easy, in that few coupling columns appear in an optimal basis. The second problem in
each pair is derived from the first by adjusting the right hand side demand and resource availability
vector so that more coupling activities are required. Hence it is more difficult. For all problems, phase
I was initiated with a basis consisting of slack variables for all resource constraints and artificial vari-
ables for demand constraints.

Table 2 describes the effect of three different pricing strategies. Pricing strategy one allowed
coupling columns to enter the basis at any iteration. It led to long running times and large working
bases since many coupling columns tended to enter the basis in phase I, even in problems for which
there were none in the optimal basis. Pricing strategy two did not allow coupling columns to enter the
basis in the first $W$ iterations unless all other columns priced out optimally. It produced a substantial
reduction in running time. Strategy three, which was the most successful, allowed no coupling columns
### Table 1. Test Problem Descriptions

<table>
<thead>
<tr>
<th>Problem numbers</th>
<th>Number of products (K)</th>
<th>Number of plants (L)</th>
<th>Number of time periods (T)</th>
<th>Number of blocks (L.T)</th>
<th>Block size</th>
<th>Number of coupling columns</th>
<th>Number of coupling rows</th>
<th>Total problem size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>7 x 22</td>
<td>210</td>
<td>6</td>
<td>90 x 174</td>
</tr>
<tr>
<td>3, 4</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>18</td>
<td>7 x 22</td>
<td>405</td>
<td>8</td>
<td>134 x 101</td>
</tr>
<tr>
<td>5, 6</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>24</td>
<td>7 x 22</td>
<td>660</td>
<td>10</td>
<td>178 x 1186</td>
</tr>
<tr>
<td>7, 8</td>
<td>5</td>
<td>3</td>
<td>10</td>
<td>30</td>
<td>7 x 22</td>
<td>975</td>
<td>12</td>
<td>222 x 1635</td>
</tr>
<tr>
<td>9, 10</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>30</td>
<td>7 x 22</td>
<td>975</td>
<td>8</td>
<td>218 x 1635</td>
</tr>
<tr>
<td>11, 12</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>40</td>
<td>7 x 22</td>
<td>1500</td>
<td>10</td>
<td>290 x 2300</td>
</tr>
<tr>
<td>13, 14</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>50</td>
<td>7 x 22</td>
<td>2125</td>
<td>12</td>
<td>362 x 3225</td>
</tr>
</tbody>
</table>

### Table 2. Effect of Pricing Strategies

<table>
<thead>
<tr>
<th>Problem number</th>
<th>Pricing strategy</th>
<th>Total iterations</th>
<th>Total time (sec)</th>
<th>Iterations (per sec)</th>
<th>Maximum size of working basis</th>
<th>Maximum number of coupling columns in basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>329</td>
<td>18.32</td>
<td>17.8</td>
<td>31</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>178</td>
<td>5.05</td>
<td>35.2</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>140</td>
<td>30.88</td>
<td>11.9</td>
<td>37</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>326</td>
<td>12.99</td>
<td>25.1</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>274</td>
<td>9.39</td>
<td>29.2</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Exceeded 60 second time limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>151</td>
<td>25.33</td>
<td>17.8</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>321</td>
<td>14.33</td>
<td>22.2</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Not attempted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>557</td>
<td>30.02</td>
<td>11.6</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>138</td>
<td>24.12</td>
<td>10.2</td>
<td>21</td>
<td>11</td>
</tr>
</tbody>
</table>

(All runs made with case 3 repartitioning attempted every 10 iterations)
<table>
<thead>
<tr>
<th>Problem number</th>
<th>Repartitioning strategy</th>
<th>Total iterations</th>
<th>Total iterations per sect</th>
<th>Number of occurrences of various cases in updating</th>
<th>Final size working columns</th>
<th>Final No. coupling columns</th>
<th>Maxi number coupling columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>123</td>
<td>1.95</td>
<td>36.0</td>
<td>82</td>
<td>96</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>126</td>
<td>5.05</td>
<td>33.2</td>
<td>81</td>
<td>97</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>138</td>
<td>6.69</td>
<td>38.5</td>
<td>81</td>
<td>97</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>6.62</td>
<td>38.5</td>
<td>81</td>
<td>97</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>299</td>
<td>7.73</td>
<td>30.8</td>
<td>155</td>
<td>144</td>
<td>45</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>311</td>
<td>7.90</td>
<td>30.8</td>
<td>155</td>
<td>144</td>
<td>45</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>321</td>
<td>10.17</td>
<td>19.9</td>
<td>113</td>
<td>209</td>
<td>38</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>391</td>
<td>15.47</td>
<td>15.5</td>
<td>301</td>
<td>350</td>
<td>84</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>333</td>
<td>16.02</td>
<td>16.0</td>
<td>192</td>
<td>229</td>
<td>62</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>306</td>
<td>16.31</td>
<td>16.3</td>
<td>188</td>
<td>220</td>
<td>55</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>321</td>
<td>16.14</td>
<td>16.5</td>
<td>638</td>
<td>385</td>
<td>200</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>403</td>
<td>18.14</td>
<td>18.1</td>
<td>656</td>
<td>376</td>
<td>175</td>
<td>77</td>
</tr>
<tr>
<td>13</td>
<td>315</td>
<td>18.73</td>
<td>18.7</td>
<td>646</td>
<td>379</td>
<td>177</td>
<td>77</td>
</tr>
<tr>
<td>14</td>
<td>651</td>
<td>19.31</td>
<td>19.3</td>
<td>650</td>
<td>353</td>
<td>171</td>
<td>85</td>
</tr>
<tr>
<td>15</td>
<td>350</td>
<td>46.45</td>
<td>46.5</td>
<td>277</td>
<td>293</td>
<td>84</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>365</td>
<td>48.13</td>
<td>48.1</td>
<td>263</td>
<td>302</td>
<td>86</td>
<td>39</td>
</tr>
<tr>
<td>17</td>
<td>365</td>
<td>46.12</td>
<td>46.1</td>
<td>263</td>
<td>302</td>
<td>86</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>372</td>
<td>46.76</td>
<td>46.8</td>
<td>265</td>
<td>292</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>19</td>
<td>1191</td>
<td>92.08</td>
<td>12.9</td>
<td>748</td>
<td>233</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1200</td>
<td>81.14</td>
<td>11.4</td>
<td>741</td>
<td>217</td>
<td>76</td>
<td>121</td>
</tr>
<tr>
<td>21</td>
<td>1208</td>
<td>89.19</td>
<td>11.5</td>
<td>756</td>
<td>225</td>
<td>81</td>
<td>24</td>
</tr>
<tr>
<td>22</td>
<td>1211</td>
<td>87.02</td>
<td>11.3</td>
<td>784</td>
<td>220</td>
<td>90</td>
<td>25</td>
</tr>
<tr>
<td>23</td>
<td>351</td>
<td>96.01</td>
<td>7.9</td>
<td>390</td>
<td>108</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>372</td>
<td>69.11</td>
<td>10.8</td>
<td>372</td>
<td>102</td>
<td>33</td>
<td>14</td>
</tr>
<tr>
<td>25</td>
<td>371</td>
<td>68.11</td>
<td>11.0</td>
<td>372</td>
<td>99</td>
<td>61</td>
<td>14</td>
</tr>
<tr>
<td>26</td>
<td>369</td>
<td>73.02</td>
<td>10.5</td>
<td>372</td>
<td>104</td>
<td>72</td>
<td>37</td>
</tr>
<tr>
<td>27</td>
<td>1064</td>
<td>147.48</td>
<td>9.9</td>
<td>787</td>
<td>238</td>
<td>81</td>
<td>0</td>
</tr>
</tbody>
</table>
to enter the basis (unless necessary) until all artificial variables have left the basis. Pricing strategy three was used for all of the remaining runs. All problems in Table 2 were solved with the repartitioning procedure of Case 3 attempted every 10 iterations (see section 4).

Table 3 shows the effects of various repartitioning strategies, i.e., strategies for employing Case 3. The strategies tested are to attempt Case 3:

1. never
2. every 10 iterations
3. every five iterations
4. every iteration
5. whenever there are at least 10 excess columns
6. whenever there are at least five excess columns.

The different strategies often gave rise to slightly different numbers of iterations, probably because different strategies result in different orderings of the rows in the problem. Hence if the basis is degenerate, ties may be broken in different ways, and different columns will leave the basis.

Using the strategy of never employing Case 3 involves a tradeoff. The computations of Case 3 never have to be performed, and Case 2c is performed less often. This is because excess columns accumulate in the non-key section so the interchange of Case 2b is more likely to succeed. However if Case 3 is not performed, then the dimension of the working basis increases whenever Case 2c is performed, and never decreases. The overall result is that the number of iterations per second is lowest among all strategies tested. Hence it is desirable to repartition the working basis periodically. Among the other strategies tested, no consistent differences emerged.

REFERENCES