NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.
ON DIAGONALIZATION METHODS
IN INTEGER PROGRAMMING

by

R. Van Slyke and R. Wets

OPERATIONS RESEARCH CENTER

INSTITUTE OF ENGINEERING RESEARCH

UNIVERSITY OF CALIFORNIA - BERKELEY
ON DIAGONALIZATION METHODS IN INTEGER PROGRAMMING

by

Richard Van Slyke and Roger Wets
Operations Research Center
University of California, Berkeley

1 August 1962

Research Report 27

This research has been partially supported by the Office of Naval Research under Contract Nonr-222(83) with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.
ON DIAGONALIZATION METHODS IN INTEGER PROGRAMMING
by Richard Van Slyke and Roger Wets

I. Introduction

An important area for improvement in existing integer programming codes is in the easy generation of efficient cutting hyperplanes; in this paper we approach this problem using a triangular canonical form. In the first part we give an algorithm based on Gomory's all-integer integer programming algorithm\(^2\), which constitutes a first step in this direction. This procedure is a practical analog of a deepest cut method discussed in the second part of the paper. In Appendix A, a brief outline and flow diagram for the algorithm are given; finally the algorithm and the deepest cut problem are illustrated by examples in appendices B and C.

We assume that we have at hand an integer program where all the coefficients and constant terms are integers. The functional is to be maximized. We write the problem in a parametric form due to Tucker:

Maximize \( x_0 \) subject to \( x_j \) integer, \( j = 0, \ldots, n; \ \ x_j \geq 0 \), \( j = 1, \ldots, n; \) and

\[
\begin{align*}
  x_0 &= b_0 + c_0 t_0 + c_1 t_1 + \ldots + c_k t_k \\
  x_1 &= b_1 + a_{10} t_0 + a_{11} t_1 + \ldots + a_{1k} t_k \\
  &\vdots \vdots \vdots \vdots \vdots \vdots \\
  x_m &= b_m + a_{m0} t_0 + a_{m1} t_1 + \ldots + a_{mk} t_k 
\end{align*}
\]

(1)

It will be convenient to assume that the above system has the property that if \( x_0, x_1, \ldots, x_m \) are non-negative integer variables then \( t_0, \ldots, t_k \) must also be non-negative integer variables. We can guarantee this, in
In general, by adding the trivial relations \( x_{i+1} = t_i, \ i = 0, \ldots, k \). In this case the problem will be of the form:

Maximize \( x_0 \) subject to \( x_j \) non-negative integers, \( j = 1, \ldots, n; \) and

\[
\begin{align*}
    x_0 &= b_0 + c_0 t_0 + c_1 t_1 + \ldots + c_k t_k \\
    x_1 &= t_0 \\
    x_2 &= t_1 \\
    &\vdots \\
    x_{k+1} &= t_k \\
    x_{k+2} &= b_{k+2} + a_{k+20} t_0 + a_{k+21} t_1 + \ldots + a_{k+2k} t_k \\
    &\vdots \\
    x_n &= b_n + a_{n0} t_0 + a_{n1} t_1 + \ldots + a_{nk} t_k
\end{align*}
\]

(1')

If in this problem \( b_i > 0, \ i = 1, 2, \ldots, n \) and \( c_j \leq 0 \) for \( j = 0, 1, \ldots, k \), the solution obtained by setting \( t_0 = t_1 = \ldots = t_k = 0 \) is optimal.

It can be assumed without loss of generality that for bounded problems the problem is in a dual feasible form, such that \( c_j < 0 \) for all \( j \). A quite general method for accomplishing this is discussed in a paper by Dantzig, Ford and Fulkerson.\(^1\)

**Two Useful Operations**

We now describe two basic operations which will frequently be used:

1. **The Cut:** Suppose

\[
\begin{align*}
    x_i &= b_i + a_{i0} t_0 + a_{i1} t_1 + \ldots + a_{il} t_l + \ldots + a_{ik} t_k
\end{align*}
\]

(2) is an equation of (1). Let
\[ J^+ = \{ j : a_{ij} > 0 \} \quad \text{and} \quad J^- = \{ j : a_{ij} \leq 0 \}. \]

If \( x_i \) is to be a non-negative variable we have the following:

\[ 0 \leq x_i = b_i + \sum_{j \in J^+} a_{ij} t_j + \sum_{j \in J^-} a_{ij} t_j \leq b_i + \sum_{j \in J^+} a_{ij} t_j. \]

If

\[ (3) \quad \tau = b_i + \sum_{j \in J^+} a_{ij} t_j \]

then \( \tau \) is a non-negative integer valued variable. Using (3) we could eliminate one of the \( t \)'s appearing in (3), say \( t_\ell \) in (1) in favor of \( \tau \). If, moreover, \( a_{i\ell} = 1 \) making the substitution \( t_\ell = \tau - b_i - \sum_{j \in J^+ \setminus \{\ell\}} a_{ij} t_j \) in (1) would leave the tableau in integers.

\[
\begin{align*}
    x_0 &= b_0 + c_0 t_0 + \ldots + c_\ell \left( \tau - b_i - \sum_{j \in J^+ \setminus \{\ell\}} a_{ij} t_j \right) + \ldots + c_{0k} t_k \\
    x_1 &= b_1 + a_{10} t_0 + \ldots + a_{1\ell} \left( \tau - b_i - \sum_{j \in J^+ \setminus \{\ell\}} a_{ij} t_j \right) + \ldots + a_{1k} t_k \\
    & \quad \vdots \\
    x_n &= b_n + a_{n0} t_0 + \ldots + a_{n\ell} \left( \tau - b_i - \sum_{j \in J^+ \setminus \{\ell\}} a_{ij} t_j \right) + \ldots + a_{nk} t_k
\end{align*}
\]

If \( a_{i\ell} > 1 \), then

-3-
\[ \tau = \frac{b_i^*}{a_{ii}} + \sum_{j \in J^+} \frac{a_{ij}^*}{a_{ii}} t_j \]

is still a non-negative integer variable where \( b_i^* \) and \( a_{ij}^* \) are the smallest integers greater than \( -b_i \) and \( a_{ij} \), respectively, which are divisible by \( a_{ii} \), where \( \left( \frac{a_{ii}^*}{a_{ii}} \right) = \left( \frac{a_{ii}^*}{a_{ii}} \right) = 1 \), so if \( a_{ii} > 1 \), we can use \((3')\) and maintain the tableau in integers.

To minimize the notation after the substitution we simply let \( \tau \) be represented by \( t_d \) in the tableau. In general we only use the substitution \((3)\) or \((3')\) when \( b_i < 0 \) and in this case call it a cut because the restriction \( \tau \geq 0 \) restricts the values that \( t_0, \ldots, t_k \) can take on, e.g., \( t_0 = t_1 = \ldots = t_k = 0 \) is not a feasible solution.

b) The Column Operation: On the other hand if we define \( \tau = \sum_j \alpha_j t_j \), where \( \alpha_j \) are non-negative integer parameters and \( \alpha_d = 1 \), and substitute for \( t_d \) we are in fact performing a column operation, because formally the substitution is equivalent to subtracting \( \alpha_i \) times the \( i \)th column of coefficients of \( (1') \) from the \( i \)th column of \( (1') \) for \( i = 0, 1, \ldots, l-1, l+1, \ldots, k \). We note that this is of the form of a cut with \( b_i = 0 \). The only thing we have to be careful about in making these substitutions for cuts and column operations is that the optimality condition for the transformed tableau is the same as for the original; more specifically that \( x_1, \ldots, x_n \geq 0 \) and integer implies that the current \( t_0, \ldots, t_k \) are non-negative integer parameters. This is easy to verify remembering that the original tableau had this property and that each new parameter is a non-negative integer weighted sum of \( t_0, \ldots, t_k \).
II. The Basic Algorithm

Ralph Gomory's all-integer algorithm concerns itself basically with making "cuts" in a systematic fashion until a transformed tableau of integers is obtained from (1) with \( c_j \leq 0, (j = 0, \ldots, k), b_i \geq 0, (i = 1, \ldots, n) \). The question is how to generate the most "efficient" cuts in some sense. Gomory has suggested\(^2\) that the efficiency may be related to the lexicographic magnitude of the pivot column in the sense that one should try to minimize the magnitude of the pivot column. Whenever we attach adjectives such as "best", "efficient", "deep" to the word "cut" we will always mean it in this way.

1. The Algorithm.

Suppose our problem is in the form:

Maximize \( x_0 \) where \( x_j \geq 0, (j = 1, \ldots, n) \); and \( x_j \) are integers \( (j = 0, \ldots, n) \)

subject to

\[
\begin{align*}
x_0 &= b_0 - t_0 \\
x_1 &= b_1 + a_{10}t_0 - t_1 \\
x_2 &= b_2 + a_{20}t_0 + a_{21}t_1 - t_2 \\
\vdots &= \vdots \\
x_{k+1} &= b_{k+1} + a_{k+1,0}t_0 + a_{k+1,1}t_1 + \cdots - t_{k+1} \\
x_{k+2} &= b_{k+2} + a_{k+2,0}t_0 + a_{k+2,1}t_1 + a_{k+2,2}t_2 + \cdots + a_{nk+1,k+1}t_{nk+1} \quad (5)
\end{align*}
\]

Then finding efficient cuts is relatively easy since the columns are already ordered lexicographically. We seek a cut \( \tau = \beta_i + \sum_j a_jt_j \) for which the
last positive coefficient, which determines the pivot column, is as far to the left as possible. This follows because we always insist on maintaining dual feasibility. It is clear that the more "efficient" a cut, the fewer columns (in general) will be affected by the pivoting. With this criterion in mind we seek to generate the cuts. We will see in the next section that every problem of the form (1') and therefore of the form (1) can be put in the form (5).

We can consider, instead of a single equation, a sum of equations weighted by positive integers to generate the cuts. To each equation say the \(i^{th}\), with a negative constant term, we associate a most efficient cut in the following sense:

(i) If

(a) the coefficient of \(t_1\) is positive we add \(a_i, k+1\) times the \(k+1^{th}\) equation to the \(i^{th}\), if this keeps the constant term negative, we go to (ii). If it makes the constant term non-negative then the original equation, the \(i^{th}\), is associated with itself.

(b) \(a_i, k+1\) is negative or zero we proceed directly to (ii).

(ii) We now have a new linear combination of the \(t\)'s where the coefficient of \(t_{k+1}\) is non-positive. Next we examine the coefficient of \(t_k\) which is \(a_i, k + a_i, k+1 a_{k+1, k}\) in case (a) or \(a_i, k\) in case (b). As in (i), if the coefficient is negative or zero we go on to consider \(t_{k-1}\). If this is not the case we add \(a_i, k + a_i, k+1 a_{k+1, k}\) times the \(k^{th}\) equation if this keeps the constant term negative. If it does not, the current linear combination is associated with the \(i^{th}\) equation, and we stop.

(iii) We continue the process until either the constant term goes non-negative and the process terminates or until a positive integer weighted
combination of equations is obtained in which the coefficient of every
\( t_j \), \( j \neq 0 \) is non-positive. If one obtains an equation with all coefficients
non-positive and with a negative constant term, there is no feasible solution.
In this way, we can associate to each equation with a negative constant term
another derived equation. We can then choose from the collection of
derived equations the one for which the last positive coefficient is farthest
to the left. From this equation we generate the cut. We then repeat the
whole process with the new tableau.

The convergence properties of this basic algorithm are closely
connected to those of Gomory's all-integer algorithm\([2]\). For completeness
we review the proof. Because of the special nature of the tableau there
will always be a decrease in the constant term in some row above the
highest row with a negative constant term. This, plus the assumption that
a feasible solution exists, is enough to guarantee convergence. To see this,
we note that for each iteration we add a lexico-negative vector to the constant
column, so that the constant column is always decreasing lexicographically.
If we indicate by a superscript the iteration numbers for the constant terms,
we have for the constant term of the functional:

\[
  b_0^{(0)} > b_0^{(1)} > \ldots > b_0^{(\nu)} > b_0^{(\nu+1)} > \ldots > b_0^{(\text{optimal, feasible})}
\]

where strict inequality can hold at most a finite number of times, i.e.,
there is a \( \nu_0 \) such that \( b_0^{(\nu)} = b_0^{(\nu_0)} \) for all \( \nu \geq \nu_0 \). For if this were
not so, since \( b_0^{(\nu)} \) decreases by an integer each time, we would have
strict inequality and the functional value would eventually become less than
the functional value of the assumed feasible solution, which is impossible
since \( x_0 \leq b_0^{(\nu)} \) for every \( \nu \). Now from \( \nu_0 \) on we examine \( b_1^{(\nu)} (\nu \geq \nu_0) \).
They satisfy the condition \( b_1^{(\nu_0+1)} \geq b_1^{(\nu_0+2)} \geq \ldots \). Suppose \( b_1^{(\nu)} \) does
not "settle down," i.e., suppose strict inequality does not hold at most a finite number of times, then eventually \( b^{(\nu)} \) becomes negative. This is a contradiction for the cut generated at this point would cause a strict decrease in \( b_0 \) which is impossible since \( \nu \geq \nu_0 \). So after a finite number of iterations \( \nu_1 \ (\geq \nu_0) \) both \( b_0 \) and \( b_1 \) have "settled down." Suppose \( b_0, \ldots, b_{l-1} \) have settled down, then if \( b_l \) decreases indefinitely it eventually goes negative. Then the cut generated at this point will imply a pivot on some column strictly to the left of the \( l \)th. In other words \( b_i \) will decrease strictly for some \( i < l \) which is a contradiction. Since the number of \( b \)'s is finite, the process terminates in a finite number of steps.

2. Basic Canonical Form.

We show in this section that any problem of form (1) or equivalently (1') can be represented in form (5). Assuming we have the problem already in form (1') we get an equivalent basic canonical form (5) to (1') by subtracting from the first row of the tableau an extra parameter \( t_{k+1} \), and adding two extra equations:

\[
\begin{align*}
    x_{n+1} & = -t_{k+1} \\
    x_{n+2} & = +t_{k+1}
\end{align*}
\]

Since \( x_{n+1} \) and \( x_{n+2} \) are required to be non-negative, \( x_{n+1}' \), \( x_{n+2}' \), and \( t_{k+1} \) must all equal zero in any feasible solution. We have then:
\[ x_0 = b_0 + c_0 t_0 + c_1 t_1 + \cdots + c_k t_k - t_{k+1} \]
\[ x_1 = t_0 \]
\[ x_2 = t_1 \]
\[ \vdots \]
\[ x_{k+1} = t_k \]
\[ x_{k+2} = b_{k+2} + a_{k+2,0} t_0 + a_{k+2,1} t_1 + \cdots + a_{k+2,k} t_k \]
\[ \vdots \]
\[ x_n = b_n + a_{n,0} t_0 + a_{n,1} t_1 + \cdots + a_{n,k} t_k \]
\[ x_{n+1} = -t_{k+1} \]
\[ x_{n+2} = +t_{k+1} \]

We then perform the following changes of variables

\[ t_j^{\text{new}} = t_{k+1} - (1+c_0) t_0 - \cdots - (1-c_{j-1}) t_{j-1} - c_j t_j - c_{j+1} t_{j+1} - \cdots - c_k t_k \]
for \( j = 1, \ldots, k+1 \) and

\[ t_0^{\text{new}}, t_{k+1}^{\text{new}} - c_0 t_0 \cdots - c_j t_j \cdots - c_k t_k \]

The transformations of variables are legitimate because the \( c_j \)'s are strictly negative; so, all the coefficients in the above equations are positive or zero and the substitutions are column operations. Performing these column operations we obtain the following form:
Max \( x_0 = b_0 - t_0 \)

\[
\begin{align*}
  x_1 & = t_0 - t_1 - \cdots - t_k + t_{k+1} \\
  x_{k+1} & = t_0 - t_1 - \cdots - t_k + t_{k+1} \\
  x_{k+2} & = b_{k+2} + a_{k+2},_0 t_0 + a_{k+2},_1 t_1 + \cdots + a_{k+2},_{k+1} t_{k+1} \\
  \vdots & \vdots \vdots \vdots \vdots \\
  x_{n+2} & = b_{n+2} + a_{n+2},_0 t_0 + a_{n+2},_1 t_1 + \cdots + a_{n+2},_{k+1} t_{k+1}
\end{align*}
\]

(8)

where for notational simplicity we do not distinguish typographically the transformed coefficients and parameters of (8) from the old ones in (6).

To prove the equivalence of (8) and (6) it suffices to write down the inverse transformation. In fact

\[
\begin{align*}
  t_0 = t_0^{\text{new}} - t_1^{\text{new}} \\
  t_1 = t_1^{\text{new}} - t_2^{\text{new}} \\
  \vdots \\
  t_k = t_k^{\text{new}} - t_{k+1}^{\text{new}}
\end{align*}
\]

Tableau (8) is a special case of the desired form (5) with the extra feature that the new \( a_{ij}, b_i, j \)'s are zero in the lower left-hand corner of the diagonal part of the matrix. The method given above is a completely general method; however, the extra variables were added only to supply a coefficient of minus one in the functional. If one already exists, say \( c_k = -1 \), the same transformations (7) will work with \( k \) replaced by \( k-1 \).

The algorithm is illustrated by an example in Appendix B.
III. Solving a Special Subproblem for the Deepest Cut

We will now discuss a special problem with interesting properties, which constitutes the basis for the deepest cut algorithm described in this section.

1. The Deepest Cut Problem.

Let us consider the following problem: Find non-negative integers \( x_j(j \neq 0) \), \( t_i \), and maximize \( x_0 \), subject to:

\[
\begin{align*}
    x_0 &= b_0 - t_0 \\
    x_1 &= b_1 + a_{10}t_0 - t_1 \\
    x_2 &= b_2 + a_{20}t_0 + a_{21}t_1 - t_2 \\
    &\vdots \\
    x_k &= b_k + a_{k,0}t_0 + a_{k,1}t_1 + \cdots + a_{k,k-1}b_{k-1} - t_k \\
    x_{k+1} &= -b_{k+1} - a_{k+1,0}t_0 - a_{k+1,1}t_1 - \cdots - a_{k,k-1}b_{k-1} + t_k
\end{align*}
\]

where all the \( b_j(j \neq 0) \) and \( a_{ij} \) are non-negative integers.

It is not difficult to see that, given this structure, the lexicographical minimum of all \( t \) vectors generating feasible \( x'\)'s is a solution to the problem. In order to make more apparent some properties of this problem let us rewrite it for \( k = 3 \).

Find non-negative integers \( x_j(j \neq 0) \), \( t_i \) and maximize \( x_0 \), subject to:
\[ \begin{align*}
    b_0 &= x_0 + t_0 \\
    b_1 &= -a_{10}t_0 + x_1 + t_1 \\
    b_2 &= -a_{20}t_0 - a_{21}t_1 + x_2 + t_2 \\
    b_3 &= -a_{30}t_0 - a_{31}t_1 - a_{32}t_2 + x_3 + t_3 \\
    x_4 &= -b_4 - a_{40}t_0 - a_{41}t_1 - a_{42}t_2 - t_3
\end{align*} \]

An equivalent statement of this problem is to consider the last equation as the objective and find the lexico-min of all the vectors \( t \) which make \( x_4 \) non-negative. The solution of this problem is an optimal solution to the initial problem. So, we have then

Objective: \( x_4 = -b_4 - a_{40}t_0 - a_{41}t_1 - a_{42}t_2 - t_3 \).

The first of the equations of the system (the original functional) is not really a constraint, since \( t_0 \) may be increased as much as necessary to achieve primal feasibility. In order to make \( x_4 \) feasible (keeping in mind that we are seeking the lexico-min vector \( t \)), we will first increase \( t_3 \) as much as possible, keeping other \( t_i \) variables at zero value. This means, with respect to the simplex method, making \( t_3 \) and \( x_0, x_1, x_2 \) basic.

If we did not succeed in reaching feasibility, we would want then to increase \( t_2 \) keeping \( t_0 \) and \( t_1 \) at zero level if this can do some good, i.e., if the coefficient of \( t_2 \) in the adjusted "objective" (\( x_4 \)) is greater than zero.

If increasing \( t_2 \) as much as possible, (which is equivalent to introducing \( t_2 \) in the basis when using the simplex method) does not suffice to make \( x_4 \geq 0 \), it means that there is no solution with \( t_0 = t_1 = 0 \). We then continue the process with \( t_1 \) and then with \( t_0 \) if necessary.
Performing these operations, there will be some \( j \) such that making \( t_j \) basic will yield a non-negative value for \( x^*_4 \) (the value of the adjusted objective). This implies that there exist a solution to the problem with \( t_0 = t_1 = \ldots = t_{j-1} = 0, \ t_j \) basic and assigning some values to basic \( t_{j+1}, t_{j+2}, \ldots, t_k \). But we are seeking the lexico-min vector \( t \) and increasing \( t_j \) as far as possible is not necessarily required. There may exist a value, say \( t^*_j \), less than the value which is assigned to \( t_j \) when we make it basic, which should suffice to make \( x^*_4 \) reach feasibility.

In the case where \( k = 3 \) \(^{(10)}\) let \( j = 1 \)

1) We introduce \( t_3 \) in the basis then the "adjusted objective" becomes:

\[
x^*_4 = x_4 + x_3 = (b_3 - b_4) + (a_{30} - a_{40})t_0 + (a_{31} - a_{41})t_1 + (a_{32} - a_{42})t_2 - x_3
\]

2) Assume \( b_3 - b_4 < 0 \)

If \( a_{32} - a_{42} \) is \( > 0 \), we then consider increasing the value of \( t_2 \) and the new "adjusted objective" will be the following linear combination

\[
x^*_4 = (a_{32} - a_{42})x_2 + x_3 + x_4
\]

But let us suppose for the sake of this discussion that \( a_{32} - a_{42} \leq 0 \).

3. We then consider increasing the value of \( t_1 \) (assuming \( a_{31} - a_{41} = \Pi_1 > 0 \)). Increasing \( t_1 \) as far as possible (or making it basic as in the simplex method) gives the following adjusted cost form

\[
x^*_4 = x_4 + x_3 + \Pi_1 x_1 = (\Pi_1 b_1 + b_3 - b_4) + (\Pi_1 a_{10} + a_{30} - a_{40})t_0 - x_1
\]

\[+ (a_{32} - a_{42})t_2 - x_3.
\]

Let us now assume that \( \Pi_1 b_1 + b_3 - b_4 > 0 \) that implies that giving to \( t_1 \) the value \( b_1 \). This is sufficient to make \( x^*_4 \) reach feasibility.

-13-
But if we examine the relation (11) we see that it suffices to give to \( t_1 \) the value

\[
- \left[ \frac{(b_3 - b_4)}{a_{31} - a_{41}} \right] = t_1^o
\]

to make \( x_4 \) feasible.

It is easy to see that the value \( t_1^o \) is less than or equal to the value assigned to \( t_1 \) basic. There \( t_1^o \) is the minimum value for \( t_1 \) so that there exist a solution to our problem with \( t_0 = 0 \); this will assure us that

\[
t_0 = 0, \ t_1 = t_1^o, \ t_2 = t_2^o, \ t_3 = t_3^o \text{ (to fix)}
\]

is the lexico-min vector \( t \) which gives the solution of the problem, \( t_2^o = 0 \) because the coefficient of \( t_2 \) was never positive in the successive "adjusted cost forms."

In order to determine the value of \( t_3 \) we fix \( t_0 = 0, \ t_1 = t_1^o, \ t_2 = t_2^o = 0 \) and modify our constant column accordingly.

\[
\bar{b}_3 = b_3 + a_{30} \cdot 0 + a_{31} t_1^o + a_{32} \cdot 0
\]

\[
\bar{b}_4 = -b_4 - a_{40} \cdot 0 - a_{41} t_1^o + a_{42} \cdot 0
\]

Repeating the same operations as described above, it is not difficult to show that \( t_3^o = -\bar{b}_4 \). This method is illustrated in Appendix B by an example.

**Properties of the Deepest Cut Problem**

1. The procedure suggested to solve the deepest cut problem is exactly the procedure used in the basic algorithm for generating cuts.
2. The procedure gives the best cut for the equations of the system, i.e., that given the criteria used for generating efficient cuts in the basic algorithm, there is no better cut possible.

3. The technique leads in a sequence of \( k \) steps to the solution. Each component of the vector \( t \) is considered once and only once to enter the basis, i.e., if at iteration \( \nu(\leq k+j+1) \) the variable \( t_j \) was not selected for entering the basis, then for all the following iterations \( \nu+1, \nu+2, \ldots \) it will never be considered again.

In the case when \( k = 3 \), we have seen that if at iteration \( \nu+2 \), \( t_2 \) has a negative coefficient, this coefficient always remains negative during the solution process, \( t_2 \) is never a candidate for entering the basis.

4. It is possible to give an upper bound for the number of steps, i.e., each variable \( t \) is considered only once for entering in the basis so the number of iterations depends only on the size of the system and is in fact at most \( k \).

2. The Deep Cut Algorithm.

If rather than dealing with the basic canonical form we had the following revised form:

Find non-negative integers \( x_j \) and \( t_i \) and maximize \( x_0 \) subject to

\[
\begin{align*}
x_0 &= b_0 - t_0 \\
x_1 &= b_1 + t_0 - t_1 \\
x_{k+1} &= b_{k+1} - t_k \\
x_{k+2} &= b_{k+2} + a_{k+2,0} t_0 + a_{k+2,1} t_1 + \ldots + a_{k+2,k+1} t_{k+1} - t_{k+2} \\
\vdots \\
x_n &= b_n + a_{n,0} t_0 + a_{n,1} t_1 + \ldots + a_{n,k+1} t_{k+1} - t_{k+2} \\
\vdots \\
x_{n+3} &= b_{n+3} + a_{n+3,0} t_0 + a_{n+3,1} t_1 + \ldots + a_{n+3,k+1} t_{k+1} - t_{k+2} \\
x_{n+4} &= -b_{n+4} - a_{n+4,0} t_0 - a_{n+4,1} t_1 - \ldots - a_{n+4,k+1} t_{k+1} + t_{k+2}
\end{align*}
\]
where all the \(a_{ij}\) and \(b_i\) are positive (except \(b_0\)), then a natural algorithm, using the results of the last section, suggests itself.

In order to obtain the revised form, starting from the basic canonical form (8), we subtract from the right-hand side of the equations

\[
\begin{align*}
x_{k+2} &= b_{k+2} + a_{k+2,0}t_0 + a_{k+2,1}t_1 + \cdots + a_{k+2,k+1}t_{k+1} \\
& \vdots \\
x_{n+2} &= b_{n+2} + a_{n+2,0}t_0 + a_{n+2,1}t_1 + \cdots + a_{n+2,k+1}t_{k+1}
\end{align*}
\]

a parameter \(t_{k+2}\), and we add two equations

\[
\begin{align*}
x_{n+3} &= -t_{k+2} \\
x_{n+4} &= +t_{k+2}
\end{align*}
\]

which have the property that \(t_{k+2}, x_{n+3}, x_{n+4}\) must all be equal to zero in any feasible solution.

We then perform column operations in order to make the coefficient of \(t_0, t_1, t_2, \ldots, t_{k+1}\) positive (except for the coefficients which will appear in the last row \(n+4\)). The equation generating the column operations has the form

\[
s = -\beta_0 t_0 - \beta_1 t_1 - \beta_2 t_2 - \cdots - \beta_j t_j + \cdots - \beta_{k+1} t_{k+1} + t_{k+2}
\]

where \(\beta_j = -\min_{i \in I_j} \{a_{ij}\}\) \(i = k+2, k+3, \ldots, n+2, j = 1, 2, 3, \ldots, k+1\).

To fix \(\beta_0\) we need some more information. We want the coefficients of \(t_0\) to be positive, but we want also to determine the coefficients of \(t_0\) so that when we substitute \(K_0 + t_0\) for \(t_0\) (where \(K_0\) is an admissible value for \(t_0\)) the constant terms become non-negative.
We know (when dealing with a maximization problem) that the optimal solution \((x_0^{o \text{ LP}})\) of the linear program is always greater than or equal to the optimal solution \((x_0^{o \text{ IP}})\) of the same problem but where integral values have to be assigned to the variables.

\[
x_0^{o \text{ LP}} \geq x_0^{o \text{ IP}}
\]

but also

\[
\left[ x_0^{o \text{ LP}} \right] \geq x_0^{o \text{ IP}}
\]

let \(K_0 = -\left[ x_0^{o \text{ LP}} \right] + b_0\) (initial \(b_0\))

we have

\[
+ x_0^{o \text{ IP}} = b_0 - t_0^o
\]

\[
- \left[ x_0^{o \text{ LP}} \right] \leq - b_0 + t_0^o
\]

\[
t_0^o \geq K_0
\]

so \(t_0^o = K_0 + t_0^i\). This means that the substitution \(K_0 + t_0^i\) for \(t_0\) is a legitimate operation.

To choose \(\beta_0\) we will assure ourselves that the coefficients of \(t_0\) become non-negative and also that when we perform the substitution (15), described above, the constant term column will contain all non-negative components (except the last one).

\[
\beta_0 = \max \left\{ \min_i a_{i0}, \max_i \left[ \frac{b_i}{k_0} \right] - a_{i0} \right\}
\]

\[
i = k+2, \ldots, n+4
\]
After performing the column operation generated by (14), i.e., pivoting on $t_{k+2}$ we then add the cut

$$s = -K_0 + t_0$$

and pivot on $t_0$.

This gives us the tableau (13), which is equivalent to our initial problem with respect to the integer valued solution.

The Algorithm

The underlying procedure is equivalent to the basic algorithm method but in this case there is only one negative constant term, $b_{n+4}$. Each "subproblem" will contain

1) the last row: $n+4$
2) one row of the set $k+2, \ldots, n+2$ (the row $n+3$ is the opposite in sign of the row $n+4$)
3) the diagonal part of the tableau.

This is exactly the structure of the deepest cut problem.

To generate the best cut for each sub-problem we can use the method of the deepest cut problem.

And among those cuts select the best one, using the same criterion as for the basic algorithm.

This algorithm can be used if, (1) the general problem can be brought into the necessary form, and (2) it can be maintained in this form throughout the pivot steps. Both operations can be performed although the maintenance of non-negative coefficients below the diagonal does involve some number of column operations. If we omit the maintenance of the non-negative coefficient, we have the basic algorithm discussed in the first part of the paper. In that case, we generate deep cuts but not the deepest cuts for the subproblems.
REFERENCES


APPENDIX A

In the following four figures a flow diagram for the basic algorithm is given. Figure 1 tells us how to get the original problem in the form (1) into (5). In Figure 2 we associate to each equation with a negative constant term a linear combination of equations in which the coefficients of the rightmost t's have been made non-positive. In Figure 3 we select the linear combination for which the last positive term is furthest from the left, and in Figure 4 we actually perform the "cut" associated with this equation. The algorithm consists of repeating steps 2-4 until all the constant terms are non-negative.

1. CANONICAL FORM

Does \( \exists x_i \ s.t. \ c_i = -1 \) ?

No

Add \(-t_{k+1}\) to functional and two equations:

\[
\begin{align*}
    x_{n+1} &= -t_{k+1} \\
    x_{n+2} &= +t_{k+1}
\end{align*}
\]

Yes

Rearrange columns and rows \( c_k = -1 \)

\[
\begin{align*}
    k &:= k - 1 \\
    n &:= n - 2
\end{align*}
\]

Set \( t_j = t'_j \) where

\[
t'_j := t_{k+1} - (1+c_0)t_0 - \ldots - (1+c_{j-1})t_{j-1} - c_j t_j - c_{j+1} t_{j+1} - \ldots - c_k t_k
\]

\( j = 0, \ldots, k + 1 \)

Go on to

2. "DERIVED EQUATION"
2. DERIVED EQUATIONS

\[ \alpha_{0j} = C_j \quad i = 1, \ldots, n+2 \]
\[ \alpha_{ij} = a_{ij} \quad j = 0, \ldots, k \]
\[ \beta_i = b_i \]

\[ j = K + 1 \]
\[ i = i + 1 \]

\[ \beta_1 < 0 \]
\[ \beta_i + \alpha_{ij} \beta_j < 0 \]

\[ \beta_i = \beta_i + \alpha_{ij} \beta_j \]
\[ \alpha_{il} = \alpha_{il} + \alpha_{ij} \alpha_{jl} \quad l = 0, \ldots, K + 1 \]
3. BEST CUT

\[ i = 1 \]
\[ J = K + 1 \]

\[ i > n + 2 \]

Yes

\[ \beta_i < 0 \]

No

Yes

\[ \sigma_{Ij} > 0 \]

Yes

\[ \sigma_{Ij} = \{ \text{least integer } \geq \sigma_{Ij} \text{ which is divisible by } a_{IJ} \} \]

No

\[ \beta_i = -\{ \text{least integer } \geq b_i \text{ divisible by } a_{IJ} \} \]

\[ \beta_i : = \beta_i / a_{IJ} \]

\[ \sigma_{Ij} = 0 \]

\[ \alpha_{Ij} = \sigma_{Ij} / \sigma_{IJ} \]

\[ j = j + 1 \]

No

Yes

\[ r_i : = \text{Max } \{ j : d_{ij} > 0 \} \]

\[ i = i + 1 \]

\[ r_i < J \]

No

Yes

\[ J = r_i \]

\[ I = i \]

\[ i = i + 1 \]

Go on to 4
4. SUBSTITUTION

\[ i = 0, \ldots, n + 2 \]
\[ b_i = b_i - b_i a_{iJ} \]

\[ i = 0, \ldots, n+2; j = 0, \ldots, k+2 \]
\[ c_j = c_j - a_{ij} a_{0J} \]
\[ a_{ij} = a_{ij} - a_{ij} a_{iJ} \quad \text{if } j \neq J \]
\[ c_J = a_{0J} \]
\[ a_{iJ} = a_{iJ} \]

Does \( \exists a \)
\( b_i < 0 \)

- Yes: RETURN TO 2
- No: TERMINATE
APPENDIX B

Basic Algorithm:

The original problem

\[
\text{Max } x_0 = 10 - 2t_0 - t_1 - t_2 \\
x_1 = t_0 \\
x_2 = t_1 \\
x_3 = t_2 \\
x_4 = 26 + t_0 - 3t_1 - 5t_2 \\
x_5 = -12 + t_0 + t_1 + t_2
\]

Normally we use special methods to generate \(a - 1\) in the objective form but in this case this is not necessary.

Make the substitutions

\[
t_{0}^{\text{new}} = t_2 + 2t_0 + t_1 \\
t_{1}^{\text{new}} = t_2 + t_0 + t_1 \\
t_{2}^{\text{new}} = t_2 + t_0
\]

we then obtain

\[
x_0 = 10 - t_0 \\
x_1 = t_0 - t_1 \\
x_2 = t_1 - t_2 \\
x_3 = - t_0 + t_1 + t_2 \\
x_4 = 26 + 6t_0 - 9t_1 - 2t_2 \\
x_5 = -12 + t_1
\]

\((S_1 = -15 + t_0)\)
To simplify the problem we make use of a convenient device. If we solve the problem without the integer constraints we get $\text{Max } x_0^{\text{LP}} = 9/12$.

Since $x_0^{\text{IP}}$ the integer programming solution is maximized over a more restricted set $\text{Max } x_0^{\text{IP}} \leq \text{Max } x_0^{\text{LP}}$. Therefore, $\text{Max } x_0^{\text{IP}} \leq -5$.

Since $x_0 = 10 - t_0 \geq +15$. This allows us to introduce the cut $S_1 = -15 + t_0$. Doing the appropriate pivot (essentially $t_0 := t_0 - 15$) we obtain:

$$
\begin{align*}
\text{Max } & \quad x_0 = -5 - t_0 \\
& \quad x_1 = 15 + t_0 - t_1 \\
& \quad x_2 = t_1 - t_2 \\
& \quad x_3 = -15 - t_0 + t_1 + t_2 \\
& \quad x_4 = 116 + 6t_0 - 9t_1 - 2t_2 \\
& \quad x_5 = -12 + t_1
\end{align*}
$$

We now start using the procedure of the basic algorithm. Beginning with $x_3 = -15 - t_0 + t_1 + t_2$

we make the rightmost positive coefficient zero by adding $x_2$

$$
\begin{align*}
& \quad x_2 + x_3 = -15 - t_0 + 2t_1
\end{align*}
$$

If we try to eliminate the $t_1$ coefficient by adding $2x_1$ we obtain a non-negative constant term. The last equation yields the cut

$$
S'_2 = -8 + t_1 \iff t_1 \geq S_1.
$$
Using \( x_5 = -12 + t_1 \) we try to eliminate \( t_1 \) but again the constant term goes non-negative so the best we can do is \( S_2 = -12 + t_1 \). The latter cut is better so we introduce \( S_2 = -12 + t_1 \). Completing the standard pivot operations our tableau becomes

\[
\begin{align*}
\text{Max} & \quad x_0 = -5 - t_0 \\
& \quad x_1 = 3 + t_0 - t_1 \\
& \quad x_2 = 12 - t_1 - t_2 \\
& \quad x_3 = -3 - t_0 + t_1 + t_2 \\
& \quad x_4 = 8 + 6t_0 - 9t_1 - 2t_2 \\
& \quad x_5 = t_1
\end{align*}
\]

Here we have only one equation with a negative constant term which implies the cut \( S_3 = -3 + t_1 + t_2 \) making the substitution we obtain the optimal solution

\[
\begin{align*}
& \quad x_0 = -5 - t_0 \\
& \quad x_1 = 3 + t_0 - t_1 \\
& \quad x_2 = 9 + 2t_1 - t_2 \\
& \quad x_3 = 0 - t_0 + t_2 \\
& \quad x_4 = 2 + 6t_0 - 7t_1 - 2t_2 \\
& \quad x_5 = 0 + t_1
\end{align*}
\]

If we did not use the linear programming solution to generate the first cut this cut would have been

\[ S_1 = -12 + t_0 \]

derived from the following linear combination: \( x_1 + x_5 \). The tableau then becomes:
\[ x_0 = -2 - t_0 \]
\[ x_1 = 12 + t_0 - t_1 \]
\[ x_2 = t_1 - t_2 \]
\[ x_3 = -12 - t_0 + t_1 + t_2 \]
\[ x_4 = 98 + 6t_0 - 9t_1 - 2t_2 \]
\[ x_5 = -12 + t_1 \]

our best following cut is: \( S_2 = -12 + t_1 \) derived from \( x_5 \).

It is not difficult to verify that no better cut is available. We then obtain:

\[ x_0 = -2 - t_0 \]
\[ x_1 = t_0 - t_1 \]
\[ x_2 = 12 + t_1 - t_2 \]
\[ x_3 = - t_0 + t_1 + t_2 \]
\[ x_4 = -10 + 6t_0 - gt_1 - 2t_2 \]
\[ x_5 = + t_1 \]

We then get the following cuts

\[ s_3 = -2 + t_0 \]
\[ s_4 = -2 + t_1 + t_2 \]
\[ s_5 = -1 + t_0 \]
\[ s_6 = -1 + t_1 + t_2 \]

and obtain the same final tableau. It is interesting to see that this required 6 cuts where the previous method only required 3 cuts.
APPENDIX C

The Deepest Cut Problem:

Find non-negative integers $x_i$ ($i = 1, 2, 3, 4$), $t_j$ ($j = 1, 2, 3$), and maximize $x_0$
subject to:

$$
\begin{align*}
x_0 &= 3 - t_0 \\
x_1 &= 1 + t_0 - t_1 \\
x_2 &= 3 + 2t_0 + 2t_1 - t_2 \\
x_3 &= 8 + 3t_1 + 6t_2 - t_3 \\
"OBJ" & \quad x_4 = -26 - t_0 - 12t_1 - 3t_2 + t_3
\end{align*}
$$

Basic variables, $x_0, x_1, x_2, x_3$.

(a) We first ask is there a solution with $t_0, t_1, t_2 = 0$? Making $t_3$
    basic and increasing as far as we can to $t_3 = 8$ we get $x_4 = -18$.
    Eliminating $t_3$ from "OBJ" we obtain

$$
\begin{align*}
x_4 + x_3 &= -18 - t_0 - 9t_1 + 3t_2 
\end{align*}
$$

(b) Since $x_4$ with $t_0, t_1, t_2 = 0$ is $-18$ we now allow $t_2$ to become
    positive and require only that $t_0, t_1 = 0$. Making $t_2$ basic and
    eliminating from the "adjusted OBJ" we get

$$
\begin{align*}
x_4 + x_3 + 3x_2 &= -9 + 5t_0 - 3t_1 
\end{align*}
$$

(c) Since the coefficient of $t_1$ is negative in the current "OBJ"
    introducing $t_1$ will only make the problem more infeasible.

(d) Finally, by making $t_0 \geq 2$ we can make the problem feasible.
    Since we want the lexico-minimum vector $t$ we take $t_0 = 2$. 
The values of the rest of the variables can be determined whether by back substitution or by repeating the procedure with \( t_0 \) no longer a variable but fixed at 2, e.g., next try to make \( x_4 \) feasible in

\[
\begin{align*}
x_0 &= 1 \\
x_1 &= 3 - t_1 \\
x_2 &= 9 + 2t_1 - t_2 \\
x_3 &= 8 + 3t_1 + 6t_2 - t_3 \\
x_4 &= -28 - 12t_1 - 3t_2 - t_3
\end{align*}
\]

Repeating the same process, we can fix \( t_1 \); then \( t_2 \), \( t_1 \) being fixed, and so on.

The complete solution is

\[
\begin{align*}
x_0 &= 1 \\
x_1 &= 3 \\
x_2 &= 0 \\
x_3 &= 7 \\
x_4 &= 0 \\
t_0 &= 2 \\
t_1 &= 0 \\
t_2 &= 9 \\
t_3 &= 55
\end{align*}
\]

Practically one can affect each row of a multiplier

\[
\begin{align*}
\pi_4^4 &= 1(\text{OBJ}), \quad \pi_3^4 = 1, \quad \pi_2^4 = (a_{32} - a_{42}) = (6 - 3) = 3, \\
\pi_1^4 &= \text{Max} \left\{ (-12 + 3 + 3\pi_2^4) = -3, \ 0 \right\} = 0.
\end{align*}
\]

which remains always the same and may be considered to some extent as playing an identical rule to the simplex multipliers.
BASIC DISTRIBUTION LIST
FOR UNCLASSIFIED TECHNICAL REPORTS

Head, Logistics and Mathematical Statistics Branch
Office of Naval Research
Washington 25, D.C.

C. O., ONR Branch Office
Navy No. 100, Box 39, F.P.O.
New York City, New York

ASTIA Document Service Center
Arlington Hall Station
Arlington 12, Virginia

Institute for Defense Analyses
Communications Research Div.
von Neumann Hall
Princeton, New Jersey

Technical Information Officer
Naval Research Laboratory
Washington 25, D.C.

C. O., ONR Branch Office
346 Broadway, New York 13, NY
Attn: J. Laderman

C. O., ONR Branch Office
1030 East Green Street
Pasadena 1, California
Attn: Dr. A. R. Laufer

Bureau of Supplies and Accounts
Code OW, Dept. of the Navy
Washington 25, D.C.

Professor Russell Ackoff
Operations Research Group
Case Institute of Technology
Cleveland 6, Ohio

Professor Kenneth J. Arrow
Serra House, Stanford University
Stanford, California

Professor G. L. Bach
Carnegie Institute of Technology
Planning and Control of Industrial Operations, Schenley Park
Pittsburgh 13, Pennsylvania

Professor A. Charnes
The Technological Institute
Northwestern University
Evanston, Illinois

Professor L. W. Cohen
Math. Dept., University of Maryland
College Park, Maryland

Professor Donald Eckman
Director, Systems Research Center
Case Institute of Technology
Cleveland, Ohio

Professor Lawrence E. Fouraker
Department of Economics
The Pennsylvania State University
State College, Pennsylvania

Professor David Gale
Dept. of Math., Brown University
Providence 12, Rhode Island

Dr. Murray Geisler
The RAND Corporation
1700 Main Street
Santa Monica, California

Professor L. Hurwicz
School of Business Administration
University of Minnesota
Minneapolis 14, Minnesota

Professor James R. Jackson
Management Sciences Research Project, Univ. of California
Los Angeles 24, California

Professor Samuel Karlin
Math. Dept., Stanford University
Stanford, California

Professor C. E. Lemke
Dept. of Mathematics
Ren-Seiger Polytechnic Institute
Troy, New York

Professor W. H. Marlow
Logistics Research Project
The George Washington University
707 - 22nd Street, N.W.
Washington 7, D.C.

Professor Oskar Morgenstern
Economics Research Project
Princeton University
92 A Nassau Street
Princeton, New Jersey
BASIC DISTRIBUTION LIST
FOR UNCLASSIFIED TECHNICAL REPORTS

Professor R. Radner
Department of Economics
University of California
Berkeley, California

Professor Stanley Reiter
Department of Economics
Purdue University
Lafayette, Indiana

Professor Murray Rosenblatt
Department of Mathematics
Brown University
Providence 12, Rhode Island

Mr. J. R. Simpson
Bureau of Supplies and Accounts
Navy Department (Code W31)
Washington 25, D. C.

Professor A. W. Tucker
Department of Mathematics
Princeton University
Princeton, New Jersey

Professor J. Wolfowitz
Department of Mathematics
Lincoln Hall, Cornell University
Ithaca 1, New York