REPRINTED FROM

NAVAL RESEARCH LOGISTICS QUARTERLY

DECEMBER 1970
VOL. 17, NO. 4

OFFICE OF NAVAL RESEARCH

NAVSO P-1278

78

A GENERALIZED UPPER BOUNDING METHOD FOR DOUBLY COUPLED LINEAR PROGRAMS

James K. Hartman

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 Webber and White [10] and Heesterman [4], 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, U 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 interation. 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 U 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

$$\text{minimize } x_{01}$$

subject to

$$\bar{B}_0 x_0 + \sum_{i=1}^{p} \bar{A}_i x_i = b_0$$

$$\bar{D}_i x_0 + \bar{B}_i x_i = b_i \qquad i = 1, \ldots, p,$$

where x_i is a vector with n_i components, all of which must be nonnegative except for x_{01}, the first component of x_0. The column corresponding to x_{01} 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:

(1)

x_0	x_1	x_2	$\cdots$	x_{p-1}	x_p		No. of rows
$\bar{B}_0$	$\bar{A}_1$	$\bar{A}_2$	$\cdots$	$\bar{A}_{p-1}$	$\bar{A}_p$	$= b_0$	m_0
$\bar{D}_1$	$\bar{B}_1$					$= b_1$	m_1
$\bar{D}_2$		$\bar{B}_2$		ϕ		$= b_2$	m_2
$\vdots$						$\vdots$	$\vdots$
$\bar{D}_{p-1}$		ϕ		$\bar{B}_{p-1}$		$= b_{p-1}$	m_{p-1}
$\bar{D}_p$					$\bar{B}_p$	$= b_p$	m_p
S_0	S_1	S_2	$\cdots$	S_{p-1}	S_p		total = M rows
No. of columns							
n_0	n_1	n_2	$\cdots$	n_{p-1}	n_p	total = N columns	

The problem has p diagonal blocks of dimension $m_i \times n_i$ and an L-shaped border consisting of m_0 coupling constraints and n_0 coupling variables. The set S_i denotes either the variables in the vector x_i 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 (1). Arranging the columns of **B** in the same order as in (1), yields the basis matrix in Figure 1 where the shaded areas contain nonzero entries. The column for x_{00} is always in the basis and will always be the first basic column.

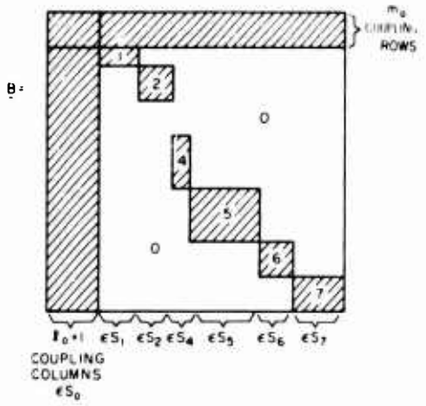


FIGURE 1

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

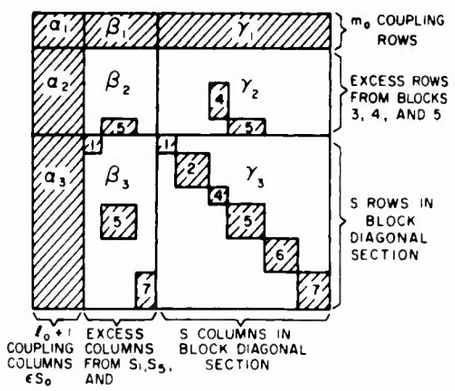


FIGURE 2

labeled the resulting nine submatrices for reference in Theorem 1. The submatrix γ_3 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 β_2.

For computational purposes the block diagonal section, γ_3 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 γ_3 of Figure 2, and M 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 $S \geq M - l_n - m_n$.

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

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

The block diagonal submatrix γ_3 is nonsingular, and hence has rank S. Consider the submatrix $[\beta_3, \gamma_3]$. It also has rank S since it has S linearly independent columns (those of γ_3) 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 $[\beta_3, \gamma_3]$ and, say, this row is an excess row from block j. By the special structure of this row (γ is zero except in block j) there must be at least one excess column in $\begin{bmatrix} \beta_2 \\ \beta_3 \end{bmatrix}$ 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 jth block of $[\beta_3, \gamma_3]$, this row and one of the excess columns can be adjoined to the jth 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 $[\beta_3, \gamma_3]$. This proves that the submatrix

$$K = \begin{bmatrix} \beta_2 & \gamma_2 \\ \beta_3 & \gamma_3 \end{bmatrix}$$

has rank S. Adjoin the first $l_n + 1$ columns to K giving the submatrix $L = \begin{bmatrix} \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{bmatrix}$. 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

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

Then $\mathbf{B}$ 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 $\mathbf{B}$ than in L. But rank $\mathbf{B} = M$, so that $M \leq S + l_n + m_n$, or $S \geq M - l_n - m_n$ 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 $\pi = c_B \mathbf{B}^{-1}$

b) Price out the nonbasic columns P_j.

$$\bar{c}_j = c_j - \pi P_j$$

and choose a column P_s with negative $\bar{c}_s$ to enter the basis if the current solution is not optimal.

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

$$\hat{P}_s = \mathbf{B}^{-1} P_s.$$

d) Determine the column to leave the basis.

$$\min_{\hat{a}_{is} > 0} \frac{x_{B_i}}{\hat{a}_{is}} = \frac{x_{B_r}}{\hat{a}_{rs}}$$

where $\hat{a}_{is}$ is component i of $\hat{P}_s$.

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

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

(2) $$\pi \mathbf{B} = c_B \text{ (solve for } \pi),$$

and

(3) $$\mathbf{B}\hat{P}_s = P_s \text{(solve for } \hat{P}_s)$$

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

Consider a nonsingular matrix $\mathbf{T}$ defined so that $\mathbf{BT} = \mathbf{R}$ is upper block triangular.

(4)
$$\begin{array}{c|c} G & H \\ \hline J & B_1 \end{array} \times \begin{array}{c|c} I & O \\ \hline V & I \end{array} = \begin{array}{c|c} B & H \\ \hline O & B_1 \end{array},$$

(B = G H / J B_1 × T = I O / V I = R = B H / O B_1)

where B_1 is the block diagonal section γ_B, G, H, and J are the remaining partitions of $\mathbf{B}$

$$\left(\text{e.g. } G = \begin{bmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \end{bmatrix}\right),$$

and

$$V = -B_1^{-1}J. \tag{5}$$

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

$$B = G + HV = G - HB_1^{-1}J. \tag{6}$$

By Theorem 1, **B** can always be partitioned so that the dimension of B is at most $m_0 + l_0$.

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

$$\pi\mathbf{R} = \pi\mathbf{BT} = c_B\mathbf{T} = (1, 0, \ldots, 0)\,\mathbf{T} = (1, 0, \ldots, 0). \tag{7}$$

The last equality holds since only x_{01} appears in the objective function, and by convention x_{01} is always the first basic variable. Since **T** is nonsingular, this multiplication will not change the solution π. Now partition π as $\pi = (\pi_0, \pi_1, \ldots, \pi_q)$, where

π_0 has $M-S$ components and is the multiplier for rows in the working basis B.

π_j has s_j components and is the multiplier for rows in the jth diagonal block B_{j1} which is $s_j \times s_j$ and nonsingular, $j = 1, \ldots, q$. From the structure of **R** in (4), Eq. (7) becomes

$$\pi_0 B = (1, 0, \ldots, 0) \quad (M-S \text{ components}), \text{ and} \tag{8}$$

$$\pi_0 H_j + \pi_j B_{j1} = 0 \quad (s_j \text{ components}) \qquad j = 1, \ldots, q, \tag{9}$$

where H_j is the submatrix of H containing the s_j columns in the jth block. Solving (8) gives

$$\pi_0 = (1, 0, \ldots, 0)B^{-1} = \text{first row of } B^{-1}, \tag{10}$$

which can be substituted into (9) to give

$$\pi_j = -\pi_0 H_j B_{j1}^{-1} \qquad j = 1, \ldots, q. \tag{11}$$

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. (3) for transforming the entering column in terms of the current basis. Define a nonsingular change of variables by

$$z = \mathbf{T}^{-1}P_s. \tag{12}$$

Then $P_s = \mathbf{T}z$, so Eq. (3) becomes

$$\mathbf{R}z = \mathbf{BT}z = B\hat{P}_s = P_s, \tag{13}$$

which is a block triangular system and hence easier to solve than (3). Partition z and P_s as $(z_0, z_1, \ldots, z_q)$ and $(P_{s0}, P_{s1}, \ldots, P_{sq})$, respectively. Then (13) can be written as

$$Bz_0 + \sum_{j=1}^{q} H_j z_j = P_{s0} \tag{14}$$

$$B_{j1} z_j = P_{sj} \qquad j = 1, \ldots, q. \tag{15}$$

Solving (15) gives

$$z_j = B_{j1}^{-1} P_{sj}, \tag{16}$$

which when substituted into (14) gives

$$z_0 = B^{-1}\left\{P_{s0} - \sum_{j=1}^{q} H_j z_j\right\}. \tag{17}$$

If P_s is not a coupling column, these formulas simplify since than $P_{sj} = 0$ 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 $\hat{P}_s$ from z using $\hat{P}_s = \mathbf{T}z$. With $\hat{P}_s$ partitioned as $(\hat{P}_{s0}, \hat{P}_{s1}, \ldots, \hat{P}_{sq})$ the structure of $\mathbf{T}$ in (4) gives

$$\hat{P}_{s0} = Iz_0 = z_0, \text{ and} \tag{18}$$

$$\hat{P}_{sj} = V_j z_0 + Iz_j = V_j z_0 + z_j \qquad j = 1, \ldots, q, \tag{19}$$

where V_j is the submatrix of V containing the rows in the jth 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 B^{-1}, B_{j1}^{-1} $(j = 1, \ldots, q)$, 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 $^*\mathbf{B}^{-1} = \mathbf{EB}^{-1}$, 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

$$^*B^{-1} = (E_1 + E_2 V)B^{-1}, \tag{20}$$

where E_1 and E_2 are partitions of $\mathbf{E}$ to be described in the proof.

PROOF: From the definition and nonsingularity of $\mathbf{R}$ and $\mathbf{T}$, in (4), (5)

$$^*\mathbf{R}^{-1} = {}^*\mathbf{T}^{-1}{}^*\mathbf{B}^{-1} = {}^*\mathbf{T}^{-1}\mathbf{EB}^{-1} = {}^*\mathbf{T}^{-1}\mathbf{ETR}^{-1}. \tag{21}$$

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

$$(22)\qquad \begin{bmatrix} {}^*B^{-1} & {}^*B^{-1}{}^*H^{-1}B_1^{-1} \\ O & {}^*B_1^{-1} \end{bmatrix} = \begin{bmatrix} I & O \\ -{}^*V & I \end{bmatrix} \begin{bmatrix} E_1 & E_2 \\ E_3 & E_4 \end{bmatrix} \begin{bmatrix} I & O \\ V & I \end{bmatrix} \begin{bmatrix} B^{-1} & -B^{-1}HB_1^{-1} \\ O & B_1^{-1} \end{bmatrix}.$$

To update the working basis inverse, note that it appears in the upper left hand partition of $^*\mathbf{R}^{-1}$. Hence deleting all but that submatrix gives the following specialization of (22):

$$(23)\qquad \begin{aligned} {}^*B^{-1} &= \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} E_1 & E_2 \\ E_3 & E_4 \end{bmatrix} \begin{bmatrix} I & O \\ V & I \end{bmatrix} \begin{bmatrix} B^{-1} \\ O \end{bmatrix} \\ &= \begin{bmatrix} E_1 & E_2 \end{bmatrix} \begin{bmatrix} B^{-1} \\ VB^{-1} \end{bmatrix} = (E_1+E_2V)B^{-1}. \end{aligned}$$

Thus E_1+E_2V is a transformation matrix for the working basis inverse.

Let the columns from the block diagonal section of $\mathbf{B}$ be called "key" columns. The remaining columns of $\mathbf{B}$ 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 $\mathbf{B}^{-1}$ is $^*\mathbf{B}^{-1}=\mathbf{E}\mathbf{B}^{-1}$. Here $\mathbf{E}$ is an elementary matrix, equal to the identity except in the rth column which is given by $[\eta_1, \eta_2, \ldots, \eta_M]'$, where

$$(24)\qquad \eta_i = \begin{cases} -\dfrac{\hat{a}_{is}}{\hat{a}_{rs}} & i=1, \ldots, M \\ & i \neq r, \end{cases}$$

$$\eta_r = \frac{1}{\hat{a}_{rs}}$$

(recall that the rth basic column is leaving the basis). Since the leaving column is non-key, partitioning $\mathbf{E}$ as in Theorem 3 gives $E_2=0$, and E_1 is an $M-S\times M-S$ elementary matrix equal to the identity except in the rth column which is $(\eta_1, \ldots, \eta_{M-S})'$.

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

$$(25)\qquad {}^*B^{-1} = (E_1+E_2V)B^{-1} = (E_1+0V)B^{-1} = E_1B^{-1}.$$

Hence 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_{jj} in B_1 will change. Thus the block inverses, B_{jj}^{-1}, require no updating. To update V recall that $^*V=-{}^*B_1^{-1}{}^*J$. Now $^*B_1^{-1}=B_1^{-1}$ as shown above, and $^*J$ is different from J only in the rth column which is replaced by the last S components of the entering column $(P_{s1}, \ldots, P_{sq})'$. Thus only the rth column of V will change, and this will be replaced by

$$-B_1^{-1}(P_{s1}, \ldots, P_{sq})' = -(z_1, \ldots, z_q)', \tag{26}$$

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_{j1}. Then if the entering column will leave B_{j1} nonsingular, a direct pivot can be performed and the basis change can be described by $^*\mathbf{B}^{-1} = \mathbf{E}\mathbf{B}^{-1}$, where $\mathbf{E}$ is an elementary column matrix. The $\mathbf{E}$ matrix differs from that in Case 1 only because the $(\eta_1, \ldots, \eta_M)'$ column is now in the key section of the matrix. Thus $E_1 = I_{M-S, M-S}$, and E_2 has only one nonzero column $(\eta_1, \ldots, \eta_{M-S})'$, which is the $r-(M-S)$th. To update the working basis inverse apply formula (20):

$$\begin{aligned} {}^*B^{-1} &= (E_1 + E_2 V)B^{-1} \\ &= \left(I + \begin{bmatrix} \eta_1 & & \\ \vdots & & 0 \\ \eta_{M-S} & & \end{bmatrix} V\right) B^{-1} \\ &= \left(I + \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_{M-S} \end{bmatrix} [v]\right) B^{-1} \\ &= B^{-1} + \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_{M-S} \end{bmatrix} vB^{-1}, \end{aligned} \tag{27}$$

where v is the $r-(M-S)$th row of V.

In the block diagonal submatrix B_1, only the $r-(M-S)$th column will change. Hence only one diagonal block inverse, say B_{j1}^{-1}, will change. B_{j1}^{-1} will be updated by a single pivot, or equivalently,

$$^*B_{j1}^{-1} = \hat{E}B_{j1}^{-1} \tag{28}$$

where $\hat{E}$ is an elementary column matrix formed from the elements of the new column. This change will leave B_{j1} 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 jth block of B_1^{-1} changes, so in $V = -B_1^{-1}J$ only the jth partition will change. The updated version is given by

$$^*V_j = -(^*B_{j1}^{-1})J_j = -\hat{E}B_{j1}^{-1}J_j = \hat{E}V_j, \tag{29}$$

so the same elementary matrix is used to update V.

CASE 2b: The column leaving the basis is a key column from block B_{j1}. 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_{j1} 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 kth basic column, then after the exchange $^*\mathbf{B} = \mathbf{B}\mathbf{E}$, where $\mathbf{E}$ is a permutation matrix, an identity matrix with the rth and kth columns exchanged. Since $\mathbf{E}^{-1} = \mathbf{E}$, $^*\mathbf{B}^{-1} = \mathbf{E}\mathbf{B}^{-1}$.

Then (20) yields ${}^*B^{-1} = (E_1 + E_2V)B^{-1}$, where E_1 is an $M-S \times M-S$ identity except for a zero in the kth diagonal position, and E_2 is zero except for a one in the kth row and $r-(M-S)$th column. Hence

(30) $${}^*B^{-1} = (E_1 + E_2V)B^{-1} = \begin{array}{|c|}\hline \begin{array}{ccccc} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & \end{array} \\ \hline \quad\quad r \quad\quad \\ \hline \begin{array}{cc} & \\ & \\ & \\ & 1 \\ & \end{array} \\ \hline \end{array} \begin{array}{l} B^{-1} \\ \leftarrow \text{row } k, \\ \\ \end{array}$$

where r is the $r-(M-S)$th row of V. This is a valid interchange if and only if $E_1 + E_2V$ is nonsingular, which is true if and only if the kth 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_{j1} will change. Hence the inverse can be updated by a simple pivot

(31) $${}^*B_{j1}^{-1} = \hat{E}B_{j1}^{-1},$$

where $\hat{E}$ is an elementary matrix. The elements needed to form the eta column in $\hat{E}$ are found in the kth column of V_j.

To update V, note that by definition ${}^*V = -{}^*B_1^{-1}{}^*J$. Only the jth block of B_1^{-1} changes, and a single column of J changes, but this column is zero outside the jth block J_j. Hence only the submatrix V_j needs to be updated,

(32) $${}^*V_j = -{}^*B_{j1}^{-1}{}^*J_j = -\hat{E}B_{j1}^{-1}{}^*J_j.$$

Now ${}^*J_j = J_j$ except in the kth column which is replaced by the leaving column which we will suppose to be the lth column in B_{j1}. Thus $B_{j1}^{-1}{}^*J_j = -V_j$ except in the kth column which becomes the lth unit vector, so that

(33) $${}^*V_j = \hat{E}V_j$$

except in the kth column which is the negative of the eta column of $\hat{E}$.

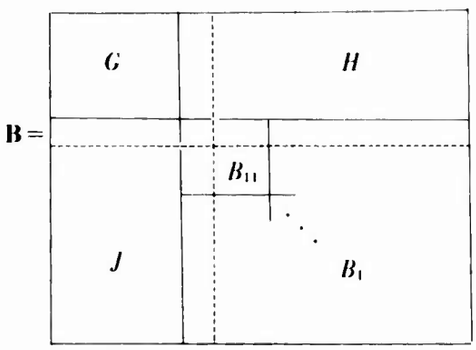
This completes the updating required for an exchange of key and non-key basic columns. After the exchange the leaving column is non-key, so Case 1 can be used to bring the new column into the basis. It should be noted that Case 1 uses the elements $\hat{a}_{is}$ of the transformed entering column (see (24)) and the vector $(z_1, \ldots, z_q)$ (see (26)). The interchange will affect these quantities. To update them interchange $\hat{a}_{rs}$ and $\hat{a}_{ks}$, and replace z_j by $\hat{E}z_j$.

CASE 2c: The column leaving the basis is a key column from block B_{j1}, and neither Case 2a nor 2b apply. When the column leaves, the new ${}^*B_{j1}$ 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_{j1} is made an excess row.

Without loss of generality assume that the pair being shifted is the first column and row of B_{11}.

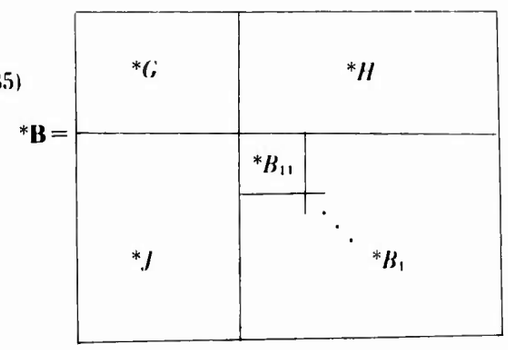
The basis matrix before and after the change are given below. All elements of the matrices are identical, only the partitioning changes.

(34)



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.

(35)



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 $^*\mathbf{B}=\mathbf{B}$, but because of the changes in partition sizes, $^*\mathbf{T}\neq\mathbf{T}$ and $^*\mathbf{R}\neq\mathbf{R}$. Hence we must compute $^*B_{11}^{-1}$, $^*B^{-1}$, and $^*V$.

$^*B_{11}^{-1}$ is computed via standard formulas. If

(36)

$$B_{11}^{-1}=\begin{array}{|c|c|}\hline W & X\\ \hline Y & Z\\ \hline\end{array}$$

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

(37)

$$^*B_{11}^{-1}=Z-\frac{YX}{W}.$$

This is possible only if $W\neq 0$ which provides a criterion for choosing the row to be shifted.†

To obtain $^*B^{-1}$ recall that $\mathbf{B}=^*\mathbf{B}$ so

(38)

$$^*\mathbf{R}^{-1}=^*\mathbf{T}^{-1}{}^*\mathbf{B}^{-1}=^*\mathbf{T}^{-1}\mathbf{B}^{-1}=^*\mathbf{T}^{-1}\mathbf{T}\mathbf{R}^{-1}.$$

†In general, if the kth column of B_{11} is being shifted, then the lth row can be shifted if the lth element of row k in B_{11}^{-1} is nonzero. There must always be at least one nonzero element in row k since B_{11}^{-1} is nonsingular.

In partitioned form this is

$$(39)\qquad \begin{array}{|c|c|}\hline {}^*B^{-1} & -{}^*B^{-1}{}^*H{}^*B_1^{-1} \\ \hline O & {}^*B_1^{-1} \\ \hline \end{array} = \begin{array}{|c|c|}\hline I & O \\ \hline -{}^*V & I \\ \hline \end{array} \begin{array}{|c|c|}\hline I & O \\ \hline V & I \\ \hline \end{array} \begin{array}{|c|c|}\hline B^{-1} & -B^{-1}HB_1^{-1} \\ \hline O & B_1^{-1} \\ \hline \end{array}.$$

To isolate ${}^*B^{-1}$ delete all but the upper left hand corner of this expression giving:

$$(40)\qquad \begin{array}{|c|}\hline {}^*B^{-1} \\ \hline \end{array} = \begin{array}{|c|c|}\hline I & O \\ \hline \end{array} \begin{array}{|c|c|}\hline I & O \\ \hline V & I \\ \hline \end{array} \begin{array}{|c|c|}\hline B^{-1} & h \\ \hline O & W \\ \hline & Y \\ \hline & O \\ \hline \end{array},$$

where h is the first column of $-B^{-1}HB_1^{-1}$ and $[W, Y]'$ is the first column of B_{11}^{-1}

$$= \begin{array}{|c|c|c|}\hline I & O & O \\ \hline v & 1 & \\ \hline \end{array} \begin{array}{|c|c|}\hline B^{-1} & h \\ \hline O & W \\ \hline & Y \\ \hline & O \\ \hline \end{array},$$

where v is the first row of V.

$$= \begin{array}{|c|c|}\hline B^{-1} & h \\ \hline vB^{-1} & vh + W \\ \hline \end{array}$$

The vector h is computed as

$$(41)\qquad \begin{aligned} h &= \text{first column of } B^{-1}HB_1^{-1} \\ &= B^{-1}H(\text{first column of } B_1^{-1}) \\ &= B^{-1}H(W \mid Y \mid O)' \\ &= B^{-1}H_1(W \mid Y)', \end{aligned}$$

where H_1 is the first partition of H.

We note that the repartitioning here changes all partitions of $\mathbf{B}$. Hence in the expression $B = G - HB_1^{-1}J$ for the working basis, all four factors change. Nevertheless, we need only add a "border" to B^{-1} to get ${}^*B^{-1}$. Starting from the equation ${}^*\mathbf{T}^{-1} = {}^*\mathbf{R}^{-1}\mathbf{R}\mathbf{T}^{-1}$ and arguing as above, a formula for updating V is obtained

$$(42)\qquad {}^{*}V=\left[\begin{array}{c|c}\hat{V}_1-\dfrac{Yv}{W} & \dfrac{Y}{W}\\ \hline \hat{V} & O\end{array}\right]$$

where v is the first row of V, $\hat{V}_1$ is the rest of the first block of V, and $\hat{V}$ is all other blocks of V. Note that ${}^{*}V$ has one less row and one more column than V 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

$$(43)\qquad \begin{array}{cc}\boxed{\alpha} & \boxed{\beta}\ \leftarrow \text{excess row}\\ \boxed{\gamma} & \boxed{B_{j1}}\ \leftarrow j\text{th diagonal block in } B_1.\\ \nwarrow & \\ \text{excess column} & \end{array}$$

Bringing the row and column into the key section would mean adding them to B_{j1} resulting in a larger jth diagonal block.

$$(44)\qquad {}^{*}B_{j1}=\left[\begin{array}{c|c}\alpha & \beta\\ \hline \gamma & B_{j1}\end{array}\right]$$

We already know B_{j1}^{-1} and we want to compute the inverse of the new block. If this inverse exists suppose that it is partitioned as

$$(45)\qquad {}^{*}B_{j1}^{-1}=\left[\begin{array}{c|c}W & X\\ \hline Y & Z\end{array}\right]$$

Then [3] the blocks are given by

$$(46)\qquad \begin{aligned} W &= 1/(\alpha-\beta B_{j1}^{-1}\gamma)\\ X &= -W\beta B_{j1}^{-1}\\ Y &= -B_{j1}^{-1}\gamma W\\ Z &= B_{j1}^{-1}-B_{j1}^{-1}\gamma X, \end{aligned}$$

and the inverse exists if and only if

$$(47)\qquad \alpha-\beta B_{j1}^{-1}\gamma\neq 0.$$

For computational purposes, note that $-B_{11}^{-1}\gamma$ is just the jth partition of the column in V corresponding to the column being moved. Hence to test for possible pairs to be moved requires computation of an inner product of β with a column of V_j.

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 ${}^*\mathbf{B}$ will be exactly the same as the elements of $\mathbf{B}$—only the partitions will change. The new working basis ${}^*B$ will become smaller. To find an expression for its inverse consider the expression

$$ {}^*\mathbf{R}^{-1} = {}^*\mathbf{T}^{-1}{}^*\mathbf{B}^{-1} = {}^*\mathbf{T}^{-1}{}^*\mathbf{B}^{-1} = {}^*\mathbf{T}^{-1}\mathbf{TR}^{-1}. \tag{48} $$

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

$$ \left[\begin{array}{c|c} {}^*B^{-1} & -{}^*B^{-1}{}^*H{}^*B_1^{-1} \\ \hline O & {}^*B_1^{-1} \end{array}\right] = \left[\begin{array}{c|c} I & O \\ \hline -{}^*V & I \end{array}\right] \left[\begin{array}{c|c} I & O \\ \hline V & I \end{array}\right] \left[\begin{array}{c|c} B^{-1} & -B^{-1}HB_1^{-1} \\ \hline O & B_1^{-1} \end{array}\right] \tag{49} $$

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

$$ {}^*B^{-1} = \left[\begin{array}{c|c} I & O \end{array}\right] \left[\begin{array}{c|c} I & O \\ \hline V & I \end{array}\right] \left[\begin{array}{c} \hat{B} \\ \hline b \\ \hline O \end{array}\right] = \left[\begin{array}{c} \hat{B} \end{array}\right] \tag{50} $$

where $\hat{B}$ is just B^{-1} 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 ${}^*V$ use the equation

$$ {}^*\mathbf{T}^{-1} = {}^*\mathbf{R}^{-1}\mathbf{B} = {}^*\mathbf{R}^{-1}\mathbf{B} = {}^*\mathbf{R}^{-1}\mathbf{RT}^{-1}. \tag{51} $$

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

$$ {}^*V \begin{bmatrix} B\phi \\ \bar{V}_1 \quad Y\phi \\ \hline \phi \end{bmatrix} \tag{52} $$

where V_1 is the first partition of V less the last column, and $\bar{V}$ is the remaining partitions of V less the last column. For this to be computed we must find ϕ which is the bottom row of the working basis less its last component ζ. The working basis is not carried along explicitly, so we must compute ϕ.

$B^{-1} = G \cdot HB$ from (6),

where G and H are partitions of the basis matrix $\mathbf{B}$ and V is updated at each iteration of the process.

$$
\begin{aligned}
(53)\qquad [\psi, \zeta] &= \text{last row of } B \\
&= \text{last row of } G + (\text{last row of } H) \cdot V \\
&= g + h \cdot V \quad (g = \text{last row of } G \\
&\qquad\qquad h = \text{last row of } H) \\
&= g + [\beta, 0] \cdot V \\
&= g + \beta \cdot V_1,
\end{aligned}
$$

and dropping the last element of this gives ψ.

In the above calculations the most complicated formulas were those for updating V in the two repartitioning procedures, Eqs. (42) and (52). An alternative procedure in these cases is to compute V from its definition $V = -B_1^{-1}J$. In each case, only one partition of V changes, say partition V_j, given by

$$
(54)\qquad {}^*V_j = -{}^*B_{j1}^{-1}{}^*J_j,
$$

with B_{j1} of dimension s_j. To compute this requires s_j^2 multiplications for each nonzero column of ${}^*J_j$, a total of at most $(M-S)\times s_j^2$ multiplications. The updating calculations require on the order of $(M-S)\times s_j$ 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_{j1} has dimension m_j (see (1)). 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 V simplify because each column of V 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 L 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 lth plant in the tth time period is hence limited by these K demand constraints and also by r constraints on locally available resources (e.g., plant capacity, labor) giving rise to a constraint block with $K + r$ rows. There are LT 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 $KLT(2L+T-3)/2$ 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 kth demand equation for one block, and a single -1 in the kth demand equation for another block. Hence they may be stored implicitly and priced out 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 $V=-B_1^{-1}J$ a coupling column has at most two nonzero partitions and these are columns of the block inverses B_{j1}^{-1}. An excess column has one nonzero partition in V. Hence it is probably best to compute V at each iteration instead of updating it. This is desirable since the most complicated update formulas are those involving V.

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 variables 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 M 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

Problem numbers	Number of products (K)	Number of plants (L)	Number of time periods (T)	Number of blocks (LT)	Block size	Number of coupling columns	Number of coupling rows	Total problem size
1, 2	5	3	4	12	7 x 22	210	6	90 x 474
3, 4	5	3	6	18	7 x 22	405	8	134 x 801
5, 6	5	3	8	24	7 x 22	660	10	178 x 1188
7, 8	5	3	10	30	7 x 22	975	12	222 x 1635
9, 10	5	5	6	30	7 x 22	975	8	218 x 1635
11, 12	5	5	8	40	7 x 22	1500	10	290 x 2380
13, 14	5	5	10	50	7 x 22	2125	12	362 x 3225

TABLE 2. Effect of Pricing Strategies

Problem number	Pricing strategy	Total iterations	Total time (sec)	Iterations (per sec)	Maximum size of working basis	Maximum number of coupling columns in basis
1	1	329	18.52	17.8	31	28
	3	178	5.05	35.2	9	3
3	1	440	36.88	11.9	37	34
	2	326	12.99	25.1	22	15
	3	274	9.39	29.2	10	4
5	1	Exceeded 60 second time limit				
	2	451	25.33	17.8	25	20
	3	321	14.43	22.2	15	6
7	1	Not attempted				
	2	557	38.07	14.6	27	21
	3	438	24.12	18.2	21	11

(All runs made with case 3 repartitioning attempted every 10 iterations)

TABLE 3. Effects of Repartitioning Strategies

Problem number	Repartitioning strategy	Total iterations	Total time (sec)	Iterations (per sec)	Number of occurrences of various cases in updating.					Final size working basis	Final No. coupling columns	Maximum size working basis	Maximum number coupling columns
					1	2a	2b	2c	3				
1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	178	4.95	36.0	82	96	25	5	0	11	0	11	3
	2	178	5.05	35.2	81	97	26	7	17	6	0	9	3
	3	178	4.60	38.7	81	97	26	7	35	6	0	9	3
	6	178	4.62	38.5	81	97	25	9	53	6	0	9	3
2	1	299	7.71	38.8	155	144	45	9	0	15	5	15	7
	2	311	7.66	40.6	163	148	50	17	31	10	5	11	8
	3	312	7.66	40.7	162	150	48	19	62	9	5	11	8
	6	311	7.72	40.3	162	149	49	20	64	10	5	12	8
5	1	322	16.17	19.9	113	209	38	9	0	19	0	19	6
	2	321	14.43	22.2	108	213	38	9	31	10	0	15	6
	3	321	13.49	23.8	108	213	38	9	64	10	0	15	6
	6	321	13.46	23.8	108	213	38	9	174	10	0	15	6
6	1	711	45.97	15.5	361	350	84	26	0	36	18	36	23
	2	733	39.81	18.4	379	354	82	63	73	22	16	25	20
	3	706	38.16	18.5	349	357	76	59	141	24	18	26	22
	6	675	36.33	18.6	322	353	69	78	426	23	17	28	23
9	1	421	24.99	16.8	192	229	62	13	0	21	4	21	12
	2	416	21.96	18.9	188	228	55	34	41	12	6	17	13
	3	416	21.66	19.2	188	228	55	34	83	10	6	17	13
	6	416	21.38	19.5	188	228	55	32	132	10	6	18	13
10	1	1023	62.14	16.5	638	385	200	25	0	33	15	33	26
	2	1034	54.13	19.1	656	378	175	77	103	21	15	29	26
	3	1025	54.78	18.7	646	379	177	77	205	21	15	26	23
	6	1023	53.01	19.3	650	373	171	85	705	21	15	27	22
11	1	570	46.35	12.3	277	293	84	15	0	25	5	25	14
	2	565	38.43	14.7	263	302	86	39	56	12	5	21	16
	3	565	38.42	14.7	263	302	86	39	113	12	5	21	16
	6	577	38.78	14.9	285	292	86	40	244	11	5	21	16
12	1	1194	92.68	12.9	748	446	233	28	0	38	17	38	23
	2	1209	84.14	14.4	764	445	217	76	121	23	16	30	26
	3	1208	89.49	13.5	766	442	225	81	241	25	17	33	27
	6	1231	87.02	14.1	784	447	220	98	900	25	18	33	26
13	1	754	96.01	7.9	390	364	108	26	0	38	7	38	19
	2	747	69.41	10.8	372	375	102	55	74	14	9	26	19
	3	751	68.44	11.0	377	374	99	61	150	14	9	26	19
	6	769	73.02	10.5	397	372	104	72	372	17	9	30	22
14	1	1364	137.38	9.9	787	577	258	81	0	43	20	43	26
	2	1390	123.29	11.3	790	600	238	79	139	28	20	33	26
	3	1389	118.84	11.7	807	582	247	82	278	27	20	34	28
	4	1351	119.74	11.3	750	601	237	89	1351	29	21	35	27
	5	1379	126.08	10.9	789	590	248	79	414	30	21	37	27
	6	1351	113.60	11.9	750	601	237	89	1066	29	21	35	27

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

[1] Dantzig, G. B. and R. M. Van Slyke, "Generalized Upper Bounded Techniques for Linear Programming," Journal of Computer and System Sciences **1**, 213–226 (1967).

[2] Grigoriadis, M. D. and K. Ritter, "A Decomposition Method for Structured Linear and Nonlinear Programs," Journal of Computer and System Sciences, **3**, 335–360 (1969).

[3] Hadley, G., *Linear Algebra* (Addison-Wesley, Reading, Mass., 1961).

[4] Heesterman, A. R. G., "Special Simplex Algorithm for Multisector Problems," Numerische Mathematik **12**, 288–306 (1968).

[5] Kaul, R. N., "An Extension of Generalized Upper Bounded Techniques for Linear Programs," ORC-65-27, Operations Research Center, University of California, Berkeley (1965).

[6] Lasdon, L. S., *Optimization Theory for Large Systems* (Macmillam Company, New York, 1970).

[7] Ritter, K., "A Decomposition Method for Linear Programming Problems with Coupling Constraints and Variables," MRC No. 739, Mathematics Research Center, University of Wisconsin (1967).

[8] Rosen, J. B., "Primal Partition Programming for Block Diagonal Matrices," Numerische Mathematik **6**, 250–260 (1964).

[9] Sakarovitch, M. and R. Saigal, "An Extension of Generalized Upper Bounding Techniques for Structured Linear Programs," SIAM Journal on Applied Mathematics **15**, 906–914 (1967).

[10] Webber, D. W. and W. W. White, "An Algorithm for Solving Large Structured Linear Programming Problems," IBM New York Scientific Center Report No. 320–2946 (1968).