O.N.R. Research Memorandum No. 128

A HYBRID-DUAL INTEGER PROGRAMMING ALGORITHM

by

Fred Glover

July, 1964

Graduate School of Industrial Administration
Carnegie Institute of Technology
Pittsburgh, Pennsylvania 15213

This paper was written as part of the contract, "Planning and Control of Industrial Operations," with the Office of Naval Research at the Graduate School of Industrial Administration, Carnegie Institute of Technology, Management Sciences Research Group. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government. Contract NONR 760(24), NR 047-048.

Ad 606955

I. Introduction

The algorithm of this paper is based on the dual linear programming algorithm ([1], [6])and the recently developed bound escalation method for ~~solving integer linear programs [3].~~ The hybrid-dual algorithm is divided into two stages. In the first stage a linear program is solved to yield an optimal solution in fractional-valued variables. The final tableau given by the dual linear programming method is then transformed and expanded into a new tableau of a readily specifiable canonical form. In the second stage of the algorithm a variant of the bound escalation method is applied to the new tableau--and to the tableaus successively derived thereafter--until one or more of a distinguished set of columns (and a corresponding set of rows) attains a predetermined configuration. At this point the indicated rows and columns are discarded, never to be recovered, and the method continues recursively with the bound escalation algorithm until the problem is solved.

It may be observed that the second stage of the hybrid-dual algorithm works almost exactly in reverse of Gomory's original integer programming method [4]. With the hybrid-dual method, once the fractional linear programming tableau is transformed and expanded, no new constraints or variables are added thereafter; instead certain variables and constraints are deleted as the solution process evolves. In contrast, the original Gomory method begins with the final linear programming tableau and progressively increases its dimension, though the tableau may be kept from exceeding a certain size by weeding out new variables when they emerge from the nonbasic set.

1. Introduction

The algorithm of this paper is based on the dual linear programming algorithm ([1], [6])and the recently developed bound escalation method for solving integer linear programs [3]. The hybrid-dual algorithm is divided into two stages. In the first stage a linear program is solved to yield an optimal solution in fractional-valued variables. The final tableau given by the dual linear programming method is then transformed and expanded into a new tableau of a readily specifiable canonical form. In the second stage of the algorithm a variant of the bound escalation method is applied to the new tableau--and to the tableaus successively derived thereafter--until one or more of a distinguished set of columns (and a corresponding set of rows) attains a predetermined configuration. At this point the indicated rows and columns are discarded, never to be recovered, and the method continues recursively with the bound escalation algorithm until the problem is solved.

It may be observed that the second stage of the hybrid-dual algorithm works almost exactly in reverse of Gomory's original integer programming method [4]. With the hybrid-dual method, once the fractional linear programming tableau is transformed and expanded, no new constraints or variables are added thereafter; instead certain variables and constraints are deleted as the solution process evolves. In contrast, the original Gomory method begins with the final linear programming tableau and progressively increases its dimension, though the tableau may be kept from exceeding a certain size by weeding out new variables when they emerge from the nonbasic set.

Two additional features of the hybrid-dual algorithm deserve mention. First, it is possible to work with a somewhat smaller tableau than the standard by following one of two modified procedures involving restricted rules of choice. Second, the hybrid-dual algorithm usually converges before all of the problem constraints are completely satisfied. When the tableau reaches a prescribed form, one portion of the bottom row may be added to another to yield the optimal solution immediately. In this way it is frequently possible to reduce the number of steps otherwise required to solve the problem.

The next section introduces the notational framework to be used in this paper, along with a brief discussion of the principal characteristics of the dual linear programming algorithm and the bound escalation method.[1] The hybrid-dual algorithm is outlined in Section III. Theorems concerning the properties of the algorithm are stated and proved, including the fundamental result that convergence to an optimal solution is assured in a finite number of steps for any bounded problem with a nonempty solution set. In Section IV an example problem is solved in two different ways to demonstrate the workings of the hybrid-dual algorithm and computational features applied to the standard tableau and also to a compressed tableau. In conclusion some computational features of the method are discussed.

II. Notation and Constituent Algorithms

The integer programming problem may be written

$$\text{Minimize} \quad wb + b_0$$

$$\text{subject to} \quad wA \geq c,$$

$$w \geq 0, \quad w \text{ integer}$$

1. It is not strictly necessary to use the dual linear programming algorithm as opposed to the primal algorithm. We have however selected the former in order to facilitate subsequent exposition.

where b is an $m \times 1$ column vector, c is a $1 \times n$ row vector, b_o is a scalar, A is an $m \times n$ matrix, and w is a $1 \times m$ row vector whose components we wish to find in order to satisfy the inequalities and optimize the minimization criterion. We assume that the components of A, b, and c are all integer. Without further loss of generality we also specify that the augmented matrix $(-b \;\; A)$ is lexicographically negative by row (see [2] and [6]). The initial tableau for the dual linear programming algorithm may then be represented as follows.

$$(1) \qquad \begin{array}{|c|c|} \hline -b_{m\times 1} & A_{m\times n} \\ \hline 0_{n\times 1} & -I_{n\times n} \\ \hline b_o & C_{1\times n} \\ \hline \end{array}$$

The first column of the tableau represents the objective function to be optimized, and each subsequent column identified a problem constraint obtained by rewriting the inequality $wA \geq c$ in the form $wA - \bar{w}I = c$, where $\bar{w} = (w_{m+1}, \ldots, w_{m+n})$ is a $1 \times n$ row vector of nonnegative "slack" variables. The dual linear programming method recursively transforms the given tableau into a new one by a sequence of pivot reduction operations. At any step, the pivot element is determined by selecting a positive entry in the current c vector, then determining the unique positive entry in the associated column which will leave the upper portions of the tableau

lexicographically negative after carrying out the pivot operation.[1] The process is completed when there are no more positive entries in the (transformed) c vector to choose from. A positive component at the base of a column whose entries are all negative or zero signals that the feasible solution set of the problem is empty.

Gaussian elimination in the dual linear programming algorithm preserves the identity of the original variables, and each tableau defines a problem exactly equivalent to the original. Hence (1) may be used to represent an arbitrary tableau obtained with the dual algorithm after reindexing. The values of the variables at the optimal solution are then obtained by setting the variables associated with the final A matrix equal to zero.

Using the same notation, the initial tableau for the bound escalation method may be depicted as follows.

(2)

$-b_{mx1}$	A_{mxn}	I_{mxm}
b_o	C_{1xn}	0_{1xm}

Here, as before, the first column represents the objective function and the successive columns identify the problem constraints. This time, however, the constraints are left as inequalities and the nonnegativity conditions are summarized explicitly in the last m columns.

1. The pivot element is always -1, obtained by multiplying the pivot column by the appropriate constant. The pivot operation then makes all other entries in the same row as the pivot element equal to zero by adding the necessary multiple of the pivot column to each of the other columns in succession.

Because the bound escalation method works somewhat differently than the linear programming algorithm, it is convenient to represent the general form of the tableau by ignoring the partition between the A matrix and the identity matrix, yielding a tableau of the form

(3)

$-b_{mx1}$	$A_{mx(m+n)}$
b_o	$C_{1x(m+n)}$

The solution to the integer programming problem is obtained with the bound escalation method when all elements of $c_{1X(m+n)}$ are nonpositive. The optimal w vector then appears as the negative of the vector originally lying beneath the I matrix.

To apply the bound escalation method it is desired to create a tableau which upon suitable indexing may be partitioned as follows.

(4)

$-b_{mx1}$	D_{pxp}	*
	$Q_{(m-p)xp}$	
b_o	d_{1xp}	*

Here D is a pXp square matrix with positive entries along the main diagonal and nonpositive entries everywhere else. Q is a matrix composed entirely of nonpositive entries, and d is a p-component row vector at least one of whose entries is positive. We are unconcerned with the portions of the tableau marked with a star.

The bound escalation method operates on such a structure to obtain a vector d* satisfying the following two conditions: (i) d - d*D is nonpositive integer, and (ii) each of the first p components of the

optimal solution vector is at least as large as the corresponding element of d^*.[1] Once d^* is determined, the bottom row of the tableau is changed by subtracting d^*P from it, where P is the matrix consisting of the first p rows of the tableau. By incorporating this procedure into a recursive process for creating the submatrices D, Q, and d, the bound escalation method converges to the optimal integer solution in a finite number of steps.[2]

It will suffice here to consider only the case in which D, Q, and d constitute a single column of the tableau. In this instance, d and D each are composed of a single positive element, d_0 and d_1 respectively, and d^* consists of the single element $\langle d_0/d_1 \rangle$.[3]

If the vector $c_{1X(m+n)}$ has at least one positive component, it is easy to create a column of the desired form by the following rules. First, select a positive entry from the bottom row of the tableau. All but one of the remaining positive entries in its column may then be eliminated by consecutively selecting any pair of them and subtracting a positive integer multiple of the row in which one appears from the row containing the other. The only restriction on this process is that the upper portion of the tableau must be kept lexicographically negative. Once a column of the desired form is obtained, it is permissible to add an integer multiple of the unique row with the positive component to any other row, provided no entries that were previously nonpositive (in the specified column) thereby become positive. More generally, row subtractions are permissible under any circumstances so long as the tableau is

1. The optimal solution vector is defined relative to the problem represented by the current tableau, and changes when the tableau is transformed.
2. D, Q, and d need not appear explicitly in the tableau, but may be generated by any nonnegative linear combination of the tableau columns (except for the first).
3. We will use $\langle x \rangle$ to denote the smallest integer greater than or equal to x.

maintained lexicographically negative, but row additions can be carried out only relative to the D, Q, d configuration in the manner indicated, except that the component d_0 of d is allowed to be zero.[3] These basic results provide all that is necessary for the sections to follow.

III. The Hybrid-Dual Algorithm.

We will let (1) represent the final tableau obtained by the dual linear programming algorithm after reindexing. Let A^f_{mXn} be the matrix each of whose entries is given by $a^f_{ij} = a_{ij} - [a_{ij}]$, and let A^w_{mxn} be the matrix with entries $a^w_{ij} = [a_{ij}]$, where a_{ij} denotes the element in the ith row andjth column of A.[1] Similarly, let c^f be the 1Xn row vector whose jth entry is $c^f_j = c_j - [c_j]$, and c^w the row vector with jth entry $c^w_j = [c_j]$. The form of the initial tableau for the hybrid-dual algorithm is then given by

(5)

$-b_{mx1}$	A^f_{mxn}	I_{mxm}	A^w_{mxn}
0_{nx1}	$-I_{nxn}$	0_{nxm}	I_{nxn}
b_o	C^f_{1xn}	0_{1xn}	C^w_{1xn}

We characterize the hybrid-dual algorithm as follows.

The hybrid-dual algorithm.

Stage 1. Solve the fractional linear programming problem with the dual linear programming method, and create the transformed tableau (5).[2]

1. The notation $[x]$ is used to indicate the greatest integer less than or equal to x.
2. It is shown in Section IV how a somewhat smaller tableau may be used by observing certain additional rules.
3. It may in fact be negative, so long as $d_0 + d_1 > 0$.

Stage 2. Apply the bound escalation method to the transformed tableau, subject to the following modifications.

A. If at any stage an element of the c^f portion of the tableau is less than zero, its column may be treated as the negative of itself except in the determination of lexicographic ordering.

B. If an element of the c^f portion of the tableau is equal to zero, and if all but one of the elements of its column are also zero, then the indicated column and the row containing the nonzero element may be removed from the tableau.

C. If an element of c^f is equal to zero, and if more than one component of its column is different from zero, the column (and some associated row) may be put in the form for removal by the method of Theorem 3 (to follow). In this fashion n rows and n columns of tableau (5) may be removed in a finite number of steps.

D. The problem is solved when the vector sum $c^f + c^w$ is non-positive integer (with deleted entries of c^f set equal to zero), and the 0_{1Xn} portion of the bottom row is nonpositive. The components of the optimal solution vector, including the values of the slack variables, are then read from the bottom row to the right of c^f, with c^w replaced by $c^f + c^w$.

To prove the validity of the algorithm we present the following Theorems and proofs.

Theorem 1. The integer programming problem summarized by tableau (1) may be solved with the bound escalation method applied to tableau (5), subject to the qualification of instruction A.

<u>Proof</u>: Each of the columns of the final dual tableau (1) represents an equality of the form

$$\sum_{i=1}^{m} a_{ij}w_i - w_{m+j} = c_j, \text{ for } j = 1, \ldots, n.$$

When this is factored into an equivalent pair of inequalities we obtain

$$\sum_i a_{ij}w_i - w_{m+j} \geqslant c_j$$

$$\sum_i (-a_{ij})w_i + w_{m+j} \geqslant -c_j$$

To represent the problem in a tableau suitable for applying the bound escalation method, we thus have[1]

(6)

$-b_{m\times 1}$	$A_{m\times n}$	$-A_{m\times n}$	$I_{(m+n)\times(m+n)}$
$0_{n\times 1}$	$-I_{n\times n}$	$I_{n\times n}$	
b_o	$C_{n\times 1}$	$-C_{n\times 1}$	$0_{1\times(m+n)}$

We assume first that $b > 0$. For the final dual linear programming tableau, each component of c is nonpositive, hence $-c \geqslant 0$. Relative to the columns whose bottom row components lie in $-c$, it is therefore possible to carry out the row additions and row subtractions indicated in the previous section. Since $b > 0$, it is permissible to subtract any positive integer multiple of a row in the $I_{n\times n}$ partition of the tableau from any row in the $-A$ partition. We first select only those

1. While the bound escalation method is guaranteed to converge for an all integer tableau, the proofs in [3] also obviously apply when A, b, and c are rational. Also, since the original problem was all integer, the slack variables added for the dual linear programming algorithm must be integer as well.

elements of $-A$ which are positive, and subtract a large enough multiple of the appropriate I_{nXn} matrix row to make each of them less than or equal to zero. Thus for each element $-a_{ij}$ of $-A$ such that $-a_{ij} > 0$, it is sufficient to subtract $< -a_{ij} >$ times the jth row of I_{nXn}, leaving $[a_{ij}] - a_{ij}$ $(= -a_{ij} - < -a_{ij} >)$ in place of the original $-a_{ij}$ entry. The effect in other portions of the tableau will be to replace a_{ij} in the A matrix by $a_{ij} - [a_{ij}]$ (for $a_{ij} < 0$), and to replace the 0 in row i and column m + j of the $I_{(m+n)X(m+n)}$ matrix by $[a_{ij}]$. When this process is carried to completion the only positive elements in the columns above the $-c$ vector will be those in the I_{nXn} matrix.

At this point, it may be possible to add integer multiples of the rows of I_{nXn} to some of the rows of (the new) $-A$. Since all elements of $-A$ must remain nonpositive after the addition, the largest multiple of the jth row of I_{nXn} that can be added to the ith row of $-A$ (for $-a_{ij} < 0$) is $[a_{ij}]$. This quantity will be zero for all elements of $-A$ that were initially nonnegative, so that the multiple may be defined entirely in terms of the initial $-A$ matrix (for $-a_{ij} < 0$). The adjustments throughout the tableau can thus be specified in exactly the same way as for $-a_{ij} > 0$. If $a_{ij} = 0$ the indicated adjustments apply trivially.

The changes in the $-c$ vector prescribed by the bound escalation method are readily determined by the remarks of the previous section. For each column j associated with an element of $-c$, $d_0 = -c_j$ and $d_1 = 1$, hence $d^* = < -c_j >$. Subtracting the appropriate multiples of the rows of the I_{nXn} matrix from the $-c$ vector yields the value $[c_j] - c_j$ $(= -c_j - < -c_j >)$ for the jth entry of the original $-c$. Carrying out the subtraction for the entire bottom row of the tableau replaces each c_j of the c vector with $c_j - [c_j]$, and replaces the

$(m + J)$-th component of the $0_{1\lambda(m+n)}$ vector by $[c_j]$, all other elements remaining unchanged.

Since all components of the $-c$ portion of the tableau are now non-positive, we will not immediately be concerned with them in applying the bound escalation method. Moreover, it is evident that the columns associated with these components will always be the negative of those columns associated with the original c vector. Hence we may use the letter to summarize the former, provided we observe that each of these columns may be treated as its negative (positioned suitably to the right).[1] Collapsing the tableau as indicated, we obtain tableau (5).

<u>Note</u>: To maintain strict lexicographic negativity, if the final tableau obtained with the dual linear programming algorithm is degenerate--that is, if some component of the final b vector is equal to zero--then the form of the starting tableau for the second stage of the hybrid-dual algorithm may have to be modified slightly. Perhaps the simplest way to accomplish the necessary changes is to begin with tableau (5) at a slightly more primative stage. Thus, if $b_k = 0$, and if a_{kJ} is the first nonzero component in the kth row of A, we define $a^f_{iJ} = a_{iJ}$, $a^w_j \quad 0$ (for $i = 1, \ldots, m$), $c^f_J = c_J$, $c^w_J = 0$. All other entries of A^f, A^w, c^f, c^w are given as before. In practice, when $b \not> 0$ the minor departure from strict lexicographic negativity which may arise from using tableau (5) as originally defined is not likely to hamper the convergence process--and may in fact speed it up.

<u>Theorem 2</u>: Rows and columns of the tableau deleted according to instruction B are irrelevant for obtaining the optimal integer solution.

---

1. The "negative" columns may in fact be ignored in the determination of lexicographic ordering since we may consider them appended at the far right of tableau (5). The linear independence of the rows of (5) assures that none of these rows will become all zero.

Proof: Each column of the tableau associated with c^f summarizes an exact equality in nonnegative variables, hence a deleted column represents an expression of the form $az = 0$. The variable z associated with the deleted row must be equal to zero, and hence the row is superfluous. Once the row is removed, the remainder of the column is obviously superfluous as well.

Theorem 3. If an element of c^f is equal to zero, then the following method will bring the assiciated column into the required form for removal, enabling the deletion of n rows and n columns of (5) in a finite number of steps.[1]

By the procedure outlined in Section II, put the indicated column in the form D, Q, d. Perform any necessary row additions with the row containing d_1 (of D) until, for each element q of Q, $d_1 + q > 0$. If all elements of Q are equal to zero, the process is completed. Otherwise, consider the negative of the column just derived, and repeat.

Proof: We will use a superscript to denote the step on which a given D, Q, d, structure is obtained in the above procedure. We note that d_1^2 (the value of d_1 on the second step) must be derived from the components of $-Q^1$. Since none of the positive elements of $-Q^1$ can be increased by the row subtractions specified in Section II for creating the D, Q, d form, it follows that $d_1^2 \leq -q^1$, where $-q^1$ is the maximum component of $-Q^1$. But the method specifies that $d_1^1 + q^1 > 0$, hence $d_1^1 > d_1^2$. Similarly, $d_1^k > d_1^{k+1} + h$, where h is a fixed positive quantity. Thus the elements of $-Q$ (Which are nonnegative and strictly less than d_1) must eventually all be driven to zero. If it is possible for the problem to be solved - in which case the elements of c^f can be forced to zero - all n of the corresponding columns (and some n rows) may therefore eventually be eliminated.

1. If all nonzero components of the column have the same sign, then by the reasoning in the proof of Theorem 2, all rows of the tableau containing the nonzero elements may be eliminated simultaneously. We note also that the optimal solution may be obtained before the indicated n rows and columns are deleted

Theorem 4. The optimal solution vector to the problem is obtained as in instruction D.

Proof: By the nature of the bound escalation method, the vector to the right of c^f (initially the $0_{1X(m+n)}$ vector of (6)) gives integer values for the original problem variables which will produce an objective function value equal to the current b_o, and undersatisfy (or oversatisfy) each of the original constraints by the positive (or negative) amount of the corresponding component of the current c^f vector. At any stage we may increment or decrement the values of the variables associated with the original $-I_{nXn}$ matrix of (6) without changing the objective function value. Suppose that it is possible to adjust the $-I_{nXn}$ variables in such a way that (a) all of the problem constraints are satisfied, and (b) all of the problem variables are nonnegative integer. Since b_o is monotone nondecreasing from one tableau to the next, it follows that the solution so obtained is optimal. But to satisfy the first n constraints as strict equalities (if no other variables are to be changed), each of the $-I_{nXn}$ variables must be set equal to the corresponding component of c^f. This may be represented by adding c^f to c^w. The conditions of instruction D then correspond to the conditions for achieving optimality, and the theorem is proved.

We may note parenthetically that, since the components of c^w are always integer, the components of c^f at the optimal solution must be integer also. Moreover, if lexicographic ordering of the tableau is rigidly maintained, the first nonzero component of c^f must be negative, since with the bound escalation method the bottom row of the tableau must be strictly increasing (lexicographically) each time it is changed.

Theorem 5. The hybrid-dual algorithm will converge in a finite number of steps for any problem containing a nonempty integer solution set bounded for optimality.

Proof: The theorem follows immediately from Theorems 1 through 4 and the convergence properties of the dual linear programming algorithm and the bound escalation method.

IV. Example Problem and Computational Experience.

To illustrate the hybrid-dual algorithm we will now solve the following problem.

$$\begin{aligned} \text{Minimize} \quad & 2w_1 + 3w_2 + 4w_3 \\ \text{subject to} \quad & -1w_1 + 2w_2 + 1w_3 \geq 7 \\ & 2w_1 + 1w_2 + 2w_3 \geq 8 \\ & 1w_1 - 1w_2 + 3w_3 \geq 8 \\ & w_1, w_2, w_3 \geq 0 \end{aligned}$$

Stage 1 (the dual linear programming algorithm):

(o)

-2	-1	2	1
-3	2	1	-1
-4	1	2	3
0	-1	0	0
0	0	-1	0
0	0	0	-1
0	7 ↑	8	8

(i)

-7/2	1/2	5/2	1/2
0	-1	0	0
-5/2	-1/2	3/2	7/2
-3/2	1/2	1/2	-1/2
0	0	-1	0
0	0	0	-1
21/2	-7/2	9/2	23/2 ↑

(ii)

-22/7	4/7	16/7	-1/7
0	-1	0	0
0	0	0	-1
-13/7	3/7	5/7	1/7
0	0	-1	0
-5/7	-1/7	3/7	2/7
131/7	-13/7	-3/7	-23/7

The optimal solution read from the final tableau is thus $w_1 = 0$, $w_2 = 13/7$, $w_3 = 23/7$, $w_4 = 0$, $w_5 = 3/7$, $w_6 = 0$, yielding an objective function value of 131/7.

The simple form of the bound escalation method outlined in Section II will suffice for the second stage. We will indicate by an arrow the column relative to which the row additions and subtractions are to be made. For definiteness, all such operations will be carried out with the lexicographically least negative row, chosen from among those in which the indicated column has a positive entry.[1]

We obtain the initial tableau for Stage 2 by putting the components of the w vector in the order $(w_1, w_4, w_6, w_2, w_5, w_3)$.

Stage 2

(o)	-22/7	4/7	2/7	6/7	1	0	0	0	2	-1
	-13/7	3/7	5/7	1/7	0	1	0	0	0	0
	-5/7	6/7	3/7	2/7	0	0	1	-1	0	0
	0	-1	0	0	0	0	0	1	0	0
	0	0	-1	0	0	0	0	0	1	0
	0	0	0	-1	0	0	0	0	0	1
	131/7	1/7	4/7	5/7 ↑	0	0	0	-2	-1	-4
(i)	-1	-2	-1	0	1	0	-3	3	2	-1
	-8/7	-3/7	2/7	-1/7	0	1	-1	1	0	0
	-5/7	6/7	3/7	2/7	0	0	1	-1	0	0
	0	-1	0	0	0	0	0	1	0	0
	0	0	-1	0	0	0	0	0	1	0
	-15/7	18/7	9/7	-1/7	0	0	3	-3	0	1
	146/7	-17/7	-5/7	-1/7 ↑	0	0	-3	1	-1	-4

1. This rule is not always the most efficient for either the simple or the more general form of the bound escalation method. With a slight embellishment the rule provides the basis for Gomory's all integer algorithm [5], which can be shown to be a variation of the bound escalation method under restricted alternatives of choice (see [3]).

In obtaining the next tableau, the fourth column becomes all zero except in the second row, hence the indicated row and column are deleted.

(ii)

-1	-2	-1	1	0	-3	3	2	-1
-3	0	1	0	2	-1	1	0	0
0	-1	0	0	0	0	1	0	0
0	0	-1	0	0	0	0	1	0
-1	3	1	0	-1	4	-4	0	1
22	-2	-1	0	-1	-2	0	-1	-4

The problem is now solved. We add the final c^f (-2 -1 0) to the final c^w (0 -1 -4), yielding the optimal solution vector (0 -1 -2 -2 -2 -4). We will not stop, however, but will continue with Stage 2 of the algorithm to illustrate the process of eliminating rows and columns from the tableau.

(iii)

-1	-1	1	0	-3	1	2	-1
-3	1	0	2	-1	1	0	0
0	-1	0	0	0	0	1	0
-1	1	0	-1	4	-1	0	1
22	-1	0	-1	-2	-2	-1	-4

(iv)

-1	1	0	-3	1	1	-1
-3	0	2	-1	1	1	0
-1	0	-1	4	-1	1	1
22	0	-1	-2	-2	-2	-4

At the optimal integer solution the vector (0 -1 -2 -2 -2 -4) in the final tableau corresponds to the negative of the reordered vector (w_1, w_4, w_6, w_2, w_5, w_3), and the objective function takes on the value 22. We have here deleted all n rows and columns (n = 3) specified according to Theorem 3.

It is generally possible to operate with n fewer rows from the start. We may in fact use only the tableau

(7)

$-b$	A^f	I	A^w
b_o	c^f	0	c^w

In order to apply the hybrid-dual method to (7), the algorithm must be modified as follows.

<u>Stage 1</u>: (As before.)

<u>Stage 2</u>: A If $c_J^f < 0$, we may replace c_J^f with $c_J^f = [c_J^f]$, c_J^w with

$c_J^w + [c_J^f]$, a_{iJ}^f with $a_{iJ}^f = [a_{iJ}^f]$, and a_{iJ}^w with $a_{iJ}^w + [a_{iJ}^f]$

for $i = 1,\ldots,m$). We do not consider the negative of the column in which c_J^f appears.

If $c_J^w > 0$, replace c_J^f with $c_J^f + c_J^w$, c_J^w with 0,

a_{iJ}^f with $a_{iJ}^f + a_{iJ}^w$, and a_{iJ}^w with 0 (for $i = 1,\ldots,m$).

B. $\not\Rightarrow$ C. If $c_J^f = 0$, the row of tableau (5) containing the Jth row of $-I_{nXn}$ may be annexed to (7), and the method of Theorem 3 used to bring the indicated column and some row into the required form for removal.

D. The solution is obtained in the same fashion as for the regular hybrid-dual algorithm.

We may justify the modified method applied to tableau (7) by observing that it is never necessary to alter the $-I_{nXn}$ portion of (5) if we elect not to add a multiple of any of the other rows to it. The adjustments of the tableau when a component of c^f is negative then become the same as those

specified by Theorem 1 in creating (5). The adjustments when a component of c^w is positive similarly result by acknowledging the implicit existence of the I_{nXn} matrix.

As with Theorem 1, the analysis is strictly correct only if $b > 0$. If some component of b becomes zero it may then be necessary to deviate from the preceding to maintain lexicographic negativity. The conditions for deviation follow the ordinary rules for applying the bound escalation method, and hence do not require additional specification. The result is that one or more of the n rows associated with the $-I_{nXn}$ matrix may have to be explicitly represented in case of degeneracy. However, we suggest as in Section III that the departure from strict lexicographic negativity brought about by ignoring degeneracy in the b vector relative to the rows of $-I_{nXn}$ will not normally interfere with convergence.[1]

To illustrate the preceding ideas more thoroughly, we will rework the example problem solved earlier. Using the condensed representation, we begin with the first tableau for Stage 2.

(0)	-22/7	4/7	2/7	6/7	1	0	0	0	2	-1		
	-13/7	3/7	5/7	1/7	0	1	0	0	0	0		
	-5/7	6/7	3/7	2/7	0	0	1	-1	0	0		
	131/7	1/7	4/7	5/7	0	0	0	-2	-1	-4		
											Col. 3	Col. 9
(1)	-1	-2	-1	0	1	0	-3	3	2	-1	0	1
	-8/7	-3/7	2/7	-1/7	0	1	-1	1	0	0	2/7	0
	-5/7	6/7	3/7	2/7	0	0	1	-1	0	0	3/7	0
	146/7	-17/7	-5/7	-1/7	0	0	-3	1	-1	-4	2/7	-2

1. If lexicographic negativity is lost in this fashion it may always be regained by inserting a column consisting of all negative coefficients immediately to the right of the b vector, with a sufficiently large negative number in the bottom row to make the constraint compatible with any optimal solution.

The two columns at the end of the tableau indicate the columns to replace column 3 and column 9 according to the rule of the modified instruction A. Selecting the new column 3 as the next column to put in the D, Q, d form, we obtain the following tableau.

											Col. 3	Col. 9
(ii)	-1	-2	0	0	1	0	-3	3	1	-1	0	1
	-3/7	-9/7	-1/7	-3/7	0	1	-2	2	0	0	6/7	-1
	-5/7	6/7	3/7	2/7	0	0	1	-1	0	0	3/7	0
	151/7	23/7	-1/7	-3/7	0	0	-4	2	-2	-4	6/7	-3

(iii)	-1	-2	0	0	1	0	-3	3	1	-1
	-3/7	-9/7	6/7	-3/7	0	1	-2	2	-1	0
	-2/7	15/7	-3/7	5/7	0	-1	3	-3	1	0
	22	-2	0	0	0	-1	-2	0	-2	-4

The problem is now solved, and the final solution vector is obtained by adding (-2 0 0) to the (0 -2 -4) vector on the bottom row, yielding (0 -1 -2 -2 -2 -4) as before.

Two points of contrast may be noted between the regular hybrid-dual algorithm and the modified method relative to the preceding example problem. First, the modified method did not do so well as the regular method in preparing columns of the tableau for elmination (though the smaller dimension of (7) served adequately to compensate for this in the present instance). Second, the modified method required an additional step to obtain the optimal solution. It may be observed that with the regular method the solution was obtained by focusing only on column 4. Following this strategy with the modified

method requires yet another step beyond the number needed above. The reason why this might be so is suggested in [3], where it is shown that it is generally desirable to make row x additions with the bound escalation method whenever possible. Suppressing the $-I_{nXn}$ matrix in effect rules out the chance for some of these additions. Thus, it may be in general that the modified method requires more steps to obtain an optimal solution than the regular method, perhaps partially offsetting the savings derived from using a smaller tableau. Moreover, the initial difference in tableau size may be a transitory one due to the superior ability of the regular hybrid-dual algorithm to eliminate rows and columns.

There is another way of working with a substantially smaller tableau than (5), while still retaining the ability to perform row additions relative to the $-I_{nXn}$ matrix. The key is to consider the columns associated with c^f one at a time, applying the method of Theorem 3 to each in succession. It is evident that the element d_o of d does not have to begin zero, but may be driven to zero exactly as the elements of $\underline{Q}$. Thus each column (and an associated row) is eliminated before moving to the next (unless the solution is obtained first). In this way only a single row associated with the $-I_{nXn}$ matrix needs to be included in the tableau at any stage. Such a procedure corresponds to the steps actually taken in solving the example problem in this paper, and seems to be about as effective as any other method for solving the simple problems on which we have tested the algorithm to date. If this method were used consistently a further savings in tableau size could be effected by letting the initial A^f matrix be equal to the A matrix of the final linear programming tableau, except for the single column to which the method

of Theorem 3 was to be applied. Since none of the operations would affect column m + j of $I_{(m+n)x(m+n)}$ of (6) until column j of A was itself operated on directly by Theorem 3, not all of the columns of $I_{(m+n)x(m+n)}$ would need to be given explicit representation. This would in effect produce a constant tableau size n - 1 rows and n - 1 columns smaller than (5) until the last column of A^f was eliminated, for an old row and column would be dropped each time a new row and column required explicit representation. A possibility that might be worth considering would be to segregate the vector to which Theorem 3 was to be applied (along with the objective function vector), until it was reduced to the indicated form for removal. Thereupon the transformation which caused the reduction could be applied to update the remainder of the tableau.

Though we have not exemplified this second method of tableau reduction, since it is relatively straightforward, we would expect it to be preferable to the modified method based on tableau (7) in many circumstances. As a general rule, however, the price of using either of the smaller tableaus is a reduction in the number of alternatives of choice, and we have yet to discover in which cases this price is greater or less.

Some comments on the form of the A^f matrix are in order. It was shown in Theorem 1 that this matrix results simply from applying the bound escalation method to each of the columns associated with the A matrix of tableau (6). It is interesting to note that the resulting constraints are in fact "Gomory cuts" of the type used in his first integer algorithm. By deriving these constraints with the basic transformations prescribed by

the bound escalation method, we have shown how each one may not only be appended to the tableau as in Gomory's first algorithm (in addition to the constraint from which it was derived), but may in fact "replace" its parent constraint, provided our rules for operating on (5) are observed.[1] Moreover, instead of using the linear programming pivot process, which may require the generation of other derivative cuts, we proceed directly with the bound escalation method in conjunction with Theorem 3, shaving off portions of the tableau as they attain the specified form.

To get a rough idea of the computational performance of the hybrid-dual algorithm we have solved eleven additional problems with the method. For comparison, these problems were also solved with the two integer programming algorithms developed by Ralph Gomory. The scope of the testing was extremely limited: none of the problems was larger than three variables and three inequalities. No real conclusions, of course, can be drawn. However, for the problems examined, the hybrid-dual method required fewer steps in almost every case than the number of pivots required by the all-integer algorithm.

1. For additional (non-Gomory) cuts which may be obtained with the transformations specified in [3], and which alternately may be used as a basis for the initial tableau of Stage 2 of the hybrid-dual algorithm, see [7]. An interesting feature of these latter cuts is that both positive and negative coefficients may attach to the variables associated with the A matrix of (6), as contrasted with the all nonnegative coefficients of A^f.

On the other hand, no marked difference was observable between the hybrid-dual algorithm and Gomory's original integer programming method, the hybrid-dual algorithm doing slightly better on some of the problems and slightly worse on others. Both methods solved 8 of the 11 problems in from one to three steps after the linear programming solution was obtained, and solved the remaining three problems within six steps after the linear programming solution. An edge should be given to Gomory's first integer programming algorithm for the problems tested, however, because of a smaller average tableau size.[1] Further investigation is needed to determine relative performances for problems of higher dimension and complexity.

It might be noted that, because of the simplicity of the problems examined, computational advantages of the general bound escalation method (in contrast to the simpler version presented in this paper) were not exploited. For problems in which the D, Q, d structure may be manufactured to exhibit certain characteristics (see, e.g., example problem 2 of [3]), the general bound escalation method by itself is appreciably superior to the algorithms discussed here. It would be interesting to know whether its incorporation into the hybrid-dual algorithm might prove similarly effective in solving still other classes of problems which have not readily yielded to either of Gomory's algorithms or to the bound escalation method in its independent form.

---

1. The second procedure outlined in this paper for reducing the dimension of (5) would have substantially eliminated this difference. We did not apply the modifed method based on tableau (7).

REFERENCES

1. Charnes, A., and Cooper, W. W., Management Models and Industrial Applications of Linear Programming, Vol. II, John Wiley and Sons, Inc., New York, 1961.

2. Dantzig, G. B., Ford, Jr., L. R., and Fulkerson, D. R., "A Primal-Dual Algorithm for Linear Programs," in Annals Study 38 Linear Inequalities and Related Systems, Kuhn and Tucker, Eds., Princeton University Press, 1956.

3. Glover, Fred, "A Bound Escalation Method for the Solution of Integer Linear Programs," Cahiers Du Centre D'Etudes de Recherche Operationnelle, Vol. 6, 1964

4. Gomory, Ralph E., "An Algorithm for Integer Solutions to Linear Programs," in Recent Advances in Mathematical Programming, Graves and Wolfe, Eds., McGraw-Hill Book Company, Inc., New York, 1963.

5. Gomory, Ralph E., "All-Integer Integer Programming Algorithm," IBM Research Report RC-189, International Business Machines Corporation Research Center, Yorktown Heights, 1960.

6. Lemke, C. E., "The Dual Method of Solving the Linear Programming Problem," Naval Research Logistics Quarterly 1, No. 1, 1954

7. Glover, Fred, "Some New Cuts for the Integer and Mixed-Integer Programming Algorithms," Office of Naval Research Memorandum, (forthcoming).