Agbadudu, Amorosu; Balachandran, Bala V.

Working Paper

Multiple Working Bases for Generalized and Variable Upper Bounding Linear Programs

Discussion Paper, No. 290

Provided in Cooperation with:
Kellogg School of Management - Center for Mathematical Studies in Economics and Management Science, Northwestern University

Suggested Citation: Agbadudu, Amorosu; Balachandran, Bala V. (1977) : Multiple Working Bases for Generalized and Variable Upper Bounding Linear Programs, Discussion Paper, No. 290, Northwestern University, Kellogg School of Management, Center for Mathematical Studies in Economics and Management Science, Evanston, IL

This Version is available at:
https://hdl.handle.net/10419/220650

Discussion Paper No. 290

MULTIPLE WORKING BASES FOR GENERALIZED AND VARIABLE UPPER BOUNDING LINEAR PROGRAMS

by

Amorosu B. Agbadudu
and
Bala V. Balachandran

July 1977

MULTIPLE WORKING BASES FOR GENERALIZED AND VARIABLE UPPER BOUNDING LINEAR PROGRAMS

by

Amorosu B. Agbadudu

and

Bala. V. Balachandran

ABSTRACT

When a linear programming model is applied to a large system it usually leads to sub-programs with special structures. Such models that are formulated frequently in production problems are linear programs with particular structures where both generalized upper bound (GUB) constraints and/or variable upper bound (VUB) constraints are present in the constraint matrix.

A variant of the revised simplex method is given for these types of linear programs with $m_1 + m_2 + m_3$ constraints where m_1, the number of regular LP constraints, m_2 and m_3 of which are GUB constraints and VUB constraints respectively. The algorithm described uses two working bases of m_1 rows and m_2 rows for pivoting, pricing and for inversion.

1. Introduction

Various applications requiring a linear programming (LP) model often lend themselves to be formulated as a large system having special structures in their constraints. In particular, if the application is related to plant location, capital budgeting, assembly line balancing and other related production problems, the linear program model has special forms of constraints referred to as generalized upper bounding (GUB) constraints and variable upper bounding (VUB) constraints. An LP is defined to have a GUB constraint [2] if that constraint is of the form

$$\sum_{s \in S_j} x_s = 1 \qquad \text{for } j = 1,2,\ldots,m_2$$

where S_j is an index set. (The cardinality and other details of set S_j are given in page 2.)

Furthermore, if an LP has the constraint of the form

$$x_j \leq x_k$$

we say that the LP has a VUB constraint. In this case x_j has a variable upper bound of x_k. The variables x_j and x_k may appear in any number of such constraints. Our definition of VUB constraints thus differ from that of Schrage [4] where only the variable x_k (and not x_j) might appear in any number VUB constraints. Thus, due to our enlarged definition to that by Schrage [4], our version of VUB constraints can accomodate a linear programming model for problems like assembly line balancing, network and job scheduling. These problems contain precedence relations which are significantly large in number where any specific variable may be upper bounded by several variables.

In this paper a methodology for solving an LP which contains both GUB and VUB constraints is proposed. Specifically we are concerned with linear programming problems of the following form:

(I): Maximize $Z = c'x$

subject to

$$\underset{\text{rows}}{m_1}\left\{ \overbrace{w_1x_1+w_2x_2+\ldots+w_{i_1}x_{i_1}}^{S_1} \overbrace{+\ldots+w_{i_2}x_{i_2}}^{S_2} \overbrace{+\ldots w_{i_r}x_{i_r}}^{S_r} \overbrace{+\ldots+w_{i_{m_2}}x_{i_{m_2}}}^{S_{m_2}} \overbrace{+\ldots w_nx_n}^{S_o} = p \right. \qquad (1)$$

$$\underset{\text{rows}}{m_2}\left\{ \begin{array}{lll} x_1+x_2+\ldots+x_{i_1} & & = 1 \\ & x_{i_1+1}+\ldots+x_{i_2} & = 1 \\ & & = 1 \\ & +\ldots+x_{i_{m_2}} & \end{array} \right\} \text{GUB} \qquad (2)$$

$$\underset{\text{rows}}{m_3}\left\{ x_j \quad \leq x_k \right\} \text{VUB} \qquad (3)$$

where x and c are n-dimensional vectors with components $(x_1,x_2,\ldots,x_n)$ and $(c_1,c_2,\ldots,c_n)$ respectively; $w_r (r = 1,2,\ldots,n)$ and p are m_1-dimensional vectors. Also we have that $m_2 \leq n$, and the variables x_j and x_k can appear in either (1) or (2) or both. Note that $|S_r| = i_r - i_{r-1}$ for $r = 1,2,\ldots,;m_2$; and $i_o = 0$. Further let S be the set of all such sets so that $S = \{S_r | r = 1,2,\ldots,m_2\}$ and $S_i \cap S_j = \emptyset$, $i \neq j$. Let S_o be the complement of S, the index set of all variables. Let the indices of variables in S_o be $m_2 + 1,\ldots n$.

There are many applications of linear programming which naturally have the form (I) or can be easily transformed to form (I). Many of these examples are found in Agbadndu [1].

2. Multiple Working Bases

The GUB problem contains the structure of (1) and (2) only. A method of solving a LP with these structures had been developed by Dantzig and Van Slyke [2]. Recently Schrage [4] considered the LP with VUB constraints and provided an algorithm dealing with a problem having structure (1) and modified form of (3). No one to our knowledge has considered the more general case of both VUB and GUB constraints put together.

In this paper we provide a procedure to solve the above problem (Form (I)) with structures (1), (2) and (3) simultaneously, in the framework of simplex algorithm, so that the two methods mentioned above may be considered as special cases. Basically we maintain two working bases which we shall refer to as "inner working basis" R and "outer working basis", H. An advantage of using two working bases as against using one is that there can be substantial saving when either the GUB or VUB constraints are significantly larger in number compared to the usual constraints. This is because we invert two smaller submatrices independently instead of a single large matrix whose dimension is the sum of dimensions of the smaller submatrices.

Matrix structure of the problem:

Let us denote by A the complete matrix of coefficients for the LP. A is partitioned as

$$A = \begin{bmatrix} \overline{W} \\ \overline{U} \\ \overline{V} \end{bmatrix} \tag{4}$$

where $\overline{W}$ has m_1 rows corresponding to the usual constraints,

$\overline{U}$ has m_2 rows corresponding to the GUB constraints,

$\overline{V}$ has m_3 rows corresponding to the VUB constraints.

Theorem I: Assume that a feasible basis B exists for this LP. Then one can always permute only the columns of this matrix so that an identity submatrix appears both in the GUB rows and VUB rows. That is, we can write the basis matrix B as:

$$B = \begin{bmatrix} W & Q & \underline{Q} \\ G & I_u & U \\ \underline{G} & V & I_v \end{bmatrix} . \tag{5}$$

Where the identity submatrices I_u and I_v have as many rows as there are GUB constraints and VUB constraints respectively. The other submatrices are as follows:

W is an $m_1 \times m_1$ matrix of coefficients of variables that are basis in the usual constraints $\overline{W}x = p$.

G is an $m_2 \times m_1$ matrix of zero's and one's corresponding to the variables that are basic in m_1 rows but appear in GUB rows.

$\underline{G}$ is an $m_3 \times m_1$ matrix of zero's, one's and one's corresponding to the variables that are basic in m_1 rows but appear in VUB rows.

Q is an $m_1 \times m_2$ matrix of coefficients of variables that are basic in GUB rows but appear in the first m_1 rows.

$\underline{Q}$ is an $m_1 \times m_3$ matrix of coefficients of variables that are basic in VUB rows but appear in m_1 rows.

U is an $m_2 \times m_3$ matrix of zero's and one's corresponding to the variables that are basic in VUB rows but appear in GUB rows.

V is an $m_3 \times m_2$ matrix of zero's, one's and one's that are basic in GUB rows but also present in VUB rows.

Proof: We note that for every GUB constraint one of the variables must be in the basis for the solution to be feasible. By GUB condition, this variable does not have a coefficient in any other GUB row. Therefore the columns can be arranged so that this single nonzero zppears on the diagonal of the submatrix within B; hence we have I_u. For any practical problem for which the RHS constant term in a VUB constraint is zero, we can without loss of generality assume the RHS is small positive number ϵ. (That is if we assume no degeneracy.) Thus, for every VUB constraint one of the variables on the LHS with positive sign must also be in basis for the solution to be feasible. Since that variable must be basic only in that row, the columns can be permuted too, so that this single nonzero appears on the diagonal of the submatrix, and hence I_v.

Corollary: Suppose the LP were to have structures (2) and (3) only. If M is the feasible basis for this LP, then one can also arrange only the columns

of this matrix so that identity submatrices appear in the diagonal. That is, we permute M as:

$$M = \begin{bmatrix} I_u & U \\ V & I_v \end{bmatrix} \quad ; \tag{6}$$

M is order L where $L = m_2 + m_3$.

3. The Working Bases

Define a nonsingular matrix $\quad \bar{T} = \begin{bmatrix} I_u & 0 \\ -V & I_v \end{bmatrix}$ (7)

such that $M\bar{T} = R^*$ is an upper block triangular matrix:

$$\underset{M}{\begin{bmatrix} I_u & U \\ V & I_v \end{bmatrix}} \times \underset{\bar{T}}{\begin{bmatrix} I_u & 0 \\ -V & I_v \end{bmatrix}} = \underset{R^*}{\begin{bmatrix} R & U \\ 0 & I_v \end{bmatrix}} \tag{8}$$

By transforming M into the form whereby the lower left submatrix of the transformation is zero, we are able to eliminate from the GUB rows the variables which are basic in the VUB rows. The matrix R will be called inner working basis. From (8) we see that

$$R = [I_u - UV] \tag{9}$$

Rewriting B matrix as

$$B = \begin{bmatrix} W & Q^* \\ G^* & M \end{bmatrix} , \tag{10}$$

where

$$Q^* = [Q, \underline{Q}]; \quad G^* = \begin{bmatrix} G \\ \underline{G} \end{bmatrix} \tag{11}$$

and M is as defined in (6). Also define a nonsingular matrix

$$T^* = \begin{bmatrix} I_{m_1} & 0 \\ D & I \end{bmatrix} \tag{12}$$

with

$$D = -M^{-1}G^* \tag{13}$$

and I, an identity matrix of order $m_2 + m_3$. Let T^* be defined in such a way that $BT^* = H^*$ is an upper block triangular matrix,

$$\underset{B}{\begin{bmatrix} W & Q^* \\ G^* & M \end{bmatrix}} \times \underset{T^*}{\begin{bmatrix} I_{m_1} & 0 \\ D & I \end{bmatrix}} = \underset{H^*}{\begin{bmatrix} H & Q^* \\ 0 & M \end{bmatrix}} \tag{14}$$

By this operation we transform B into the form whereby the lower left submatrix of the transformation is zero. And thus we are able to eliminate <u>simultaneously</u> from both the GUB rows and VUB rows the variables which are basic in those two sets of rows.

The matrix H is central to the procedure and will be called the <u>outer working basis</u>. As it can be seen below, H is of order m_1, the number of rows corresponding to the main constraints. From (13) and (14) we see that

$$H = [W - Q^*M^{-1}G^*] \tag{15}$$

Theorem 2: The working bases H and R are nonsingular.

Proof: We shall prove this for H. (The proof for R follows the similar pattern.) Suppose the columns of H are dependent then so are the first m_1 columns of BT^*, since they are formed by adding zeros to the columns of H. Thus BT^* is singular. Hence $\det(BT^*) = 0$. (16)

But $\det(BT^*) = \det(B)\det(T^*)$. Since $\det(T^*) = 1$ it follows that $\det(BT^*) = 0 \rightarrow \det(B) = 0$. But B is a basis matrix, hence nonsingular, so we have contradiction.

Q.E.D.

Recall that $M = \begin{bmatrix} I_u & U \\ V & I_v \end{bmatrix}$

Hence $$M^{-1} = \begin{bmatrix} R^{-1} & -R^{-1}U \\ -VR^{-1} & I_V + VR^{-1}U \end{bmatrix} , \quad (17)$$

Where $$R^{-1} = (I_U - UV)^{-1} . \quad (18)$$

Therefore $$Q^*M^{-1} = [(Q - \underline{Q}V)R^{-1} \quad \underline{Q} - (Q - \underline{Q}V)R^{-1}U] \quad (19)$$

And $$M^{-1}G^* = \begin{bmatrix} R^{-1}(G - U\underline{G}) \\ -VR^{-1}(G - U\underline{G}) + \underline{G} \end{bmatrix} ; \quad (20)$$

So that $$Q^*M^{-1}G^* = \underline{QG} + (Q - \underline{Q}V)R^{-1}(G - U\underline{G}) \quad (21)$$

Hence $$H = [W - \underline{QG} - (Q - \underline{Q}V)R^{-1}(G - U\underline{G})] \quad (22)$$

and it is $m_1 \times m_1$ matrix.

Our approach to the solution of the problems of form (I) is to adapt the revised primal simplex method with special attention to the multiple bases.

The simplex algorithm [5] involves the following steps at each iteration:

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

(b) Price out the nonbasic columns a_j,

$$\pi a_j - C_j = Z_j - C_j$$

and choose a column a_j with most negative to enter the basis if the current solution is not optimal.

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

$$\hat{a}_j = B^{-1} a_j.$$

(d) Determine the column to leave the basis,

$$\min_{\hat{a}_{ij}>0} \frac{x_{Bi}}{\hat{a}_{ij}} = \frac{x_{Br}}{\hat{a}_{rj}}$$

where $\hat{a}_{ij}$ is component i of $\hat{a}_j$.

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

Steps (a) and (c) involve multiplication by the basis matrix B^{-1}. Even though B has special structure, B^{-1} is essentially dense with non-zero elements. So if one wants to maintain B^{-1} for large problems it will require extensively storage and computation. Instead we express B^{-1} in terms of other smaller matrices.

Adaptation of Simplex Multipliers $\pi = C_B B^{-1}$ in the Algorithm.

Let us partition π as $(\pi^1 \pi^L)$ where π^1 is the first m_1 vector of simplex multipliers for the first m_1 rows (corresponding to the usual constraints), $\pi^L = (\pi^2 \pi^3)$ is the last m_2 and m_3 vectors of simplex multipliers (corresponding to GUB rows and VUB rows respectively). Further

partition the price coefficients of the basis variables, C_B, as $(C^1 C^L)$, where C^1 is the price vector of coefficients for the basic variables in the first m_1 constraints and $C^L = (C^2 C^3)$ are the price vectors (corresponding to GUB and VUB rows respectively). To find the simplex multipliers, solve the equation:

$$\pi B = C_B \tag{23}$$

Multiplying both sides by T^* we obtain

$$\pi BT^* = C_B T^*,$$

but BT^* by definition is H^*

$$\therefore \pi H^* = C_B T^*$$

that is, $(\pi^1 \pi^L)\begin{pmatrix} H & Q^* \\ 0 & M \end{pmatrix} = (C^1 C^L)\begin{pmatrix} I & 0 \\ D & I \end{pmatrix}$,

Or $$\pi^1 H = C^1 - C^L M^{-1} G^* \tag{24}$$

and $$\pi^1 Q^* + \pi^L M = C^L \tag{25}$$

From (8) $$\pi^1 = (C^1 - C^L M^{-1} G^*) H^{-1} \tag{26}$$

Substituting this value into (25) we have that

$$\pi^L = (C^L - \pi^1 Q^*) M^{-1} = C^L M^{-1} - \pi^1 Q^* M^{-1} \tag{27}$$

Using expression (17) for M^{-1}, we see that only the inverse of R, the inner working basis is required for the terms M^{-1}, $Q^* M^{-1}$ and $G^* M^{-1}$. Thus to obtain the simplex multipliers it suffices to know the inverse of the working bases only.

The evaluation of reduced costs for non-basic variables is made by pre-multiplying a non-basic column by $C_B B^{-1}$ or $(\pi^1 \pi^L)$ as given by expressions (26) and (27). Because the last $L = m_2 + m_3$ elements of the non-basic column are e_i^G and V_j, the calculation of $C_B B^{-1} a_j - C_j$ (that is, $Z_j - C_j$) reduces to

$$C_B B^{-1} a_j - C_j = \pi' w_j + \pi^2 e_i^G + \pi^3 V_j - C_j \tag{28}$$

where

$$\pi^1 = [C^1 - C^3 \underline{G} - C^2 R^{-1}(G - U\underline{G}) + C^3 V R^{-1}(G - U\underline{G})] H^{-1}$$

$$\pi^2 = [(C^2 - C^3 V) - \pi^1 (Q - \underline{Q}V)] R^{-1}$$

$$\pi^3 = C^3 - \pi^1 \underline{Q} - \pi^2 U$$

i.e., by using equations (19), (20), (21).

Analysis of Incoming Vector a_j

Let us partition the vector a_j as $a_j = \begin{bmatrix} w_j \\ e_i^* \end{bmatrix}$. Here w_j is the first m_1 component of a_j (which is a part of a_j corresponding to the first m_1 rows) and e_i^* is the last L components of a_j; it is that part of a_j that corresponds to the last $m_2 + m_3$ rows. From step (c) we see that

$$\hat{a}_j = B^{-1} a_j \tag{29}$$

Since T^* is nonsingular T^* inverse exists. So if we premultiply both sides of the above expression by T^{*-1} we get $T^{*-1}\hat{a}_j = T^{*-1} B^{-1} \hat{a}_j = H^{*-1} a_j$

(since $BT^{*} = H^{*}$) that is,

$$\begin{bmatrix} I & 0 \\ -D & I \end{bmatrix} \begin{bmatrix} w_j \\ \hat{e}_i^{*} \end{bmatrix} = \begin{bmatrix} H^{-1} & H^{-1}Q^{*}M^{-1} \\ 0 & M^{-1} \end{bmatrix} \begin{bmatrix} \alpha_j \\ e_i^{*} \end{bmatrix} \tag{30}$$

or

$$\hat{w}_j = H^{-1}w_j - H^{-1}Q^{*}M^{-1}e_i^{*} = H^{-1}(w_j - Q^{*}M^{-1}e_i^{*}) \tag{31}$$

and

$$M^{-1}G^{*}\hat{w}_j + \hat{e}_i^{*} = M^{-1}e_i^{*} \tag{32}$$

Substituting the value $\hat{w}_j$ of (31) into (32) we have

$$\hat{e}_i^{*} = M^{-1}e_i^{*} - M^{-1}GH^{-1}(w_j - Q^{*}M^{-1}e_i^{*}) \tag{33}$$

Therefore

$$\hat{a}_j = B^{-1}a_j = \begin{bmatrix} H(w_j - Q^{*}M^{-1}e_i^{*}) \\ M^{-1}e_i^{*} - M^{-1}GH^{-1}(w_j - Q^{*}M^{-1}e_i^{*}) \end{bmatrix} \tag{34}$$

Where

$$e_i^{*} = \begin{bmatrix} e_i^{G} \\ V_j \end{bmatrix} \tag{35}$$

is partitioned in such a way that the first m_2 components e_i^{G} is a unit vector of order m_2 the number of GUB rows, and i is the index of

the set $S_i (i = 1,2,\ldots,m_2)$ which contains the incoming variable. The last m_3 components V_j is the column of $\bar{V}$ that contains x_j.

Now consider the first m_1 rows of $B^{-1}a_j$ in (34). We note that from equation (19)

$$Q^* M^{-1} = [(Q - \underline{Q}V)R^{-1}\underline{Q} - (Q - \underline{Q}V)R^{-1}U]. \quad \text{Hence}$$

$$Q^* M^{-1} e_i^* = [(Q - \underline{Q}V)R^{-1}\underline{Q} - (Q - QV)R^{-1}U] \begin{bmatrix} e_i^G \\ V_j \end{bmatrix}$$

$$= (Q - \underline{Q}V)R^{-1}e_i^G + \underline{Q}V_j^3 - (Q - \underline{Q}V)R^{-1}UV_j \tag{36}$$

The nonzero elements of V_j are the rows of $\bar{V}$ where x_j appear. But the corresponding columns of $\underline{Q}$ and U are zero, since x_j is the entering variable. Therefore the products $\underline{Q}V_j$ and UV_j are all zero. Hence,

$$Q^* M^{-1} e_i^* = (Q - \underline{Q}V)R^{-1}e_i^G \tag{37}$$

Also from equation (20)

$$M^{-1}G^* = \begin{bmatrix} R^{-1}(G - U\underline{G}) \\ - VR^{-1}(G - U\underline{G}) + \underline{G} \end{bmatrix} \tag{38}$$

We also have from equations (17) and (35), that

$$M^{-1}e_i^* = \begin{bmatrix} R^{-1} & - R^{-1}U \\ - VR^{-1} & I_v + VR^{-1}U \end{bmatrix} \begin{bmatrix} e_i^G \\ V_j^3 \end{bmatrix} = \begin{bmatrix} R^{-1}e_i^G \\ V_j - VR^{-1}e_i^G \end{bmatrix} \tag{39}$$

Hence using equations (38) and (39) we have that

$$M^{-1}e_i^* - M^{-1}G^*H^{-1}(w_j - Q^*M^{-1}e_i^G)$$

$$= \begin{bmatrix} R^{-1}e_i^G & R^{-1}(G - U\underline{G}) \\ - & \\ V_j - VR^{-1}e_i^G & - VR^{-1}(G - U\underline{G}) \end{bmatrix} [H^{-1}(w_j - (Q - \underline{Q}V)R^{-1}e_i^G]$$

$$= \begin{bmatrix} R^{-1}e_i^G - R^{-1}(G - U\underline{G})H^{-1}[w_j - (Q - \underline{Q}V)R^{-1}e_i^G] \\ - VR^{-1}e_i^G + [VR^{-1}(G - U\underline{G}) - \underline{G}]H^{-1}(w_j - (Q - \underline{Q}V)R^{-1}e_i^G + V_j \end{bmatrix} \quad (40)$$

$$B^{-1}a_j = \begin{bmatrix} H^{-1}(w_j - (Q - \underline{Q}V)R^{-1}e_i^G \\ R^{-1}e_i^G - R^{-1}(G - U\underline{G})H^{-1}(w_j - Q - (\underline{Q}V)R^{-1}e_i^G \\ - VR^{-1}e_i^G + [VR^{-1}(G - U\underline{G}) - \underline{G}]H^{-1}(w_j - (Q - \underline{Q}V)R^{-1}e_i^G + V_j \end{bmatrix} \quad (41)$$

As we can see from the above expression, in determining a_j all that is needed here is to know the inverses of the working bases.

The Adaptation Of The Right Hand Side (Current Value of b).

Let us now determine the right hand side of our equation at each iteration. Let b be the value of this r.h.s. Further b can be partitioned as

$$b = \begin{bmatrix} p \\ 1^* \end{bmatrix}$$

where p corresponds to the first m_1 rows, and 1^* is $m_2 + m_3$ column vector with its first m_2 components having values 1 and the last m_3 components that corresponds to VUB rows have value zero. That is,

$$1^* = \begin{bmatrix} e \\ 0 \end{bmatrix} \tag{42}$$

where e is vector of one's (order m_2) and 0 is vector of zeros (order m_3).

The current value of b denoted by $\hat{b}$ is given by

$$\hat{b} = B^{-1}b \tag{43}$$

Just as in the case of the last section it can be shown that

$$B^{-1}b = \begin{bmatrix} H^{-1} & H^{-1}Q^*M^{-1} \\ -M^{-1}G^*H^{-1} & M^{-1} + M^{-1}G^*H^{-1}Q^*M^{-1} \end{bmatrix} \begin{bmatrix} p \\ 1^* \end{bmatrix}$$

or

$$B^{-1}b = \begin{bmatrix} H^{-1}(p - Q^*M^{-1}1^*) \\ M^{-1}1^* - M^{-1}G^*H^{-1}(p - Q^*M^{-1}1^*) \end{bmatrix} \tag{44}$$

Since $Q^*M^{-1} = [(Q - \underline{Q}V)R^{-1}\ \underline{Q} - (Q - \underline{Q}V)R^{-1}U]$, using Equation (42)

$$Q^*M^{-1}1^* = (Q - \underline{Q}V)R^{-1}e \tag{45}$$

Similarly

$$M^{-1}1^* = \begin{bmatrix} R^{-1} \\ VR^{-1}e \end{bmatrix} \tag{46}$$

Thus the first m_1 components of $B^{-1}b$ is $H^{-1}(P - (Q - \underline{Q}V)R^{-1}e)$ and the last $m_2 + m_3$ components of $B^{-1}b$ become

$$\begin{bmatrix} R^{-1}e - R^{-1}(G - U\underline{G})H^{-1}(p - (Q - \underline{Q}V)R^{-1}e) \\ \\ - VR^{-1}e + [VR^{-1}(G - U\underline{G})H^{-1}(p - (Q - \underline{Q}V)R^{-1}e) \end{bmatrix}$$

Hence the current value of b, is given by the expression $\hat{b} = B^{-1}b$ where

$$B^{-1}b = \begin{bmatrix} H^{-1}(p - (Q - \underline{Q}V)R^{-1}e \\ \\ R^{-1}e - R^{-1}(G - U\underline{G})H^{-1}[p - (Q - \underline{Q}V)R^{-1}e \\ \\ -VR^{-1}e + [VR^{-1}(G - U\underline{G}) + \underline{G}]H^{-1}(p - \underline{Q}V)R^{-1}e \end{bmatrix} \tag{47}$$

Thus like the previous cases, what we need are the inverses of the working bases.

4. Updating Basis and Pivoting Arrangement.

Although pricing out of columns to select the entering column and choosing the column to leave the basis are done as in the ordinary revised simplex method, the updating requirements are somewhat different from those of the standard simplex algorithm. This is because not the entire B^{-1} is needed; rather only the inverse of the working bases H and/or R (sometimes one or none) are needed.

In all cases if we represent the slack variables associated with VUB constraints as basic, and artificial variables associated with GUB constraints

as basic, and choosing the basic variables for the first m_1 constraints in the usual way, then submatrices Q, U, G and V will be null matrices to start with (see equation (5)). The updating procedure here is to keep these matrices null throughout the entire iterative process. With this arrangement, the first $m_1 + m_2$ rows of $\hat{a}_j$ is given by

$$\hat{a}_j = B^{-1}a_j = \begin{bmatrix} H^{-1}(w_j - QR^{-1}e_i) \\ \\ R^{-1}e_i - R^{-1}GH^{-1}(w_j - QR^{-1}e_i \end{bmatrix}$$

and

$$\hat{b} = B^{-1}b = \begin{bmatrix} H^{-1}(p - QR^{-1}e) \\ \\ R^{-1}e - R^{-1}GH^{-1}(p - QR^{-1}e) \end{bmatrix}$$

The corresponding entries for m_3 rows will be discussed later on.

Now consider the basic variable that is being driven to zero. The position of this variable can be in one of the main rows, or in one of the GUB rows, or in one VUB rows. Consider the following cases:

(1) The departing variable is basic in one of the main rows

(2) The departing variable is basic in one of the GUB rows and

  (a) the entering variable is from the same row, or

  (b) otherwise.

(3) The departing variable is basic in one of VUB rows, and

  (a) the entering variable is from the same row, or

  (b) otherwise.

We note that if we introduce the slack variable, y_v we can write the VUB constraint $x_j \leq x_k$ as

$$\text{row V:}\quad x_j + y_v - x_k = 0 \tag{48}$$

We can further <u>impose</u> here that always x_j or y_v be basic in row V. With this assumption, whenever x_j is basic in any VUB rows and x_k basic in either GUB or Main rows, x_j can be substituted for from the expression:

$$x_j = x_k - y_v.$$

In this case, the coefficient of x_k, w_k is replaced by $w_k + w_j$ and the coefficient of y_v, 0 is replaced by $-w_j$ and so on. (Similar replacements with U's can be made for the last m_2 rows, e.g. $U_k + U_j$ for U_k and $-U_j$ for 0 respectively.)

<u>Remark.</u> When a basic variable leaves either a GUB row or a VUB row, it is possible for the entering variable to have a zero coefficient in that row. In this case we bring in a basic variable from only a main row, and make that basic variable from the main row in the VUB or GUB row considered. Such variable we shall hereafter refer to as a "suitable variable."

Let d = departing column number associated with departing variable x_d

e = entering column number associated with entering variable x_e

p = row in which the departing column (or variable) is basic

t = column of a "suitable variable" x_t that is basic in a main row

r = row in which x_t is basic

q = row in which x_k may be basic.

(Notice that q can be found between m_1 through m_2 rows only.)

Case (1). First compute column $\hat{e} = e - QR^{-1}e_i$. Then replace column d in H, G by column $\hat{e}$. Pivot on column d with row p as pivoting row to give a new inverse of H. R is unaffected in this operation which is confined to first m_1 rows. Here e_i is same as e_i^G.

Case (2). (a) Simply replace column d in Q and R by column e. Pivot on column d in matrix R with row p as pivot row. This gives a new R^{-1}. H is unaffected in this operation which is confined with GUB rows only.

(b) Replace column d in Q and R by column t. Pivot on column d on matrix R using row p as pivot row to obtain a new R^{-1}. Compute $\hat{e} = e - QR^{-1}e_i$; then replace column t of H and G by column $\hat{e}$. Pivot column t of matrix H using row r as pivoting row to obtain a new H^{-1}. Here we have two separate pivots on two different bases.

Case (3) (a); (i). $x_k \notin$ GUB set and it is not basic. Simply replace column d in I_v by column e. No pivot is done here.

(ii). $x_k \notin$ GUB set but it is basic. Replace column d of I_v by column e and update as follows:

$$\hat{w}_k = w_k + w_e \quad \text{and} \quad \hat{U}_k = U_k + U_e$$

$$\hat{w}_d = - w_e \quad \text{and} \quad \hat{U}_d = - U_e$$

$$\hat{w}_e = 0 \quad \text{and} \quad \hat{U}_e = 0$$

H is pivoted on column k using $q^{\underline{th}}$ row as pivot row to give a new H inverse. R is not affected.

(iii). $x_k \in$ GUB set. Replace column d of I_v by column e, and update as in case (3a) (ii). However H is not affected here, but R is affected if x_k is basic in a GUB row. In that case a pivot on column k using $q^{\underline{th}}$ row as a pivot row will give a new R inverse.

3(b),(i). $x_k \notin$ GUB and it is not basic. Replace column d by column t in I_v. Compute $\hat{e} = e - QR^{-1}e_i$. Replace column t in H and G by column $\hat{e}$. Using $r^{\underline{th}}$ row as pivot row, pivot on column t of H to get a new inverse of H.

(ii). $x_k \notin$ GUB but it is basic. Do the same replacement as in (3b),(i) and update as follows:

$$\hat{w}_k = w_k + w_j \qquad \text{and} \qquad \hat{U}_k = U_k + U_j$$

$$\hat{w}_d = w_t \qquad \text{and} \qquad \hat{U}_d = U_t$$

$$\hat{w}_t = 0 \qquad \text{and} \qquad \hat{U}_t = 0$$

(Also compute $\hat{e} = e - QR^{-1}e_i$.) Two pivots are needed here. One in column t using $r^{\underline{th}}$ row and the second in column K using $q^{\underline{th}}$ row as pivot row.

(iii). $x_k \in$ GUB. Replace column d by column t in I_v and replace column t by column e in matrices Q and R. R does not change from previous iteration. Update as in case 3(b),(ii). If x_k is basic in one of the GUB rows then one is needed for R too, using column k and row q.

The above updating process thus replaces numerous matrix multiplications by matrix additions. This savings in operations and reduction in size of the basis is found to be computationally very significant [1].

Transformation of the Entering Column: - VUB Rows.

Three cases are noted here:

(a) x_k is not basic. Then $B^{-1}a_j = V_j$

(b) x_k is basic in row r (main or GUB row).

$$B^{-1}a_j = w_{rj} + V_j \quad (\text{or } U_{rj} + V_j \text{ if r is in a GUB row}).$$

(c) x_k is basic in row r

x_j is basic in row t

$B^{-1}a_j = w_{rj} - w_{tj}$ (or $U_{rj} - U_{tj}$ if r is in a GUB row).

The corresponding current value of the right-hand side,

$\hat{b} = B^{-1}b$ are given by

(a) $B^{-1}b = 0$

(b) $B^{-1}b = w_{ro}$

(c) $B^{-1}b = w_{ro} - w_{to}$

respectively, where right-hand side is column zero.

5. Numerical Results

Computer code for the algorithm was written in APL.[1/] Twenty randomly generated problems of the form I were solved. Computations per iteration [2/] (following the usual operation count practice, e.g. Hadley [3]), number of iteration per problem as well as storage requirement of these problems are shown. These are compared against the result obtained by solving the same problem by the standard revised simplex method. As expected, these show that our algorithm is actually more efficient for problems of form I. The results are presented in the table below.

1/ APLUM System -- running on the CDC 6400, Computer Centre, Northwestern University.

2/ For detailed computation of number of operation per iteration see [1].

Algorithm	Group	Number of Problem in the Group	Number of Variables per Problem	Number of Constraints per Problem	Minimum Storage Requirements	Average Number of Iteration per Problem	Number of Operation per Iteration	Average Computing Time (CPU seconds)
Revised Simplex	1	10	39	19	361	19.3	1159	1.959
	2	10	55	25	625	18.4	2075	2.788
The Algorithm	1	10	39	19	235	9.4	563	0.867
	2	10	55	25	335	10.3	860	1.523

REFERENCES

[1] AGBADUDU, A. B., "Large Scale Systems with Generalized Upper Bounding, Variable Upper Bounding and Generalized Variable Upper Bounding Constraints." Unpublished Ph.D. dissertation submitted to the Northwestern Graduate School, June 1977.

[2] DANTZIG, G. B. and SLYKE, R. M., "Generalized Upper Bound Techniques," Journal of Computer and System Sciences, 1, 1967.

[3] HADLEY, G., Linear Programming, Addison-Wesley, Reading, Ma., 1962.

[4] SCHRAGE, L., "Implicit Representation of Variable Upper Bounds in Linear Programming," Mathematical Programming Study 4, 1975.

[5] SIMONNARD, M., Linear Programming, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1966.