ZERO-ONE INTEGER PROGRAMMING

763

by

RATNAM VENKATA CHITTURI

B. S.(Mech. Engg), Andhra University
Waltair, India, 1964

A MASTER'S REPORT

submitted in the partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1967

Approved by:

Dr. Frank A. Tillman

Major Professor

TABLE OF CONTENTS

	Page
INTRODUCTION	1
ZERO-ONE INTEGER PROGRAMMING PROBLEM FORMULATION	9
CUTTING PLANE METHODS	11
PARALLEL SHIFTS OF THE OBJECTIVE FUNCTION HYPER PLANE	22
COMBINATORIAL METHODS	41
ADDITIVE ALGORITHM OF BALAS (WITH SOME MODIFICATIONS)	62
SUMMARY	94
CONCLUSION	95
ACKNOWLEDGEMENTS	97
REFERENCES	98
APPENDICES	100
APPENDIX I	101
APPENDIX II	109
APPENDIX III	116
APPENDIX IV	127

INTRODUCTION

Linear programming was developed in the early 50's. Since then, it has become a powerful mathematical tool in solving a number of military, economic, and industrial problems. Primarily, linear programming helps in selecting a best schedule of actions among many conflicting alternatives and may be applied to systems having the following characteristics:

1. Some objective to be optimized, such as maximum profit or minimum cost.
2. There are a number of variables to be considered simultaneously.
3. In addition to many variables, there are a number of constraints on the system which the solution must satisfy. These restrictions are linear and may represent such quantities as production capacity or limited sales.

In general, linear programming problems are solved by the simplex method. In some situations, the simplex solution which allows for fractional values of the variables may not provide the solution that is desired. For example, in an airline scheduling, it makes little sense to speak of an optimal solution in terms of fractional units of planes or people. It is important to note that the integer solution obtained by rounding off the fractional values does not usually constitute an optimal integer solution. This fact led to the development of integer linear programming in 1959, which guarantees an optimal integer solution.

In some situations, the variables in the integer solution may take only a zero or one value. This may occur in scheduling and machine sequencing. Because of the special form of the solution and because of computational difficulties with the current integer programming algorithm, much work has been done to find an efficient algorithm for solving the zero-one integer programming problem. The purpose of this paper is to review some of the better known zero-one algorithms and discuss their respective advantages and disadvantages. A number of practical problems can be classified as zero-one integer programming problems. The following are examples of these type of problems:

a. Scheduling of jobs through a production facility.

b. The machine loading or sequencing problem.

c. Problems where decision A or B must be made, but not both A and B.

These problems are special cases of linear programming problems in which the decisions are of the "either - or" type and where a linear objective function is to be optimized and the system is subject to restraints. To simplify the discussion, the zero-one problem is presented in the manner of the linear programming (L.P.) problem. As mentioned earlier, the objective of a linear programming problem is to maximize or minimize a linear objective function subject to a set of linear equality or inequality constraints governing the system.

Linear Programming

The general form of a linear programming problem is stated as:

$$\begin{aligned} &\text{minimize} && Z = CX \\ &\text{subject to} && AX \gtreqless b \\ & && X \geq 0 \end{aligned} \qquad (1)$$

where C, X and b are vectors and A is an n-dimensional matrix and Z is a scalar.

From elementary algebra, it is known that a system of linear equalities either has a unique solution or has no solution at all. However, a system of linear inequalities may have an infinite number of solutions or a finite number of basic solutions. A basic solution to a linear programming problem having m constraints and n variables, where $n > m$, is the one in which, at most, m variables take values greater than zero and the remaining variables are set equal to zero. It is important to note that a solution must be non-negative or feasible, that is $x_j \geq 0$, $j = 1, 2, 3, \ldots, n$. A feasible solution which yields a maximum or minimum value for the objective function is called an optimal solution to the linear programming problem. If X' is an optimal feasible solution, then $Z^* = CX'$, $X' \geq 0$ and $AX' - b \geq 0$.

where Z^* = the optimal value of the objective function.

Generally linear programming problems are solved by the simplex method developed by Dantzig [4].

Integer Programming

A linear programming problem becomes an integer programming problem when one or more of the variables are required to have integer solution values. Hence it can be shown that the problem (1) becomes an integer programming problem when the following restriction is added:

$$x_j = \{x_j : x_j = 0, 1, 2, \ldots ; j \in J\} \qquad (2)$$

where $J \subseteq N$, $N = \{1, 2, 3, \ldots , n\}$.

Beale Equations

To better understand the methods of integer programming, it is helpful to review Beale Equations. E. M. L. Beale [3] was the first to represent the linear programming problem by equalities in non-negative variables. If a slack vector X_2 is introduced into problem (1), it can be rewritten as

$$\begin{aligned} &\text{minimize} && a_{oo} = a_o X_1 \\ &\text{subject to} && A_1 X_1 + I X_2 = b \\ & && X_1, X_2 \quad 0 \end{aligned} \qquad (3)$$

where $a_o = (a_{oj})$ is an 1 x n row vector,
$X_1 = (x_j)$ is an n x 1 column vector

---

*In this paper $\subseteq$ is meant to be for inclusion, where as $\subset$ is for strict inclusion.

$X_2 = (x_{n+i})$ is an m x 1 slack (column) vector

$A_1 = (a_{ij})$ is an m x n matrix of coefficients

I = an m x m identity matrix

$b = (b_i)$ is an m x 1 requirement vector.

By rearranging the vectors in the above problem, it is put in the following form.

$$\text{Minimize} \qquad x_o = a_{oo} - a_o X_1$$

$$\text{subject to} \qquad X_2 = b - A_1 X_1$$

$$X_1, X_2 \geq 0.$$

where x_o is the value of the objective function.

Now an integer programming problem can be stated in Beale Equation form as follows: Find $x_o, x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}$ with x_j, $j \in J$ integer valued, which

$$\text{minimize} \qquad x_o = a_{oo} + \sum_{j=1}^{n} a_{oj}(-x_j)$$

$$x_{n+i} = a_{io} + \sum_{j=1}^{n} a_{ij}(-x_j) \quad 0 \qquad (4)$$

$$x_j \geq 0$$

$$x_j \text{ an integer} \quad j \in J, \quad J \subseteq N$$

where x_j = non-basic variables

x_{n+i} = basic variables

a_{ij} = coefficients with integer values

$i = 1, 2, 3, \ldots, m$

$j = 1, 2, 3, \ldots, n.$

This problem can be solved by the simplex method and if the solution obtained has the integer property, it is the optimal integer solution. This is usually not the case, thus other methods are required to solve this problem. The most common method is a modification to the simplex method, due to Gomory [10,11] which requires the addition of additional constraints to the optimal simplex solution and force the solution to be integer valued. This is one of the cutting plane methods which will be discussed later in detail.

Zero-one Integer Programming

A special case of integer linear programming is called zero-one integer linear programming which is the author's prime concern in this paper. This special case occurs when one or more of the variables are restricted to the integers 0 or 1. For simplicity, this zero-one integer linear programming is called the zero-one problem. Now it is evident that problem (2) becomes a zero-one problem when the following restriction is added to it.

$$x_j = x_j : x_j = 0, 1 ; \quad j \in J \quad . \qquad (5)$$

where $J \subseteq N$, $\{N = 1, 2, 3, \ldots, n\}$.

Due to the special nature of the problem, the solution space is

restricted to the unit cube in hyper-space for those variables which are restricted to be 0 or 1. Thers are a number of algorithms for solving these zero-one problems and they are classified into the following four groups. They are

1. Cutting plane methods.
2. Parallel shifts of the objective function hyper-plane.
3. Boolen Algebra methods.
4. Combinatorial methods.

Literature Survey

Gomory [10,11] was successful in solving problem (2) by using the cutting plane methods. He developed two algorithms which guarantee an integer optimal (if one exists) solution in a finite number of steps. These algorithms can be utilized to solve zero-one problems simply by adding the additional restriction denoted by (5). The second method involves in obtaining a non-integer optimal solution by simplex method, and then adding restraints which forces the solution to integer that is 0 or 1, and resolving the problem. This additional restraint, at each iteration, is obtained from the objective function. The idea is to force the objective function value to be integer and then find the integer solution (if it exists) which corresponds to it. This method was developed by Elmaghraby [5].

The boolen algebra methods are due to R. Fortet [14] and R. Camon [13]. These are not discussed in this paper because

of their special nature. The principal combinatorial methods which are used for solving zero-one problems, are due to Balas [1] and Glover [8,9].

Purpose

In this paper, an attempt is made to review some of the above methods which include the cutting plane, parallel shifts of the objective function hyper-plane and combinatorial methods. The formulation of a zero-one problem is discussed in the first section. The second section is devoted to the cutting plane methods of Gomory [10,11]. Elmaghraby's method [5] is discussed in the third section. The remaining sections are devoted to a general discussion of the recently developed combinatorial approach and the particular algorithm of Balas [1].

ZERO-ONE INTEGER PROGRAMMING PROBLEM FORMULATION

The general form of the zero-one integer programming problem may be stated as follows:

$$\begin{aligned} &\text{minimize} && C' X' \\ &\text{subject to} && A' X' \gtrless b' \\ & && x_j' = 0 \text{ or } 1 \\ & && j = 1, 2, 3, \ldots, n. \end{aligned} \tag{6}$$

In this discussion, the constraints are transformed into inequalities of the form ($\leq$) and all the coefficients in the objective function, which is to be minimized, are transformed to be positive. This is done in the following manner:

a. Replace all exact equations by two inequalities, one a greater than or equal to ($\geq$) and the other, a less than or equal to ($\leq$).

b. Multiply all greater than or equal to ($\geq$) inequalities by (-1).

c. Set

$$x_j = \begin{cases} x_j' & \text{for } C_j' \geq 0 \text{ when minimizing, for} \\ & \quad C_j' \leq 0 \text{ when maximizing.} \\ 1 - x_j' & \text{for } C_j' \leq 0 \text{ when minimizing, for} \\ & \quad C_j' \geq 0 \text{ when maximizing.} \end{cases}$$

Finally by introducing an m-component non-negative slack vector Y, the problem may be restated in the desired form, that is, find X which

$$
\begin{aligned}
&\text{minimizes} && Z = CX \\
&\text{subject to} && AX + Y = b \\
& && X,\ Y \geq 0 \\
& && x_j = 0 \text{ or } 1\ , \quad j \in N
\end{aligned}
\tag{7}
$$

where $X = (x_j)$ is an n x 1 column vector

$C = (c_j)$ is a 1 x n row vector with $C \geq 0$

$A = (a_{ij})$ is an m x n matrix

$b = (b_i)$ is a m x 1 column vector

$Y = (y_i)$ is a m x 1 column vector

N = a set of indices j for the variables x_j

$= \{ 1, 2, 3, \ldots, n \}$.

It is noted that all the algorithms described in this paper require the problems be dual-feasible that is all $C_j \geq 0$. The discussion of the zero-one algorithms to be presented in the following sections, is based on the above formulation (7) of the problem.

CUTTING PLANE METHODS

Gomory [10,11] developed two algorithms for obtaining an optimal solution to a general integer linear programming problem. In these algorithms, the solution obtained is integer not necessarily zero or one. The first method, called All Integer Method, was initially developed in 1958. The second method, called All Integer Integer Method, was developed in 1960. The basic requirement of both the methods is that any problem to be solved has to be dual-feasible and must have the coefficient matrix A in integers. It is interesting to note that the second method maintains integers in the tableaus throughout all iterations, as all the computations are done by additions and subtractions only. However, only the first method, that is All Integer Method will be discussed in this paper. For the second method the reader is referred to [11].

The All Integer Method is a cutting plane method. Principally all cutting plane methods use the simplex algorithm to obtain a non-integer optimal solution. Some type of a constraint generation technique is then utilized to reduce the solution space so that the integer optimal solution is obtained. These added constraints cut into the solution space as deeply as possible without excluding any integer solutions. This is the reason these methods are called cutting plane methods. As mentioned earlier, only the All Integer Method will be described in this paper.

To use this algorithm it is necessary to formulate the

problem according to the Beale Equation form (4). One must remember that if a given problem is not dual-feasible, then it is necessary to reformulate the given problem into a dual-feasible problem as explained in the previous section. By adjoining the constraints $x_j = (-1)\ (-x_j)$, $j \in N$, the above problem can be stated as

$$\text{minimize} \quad Z = z_o + \sum_{j=1}^{n} C_j(-x_j)$$

$$\text{subject to} \quad y_i = x_{n+i} = b_i + \sum_{j=1}^{n} a_{ij}(-x_j)$$

$$y_{m+j} = x_{m+n+j} = 1 - 1\ (-x_j)$$

$$x_j = 0 - 1\ (-x_j)$$

$$x_j \geq 0 , \quad \text{all } j$$

This problem may be exhibited in a tableau form. This is shown in tableau 1. The variables $z, y_1, y_2, \ldots, y_{m+n}$ are basic variables where as $x_1, x_2, \ldots, x_n$ are non-basic variables. Since the simplex solution puts no upper bound on the variables, it is necessary to include constraints $x_j \leq 1$, $j \in N$ which represent upperbounds on the variables for the zero-one problem. These constraints are included in the tableau 1.

The linear programming problem displayed in tableau 1 is solved by using the simplex method. If the optimal solution is integer valued, it is then the feasible and optimal solution to the zero-one problem. Otherwise, a new constraint has to be

Tableau 1

	1	$-x_1$	$-x_2$	. . .	$-x_n$
$Z =$	z_o	c_1	c_2	. . .	c_n
$y_1 =$	b_1	a_{11}	a_{12}	. . .	a_{1n}
$y_2 =$	b_2	a_{21}	a_{22}	. . .	a_{2n}
.	.	.	.		.
.	.	.	.		.
.	.	.	.		.
$y_m =$	b_m	a_{m1}	a_{m2}	. . .	a_{mn}
$y_{m+1} =$	1	1	0	. . .	0
$y_{m+2} =$	1	0	1	. . .	0
.	.	.	.		.
.	.	.	.		.
.	.	.	.		.
$y_{m+n} =$	1	0	0	. . .	1
$x_1 =$	0	-1	0	. . .	0
$x_2 =$	0	0	-1	. . .	0
.	.	.	.		.
.	.	.	.		.
.	.	.	.		.
$x_n =$	0	0	0		-1

obtained to exclude the non-integer portions of the optimal solution, but not any of the feasible solutions to the zero-one problem. To obtain such a constraint, it is necessary to reformulate the problem at the optimal or final iteration. Let B be the basis matrix (the matrix composed of the columns of A corresponding to x_j in the solution) for the optimal solution and let R be the set of j corresponding to the non-basic variables (the variables x_j which are not in the solution and which are set equal to zero). The coefficients a_{ij} in the matrix A are transformed at each iteration in obtaining the current basic solution. This transformation is referred to as updating. Denote y_j as the updated column vectors at the final iteration and Y_o as the updated requirement vector (often called as the right hand side vector). Also let X_B be a vector containing basic variables for the optimal solution. The set of variables in the solution which may take values other than zero is called the basis. The number of variables in a basis corresponds to the number of restrictions in the problem. At the initial iteration, the slack vector Y, constitutes the basis. This basis changes at each iteration as a variable is removed from the basis while another non-basic variable enters the basis. This change results in an increase in the objective function value for a maximization problem and a decrease in the value for the minimization problem. This increase or decrease occurs until the optimal solution is reached. The optimal solution is as follows

$$y_j = B^{-1} a_j \qquad j \in N$$
$$Y_o = B^{-1} b$$
$$X_B = Y_o$$

Furthermore, any feasible solution X must satisfy

$$X_B = Y_o + \sum_{j \in R} y_j (-x_j) \tag{8}$$

There is only one solution to (8) with $x_j = 0$, $j \in R$, which is the basic feasible solution $X_B = Y_o$, $X_R = \{ x_j = 0; \ j \in R \}$. Suppose that not all components of Y_o are integers, in particular assume y_{uo} is not an integer. Then consider the u-th equation of (8) which is

$$x_{B_u} = y_{uo} + \sum_{j \in R} y_{uj} (-X_j) \tag{9}$$

Now let

$$\begin{aligned} y_{uj} &= \delta_{uj} + f_{uj} \ , \quad j \in R \\ y_{uo} &= \delta_{uo} + f_{uo} \end{aligned} \tag{10}$$

where δ_{uj} is the <u>largest</u> integer less than or equal to y_{uj}, $j \in R$ and $j = 0$. If $y_{uj} > 0$, then $\delta_{uj} \geq 0$ and $0 \leq f_{uj} < 1$. On the other hand if $y_{uj} < 0$ then $\delta_{uj} \leq 0$, but the fraction f_{uj} remains positive, that is $0 \leq f_{uj} < 1$. Hence f_{uj} is always greater than or equal to 0. Furthermore $f_{uo} > 0$ by the assumption that y_{uo} is not an integer and from the fact that the

solution is feasible, y_{uo} is a positive real number. Substituting (10) into (9), we obtain

$$x_{B_u} = \delta_{uo} + \sum_{j \in R} \delta_{uj}(-x_j) + f_{uo} + \sum_{j \in R} f_{uj}(-x_j)$$

or

$$x_{B_u} - \delta_{uo} - \sum_{j \in R} \delta_{uj}(-x_j) = f_{uo} + \sum_{j \ R} f_{uj}(-x_j) \quad (11)$$

Now for any integer feasible solution to (8) (that is all $x_j \geq 0$, $j \in N$ are integers) which may not be optimal or necessarily basic (that is some $x_j > 0$, $j \in R$) to the original problem, the left hand side of the equation (11)

$$x_{B_u} - \quad uo - \sum_{j \in R} \delta_{uj}(-x_j) \quad (12)$$

will be an integer. And the quantity in (12) may not necessarily be non-negative because δ_{uj} is an integer and may be less than zero and x_j, $j \in R$ is a non-negative integer from the feasible solution. Since (12) has an integer value and from equation (11), we find

$$f_{uo} + \sum_{j \in R} f_{uj}(-x_j) \quad (13)$$

must also be an integer. Now $\sum_{j \in R} f_{uj}(-x_j)$ cannot be positive, since $f_{uj} \geq 0$ and $x_j \geq 0$. Thus from the fact that (13) is an integer and since $0 < f_{uo} < 1$ along with the fact that

$\sum_{j \in R} f_{uj}(-x_j) \leq 0$, the quantity in (12) cannot be a positive integer. Therefore every feasible solution to an integer problem (that is non-negative and integer valued) must satisfy

$$f_{uo} + \sum_{j \in R} f_{uj}(-x_j) \leq 0 \qquad (14)$$

or

$$-f_{uo} - \sum_{j \in R} f_{uj}(-x_j) \geq 0 \qquad (15)$$

Clearly the optimal solution to a linear programming problem does not satisfy (15), since $x_j = 0$; $j \in R$. Thus if (15) is added to the linear programming problem, the new set of feasible solutions will be smaller than that for the original linear programming problem, but still contains all feasible solutions to the integer problem. The procedure then is to attach (15) to the linear programming problem and solve the resulting problem which now has (m+1) constraints.

Let S_1 be the slack for (15); the subscript 1 indicates that it is the slack variable for the first cut annexed to the linear programming problem. Hence,

$$S_1 = -f_{uo} - \sum_{j \in R} f_{uj}(-x_j) \qquad (16)$$

However, it was observed that for any integer solution to (8), (13) must be an integer; hence in equation (16), S_1 will be an integer. Thus one need not be concerned about the fact that the

new variable introduced into the problem may not be integer. Note that the basis matrix for the augmented problem is now

$$B_1 = \begin{bmatrix} B & \bar{0} \\ \bar{0}' & 1 \end{bmatrix} \quad \text{and} \quad B_1^{-1} = \begin{bmatrix} B^{-1} & \bar{0} \\ \bar{0}' & 1 \end{bmatrix}$$

where $e_{m+1} = \begin{bmatrix} \bar{0} \\ 1 \end{bmatrix}$ is the activity vector corresponding to S_1. Thus a basic solution to the augmented problem is

$$B_1^{-1} \begin{bmatrix} b \\ -f_{uo} \end{bmatrix} = \begin{bmatrix} Y_o \\ -f_{uo} \end{bmatrix}$$

This basic solution is not feasible, since $-f_{uo} < 0$. The vector containing the cost coefficients corresponding to B_1 is $(C_B , 0)$. Thus, the $Z_j - C_j$ for the augmented problem are precisely the same as those for the original problem. The y_j, $j \in R$; for the augmented problem (denoted by y_j^1 for the first cut) are

$$y_j^1 = B_1^{-1} \begin{bmatrix} a_j \\ -f_{uj} \end{bmatrix} = \begin{bmatrix} y_j \\ -f_{uj} \end{bmatrix} \qquad (17)$$

Therefore the current $Z_j - C_j$ for the new problem are precisely the same as those for the optimal solution of the original problem. As a result, we have a basic optimal solution (all $Z_j - C_j \geq 0$), but not feasible to the augmented problem. Hence the

dual simplex algorithm is applied to obtain an optimal feasible solution. The initial simplex tableau for the augmented problem is obtained from (17) by adding another row to the tableau corresponding to the optimal solution of the original linear programming problem. The quantity $(-f_{uo})$ is entered in this new row in the solution or Y_o column and $(-f_{uj})$ are entered in the $j \in R$ columns. Dual simplex algorithm is then applied to the augmented problem until a feasible solution is obtained. If this solution has the required integer property, it is then the optimal feasible solution to the integer problem. Otherwise the process is repeated by adding another constraint similar to (16) until an integer feasible solution is obtained. Now the iterative procedure for obtaining a zero-one solution is summarized below.

Step 1. Formulate the zero-one problem into a dual-feasible problem as described in the last section. Arrange the problem in a tableau form similar to tableau 1.

Step 2. Obtain an optimal solution using the simplex method. If the solution is in integers (0 or 1), it is the optimal solution to the given problem and the process terminates. Otherwise go to step 3.

Step 3. If more than one component of Y_o is non-integral select the component for which f_{uo} is the largest. This allows us to make the largest possible cut, but it should be noted that it may not lead to the integer solution in the minimum number of cuts. From the row corresponding to maximum f_{uo}, form the

constraint

$$S_1 = - f_{uo} - \sum_{j \in R} f_{uj}(-x_j) \qquad (18)$$

and augment to the last tableau (in the first iteration of the integer algorithm, the last tableau is the one corresponding to the optimal solution of the original linear programming problem). The large f_{uo} ensures a deep cut, but does not exclude any of the integer solutions.

In the dual simplex algorithm, the variable which is most negative is normally selected to leave the basis. However since there is only one negative basic variable $S_1 = -f_{uo}$, it is selected to leave. Let the augmented row be denoted as row r. Thus $x_{B_r} = S_1$ and it leaves the basis.

Step 4. In the dual simplex algorithm, the variable selected to enter the basis is determined by computing

$$\frac{c_k}{a_{rk}} = \min \left| \frac{c_j}{a_{rj}} \right| , \; a_{rj} < 0 . \qquad (19)$$

The column k for which equation (19) holds, is selected to enter the basis.

Step 5. Form a new tableau, where x_k replaces x_{B_r} in the basic solution to the augmented problem. The new solution may not be primal feasible, in which case the dual simplex algorithm is again applied until the problem is primal feasible. If

the solution is integer valued, it is then optimal and feasible to the original integer problem; otherwise return to step 3.

Note. In Gomory's method, when a constraint is generated, the prior ones can be disregarded since they become redundant in the reduced solution space. Thus when a second constraint is generated, the first one can be ignored; when a third constraint is generated, the second one can be disregarded and so on. Also when the slack variables (corresponding to the added constraints) appear in the basis with a positive value, they may be disregarded from there on; that is the corresponding row and column may be removed from tableau.

An example problem is solved in Appendix I, illustrating this algorithm. The author feels that "the all integer method" described in this section is not a very efficient method for solving zero-one problems, since it requires the addition of n-constraints ($x_j \leq 1$, $j \in N$) which makes the problem unusually large. A problem of size m x n becomes (m+n)xn and hence requires excessive computational time. Some of the methods presented in the following sections appear to be more efficient for solving these problems.

PARALLEL SHIFTS OF THE OBJECTIVE FUNCTION HYPER PLANE

In 1963, Elmaghraby [5], developed a method by which an optimal solution can be obtained to the zero-one problem. As in cutting plane method of Gomory [10], he uses the simplex method to obtain a non-integer optimal solution. It was mentioned earlier in Gomory's all integer method that it is necessary to add the upperbound constraints $x_j \leq 1$, $j \in N$; and as a consequence makes it inefficient for large problems. However in Elmaghraby's method, this inefficiency was eliminated to a large extent by using a modified version of the simplex method called upperbound technique developed in 1954.

In an ordinary simplex method, the variables are lower bounded by zero and hence the variables never become negative. Similarly in the upperbound technique, the variables do not exceed the upperbound. For example, if the variables are upperbounded by 1 as in the case of the zero-one problem, the variables do not exceed one. This is the characteristic of the upperbound technique. However the computational time for the upperbound algorithm is usually greater than that for the simplex algorithm. The upperbound technique will not be explained except as it applies to the method of Elmaghraby. For those who are not familiar with it, a brief review is given in Appendix II and for a more detailed discussion, the reader is referred to any standard text on linear programming [4,12].

To facilitate the discussion, the zero-one problem (7) is restated as follows:

$$\begin{aligned} &\text{minimize} && Z = CX \\ &\text{subject to} && AX + Y = b \\ &&& X,\ Y \geq 0 \\ &&& x_j = 0 \text{ or } 1\ ,\ j \in J,\ J \subseteq N \end{aligned} \tag{21}$$

where all dimensions remain the same as before and where the solution space of the problem as stated in (21) is an n-dimensional space W. Thus one of the extreme points of W represents the optimal solution to the linear programming problem from (21). If the integer restriction is removed in (21), we have the following problem:

$$\begin{aligned} &\text{minimize} && Z = CX \\ &\text{subject to} && AX + Y = b \\ &&& X \leq 1 \\ &&& X,\ Y \geq 0\ . \end{aligned} \tag{22}$$

Since (22) is a linear programming problem, an optimal solution can be obtained using the upperbound version of the simplex method. Thus having solved problem (22) and if, in the optimal tableau F_o, the components of X have the integer property, then the optimal solution to (22) is also optimal for (21). Otherwise a new constraint is generated to force the solution to an integer solution.

As discussed in the previous section, Gomory uses one of the problem constraints to generate the new constraint, $S_1 = - f_{uo} - f_{uj}(-x_j)$, which reduces the solution space, but do not exclude any of the integer solutions. In theory, this ensures

that an optimal integer solution is reached in a finite number of steps. However, Elmaghraby uses a different approach. Instead of formulating the additional constraints from one of the problem constraints, he uses the objective function to obtain his new constraint. He claims that this is an efficient approach. The new constraint to be added to problem (22) is parallel to the objective function hyper plane $Z = CX$. The new problem is resolved and if the optimal solution obtained has the integer property (and is feasible to (21)), it is then the optimal solution to the integer problem. Otherwise another constraint parallel to the first one is obtained and the process is repeated until an integer feasible solution is obtained.

In summary, the Elmaghraby's method proceeds in the following manner. Initially the zero-one problem, without integer restriction, is solved using the upperbound version of the simplex method. If the optimal solution obtained is integer, it is then the optimal integer solution to the zero-one problem and the process terminates. Otherwise the objective function hyperplane is shifted in steps (parallel to itself) and the solution is checked at each step. Whenever the solution is integer, it is then the optimal integer solution to the original problem and the process terminates.

Before going through the determination of this additional constraint, it is of interest to understand the concept lying behind the Elmaghraby's method. In general, an integer problem may have an integer optimal solution inside the solution space .

W. Thus any of the extreme points may not constitute an integer optimal solution. To illustrate this, consider a solution space W defined by the two dimensional extreme points (0,0), (3,0), (2.6, 1.4), (1.5, 2.0) and (0, 2.5) as shown in Fig. 1. In fact there are only two integer extreme points and they are (0,0) and (3,0). The other integer points inside the space W are (0,1), (0,2), (1,0), (1,1), (1,2), (2,0) and (2,1). The integer optimal solution may correspond to one of these integer points inside the solution space W. Thus an optimal solution to the integer problem may not necessarily coincide with one of the extreme points.

However, the solution space W of a zero-one problem must be within a unit hyper cube in an n-dimensional plane, since the variables cannot exceed the value 1. A unit hyper cube does not have any integer points inside its space and as a consequence, the solution space W of a zero-one problem does not have any interior integer points. Thus if there exists an integer point in W, it must be a corner point, that is it must be an extreme point. From this discussion, it is evident that one of the basic solutions muet constitute an optimal solution to the zero-one problem (note that an optimal solution to an integer problem may not necessarily be basic). This property is the special nature of the zero-one problem which makes the solution process simple.

If we recall the procedure of the simplex method, it is observed that the basis changes at each iteration with an improvement (ΔZ) in the objective function. Intuitively it means that the solution point is moving from one extreme point to another

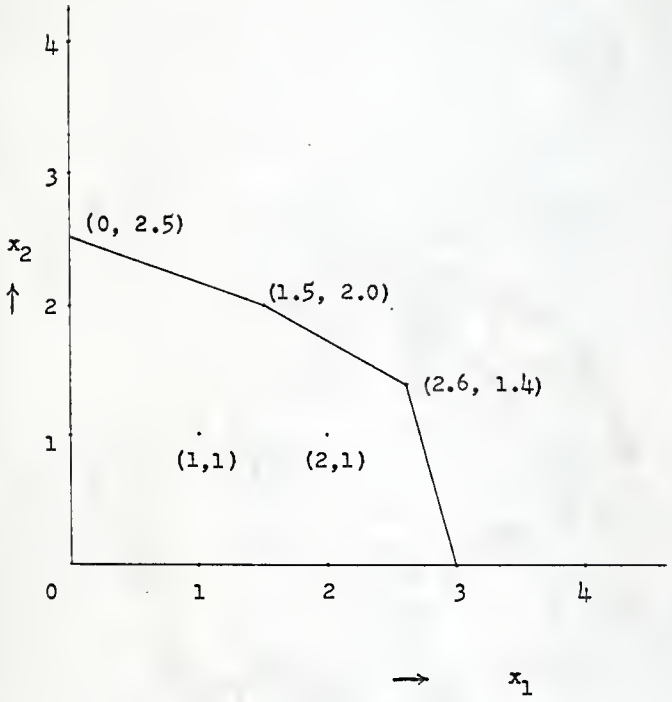

Fig. 1. Two dimensional solution space for a hypothetical problem.

extreme point in the forward direction (that is, with improved Z). This suggests that if the optimal solution to the linear programming problem derived from the zero-one problem is not integer, then the nearest integer point from the optimal solution point which is clearly an extreme point, must be an optimal solution to the zero-one problem. Now it is necessary to develop a method by which the nearest integer point can be reached.

After getting a non-integer optimal solution to the zero-one problem, if the solution point is moved backward from one extreme point to the next nearest extreme point until an integer point is found, it is then the optimal integer solution to the zero-one problem. For any extreme point, there may exist other extreme points having the same objective function value. Hence all these extreme points having the same objective function value lie on the same objective function hyper plane and they are called alternate extreme points. This suggests that even though an extreme point (corresponding to the present solution) is not integer, one of its alternate extreme points may be an integer. Thus, it is necessary to search all alternate solutions and check for an integer solution.

So far it is observed that, for any solution, it is first necessary to search all alternate solutions to obtain an integer solution and if none exists, the solution point must be moved to the next nearest extreme point. Now it is necessary to find how this nearest extreme point may be reached. Since moving the solution point from one extreme point to another extreme point

(but not alternate) corresponds to a change in the basis and the objective function value, the extreme point corresponding to the minimum change in the objective function is the next nearest extreme point. If this minimum change is denoted by d, then d represents the distance of the next nearest extreme point from the present extreme point and often called depth of cut.

Also each alternate extreme point may have a different nearest extreme point. Hence if there are L alternate extreme points, then there are L nearest extreme points that is there are L distances d_k , $k = 1, 2, 3, \ldots, L$, where d_k is the distance from the k^{th} alternate extreme point to its next nearest extreme point. Thus if d^* is the minimum of all these distances d_k ; $k = 1, 2, \ldots, L$, the extreme point corresponding to d^* is then the next nearest extreme point to which the solution point must be moved. Intuitively this extreme point (corresponding to d^*) is the nearest extreme point from the objective function hyper plane. This ensures that no extreme point is excluded from the search. In one sense d^* represents the least change in the objective function value from the present basic solution to the next basic solution (in other words, no other basic solution results in a change in the objective function value less than d^*). In another sense d^* represents the maximum change in the objective function value before the present basic solution becomes infeasible; that is, if the proposed change d is less than d^*, the basis remains the same and if d is greater than d^*, then the basis becomes infeasible.

As a consequence, Elmaghraby developed the additional constraint from the objective function.

To determine how this additional constraint is to be obtained, consider the following equation:

$$Z = z_o + \sum_{j \in R} (z_j - c_j)\, x_j + \sum_{j \in R} (z_j - c_j)\, y_j$$

$$= z_o + \sum_{j \in R} V_j\, x_j + \sum_{j \in R} V_j\, y_j \qquad (23)$$

where

$$z_o = \sum_{j=1}^{n} c_j\, x_j$$

and

R = the set of indices corresponding to the non-basic variables,

N = the set of indicies of all variables.

$$= \{ 1, 2, 3, \ldots, n, n+1, \ldots n+m \}$$

It is clear that in the optimal solution, $x_j = 0$, $j \in R$. Hence $Z = z_o$. Now formulating the objective function as a constraint, we obtain

$$\sum_{j \in N} c_j\, x_j + \sum_{j \in N} c_j\, y_j + D = z_o \qquad (24)$$

where D is a slack variable with zero cost and $D \geq 0$. If the problem (22) is augmented with the equation (24) and solved by

the simplex method (the upperbound technique is implied), the corresponding equation (24) in F_o- tableau (the optimal tableau) will be

$$\sum_{j \in N} (c_j - z_j)x_j + \sum_{j \in N} (c_j - z_j)y_j + D = 0$$

or

$$- \sum_{j \in R} v_j x_j - \sum_{j \in R} v_j y_j + D = 0 \qquad (25)$$

where

$$v_j = z_j - c_j$$

at the initial iteration $D = Z_o$ and at the final iteration $D = 0$. This results from the fact that at each iteration the value of the objective function approaches z_o. In addition in the optimal solution,

$$x_j , Y_j \geq 0 ; v_j = 0 , \quad j \in B$$

and

$$x_j = y_j = 0 ; v_j = 0 , \quad j \in R$$

Also

$$\sum_{j \in B} c_j x_j + \sum_{j \in B} c_j y_j = z_o \qquad (25a)$$

and

$$\sum_{j \in R} c_j x_j + \sum_{j \in R} c_j y_j = 0$$

where

$B \cup R = N$ ("U" means union)

B = Basis.

The substitution of this solution (25a) into equation (24) results in $D = 0$. It is noted that the original constraint equation (24) has the final form of equation (25) in the optimal solution tableau F_o.

Thus from equation (25), the following constraint is obtained

$$- \sum_{j \in R} v_j x_j - \sum_{j \in R} v_j y_j + D = - d \qquad (26)$$

where d = a constant ≥ 0 to be determined later and it represents the depth of cut perpendicular to the objective function hyper plane so as to reach the next nearest extreme point. Equation (26) is referred to as the D-equation. Now the equation (25) in F_o- tableau is replaced by the D-equation, where the new tableau with the D-equation is denoted as the F-1 tableau. For $d = 0$, the F-1 tableau is equivalent to F-0 tableau and the solution is feasible and optimal to the problem (22). However for $d > 0$, the solution in the F-1 tableau is not feasible, since $D = -d < 0$. The added constraint (D-equation) reduces the convex set W and excludes the optimal non-integer

solution to the zero-one problem. This is the same concept used in the cutting plane method. In the cutting plane method, a constraint from $AX + Y = b$ is used to generate the new constraint, where as in the method of this section, the objective function is used to generate the new constraint.

As mentioned earlier, d represents the depth of cut perpendicular to the objective function plane and moves the solution point to the nearest extreme point. Hence d_1 represents the distance between the first D-equation plane and the objective function plane, d_2 the distance between the first D-equation plane and the second D-equation plane and so on. Thus d represents the decrease in the objective function value z_o (for a maximization problem) between two iterations and the sum of all these distances represent the total decrease in the objective function value (that is the decrease in objective function from the non-integer optimal solution to the integer optimal solution). From this it is evident that d should be as large as possible in order to obtain an extreme point which is closest to the present objective function hyper plane. The basis changes at the next extreme point. The first time a basis becomes infeasible; it represents that a second extreme point is obtained. This happens when $d = d^*$. Hence d must be chosen as large as possible and still maintain a feasible solution. Thus if d^* is the maximum decrease in the objective function before the solution becomes infeasible, then if $d \leq d^*$, there is no necessity to change the basis, while if $d \geq d^*$, the basic solution becomes

infeasible and hence the basis has to be changed. Now to determine d^*, let

X_{B_o} = Basis for the non-integer optimal solution.

C_{B_o} = Price vector corresponding to the basis X_B.

$z_o = C_{B_o} X_{B_o}$.

X'_{B_o} = The new basis when $d = d^* + \epsilon$, ϵ being a small positive quantity.

C'_{B_o} = Price vector for X'_{B_o}.

$z'_o = C'_{B_o} X'_{B_o}$.

x_{B_r} = Variable leaving from X_{B_o}.

x_k = Variable entering the new basis X'_{B_o}.

Assuming that x_k enters the basis, then the new value of z_o is

$$z_o = C'_{B_o} X'_{B_o}$$

$$= \sum_{i \neq r} C_{B_i} x'_{B_i} + C_k x'_k$$

$$= \sum_{i \in M} C_{B_i} x'_{B_i} + C_k x'_k - C_{B_r} x'_{B_r}$$

$$= \sum_{i \in M} C_{B_i} \left(x_{B_i} - \frac{b_r y_{ik}}{y_{rk}}\right) + C_k \frac{b_r}{y_{rk}} - C_{B_r}\left(x_{B_r} - \frac{b_r y_{rk}}{y_{rk}}\right)$$

$$= \sum_{i \in M} C_{B_i} x_{B_i} - \frac{b_r}{y_{rk}} \sum_{i \in M} C_{B_i} y_{ik} + C_k \frac{b_r}{y_{rk}}$$

$$= z_o - \frac{b_r}{y_{rk}} z_k + C_k \frac{b_r}{y_{rk}}$$

$$= z_o - \frac{b_r}{y_{rk}} (z_k - C_k)$$

$$= z_o - \frac{b_r}{y_{rk}} V_k$$

$$= z_o - d^*$$

where

$$b_r = \begin{cases} b_r & \text{if } y_{rk} > 0 \\ b_r - B_r & \text{if } y_{rk} < 0, \ B_r \text{ being the upper-bound for } x_{B_r}. \end{cases}$$

The change in the objective function $\frac{b_r}{y_{rk}} V_k$ is set equal to d^*. For the basis X_B' to remain feasible,

$$\frac{b_r}{y_{rk}} V_k = \min_i \left[\min_j \left\{ \frac{b_i V_j}{y_{ij}}, \ y_{ij} > 0 \text{ and } \frac{(b_i - B_i)V_j}{y_{ij}}, \ y_{ij} < 0 \right\} \right]$$

where

B_i = upperbound of x_{B_i}.

However d^* has to be determined over all the optimal solutions (alternate). Suppose that for some $j \in R$, $V_j = 0$. This indicates that there are alternate optimal solutions to F_o. Let S be the set of alternate extreme points for the objective function hyper plane passing through F_o, the extreme point of the convex set W, where $S = \{ P_1, P_2, P_3, \ldots, P_L \}$. The extreme points $P_k \in S$ are obviously extreme points of the original solution space W and each is represented by a unique basis[1] B_k, $k = 1, 2, 3, \ldots, L$. For each basis B_k, row $i \in B_k$, non-basic variable x_j, $-V_j < 0$ and $y_{ij} \neq 0$, determine d_k. This is done from the F-1 tableau as follows:

$$d_k = \min_i \left[\min_j \left\{ \frac{b_i V_j}{y_{ij}}, \; y_{ij} > 0 \text{ and } \frac{(b_i - B_i)V_j}{y_{ij}}, \; y_{ij} < 0 \right\} \right] > 0 \qquad (27)$$

where

$i = 1, 2, 3, \ldots, m$

$j \in R$

---

[1] Except in the case of a degenerate basis ($b_i = a_{io} = 0$) and positive entry y_{ij} that can serve as a pivot for some vector j whose $V_j = 0$.

B_i = upperbound on x_{B_i}

$y_j = B^{-1} a_j$

Thus this determines the maximum change in the b vector that is permissible and yet remain feasible. And d_k represents the corresponding change in the objective function value. Also, it is important to note that no basic solution to the original problem must be excluded by the addition of new constraint. To ensure this, all the alternate optimal solutions have to be considered in determining d^*. Hence

$$d^* = \min_k \left\{ d_k > 0 \right\}, \quad k = 1, 2, 3, \ldots, L. \qquad (28)$$

Thus having determined d^*, the algorithm proceeds in the following manner. d^* is determined for some row r and some column k with $-V_k < 0$ for some basis B_k. Now, y_{rk} is the pivot element in B_k tableau, x_{B_r} is the variable leaving the basis B_k and x_k is the variable entering the basis. The simplex algorithm is applied to obtain a new tableau. Then the new solution to the augmented problem (22) is again feasible. Note that d in the equation (26) becomes zero, because of the manner in which it was determined. If the new solution is feasible to problem (21), it is also optimal. Otherwise, another constraint of the form (26) is augmented to the F-1 tableau and the problem is resolved by the simplex method. The process is repeated until an optimal integer solution is obtained.

The algorithm is summarized as follows:

Step 1. Obtain an optimal solution to the zero-one problem without the integer restrictions (problem (22)) using the simplex method and upperbound technique. If the solution that is obtained is a feasible integer solution to (22) then it is also optimal to (21) and the algorithm ends. Otherwise proceed to step 2.

Step 2. Form the D-equation as explained earlier. Determine d^*, the pivot row r and the pivot column k. Add the constraint

$$- \sum_{j \in R} V_j x_j - \sum_{j \in R} V_j y_j + D = -d$$

to F-0 tableau, where $d = d^* = \min_k \ (d_k > 0)$.

Note. It is necessary to remember that $d^* = \min_k \{ d_k \}$, $k = 1, 2, 3, \ldots, L$ such that $d_k > 0$. This indicates that those extreme points for which $d_k = 0$ are to be excluded in determining d^*, d_1 can become zero in two ways.

a. $d_k = \dfrac{b_i V_j}{y_{ij}} = 0$, $-V_j < 0$ and $y_{ij} > 0$

Hence $d_k = 0$, if and only if $b_i = 0$.

This indicates that the solution is degenerate. Hence if $d > 0$, then the variable x_{B_i} goes negative and the solution becomes infeasible. Thus it is necessary to remove x_{B_i} from the basis and eliminate degeneracy. Exclude this solution in determining

d^* and proceed to the alternate optimal solutions.

$$\text{b.}\quad d_k = \frac{(b_i - B_i)V_j}{y_{ij}} = 0\ ,\quad -V_j < 0 \quad \text{and} \quad y_{ij} < 0.$$

This happens if and only if $b_i = B_i = 1$ (note that for slack variables the upperbound = $+\infty$). This indicates that if $d > 0$, the basic variable x_{B_i} would exceed its upperbound. Thus it is necessary to remove x_{B_i} from the basis. Since it has to become non-basic at its upperbound (in standard simplex, all non-basic variables are set equal to zero), make transformation $x_{B_i} = 1 - x'_{B_i}$ and change the sign of the price coefficient C_{B_i} of x_{B_i} (that it becomes $-C_{B_i}$ instead of C_{B_i}). Now x'_{B_i} replaces x_{B_i} with zero value. As a consequence, the solution becomes degenerate and hence eliminate degeneracy by pivoting on the variable x_{B_i}. This is illustrated in the example problem in Appendix III.

Step 3. It was noted previously that if

$$d^* = \frac{b_r V_k}{y_{rk}}\ ,$$

then x_{B_r} leaves the basis and the variable x_k enters the basis. However, this criterion (for replacing x_{B_r} by x_k in the basis) cannot be applied since the problem (22) augmented with the D-equation is not primal feasible. Consider the

augmented constraint (D-equation),

$$- \sum_{j \in R} V_j x_j - \sum_{j \in R} V_j y_j + D = -d$$

The requirements for a problem to be primal feasible is that the $b_i \geq 0$ for all i. However corresponding to the above constraint, $b_{m+1} = -d < 0$. Therefore the ordinary simplex algorithm cannot be applied to solve the new augmented problem (22). Hence it is necessary to use the dual-simplex algorithm. In this algorithm, the criterion for selecting a variable to leave the basis is to select the row corresponding to the most negative b_i. The only equation which has a negative $b_i (i = m+1)$ is the D-equation. Therefore the pivot row is the one corresponding to the D-equation. However the variable entering the basis is x_k as determined from

$$d^* = \frac{b_r V_k}{y_{rk}}$$

Hence the pivot row is the (m+1)th row (the added D-equation) and the pivot column is k. The usual simplex method is now applied to obtain the new tableau.

Step 4. The new solution is again feasible to the augmented problem. If this solution is a feasible solution to problem (21), it is then the optimal solution. If not, return to step 2 and repeat the process until an optimal zero-one solution is obtained.

Thus a different approach for the solution of a zero-one

problem is introduced in this section. An example problem is solved in Appendix III. It seems that the algorithm of this section is an efficient method for obtaining an optimal zero-one solution, provided the problem does not have many alternate optimal solutions. However this information is not available beforehand. Thus the computational time may become excessive if this is the case. This uncertainity regarding the computational time has led to the development of the combinatorial methods which are introduced in the following sections.

COMBINATORIAL METHODS

The two methods discussed earlier use the simplex method as a basis to obtain an optimal zero-one solution. Basically both methods use the simplex to obtain a non-integer optimal solution and then try to force the solution to an integer optimal solution. However the approach discussed in this section is an enumerative procedure which consists of evaluating all or a subset of the 2^n possible solutions and selecting the one which provides the best solution. The problem of the previous section is again stated as follows:

$$\begin{aligned} &\text{minimize} && z = CX \\ &\text{subject to} && \text{i, } AX + Y = b \\ & && \text{ii, } x_j = 0 \text{ or } 1,\ j \in N \\ & && \text{iii, } X, Y \geq 0. \end{aligned} \tag{29}$$

Because there are n variables and each may take the value zero or one, then are 2^n possible solutions to this problem. The above problem is labeled 'P'. There exists an (n+m) dimensional vector $U = (X, Y)$ which is called a solution, if it satisfies i, and iii, constraints; a feasible solution, if it satisfies i, ii, and iii, constraints and an optimal feasible solution if it satisfies (29). In the vector $U = (X, Y)$, X is the activity vector and Y is the slack vector. The exhaustive enumeration technique consists of enumerating all 2^n possible solutions explicitly. Truncated or partial enumeration technique consists of enumerating only those groups of solutions which are feasible

and which could lead to a better solution than those previously evaluated. The solutions to problem (29) can be represented by means of a solution tree. The branches or arcs of the tree are joined together by nodes. Hence the junction of any two branches is a node. A value of 0 or 1 is assigned to various components of the activity vector X which then forms the branches of the tree. There is a solution to the original problem associated with each node. In some methods, an auxiliary problem is substituted for the original problem at each node and it turns out that a solution to the original problem is also a solution to the auxiliary problem.

The following definitions and conventions which are used in the discussion are now given.

Chain: A chain is a path through two or more nodes.

In special situations, a chain may consist of only one node. If there is a chain passing from node h to node k, h is called a predecessor of k and k is called a successor of h.

Arc: A path connecting two nodes is called an arc.

If (h, k) is an arc of the tree, then h is an immediate predecessor of k and k is an immediate successor of h. The solution tree has the following properties:

1. The initial node of any chain is the node 0.
2. Each node except node 0 has a unique immediate predecessor.
3. A chain has the property that for any two nodes h and k in the tree, there can be only one chain

having h and k as end points.

4. If there exists a chain from h to k, this chain then includes the node 0.
5. Each arc of the tree represents the assignment of a specific value ($x_j = 0$ or 1) to some component of the activity vector X.
6. From any node k, there can be only two distinct arcs (k, h) and (k, V) where $h \neq V$.
7. If an x_j, $j \in N$ is chosen to be assigned a value on the two arcs leaving node k (denote this variable by g_k), then the variable $x_j = g_k = 0$ is assigned for one arc and $x_j = g_k = 1$ is assigned for the other arc.
8. If h and k are two distinct nodes, $h \neq k$, lying on the same chain from node 0, the variable g_k at node k must not reappear at node h, thus $g_h \neq g_k$.
9. For each arc of the chain from node 0 to node k, some x_j, $j \in N$ (not being assigned before along this chain) is assigned a value zero or one. The vector u^k is then defined to be the vector containing the set of variables from the vector X that were assigned from node 0 to node k.

Note that the variable x_j, $j = 1, 2, 3, \ldots, n$ is assigned to node k, $k = 0, 1, 2, \ldots, P$ and k need not correspond to j. Also it is noted that the chain joining nodes 0 and k may not necessarily contain all nodes V, between 0 and k, where

V = 0, 1, 2, . . . , k.

For example, consider a chain joining the nodes 0, 3, 6, 7, 15 and 20 and where there are 10 variables in the vector X. Let the $x_j = 0$ and 1, $j = 9$ be assigned to the two arcs leaving node 3 respectively. Then $g_3 = x_9 = 0$ is assigned along one arc from node 3 and $g_3 = x_9 = 1$ is assigned to the other arc. Thus the index j does not correspond to the index k. Also k may not be a continuous integer along a chain. This is evident from the above example in which k = 0, 3, 6, 7, 15 and 20 along the chain under consideration. Now it is useful to define another index r which takes continuous integers along a chain. The index r = 1 represents the first node along a chain, the index r = 2 represents the second node, and so on. Thus in the above example r = 1 corresponds to k = 0, r = 2 corresponds to k = 3, r = 3 corresponds to k = 6, r = 4 corresponds to k = 7 and so on.

If $x_2 = 0$ is assigned on the arc (0,3), $x_9 = 1$ is assigned on the arc (3,6) and $x_5 = 1$ is assigned on the arc (6,7), then

$$u^0 = \{ x_2 \} \quad , \quad \bar{u}^0 = \{ x_2 = 0 \}$$

$$u^3 = \{ x_2, x_9 \} \quad , \quad \bar{u}^3 = \{ x_2 = 0, \; x_9 = 1 \}$$

$$u^9 = \{ x_2, x_9, x_5 \} , \quad \bar{u}^9 = \{ x_2 = 0, \; x_9 = 1, \; x_5 = 1 \} .$$

Now using the index r, we define u_r^k as the variable that has been assigned a value at r^{th} node (the number of node in serial order from node 0) along the chain joining nodes 0 and k. Hence

for $k = 3$, there are 2 nodes along the chain that is $r = 1, 2$ and $u_1^3 = x_2$ and $u_2^3 = x_9$. And for $k = 15$, $r = 1, 2, 3, 4, 5$ and $u_1^{15} = x_2$, $u_2^{15} = x_9$, $u_3^{15} = x_5$ and so on. Also the corresponding assigned values are denoted by $\bar{u}_r^k$ and as a consequence $u_r^k = \bar{u}_r^k$.

Thus $u^k = \bar{u}^k$ is the solution to the problem associated with the node k. This problem is denoted as P^k and is stated as

$$\begin{aligned} &\text{minimize} && C^k X^k + C_o^k \\ &\text{subject to} && A^k X^k + Y^k = b^k \\ & && X^k = 0 \text{ or } 1 \\ & && Y^k \geq 0 \ . \end{aligned} \tag{30}$$

where

X^k is the vector obtained from the vector X by deleting the assigned variables which are contained in the vector u^k.

A^k is the submatrix of A obtained by deleting the columns associated with the variables in u^k.

The problem P^k is obtained by adjusting the constraints to account for the previously assigned variables x_j , $j \in N$. The solution procedure continues by assigning values to the remaining unassigned variables according to the adjusted problem. Let m^k be the components of C, the cost coefficient vector, associated with the assigned variables x_j which make up u^k and let M^k be

the columns of A associated with the same variables. Now

$$c_o^k = m^k \bar{u}^k$$

$$b^k = b - M^k \bar{u}^k$$

and

$$y^k = b^k - A^k X^k .$$

Consider the following definitions which pertain to the solution at the node k. Let

T_k be the set of indicies of the variables in the vector u^k.

J_k be the set of indicies of the variables from T_k, which were assigned a value 1.

H_k be the set of indicies of the variables from T_k, which were assigned a value 0.

Since the variables can only take a value 0 or 1,

$$J_k \cup H_k = T_k \tag{31}$$

Also since the variables not in the vector u^k, are free to be assigned a value either 0 or 1, they are called free variables. And hence X^k is a vector consisting of free variables. Let Q_k be the set of indicies of the variables in X^k. Therefore

$$T_k \cup Q_k = (J_k \cup H_k) \cup Q_k = N \tag{32}$$

where

$$N = \{ j, x_j ; j = 1, 2, 3, \ldots, n \}$$

The solution to the problem (30) (P^k) at node k can be represented

by $U^k = (u^k, X^k, Y^k)$

where

$$Y^k = b - M^k \bar{u}^k \geq 0$$

$$X^k = 0 \text{ or } 1 \qquad (33)$$

and

$$u^k = \bar{u}^k$$

In other words

$$x_j = \begin{cases} 1 & , j \in J_k \\ 0 & , j \in H_k \\ 0 \text{ or } 1 & , j \in Q_k \end{cases}$$

and (34)

$$Y^k = b - \sum_{j \in J_k} A_j$$

To understand the above discussion more thoroughly, let us consider the following example and Fig. 2.

Minimize

$$5x_1 + 7x_2 + 10x_3 + 3x_4$$

subject to

$$\begin{aligned} -x_1 + 3x_2 - 5x_3 - x_4 + y_1 &= -2 \\ 2x_1 - 3x_2 + 3x_3 + 2x_4 + y_2 &= 0 \\ -2x_2 - 4x_3 + 5x_4 + y_3 &= -5 \end{aligned}$$

$$x_j = 0 \text{ or } 1 \ , \ j \in N = \{1, 2, 3, 4\} \qquad (35)$$

$$Y \geq 0 \quad , \text{ that is, } y_1 = y_2 = y_3 \quad 0.$$

This is a dual-feasible problem since $c_j > 0$ for all j. Suppose the initial solution is $U^o = \{ u^o, X^o, Y^o \}$ (Since no variables are assigned a value, the set u^o is empty, that is $T_k = \emptyset$), where

$$X^o = X = 0$$

$$Y^o = b$$

$\emptyset$ = empty set.

In other words

$$U^o = \{ x_1 = x_2 = x_3 = x_4 = 0,\ Y_1^o = -2,$$

$$y_2^o = 0,\ y_3^o = -5 \}$$

$$J_o = H_o = T_o = \emptyset$$

$$Q_o = N = \{1, 2, 3, 4\}$$

In Fig. 2, beginning with node o, the succeeding nodes of the tree are numbered in an order in which they might be generated by solving the problem with some hypothetical algorithm. (The details of a well known algorithm by Balas (1) for solving the problem is to be covered in the next section of this paper.) At node 0, the problem is the original problem (35) as stated above. Proceeding from node 0 to node 1, assume the algorithm indicates that the variable x_3 is selected to be set equal to 1. Therefore at node 1,

$$J_1 = \{ 3 \}, \qquad H_1 = \emptyset, \qquad Q_1 = \{1, 2, 4\}$$

$$T_1 = J_1 \ U \ H_1 = \{ 3 \}.$$

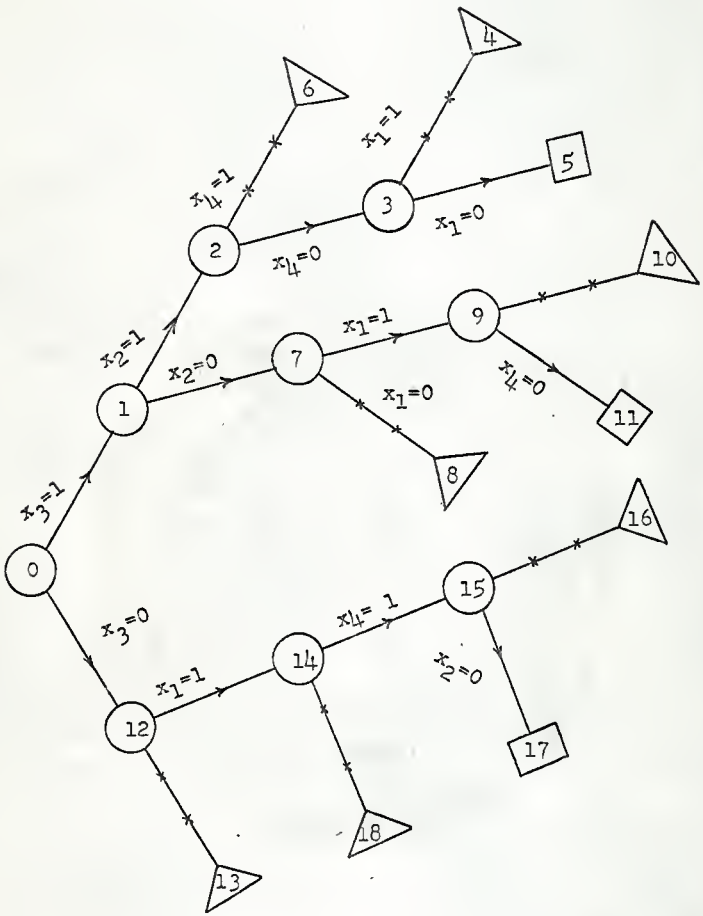

Fig. 2. Solution tree of an example problem.

Hence the solution at node 1 and consequently the solution to P^0 is

$$x_j = \begin{cases} 1 & j \in J_1 = \{3\} \\ 0 & j \in Q_1 = \{1, 2, 4\} \end{cases} .$$

$$Y^1 = b - A_3$$

$$z_1 = C_o{}^k = c_3 x_3 = 10$$

or

$$U^1 = \{u^1, X^1, Y^1\}$$

$$= \{x_3 = 1,\ x_1 = x_2 = x_4 = 0,\ y_1^1 = 3,\ y_2^1 = -3,\ y_3^1 = -1\}$$

Now the problem associated with node 1 is P^1 and is stated as:

$$\begin{aligned} &\text{minimize} && z = C^1X^1 + 10 \\ &\text{subject to} && A^1X^1 + Y^1 = b^1 \\ & && X^1 = 0 \text{ or } 1 \\ & && Y^1 \geq 0. \end{aligned}$$

where

A^1 is the submatrix of A obtained by deleting A_3 from A $[A^1 = (A_1, A_2, A_4)]$.

X^1 is the vector obtained from X by deleting x_3.

Y^1 is the slack vector corresponding to problem P^1.

$b^1 = b - M^1 \bar{u}^1$

$= b - A_3$, since $M^1 = A_3$ and $\bar{u}^1 = 1$.

Hence the example becomes

minimize

$$5x_1 + 7x_2 + 3x_4 + 10$$

subject to

$$\begin{aligned} -x_1 + 3x_2 - x_4 + y_1^1 &= 3 \\ 2x_1 - 3x_2 + 2x_4 + y_2^1 &= -3 \\ -2x_2 + 5x_4 + y_3^1 &= -1 \end{aligned} \qquad (36)$$

$$x_1, x_2, x_4 = 0 \text{ or } 1, \; Y^1 \geq 0$$

Proceeding to the next node 2, assume the variable x_2 is selected and assigned a value 1 (according to the hypothetical algorithm). Thus at node 2, $x_3 = 1$ and $x_2 = 1$.

$$J_2 = \{ 2, 3 \}, \quad H_2 = \emptyset, \quad T_2 = J_2 \cup H_2 = \{ 2, 3 \},$$

$$Q_2 = \{ 1, 4 \}.$$

The solution at node 2 is

$$x_j = \begin{cases} 1, & j \in J_2 \\ 0, & j \in Q_2 \end{cases}$$

$$Y^2 = Y^1 - A_2 = Y^0 - A_3 - A_2$$

or

$$U^2 = \{ u^2, X^2, Y^2 \}$$

$$= \{ x_3 = x_2 = 1, \; x_1 = x_4 = 0, \; y_1^2 = 0, \; y_2^2 = 0, \; y_3^2 = 1 \}.$$

The new problem at node 2 which is P^2 is

$$\begin{aligned} \text{minimize} \quad & z = 5x_1 + 3x_4 + 17 \\ \text{subject to} \quad & -x_1 - x_4 + y_1^2 = 0 \\ & 2x_1 + 2x_4 + y_2^2 = 0 \\ & 5x_4 + y_3^2 = 1 \end{aligned}$$

$$x_1, x_4 = 0 \text{ or } 1; \; Y^2 \geq 0.$$

Proceeding to node 3, by the hypothetical algorithm, assume x_4 is assigned a value $x_4 = 0$, which results in

$$J_3 = \{ 2, 3 \}, \qquad H_3 = \{ 4 \}$$

$$T_3 = J_3 \cup H_3 = \{ 2, 3, 4 \} \text{ and } Q_3 = \{ 1 \}.$$

The solution at node 3 is

$$x_j = \begin{cases} 1, & j \in J_3 \\ 0, & j \in H_3 \\ 0, & j \in Q_3 \end{cases}$$

$$Y^3 = Y^2 - A_4 x_4 = Y^2$$

or

$$\begin{aligned} U^3 &= \{ u^3, X^3, Y^3 \} \\ &= \{ x_3 = x_2 = 1, \; x_4 = 0, \; x_1 = 0, \; Y^3 = Y^2 \} \end{aligned}$$

Again the new problem at node 3 which is P^3 is

$$\text{minimize} \qquad z = 5x_1 + 17$$

$$\text{subject to} \quad \begin{aligned} -x_1 + y_1^3 &= 0 \\ 2x_1 + y_2^3 &= 0 \\ + y_3^3 &= 1 \end{aligned}$$

$$x_1 = 0 \text{ or } 1,$$

$$Y^3 \geq 0 .$$

Assume that the tests applied to the node 3 (from the hypothetical algorithm) reveal that if $x_1 = 1$, some constraint is violated and hence the solution is infeasible at this point in the tree. This is indicated by putting crossed marks on the arcs such as on arc (3,4) of Fig. 2. A triangle is used instead of a circle (which represents a node) to indicate that one cannot proceed along that path. Thus node 4 is closed and hence another node such as 5 is tested. Starting at node 3, we would proceed to node 5 by setting $x_1 = 0$. The solution at this node is

$$x_j = \begin{cases} 1 \ , & j \in J_5 = \{3,2\} \\ 0 \ , & j \in H_5 = \{1,4\} \end{cases}$$

$$Y^5 = Y^4 - A_1 x_1 = Y^4 = Y^3$$

or

$$\begin{aligned} U^5 &= \{ u^5, x^5, Y^5 \} \\ &= \{ x_3 = x_2 = 1, \ x_1 = x_4 = 0, \ Y^5 = Y^4 = Y^3 \} . \end{aligned}$$

The solution is tested for feasibility and it is feasible at node 5. This is indicated by a square. Note that the solution is feasible at node 2 itself, but the search continued in

the hope of getting a better solution until all possibilities are exhausted along that chain. It is important to note that when the appropriate tests, applied to a node, indicates a 'stop signal' such as no feasible solution along that chain, no further search is needed along that chain. It is then necessary to backtrack to a node at which further search is possible along a different chain. Hence if this occurs at node o (that is in the beginning), all possible solutions have been implicitly evaluated and the search terminates. Otherwise the search proceeds as explained earlier. At the end, all feasible solutions are compared and the one which gives the minimum value for the objective function is taken as the optimal solution to the given problem. In the example problem, the feasible solutions at nodes 5, 11 and 17 would be compared and the best one would be selected.

Trial Solutions

In general, it is not always necessary to assign values to all the x_j's in X along a chain before checking for feasibility. Some combinatorial methods [1, 8] use trial solutions to reduce the search process and speed up the computations. These trial solutions may yield a feasible or even an optimal solution to original problem 'P', before all the values for the components of X have been specified in the solution tree. The trial solutions to problem (30) are denoted by $\bar{X}^k$ at node k. Upon specifying a trial solution $\bar{X}^k$ to problem (30), a corresponding trial solution is determined for the original problem P from the

relations $X^k = \bar{X}^k$ and $u^k = \bar{u}^k$. This trial solution to P is denoted by X(k). There are four principal types of trial solutions to problem (30) P^k that are considered.

i, $X^k = \bar{X}^k = 0$ See references (1) and (8).

ii, $X^k = \bar{X}^k = \lambda$, $0 \leq \lambda \leq 1$ See reference (9)

iii, $X^k = \bar{X}^k$

where

$\bar{X}^k$ is an optimal solution to problem (30) or to a problem that results by relaxing some of the constraints of problem (30) or by adding some additional constraints to the problem (30).

iv, $X^k = \bar{X}^k$, which is the same as in ii, or iii, except the non-integer components of $\bar{X}^k$ are "rounded off" to integer values.

The above trial solutions are used in various algorithms to test for different things. A trial solution determined by i, can be used to check for feasibility to P^k which has to be true for a solution to be feasible to P. The feasibility of $\bar{X}^k = 0$ for P^k is assured by $b^k \geq 0$. The trial solution determined by iii, is useful, because it yields a tost for indication which solutions should not be considered further. Thus if tho trial solution $\bar{X}^k$ to P^k also yields a feasible solution X(k) to P, then there can be no other better solution along this chain satisfying $u^k = \bar{u}^k$ which improves the objective function ($c \geq 0$). Hence node k may be disregarded in the process of adding new arcs to the tree. Another feature of the trial solution determined by iii, is that it provides the basis for the "least cost" solution,

so that other solutions which yields a cost which is greater than the "least cost" can also be disregarded.

Open and Closed Nodes

From the above discussion, the nodes of a partially generated tree can be divided into two classes. A node k will be said to be closed if a test determines that there is no feasible solution beyond that node. For example, this occurs when the arcs (k,q) and (k,j) exhaust the two values of the variable g_k = 0 and 1. There would then be no feasible solution to P^k, since all solutions have been implicitly enumerated beyond the node k. Thus it is necessary to backtrack to node (k-1). The efficiency of any truncated or partial enumeration technique largely depends on its ability to carry out this one test that is to determine that the problem P^k has no feasible solution. A node is termed open if it is not closed. If a node is closed, there is no need to search the path beyond this node.

Redefinition of the Constraint Set

A partial enumerative algorithm can be made more efficient by adding additional constraints to the problem P^k or by relaxing some of the original constraints. The reasons for this may be as follows:

1. It may be possible to adjoin certain constraints to the problem P^k that are implied by the constraint set $A^k X^k + Y^k = b^k$ and the integer restrictions

on x_j. This would be revealed from the algorithm used to solve the problem.

For example, the algorithm might reveal that a component x_j^k of X^k cannot be 0; then x_j^k must be equal to 1 and this restriction would be added to the problem P^k.

2. It may be possible to omit some of the constraints of $A^k X^k + Y^k = b^k$, either because they are redundant or because they may not assist in obtaining a feasible solution to the problem P^k.

3. Once an arc from node k has been generated it need not be generated again at a later stage along that path. To avoid this, it is useful, for notational purposes, to conceive of those arcs as being stated in the form of constraints explicitly associated with the node k. For example, $u^k = \bar{u}^k$ would be added as a constraint set.

4. Information obtained after the generation of node k may yield other constraints applicable to that node. For example, as indicated above, by adjoining the constraint $CX < CX^*$ where X^* denotes the best feasible solution found up to this point ($C^k X^k + C_o^k < CX^*$ at node k).

General Procedure for Generating a Sequence of Solutions

Let us summarize briefly what has been discussed so far.

At node k, the partial solution to problem P can be stated as

$$x_j = \begin{cases} 1, & j \in J_k \\ 0, & j \in H_k \end{cases}$$

and

$$x_j = 0 \text{ or } 1, \quad j \in Q_k .$$

Since the R variables in the set Q_k can take either 0 or 1, there are still 2^R solutions to be implicitly or explicitly enumerated at node k. Note that R is the number of elements in the set Q_k. For example, if we take node 14 in Fig. 2, the partial solution obtained is ($x_3 = 0$, $x_1 = 1$). Therefore if we explicitly enumerate all the solutions at node 14, we get

$$(x_3 = 0, \quad x_1 = 1, \quad x_4 = 1, \quad x_2 = 1)$$

$$(x_3 = 0, \quad x_1 = 1, \quad x_4 = 1, \quad x_2 = 0)$$

$$(x_3 = 0, \quad x_1 = 1, \quad x_4 = 0, \quad x_2 = 1)$$

$$(x_3 = 0, \quad x_1 = 1, \quad x_4 = 0, \quad x_2 = 0)$$

Since there are two free variables at node (14), there are $2^2 = 4$ possible solutions at that node. Suppose a check reveals that the node (14) is closed, then there is no need to enumerate the above 4 solutions explicitly. When it is known that the node (14) is closed, all four possible solutions beyond this node have been enumerated implicitly. Hence it is necessary to backtrack to a open node. Let the set of values that the variable x_j,

$j \in Q_k$ can take be S_k where $S_k = \{0,1\}$. If an x_j (g_k) from X^k is selected to be assigned a value equal to zero at node k, then the number zero is cancelled from the set S_k, in other words, $S_k = \{1\}$. Hence whenever $S_k = \emptyset$, it is meant that all solutions have been exhausted at node k and the node k is closed. Conversely whenever the node k is closed, $S_k = \emptyset$. If a feasible solution is obtained for the original dual-feasible minimization problem, the succeeding nodes on that path are closed, that is $S_{k1} = \emptyset$ where $k^1 > k$ along that path. This states that the first feasible solution is the "least cost" solution on that path. Consequently there is no need to search further.

Now the general procedure of the enumerative methods are summarized as follows:

1. The starting node 0 is specified and the tree is constructed according to the algorithm.
2. If there are no free variables that can be selected to enter the solution, the process terminates. Otherwise go to step 3.
3. If all nodes in the tree are closed, the process terminates. (If the node 0 is closed, the solution is trivial and unique and it is $U^o = (u^o, X^o, Y^o)$). Otherwise go to step 4.

4a. Select an open node k.

b. Select a variable from X^k for which a value is to be assigned.

5. Now form a new arc leaving the node k by assigning

a value to x_j (g_k) from S_k. Test the trial solution for feasibility.

6. If node (k+1) is not closed, and Q_{k+1} is not empty, return to step 4. Otherwise go to step 7.

7. If $S_k = \emptyset$, backtrack to node (k-1) and return to step 3. Otherwise go to step 4b.

The above procedure is shown in a flow chart on the next page. Thus the underlying approach to combinatorial methods has been introduced in this section. An example of a specific combinatorial method is discussed in the next section.

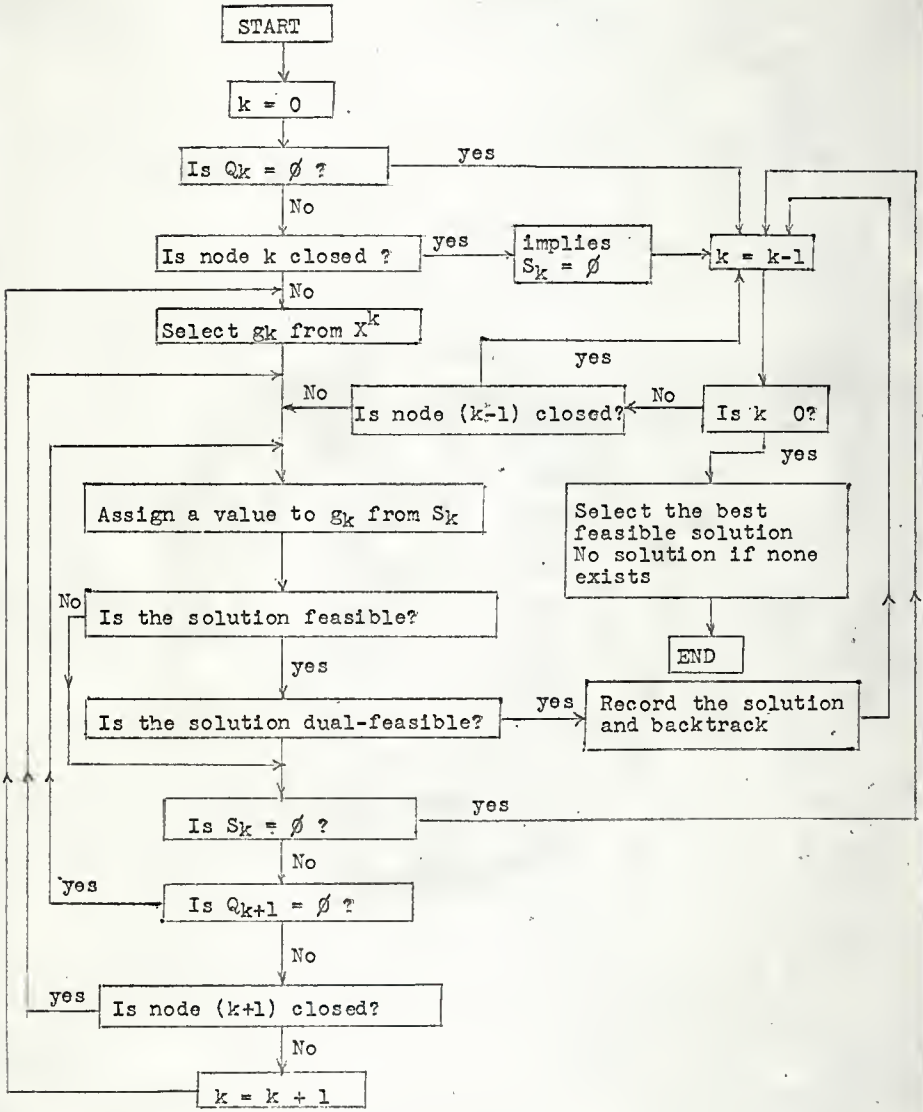

Fig. 3. A Flow Chart for the general combinatorial procedure.

ADDITIVE ALGORITHM OF BALAS (WITH SOME MODIFICATIONS)

In the previous section, the general features of combinatorial algorithms were discussed. In the discussion, it was assumed that some hypothetical algorithm dictated the steps of the solution procedure. In this section one of the combinatorial algorithms due to Balas [1] is presented for solving the zero-one problem. Another algorithm due to Glover [8] is very similar, but will not be discussed in this paper.

Balas' algorithm is applicable to problems which can be formulated as problem "P" which is dual-feasible ($C \geq 0$). Balas uses the trial solution $\overline{X}^k = 0$ that is $\{ x_j = 0,\ j \in Q_k \}$ for problem P^k at node k and if this results in a feasible solution to P^k, then a feasible solution has been found to the original problem P. This then closes the node k. This feasible solution is an optimal solution to P^k and a local optimal solution to P.

To determine if the solution to P^k is feasible, the algorithm proceeds in the following manner. The current solution at node k, $u^k = \overline{u}^k$ is substituted into the original problem to obtain

$$A^k X^k + M^k \overline{u}^k + Y^k = b$$

or

$$A^k X^k + Y^k = b - M^k \overline{u}^k = b^k . \qquad (37)$$

Now this trial solution to (37) $X^k = \overline{X}^k = 0$ results in either

a. an optimal feasible solution to P^k, if $Y^k = b^k \geq 0$.

b. an infeasible solution to P^k if any $y_i^k = b_i^k < 0$.

Note that if the optimal feasible solution to P^k is substituted into P, it is feasible, but may not be optimal.

As was pointed out above, if the solution corresponds to a., then node k is closed. On the other hand if the solution results in b., that is if some $b_i^k < 0$, then the coefficients in those equations are checked to determine if

$$\sum_j \overline{a_{ij}}^k > b_i^k \qquad (38)$$

where $\sum \overline{a_{ij}}^k$ are the sum of all negative a_{ij}^k in the ith row. If (38) holds, then the problem P^k does not have a feasible solution, because the b_i^k can only become positive if the sum of these negative coefficients can offset the degree of negativity in b_i^k. That if $\sum_j \overline{a_{ij}}^k > b_i^k$, it is impossible to obtain a feasible solution, thus there is no feasible solution to P^k and node k is closed. This points out that node k is open if it is at all possible to obtain a feasible solution along this path.

From the above discussion, a necessary condition for a node k to be open is that the relationship $\sum_j \overline{a}_j^k \leq b^k$ must hold.

The search process continues until a feasible solution or another stop signal is encountered. The third signal indicating that a node k is closed, arises from the situation in which the following relationship holds:

$$c_j^k + c_o^k > CX^* , \quad j \in Q_k . \tag{39}$$

That is, if CX^* is the value of the objective function for the best feasible solution obtained so far, and if the value z at node (k+1) exceeds CX^*, then node k is closed, because any other solution to P exceeds CX^* on this particular path by virtue of the problem formulation.

The fourth and the final stop signal for closing node k arises from the situation where

$$b^k < 0 \text{ and } a_j^k \geq 0 \text{ for all } j \in Q_k \tag{40}$$

This indicates that it is impossible to make $b^k \geq 0$, since all $a_j^k \geq 0$, thus there is no reason to continue on this path. In summary then the stop signals for closing node k in Balas algorithm arise when:

1. A trial solution $\bar{X}^k = 0$ yields a feasible solution to P^k.
2. The trial solution $\bar{X}^k = 0$ does not yield a feasible solution but the relation $\sum_j \bar{a}_{ij}^k > b_i^k$ holds for some row with a negative b_i^k.
3. The relation $c_j^k + c_o^k > CX^*$ holds.
4. A situation arises where $b^k < 0$ and $a_j^k \geq 0$ for all $j \in Q_k$.

A possible modification to Balas' algorithm as suggested by Glover [9] is now discussed. This modification reduces the number of combinations to be searched and accelerates the solu-

tion process. In the previous section, it was mentioned that the addition of some additional constraints might be desirable when certain information is implied in the constraint set $A^k X^k + Y^k = b^k$. To understand this more clearly, consider the equation $2x_1 - 3x_2 + 2x_4 \leq -3$. It is impossible to have $x_2 = 0$ and still obtain a feasible solution. Hence x_2 must take a value 1. Now consider a second inequality of the form

$$-3x_1 - 2x_2 - 4x_3 + 5x_4 \leq -8 \qquad (41)$$

where all variables except x_4 must be equal to 1 to have a feasible solution. Note that the sum of the negative coefficients do not exceed the right hand side (relation (38) does not hold), that is $\sum_j \bar{a}_j^k < b^k$ where $\bar{a}_j^k$ are the negative coefficients. Thus an additional constraint $x_2 = 1$ is added to the first inequality and the constraint $x_1 = x_2 = x_3 = 1$ is added to the second inequality. It is evident that this addition accelerates the solution process. Now a couple of rules which exploit the above information (if exists) will be discussed. Note that they are not the stop signals, but yield some information (which can be expressed in the form of additional constraints) that is implied in the constraint set of the problem P^k and speed up the computation.

If a situation occurs where a node k is not closed by (38) but would be closed by the following relation

$$\sum_{j} \bar{a}_j^k - \bar{a}_q^k > b^k \qquad (42)$$

where

$$\bar{a}_q^k = \max_j (\bar{a}_j^k) \quad \text{and} \quad b^k < 0 .$$

Relation (42) then implies that $x_j^k = 1$ for all j corresponding to $a_j^k < 0$. Otherwise there would be no feasible solution to P^k. In the above inequality (41), $\sum \bar{a}_j^k - \bar{a}_q^k = -9 - (-2) = -7 > -8$. Therefore $x_1 = x_2 = x_3 = 1$. There is also a corresponding situation where all x_j^k would equal to zero, that is $x_j^k = 0$ for all j for which $a_j^k > 0$. This would occur when relation (38) is not satisfied, but the following relation holds

$$\sum_{j} \bar{a}_j^k + a_s^k > b^k \qquad (43)$$

where

$$a_s^k = \min_j \left\{ a_j^k \geq 0 \right\} \quad \text{and} \quad b^k < 0 .$$

To observe this consider the inequality

$$2x_1 - 2x_2 - 4x_3 + 5x_4 \leq -5 \qquad (44)$$

$$\sum \bar{a}_j^k + a_s^k = (-2 - 4) + 2 = -4 > -5 .$$

Hence $x_j^k = 0$, for j where $a_j^k > 0$, thus $x_1 = x_4 = 0$ as implied in the inequality (43). Note that the relations (42) and (43) are suggested by Glover /9/. Thus this implied information is included in u^k which is used as a constraint set $u^k = \bar{u}^k$

at the succeeding nodes.

Balas algorithm for solving a zero-one integer linear programming problem utilizes the 4 stop signals and the modifications discussed above. The algorithm follows the procedure of the general combinatorial approach discussed in the previous section. A brief summary of the algorithm is given to give the reader a better understanding of the procedure. A more thorough treatment of the details is presented later along with the steps of the algorithm.

Balas Algorithm

Balas algorithm starts with a dual-feasible ($C \geq 0$) linear programming problem P^o with the initial solution $U^o = (u^o, X^o, Y^o) = (0, 0, b)$. This solution corresponds to node 0. Throughout the algorithm, the activity vector X remains non-basic and the slack vector Y remains basic. Initially, $X^o = 0$ and $Y^o = b$. Since a trial solution to P^k is feasible when $Y^k = b^k \geq 0$, the goal is to obtain a non-negative basis vector. This is the criterion used in the algorithm to form a new arc at node k. The set of variables from N which reduce the negativity of b^k (that is force the b_i^k's to be positive) form a subset N_s, called the set of improving vectors. The variable which reduces the negativity condition of the basis vector b^k the greatest amount is selected to enter the solution vector u^k.

The variables from N_s are then introduced one at a time into the solution in an attempt to force b^k to be non-negative

and obtain a feasible solution to P^k. The method for doing this is explained as follows: calculate the vector b^{k+1} by introducing each x_j^k, $j \in N_s$ into the solution to see which variable should be selected for assigning a value 1 at node k. Thus for each j in N_s, there is a corresponding vector b^{k+1}, which is calculated from the relation,

$$b^{k+1} = b^k - a_j^k \tag{45}$$

Now define the set

$$V_j^k = \sum_{i \in M} \bar{b}_i^{k+1}, \quad j \in N_s \tag{46}$$

where $\sum \bar{b}_i^{k+1}$ is the sum of all negative b_i^{k+1}. Compute V_j^k for each $j \in N_s$ for which b^{k+1} is also calculated. Compare all V_j^k, $j \in N_s$ and select the variable x_j^k associated with the $\max V_j^k$, $j \in N_s$ and set equal to $x_{j_{k+1}}^k$, that is

$$x_{j_{k+1}}^k = \left\{ x_j^k : \max V_j^k, \quad j \in N_s \right\} \tag{47}$$

is selected to enter the solution with a value equal to 1. Intuitively this means that the value V_j^k is computed as though each variable x_j, $j \in N_s$ is introduced into the solution to reach node (k+1) and all the resulting solutions (there are as many solutions as there are number of j's in N_s) are compared

to select the one which has the least negativity which reflects in V_j^k. This criteria for selection of variables with the solution seems to reduce the negativity condition of the vector b^k at the fastest rate.

The above discussion is summarized as follows: If for some i, $y_i < 0$, the variable $x_{j_{k+1}}^k$ is selected from X^k by relation (47) and introduced into the solution. This point is illustrated as follows: If $k = 0$,

$$x_{j_1}^0 = \left\{ x_j^0 : \max_j V_j^0 , \; j \in N_s \right\}$$

where

$$J_1 = \left\{ j_1 \right\}, \quad H_1 = \emptyset , \quad T_1 = \left\{ j_1 \right\} \text{ and } Q_1 = N - \left\{ j_1 \right\} .$$

The solution is then

$$U^1 = (u^1, X^1, Y^1),$$

where

$$u^1 = (x_{j_1}) , \quad X^1 = 0$$

and

$$Y^1 = b^0 - A_{j_1}^0 = b^1.$$

The problem P^1 is

$$\text{minimize} \quad C^1 X^1 + c_{j_1}$$

$$\text{subject to} \quad A^1 X^1 + Y^1 = b^1$$

$$x_j^1 = 0 \text{ or } 1 \qquad j \in Q_1$$

$$x_j = 1 \qquad j = j_1$$

$$Y^1 \geq 0 .$$

If the solution vector U^1 still contains some negative components, the algorithm continues and problem P^1 is solved. Another variable $x^1_{j_2}$ is selected from X^1, as determined by (45), (46) and (47) and the solution to problem P^1 is

$$U^2 = (u^2, X^2, Y^2) = (x_{j_1} = x_{j_2} = 1, X^2 = 0, Y^2 = Y^1 - A^1_{j_2}).$$

If this solution U^2 is not feasible, then the problem P^2 is formulated as follows:

$$\text{minimize} \quad C^2 X^2 + c_{j_1} + c_{j_2}$$

$$\text{subject to} \quad A^2 X^2 + Y^2 = b^2 = Y^1 - A^1_{j_2}$$

$$x_j = 1, \; j \in J_2, \; J_2 = \{ j_1 , j_2 \}$$

$$x_j = 0 \text{ or } 1 , \quad j \in Q_2$$

$$Y^2 \geq 0 .$$

This procedure is repeated until the solution U^s is feasible or evidence is obtained that such a solution to P^s does not exist. If in the process, a non-negative solution $U^s = (u^s, X^s, Y^s)$ is obtained, it is then an optimal solution to problem P^s. It is noted that this is a feasible solution but it may

not be the optimal solution to the original problem P. A non-negative solution U^s closes the node s and the search backtracks to another open node P ($P \prec s$) and the search continues along some other chain. The new value of the objective function z_s is checked to determine if $z_s \prec z^*$ (z^* corresponds to the best feasible solution obtained up to this point) and if so, then z_s is set equal to z^*, $z^* = z_s$. The search continues for a better value of z^* until all nodes are investigated. During the algorithm, only the vector b is changed at each node and the coefficient matrix A remains unchanged. This preceeding discussion briefly summarizes the algorithm of Balas. Before proceeding to a more detailed discussion, it is necessary to define a number of quantities. This is done in the following section.

Notations and Definitions

The following notations and definitions are used in the discussion of Balas algorithm. A solution sequence is represented by $U^o = (u^o, X^o, Y^o)$, $U^1 = (u^1, X^1, Y^1)$, . . . , $U^p = (u^p, X^p, Y^p)$, . . . , $U^s = (u^s, X^s, Y^s)$, . . . ; where u, X, and Y with superscripts represent the solution vectors of a particular solution in the sequence. U^o is the solution at node 0, U^1 is the solution at node 1 and so on. The nodes, as they are generated in the algorithm, are numbered in ascending order starting from 0. As mentioned earlier, there is a solution associated with each node. All these solutions, in order, form a solution-sequence. Thus X^p represents a vector which contains

all free variables (those which have not been assigned a value of zero or one) at node p, u^p is the vector consisting of the variable which were previously assigned a value of zero or one and which form the chain connecting the nodes 0 and p. The sequence of solutions which are generated by Balas algorithm are denoted according to the above notation. Thus the s-th term of this sequence is denoted as $U^s = U(j_1, j_2, j_3, \ldots, j_r) = (u^s, X^s, Y^s)$, where $(j_1, j_2, \ldots, j_r)$ forms the set J_s, that is, $J_s = \{ j \mid j \in N, x_j = 1 \}$.

The variables which were previously assigned a value zero are contained in the set H_s where H_s is defined by

$$H_s = \{ j \mid j \in N, x_j = 0 \} .$$

In the solution U^s, u^s is the vector which consists of the set of variables whose indicies are in T_s where T_s is defined by

$$T_s = J_s \cup H_s$$

and where

$$X^s = 0 ,$$

$$Y^s = b - \sum_{j \in J_s} A_j .$$

The value of the objective function for the solution U^s is denoted by z_s. Let Z_s be the set of z for <u>all feasible</u> solutions obtained prior to and including the solution at node s, that is,

$$Z_s = \{ z_p \mid p \leq s, \; U^p \geq 0 \} \tag{48}$$

Note that $U^p \geq 0$ indicates that the solution is feasible at node p. If the set Z_s is not empty, then the solution associated with the smallest element is termed the "best" solution and has a value equal to Z^*. On the other hand if Z_s is empty, then there is no feasible solution to P in this sequence of solutions and $Z^* = \infty$. Hence

$$Z^* = \begin{cases} \infty & \text{if } Z_s = \emptyset \\ \min \; z_p \mid p \prec s & \text{if } Z \neq \emptyset . \end{cases} \tag{49}$$

denotes the "best" value of the objective function up to this point.

Now it is of interest to determine the set N_s which represent the set of vectors which can improve the solution. At each node k, a set of values of V_j^k for each $j \mid j \in N_k$ is calculated for the solution U^k, where N_k is the set of improving vectors at node k. These V_j^k will be used for determining which variable from the set N_k is to be introduced into the solution. When a variable is introduced into the solution, the corresponding V_j^k is cancelled from the sot N_k. Also more V_j^k's may be cancelled at subsequent nodes as more information is gained. Thus the set N_k at node k may get reduced in later iterations of the algorithm. In essence the set of unassigned variables at node k which were thought to be helpful to improve

the solution U^k, is subject to be reduced at nodes beyond k. At subsequent nodes it becomes clear that some of the variables from N_k are not useful anymore to improve the solution U^k at node k and consequently they will be cancelled from N_k. This is particularly helpful when backtracking to this node k and searching along other branches of the tree. The variables that were eliminated from node k need not be considered later on other branches of the tree, which include node k. The rules which permit this cancellation will be explained later when the procedure of the algorithm is summarized.

A new set C_k^s $(k < s)$ is introduced which includes the set of j's that correspond to those V_j^k's that are cancelled from N_k starting at node (k+1) till reaching node s. That is the variables that are cancelled from N_k at node k from the information gained at nodes k+1 through s comprise the set C_k^s. The determination of the C_k^s is illustrated in the following example. Suppose that $N_2 = \{2, 3, 5, 6\}$ and that V_j^2 are calculated for j = 2, 3, 5, and 6. For the solution U^3, j_3 is selected such that $V_{j3}^2 = \max V_j^2$. Thus at iteration 3, j_3 is cancelled from N_2 and included in the set C_k^s. Hence $C_2^3 = \{j_3\}$. Similarly additional elements from N_2 may be cancelled in iteration 4 and so on up to s. Thus all the cancelled elements from iteration 3 to s are included in the set C_2^s. For example, assume that $j_3 = 3$ and elements 5 and 6 have been cancelled from N_2 prior to reaching the 9th iteration (node); therefore $C_2^9 = \{3, 5, 6\}$. From the above discussion $C_k^k = \emptyset$, because elements may only be cancelled at subsequent

nodes (iterations). Thus by definition $C_k^k = \emptyset$, an empty set.

In similar manner, a new set C^s is introduced which contains the set of those j's from all of the sets N_p, where $p \prec s$ and $J_p \subset J_s$ (J_p is the set of j's corresponding to $x_j = 1$ at node p) have been cancelled prior to obtaining the solution U^3, that is

$$C^s = U_p \; J_p \subset J_s \; C_p^s \tag{50}$$

Stated another way, C^s is the union of all C_p^s for p = 0, 1, 2, . . . , (s-1) such that J_p is strictly contained in J_s. For example consider node 9 in Fig. 1. It is necessary to determine the values of p for which $J_p \subset J$. Since

$$s = 9,\ J_o \subset J_9,\ J_1 \subset J_9 ,$$

$$J_2,\ J_3,\ J_4,\ J_5,\ J_6,\ J_8 \not\subset J_9, \text{ and } J_7 \subset J_9$$

Hence for p = 0, 1, and 7, $J_p \subset J_s$. Therefore

$$C^9 = C_o^9 \; U \; C_1^9 \; U \; C_7^9 \qquad (p \prec s).$$

This is intuitively clear, since at any node s, in determining C^s, we are concerned only with the nodes which lie on the chain joining nodes 0 and s, that is the path being investigated for a feasible solution. Thus to obtain C^s at node s, it is necessary to obtain a set which is the union of the sets C_p^s at all previous nodes along the chain joining nodes 0 and s. In reference to Fig. 2, it is clear that the nodes 2, 3, 4, 5, 6 and

8 are not predecessors (preceeding nodes) of node 9, where as nodes 0, 1 and 7 are predecessors.

Determination of Set of Improving Vectors N_s

Now the method for determining the set of improving vectors N_s for improving the solution U^s will be discussed. This is the set of variables which may possibly yield a better feasible solution to problem P^s if introduced into the solution. A new set R^s is defined as the complement of N_s where $R^s + N_s = N$. This set of variables from R^s cannot improve the solution U^s at node s. Thus $N - R^s = N_s$ the set of improving vectors for the solution U^{s+1}.

As mentioned earlier, C^s corresponds to the set of vectors which will not improve the solution. It consists of the j's for the x_j which are in the solution and as well as those which are subsequently determined undesirable for the solution and whose V_j^k $(k < s)$ are cancelled. It is obvious then that C^s is included in R^s.

Another set of variables which will not improve the solution are those which may not improve the objective function. Relation (39) can be utilized to obtain this set, denoted as D_s, which correspond to the $x_j = 1$, $j \in (N - C^s)$ for which the objective function will exceed the Z^* value. Thus if z_s is the value of the objective function for the solution U^s, then

$$D_s = \left\{ j \mid j \in (N - C^s),\ z_{s+1} = z_s + c_j \geq Z^* \right\} \qquad (51)$$

Similarly another set of variables exist which cannot lead to a feasible solution. This set denoted by E_s, corresponds to relation (40) and consists of those variables $x_j = 1$, $j \in [N - (C^s \cup D_s)]$ which would not force the negative $y_i^s < 0$ towards a positive value that is $y_i^{s+1} \not> y_i^s$. Thus

$$E_s = \{ j \mid j \in N - (C^s \cup D_s),\ y_i^s < 0 \text{ and } A_j \geq 0 \}. \quad (52)$$

Another way of considering E_s is to note that the vector b^s must be non-negative in order for a solution to U^{s+1} to be feasible. But as stated above, the variables in the vector E_s tends to increase the negativity, since $A_j \geq 0$. Thus the variables in E_s do not lead to a feasible solution and are placed in the same category as the variables in C^s and D_s. Now the set R^s corresponding to the variables which will not improve the solution, is defined as

$$R^s = C^s \cup D_s \cup E_s \quad (53)$$

and as a consequence the set of variables in N which may improve the solution are defined by

$$N_s = N - R^s = N - (C^s \cup D_s \cup E_s). \quad (54)$$

It is noted that each node k has a set of improving vectors N_k according to (54). This set may be reduced as more information is obtained at succeeding nodes. This N_k is continuously

being updated during the solution process.

The sets as stated in relation (54) are determined as the algorithm proceeds forward from one node to the next. If on the other hand a stop signal is encountered at some node s, it is necessary to backtrack to an open node k ($k \prec s$).

There may exist a set D_k^s corresponding to the set of x_j, $j \in (N_s - C_k^s)$ such that if x_j were introduced into the solution at their upperbounds (that is $x_j = 1$), the solution U^{s+1} from node k is not as good as the best solution obtained up to this point, that is Z_{s+1} would exceed Z^* where $J_{s+1} = J_k \cup \{ j \}$. Thus it is possible to eliminate the set

$$D_k^s = \left\{ j \mid j \in (N - C_k^s),\ z_k + c_j = z_{j+1} \geqslant Z^* \right\} \qquad (55)$$

in the search for a better solution.

Thus as the algorithm progresses, it is necessary to consider the set N_k (54), the set of improving vectors for improving the solution U^k at node k. However when backtracking to node k from nodes at a later stage, the information gained from nodes k through s indicate that some of the variables in N_k do not actually improve the solution U^k at node k. As discussed earlier, this is evident from the sets C_k^s and D_k^s which comprise of the variables that were found to be not useful in improving the solution U^k at node k. Thus it is necessary to define another set N_k^s, the new set of improving vectors at node k ($k \prec s$) after backtracking from node s. Only the set N_k^s is

considered in obtaining the solution U^{s+1} at node k and this set is determined by

$$N_k^s = N_k - (C_k^s \cup D_k^s), \quad k \prec s \quad \text{and} \quad J_k \subset J_s . \qquad (56)$$

The sets N_s and N_k^s play a central role in this algorithm. Whenever a solution U^s is reached, only the improving vectors for that solution are considered for introducing into the solution. Whenever the set N_s is void at the node s, this is intrepreted as a 'stop signal', which means that no feasible solution U^t exists such that $J_s \subset J_t$ and $z_t \prec Z^{\#}$, thus the node t is closed. In this situation, the algorithm backtracks to an open node k on the same chain, where the set of improving vectors N_k^s is to be considered for improving the solution U^s.

The four stop signals previously discussed are incorporated in the method for determining N_s and N_k^s. Encountering a stop signal results in either N_s or N_k^s being empty ($= \emptyset$). This is explained below in the steps of the algorithm which will now be presented. Finally an example problem is solved illustrating the technique in Appendix IV.

The Algorithm

Before proceeding with the algorithm, a review of some of the sets used will now be presented. Suppose that at the s-th iteration, the solution is

$$U^s = U(j_1, j_2, \ldots, j_r) = (u^s, X^s, Y^s)$$

and

J_s = a set of j for which $x_j = 1$ at node s.

H_s = a set of j for which $x_j = 0$ at node s.

T_s = a set of j for which x_j were assigned a value from node 0 to node s along the chain connecting these two nodes.

$= J_s \cup H_s$

Q_s = a set of j for which x_j are free to be assigned a value at node s.

$= N - T_s$

N_s = the set of j for which x_j may improve the solution U^s.

C_k^s = the set of j corresponding to V_j^k that were cancelled from node k to node s.

$$V_j^k = \sum_{i \in M} \bar{b}_i^{k+1} = \sum_{i \in M} b_i^k - a_{ij}^k, \quad j \in N_k$$

$$C^s = \cup_p \mid J_p \subset J_s \quad C_p^s$$

= the set of j for which all $V_j^p (p \mid J_p \subset J_s)$ are cancelled along the chain joining nodes 0 and s before reaching node s.

D_s = the set of j for which if x_j were introduced into the solution will not improve z.

E_s = the set of j for those x_j that, if introduced into the solution, will not lead to a feasible

solution.

D_k^s = the set similar to D_s, but considered at node k in backtracking.

N_k^s = the set similar to N_s, but considered to be desirable to improve the solution in backtracking to node k.

M = the set of constraints = $\{1, 2, 3, \ldots, m\}$.

N = the set of j corresponding to x_j in X = $\{1, 2, \ldots, n\}$.

$$x_j^s = \begin{cases} 1, & j \in J_s \\ 0, & j \in H_s \\ 0, & j \in (N - T_s) \end{cases}$$

$$Y^s = b - \sum_{j \in J_s} A_j$$

and

$$z_s = \sum_{j \in J_s} c_j .$$

Now the procedure at s = 0 starts with the initial solution U^o, in which $X^o = 0$, $Y^o = b$ and $z_o = 0$. The procedure then continues as follows:

Step 1. Check $y_i^s \geq 0$, $i \in M$.

1a. If $y_i^s \geq 0$, $i \in M$, set $z_s = Z^*$. If this happens for U^o, then U^o is the unique solution and the algorithm ends. Otherwise, this indicates that a feasible solution for P^s exists

and $N_s = \emptyset$. This is true since the solution is dual-feasible ($c_j \geq 0$ for all j). Now backtrack to a preceeding node k and form the sets D_k^s for all $k \prec s$, starting with the immediate predecessor. Cancel all V_j^k for $j \in D_k^s$ where

$$D_k^s = \left\{ j \mid j \in (N - (C_k^s), z_s + c_j \geq Z^* \right\}$$

Since the set D_k^s is not useful in improving the solution it is necessary to cancel D_k^s from N_k. Now pass to step 5.

1b. If there exists an i, such that $y_i^s \prec 0$, pass to step 2.

Step 2. Obtain the set of improving vectors N_s for the node s, from

$$N_s = N - (C^s \cup D_s \cup F_s)$$

where

$$C^s = \cup_p \mid J_p \subset J_s \quad C_p^s$$

$$D_s = \left\{ j \mid j \in (N - C^s), \quad z_s + c_j \geq Z^* \right\}$$

$$E_s = \left\{ j \mid j \in [N - (C^s \cup D_s)], \quad y_i^s \prec 0 \quad \text{and} \right.$$

$$\left. A_j \geq 0 \right\}.$$

2a. If $N_s = \emptyset$, there are no improving vectors for U^s, or the node s is closed. Hence pass to step 5.

2b. If $N_s = \emptyset$, pass to step 3.

Step 3. Check the relations

$$\sum_{j \in N_s} \bar{a}_{ij} \leq y_i^s \quad (i \mid y_i^s < 0) \tag{57}$$

where $\bar{a}_{ij}$ are the negative elements of A.

3a. If there exists an i for which the above relation (57) does not hold, pass to step 5. This means the node s is closed. The above relation is derived from relation (38).

3b. If all relations in (57) hold, then check the relation

$$\sum_{j \in N_s} \bar{a}_{ij} + \min (a_{ij} \geq 0) \leq y_i^s , \ (i \mid y_i < 0) \tag{58}$$

3b. i, If all the relations in (58) hold, pass to step 3c.

ii, If one or more of the relations in (58) do not hold, then let M_1 be the subset of M for which (58) does not hold and cancel all those j, $j \in N_s$ where $a_{ij} \geq 0$ and $i \in M_1$. Add this set M_1 to H_s and C_s^{s+1}. This states that $x_j^s = 0$, for $j \in N_s$ where $a_{ij} \geq 0$ and $i \in M_1$. Now pass to step 3c.

3c. Check the relation

$$\sum_{j \in N_s} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i^s \ , \ (i \mid y_i < 0) \qquad (59)$$

3c. i, If all relations in (56) hold, compute the values V_j^s for all $j \in N_s$, and choose j_{s+1} so that

$$V_{j_{s+1}}^s = \max_{j \in N_s} \left\{ V_j^s \right\}$$

where

$$V_j^s = \sum_{i \in M} \bar{b}_i^{k+1}$$

cancel $V_{j_{s+1}}^s$ from N_s and pass to step 8.

ii, If all relations in (57) hold, and there exists a subset M^s of M such that the relations in (59) do not hold for $i \in M^s$, pass to step 4.

Step 4. In this step it is necessary to force all the variables which have negative coefficients in the subset M^s into the solution at the same time. This is clear from the relations

$$y_i^{s+1} = y_i^s - \sum_{j \in N_s} \bar{a}_{ij} \geq 0 \ , \ i \in M^s \qquad (60)$$

and

$$y_i^{s+1} = y_i^s - \sum_{j \in N_s} \bar{a}_{ij} - \max (a_{ij} < 0) \leq 0, \ i \in M^s \qquad (61)$$

The trial solution is feasible if all x_j for which $a_{ij} < 0$ are introduced into the solution simultaneously. This is evident from relation (60). On the other hand if all the x_j are not introduced for which $a_{ij} < 0$, the trial solution is not feasible. This is evident from relation (61).

Before introducing the above mentioned variables into the solution, it is necessary to check if $z_{s+1} > Z^*$. Hence the relation

$$z_s + \sum_{j \in F_s} c_j < Z^* \qquad (62)$$

is checked, where $F_s = \left\{ j \mid j \in N_s , \; a_{ij} < 0 \right\}$ for at least one $i \in M^s$. To obtain a feasible solution along the current branch, it is necessary to set $x_j = 1, \; j \in F_s$.

4a. If (62) holds, cancel the V_j^s for all $j \in F_s$ from N_s. When (62) holds, z_{s+1} is less than the least cost obtained thus far, therefore z_{s+1} will become the 'least cost' Z^*. Hence set

$$J_{s+1} = J_s \cup F_s$$

compute

$$z_{s+1} = z_s + \sum_{j \in F_s} c_j$$

and the slack variables

$$y_i^{s+1} = y_i^s - \sum_{j \in F_s} a_{ij} , \quad i \in M$$

for the solution

$$U^{s+1} = (u^{s+1}, X^{s+1}, Y^{s+1}).$$

The value of x_j in the vector u^{s+1} will be either 1 or 0, depending whether or not it is in J_s or H_s respectively. In other words

$$x_j = \begin{cases} 1 , & j \in J_s \\ 0 , & j \in H_s \end{cases}$$

$$X^{s+1} = 0 , \quad Y^{s+1} = b^{s+1} .$$

Now pass to the next iteration by starting again at step 1.

4b. If (62) does not hold, it means this solution increases the value of the objective function beyond the least cost, therefore cancel all the V_j^s for $j \in N_s$. This situation indicates that the node s is closed. Now pass to step 5.

Step 5. This step arises from situations 1a, 2a, 3a and 4b. Since the node s is closed in all these cases, $N_s = \emptyset$. Hence it is necessary to backtrack to a predecessor k (preceeding node k) where $k \mid J_k \subset J_s$ and search for a better solution. By proceeding from 1a to this step, it means a feasible solution exists and it is necessary to search for a better solution. By proceeding from 2a, 3a, or 4b, to this step, it indi-

cates that there is no feasible solution along the current path. At this point all possible solutions have been enumerated implicitly by this fact. Hence it is necessary to search elsewhere for a feasible solution.

Now to determine which node k to backtrack to, compute the set of improving vectors $N_k^s = N_k - (C_k^s \cup D_k^s)$ at each preceeding node k where $k \prec s$ and $J_k \subset J_s$ in the decreasing order of k until either of the following occurs:

1. a number k_1 is found such that $J_{k_1} \subset J_s$ and $N_{k_1}^s \neq \emptyset$ or

2. $N_k^s = \emptyset$ for all k such that $J_k \subset J_s$.

5a. If $N_k^s = \emptyset$ for all k where $J_k \subset J_s$, it indicates that all nodes are closed and the algorithm has come to an end. In this case, if $Z_s = \emptyset$, P has no feasible solution. On the other hand if $Z_s = \emptyset$, then the least cost Z^* is the optimum value of the objective function.

5b. If $N_k^s = \emptyset$ for a particular node $k = k_1$, $J_{k_1} \subset J_s$, pass to step 6.

Step 6. Check the relations

$$\sum_{j \in N_k^s} \bar{a}_{ij} \leq y_i^k, \quad (i \mid y_i^k \leq 0) \tag{63}$$

for $k = k_1$. Here again, it is necessary to check whether node

k_1 is closed.

6a. If any of the relations in (63) do hold for $k = k_1$, it is clear that the node k_1 is closed. Hence cancel $V_j^{k_1}$ for all $j \in N_{k_1}^s$ and repeat step 5 for $k \prec k_1$, noting k_2 instead of k_1 in steps 5 and 6. Whenever step 5 is repeated for $k \prec k_\alpha$, note $k_{\alpha+1}$ instead of k_2 in steps 5 and 6.

If (63) does not hold for any k such that $N_k^s = \emptyset$, $k \prec s$, the algorithm has ended, with the same conclusion as in 5a.

6b. If all relations (63) hold, check the relations

$$\sum_{j \in N_{k_1}^s} \bar{a}_{ij} + \min (a_{ij} > 0) \leq y_i^{k_1} \quad (i \mid y_i^k \prec 0) \qquad (64)$$

then proceed as follows:

6b. i, If all the relations (64) hold pass to step 6c.

ii, If one or more of the relations (64) do not hold, let M_1 be the subset of M for which (64) does not hold, then cancel all those

$$\left\{ j \mid j \in N_k^s , a_{ij} \geq 0 , i \in M_1 \right\} \text{ from } N_k^s$$

Add this set of the cancelled values to H_s and C_s^{s+1}. This states that $x_j^s = 0$, $j \in N_k^s$ and $a_{ij} \geq 0$, $i \in M_1$, hence pass to step 6c.

6c. Check the relation

$$\sum_{j \in N_k^s} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i^s \quad (i \mid y_i^s < 0) \qquad (65)$$

and proceed as follows:

6c. i, If all relations (65) hold for $k = k_1$, compute the values $V_j^{k_1}$ for all $j \in N_{k_1}^s$ and select a j_{s+1} such that

$$V_{j_{s+1}}^{k_1} = \max_{j \in N_{k_1}^s} \left\{ V_j^{k_1} \right\} . \qquad (66)$$

Cancel $\left\{ j_{s+1} \right\}$ from $N_{k_1}^s$ and pass to step 8.

ii, If all the relations in (63) hold, and there exists a subset $M_{k_1}^s$ of M for which the relation (65) does not hold, pass to step 7.

Step 7. This is the same situation as in step 4. Therefore if

$$F_{k_1}^s = \left\{ j \mid j \in N_{k_1}^s, \ a_{ij} < 0 \right\}$$

for at least one $i \in M_{k_1}^s$, then it is necessary to introduce all the variables in $F_{k_1}^s$ into the solution simultaneously. Before doing this, check the relation

$$z_{k_1} + \sum_{j \in F_{k_1}^s} c_j < Z^* \qquad (67)$$

and proceed as follows:

7a. If (67) holds, cancel $V_j^{k_1}$ for all $j \in F_{k_1}^s$. Set

$$J_{s+1} = J_{k_1} \cup F_{k_1}^s$$

Compute the value, of the objective function

$$z_{s+1} = z_{k_1} + \sum_{j \in F_{k_1}^s} c_j \qquad (68)$$

and of the slack variables

$$Y_1^{s+1} = y_1^{k_1} - \sum_{j \in F_{k_1}^s} a_{ij}, \quad i \in M \qquad (69)$$

for the new solution

$$U^{s+1} = (u^{s+1}, X^{s+1}, Y^{s+1})$$

where the vector u^{s+1} is determined by

$$x_j = \begin{cases} 1, & j \in J_{s+1} \\ 0, & j \in H_{s+1} \end{cases}$$

$$X^{s+1} = 0$$

Pass to the next iteration starting again at step 2.

7b. If (67) does not hold, cancel the values $V_j^{k_1}$ for all $j \in N_{k_1}^s$ and repeat step 5, for $k < k_1$. If no $k < k_1$ exists, that is if $k_1 = 0$, the algorithm has ended with the same conclusion as in 5a.

Step 8. This step arises from 3c and 6c. In both situations, a single variable enters the solution.

In 3c, the situation is encountered where a single variable is introduced into the solution at node s and we proceed to node (s+1) on the same branch or path. However in 6c, the situation is encountered where the algorithm backtracks to a node k ($k < s$) and a single variable is introduced into the solution at node k and we proceed to node (s+1) along a new branch.

Now set

$$J_{s+1} = J_p \cup \{ j_{s+1} \}$$

where

$$p = \begin{cases} p = s, \text{ by proceeding from 3c to this step} \\ p = \text{ the last value cancolled, } V_{j_{s+1}}^p \\ = k_1 \qquad (p < s) \end{cases}$$

Compute the value, of the objectivo function and the slack variables for the new solution U^{s+1} at node (s+1), as follows:

$$z_{s+1} = z_p + c_{j_{s+1}} = \sum_{j \in J_{s+1}} c_j \tag{70}$$

and

$$y_i^{s+1} = y_i^p - a_{ij_{s+1}}$$

$$= b_i - \sum_{j \in J_{s+1}} a_{ij}, \quad i \in M \tag{71}$$

$$X^{s+1} = 0$$

$$u^{s+1} = \bar{u}^{s+1}$$

where $\bar{u}^{s+1}$ is determined by

$$x_j = \begin{cases} 1, & j \in J_{s+1} \\ 0, & j \in H_{s+1} \end{cases}$$

Therefore

$$U^{s+1} = (u^{s+1}, X^{s+1}, Y^{s+1})$$

and now pass to the next iteration starting again at step 1.

Note. The algorithm in the section yields one optimal solution (if such a solution exists). However by setting $>$ instead of $\geqslant$ in (51) and (55) and $\leqslant$ instead of $<$ in (62) and (67), it gives all existing optimal solutions.

This algorithm seems to be an efficient method for solving zero-one problems when there are few variables. Freeman [6] reported very good results when there are 30 or less variables. However this approach becomes less efficient, as the number of

variables increase. Recently Glover [8] developed "Multiphase dual-algorithm" which parallels Balas algorithm. He claims that his algorithm is more efficient than a number of other methods in solving a number of test problems.

SUMMARY

Linear programming problems can be solved by using either the standard simplex method or the dual simplex method. However these methods do not yield optimal solutions to problems, where integer solutions are desired. It is necessary to solve these integer programming problems by using different techniques such as Gomory's cutting plane method. There is also a special class of integer programming problems which require zero-one integer solutions. Gomory's cutting plane methods can be used to solve these zero-one problems, but they are somewhat inefficient. Other methods for solving these problems utilize the special structure of the zero-one problems. This paper investigates the various approaches that were developed to solve zero-one problems which are divided into three different categories as follows:

1. Cutting plane methods.
2. Parallel shifts of the objective function hyperplane.
3. Combinatorial methods.

A brief survey of each of the three approaches is presented in this paper. Most of the discussion is devoted to the combinatorial methods which the author believes are most efficient. Gomory's cutting plane method is presented along with Elmaghraby's method which falls into the second category. A general combinatorial approach is presented followed by a specific combinatorial algorithm developed by Balas. Each algorithm is described in detail and the solution process of two problems is illustrated in appendices.

CONCLUSION

It is interesting to note that both the cutting plane method and parallel shifting of the objective function hyperplane method, use additional constraints to cut the solution space W in order to exclude as many of the non-integer solutions as possible, but not any of the integer solutions are excluded. Gomory's cutting plane method generates the additional constraint from one of the problem constraint, where as Elmaghraby's method generates this additional constraint from the objective function.

As explained earlier, Gomory's method is very inefficient to solve zero-one problem, since the problem size is increased when constraints of the form $x_j \leq 1$, $j \in N$ are added to the original problem. The computational time increases very rapidly as the number of variables increase. Elmaghraby's method seems to be more efficient when compared with the cutting plane method. The upperbound technique which is incorporated in this method takes care of the upperbounded constraints $x_j \leq 1$, $j \in N$ and hence no constraints of this form are needed. But this method is inefficient from the fact that all alternate optimal solutions have to be considered.

The combinatorial approach seems to yield very good results. Its efficiency depends on the tests which exclude the non-feasible solutions. Freeman [6] modified Balas algorithm [1] to include some of the tests developed by Glover [8] and reported very good results when there are less than 30 variables. Balas method seems to be somewhat less efficient with more than 30

variables. In conclusion, more research is needed on zero-one integer programming algorithms since none are well suited for solving large practical problems.

ACKNOWLEDGEMENTS

I am taking this opportunity to extend my sincere thanks to Dr. F. A. Tillman, my major professor and Head of the Industrial Engineering Department, without whose encouragement, words of confidence and valuable guidance, this paper would not have been successful. I also wish to extend my sincere thanks to Dr. S. A. Konz and Dr. L. E. Grosh, for their valuable suggestions.

REFERENCES

1. Balas, Egon., "An Additive Algorithm for Solving Linear Programs with Zero-one Variables," Operations Research, July-August, 1965.

2. Balanski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12, No. 3, Nov. 1965.

3. Beale, E. M. L., "Survey of Integer Programming," Operations Research Quarterly, Vol. 16, No. 2, June 1965.

4. Dantzig, G. B., "Linear Programming and Extensions," Princeton University Press, 1963.

5. Elmaghraby, S. E., "An Algorithm for the Solution of the 'Zero-one' problem of Integer Linear Programming," Department of Industrial Administration, Yale University, May, 1963.

6. Freeman, N. J., "Computational Experience with the Balas Integer Programming Algorithm," RAND, P-3241, Santa Monica, California, October 1965.

7. Goeffrion, A., "Implicit Enumeration and the 0-1 Algorithm of Balas," RAND, RM-4783-PR, Santa Monica, California, February 1966.

8. Glover, Fred., "A Multiphase-dual Algorithm for the Zero-one Integer Programming Problem," Operations Research, Nov.-Dec. 1965.

9. Glover, Fred., "Truncated Enumeration Methods for Solving Pure and Mixed Integer Linear Programs," WP-27, Operations Research Center, University of California, Berkely, May 1966.

10. Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," Princeton-IBM Mathematics Research Project, Technical Report No. 1, Nov. 17, 1958.

11. Gomory, R. E., "All-Integer Integer Programming Algorithm," IBM Research Report RC-189, Jan. 29, 1960.

12. Hadley, G., "Non Linear and Dynamic Programming," Addison-Wesley Publishing Co., Inc., 1962.

13. Camion, P., "Une me'thods de resolution par l'alge'bre de Boole des proble'mes combinatoires ou interviennent des entiers," Cashiers du Centre D' E'tudes de Recherche, Operationnelle, Brussels, 1960.

14. Fortet, R., "Applications de l'alge'bre de Boole en recherche ope'rationnelle," Revue Francaise de Recherche Ope'rationnelle 4, 17-25 (1960).

A P P E N D I C E S

APPENDIX I

Problem

Minimize $2x_1 + x_2 + 4x_3$

subject to
$$-x_1 + x_2 - 2x_3 \leq 0$$
$$2x_1 + 2x_2 + x_3 \leq -1$$
$$x_1 + x_2 + x_3 \leq 2$$
$$x_1, \quad x_2, \quad x_3 = 0 \text{ or } 1$$

Solution

Converting the above problem into Beale Equation form, we obtain

Maximize
$$z = z_o + 2(-x_1) + (-x_2) + 4(-x_3)$$
$$y_1 = 0 - (-x_1) + (-x_2) - 2(-x_3)$$
$$y_2 = -1 + 2(-x_1) - 2(-x_2) + (-x_3)$$
$$y_3 = 2 + (-x_1) + (-x_2) + (-x_3)$$
$$y_4 = 1 + (-x_1) + 0 + 0$$
$$y_5 = 1 + 0 + (-x_2) + 0$$
$$y_6 = 1 + 0 + 0 + (-x_3)$$
$$x_1 = 0 - (-x_1) + 0 + 0$$
$$x_2 = 0 + 0 - (-x_2) + 0$$
$$x_3 = 0 + 0 + 0 - (-x_3)$$

Now arranging this problem in a tableau form similar to tableau

1, we obtain

		0	1	2	3	
		$-b$	$-x_1$	$-x_2$	$-x_3$	
0	z	0	2	1	4	
1	y_1	0	-1	1	-2	
2	y_2	-1	2	-2*	1	←
3	y_3	2	1	1	1	
4	y_4	1	1	0	0	
5	y_5	1	0	1	0	
6	y_6	1	0	0	1	
7	x_1	0	-1	0	0	
8	x_2	0	0	-1	0	
9	x_3	0	0	0	-1	
				↑		

The ordinary simplex method cannot be applied to the above tableau since it is not primal feasible, that is all $b_i \not\geq 0$, $i \in M$ which is a necessary condition to solve any linear programming problem using this method. Hence the dual simplex algorithm is applied to obtain a non-integer optimal solution. The dual simplex algorithm starts with an initial solution $Y = b$, $X = 0$ and $z = 0$, where Y is called the basis vector, X non-basic vector and z the value of the objective function. At each iteration, a non-basic variable replaces a basic variable with an improvment in the objective function value. The row corresponding to the basic variable leaving the basis is called the

pivot row and the column corresponding to the variable entering the basis is called the pivoted column. This change of basis is done by transformation of the tableau applying the procedure of the simplex method. The pivot row is denoted by r and the pivot column by k. The selection of pivot row is made by choosing the most negative b_i, that $b_r = \min b_i$. Thus x_{B_r}, the variable from the basis X_B corresponding to the row r, leaves the basis. The pivot column k is determined by

$$\frac{c_k}{a_{rk}} = \min \left| \frac{c_j}{a_{zj}} \right| , \quad a_{2j} < 0$$

This process of changing basis and improving z-value continues till a primal feasible solution, that is all y_i (transformed value of b_i) ≥ 0, is obtained. The solution then represents an optimal feasible solution to the linear programming problem, but not to the zero-one problem. The iterative procedure is now as follows:

Iteration 1. The pivot row corresponds to the most negative b_i, i = 1, 2, 3, 4, 5 and 6. As there is only one negative b_i, i = 2, the pivot row r = 2 for which $b_r = -1$. The pivot column k is determined by

$$\frac{c_k}{a_{rk}} = \min \left| \frac{c_j}{a_{2j}} \right| , \quad j \mid a_{2j} < 0$$

Again there is only one negative a_{2j}, j = 2. Hence the pivot column k = 2 and the pivot element $a_{rk} = a_{22} = -2$. In the

tableau, the pivot row and pivot column are shown by arrows, while the pivot element is shown with a star. Thus x_2 replaces y_2 in the basis. The new tableau is obtained through the following relations

$$A_j' = A_j - \frac{A_k}{a_{rk}} \cdot a_{rj} \qquad j \neq k, \text{ that is } j = 0,1,3.$$

and

$$A_j' = \frac{A_j}{|a_{rk}|} \qquad j = k = 2 .$$

where A_j is a $(m + 2n + 1) \times 1$ column vector of elements a_{ij}, $i = 0, 1, 2, \ldots, m + 2n$. Through the above transformation, the following tableau is obtained.

	$-b$	$-x_1$	$-y_2$	$-x_3$	
z	-1/2	3	1/2	9/2	
y_1	-1/2	0	1/2	-3/2*	←
y_2	0	0	-1	0	
y_3	3/2	2	1/2	3/2	
y_4	1	1	0	0	
y_5	1/2	1	1/2	1/2	
y_6	1	0	0	1	
x_1	0	-1	0	0	
x_2	1/2	-1	-1/2	-1/2	
x_3	0	0	0	-1	
				↑	

Iteration 2. At each iteration, it is necessary to check if the solution is primal feasible. However, the solution in the new tableau is not primal feasible since $b_1 = -1/2$. Also since this is only the negative element in the b vector, it is the pivot row r. Furthermore, there is only one negative element a_{ij} $(j = 1, 2, 3)$ in the pivot row and it is $a_{13} = -3/2$. Hence $r = 1$ and $k = 3$, while the pivot element $a_{rk} = a_{13} = -3/2$. The transformed tableau is shown below.

	$-b$	$-x_1$	$-y_2$	$-y_1$	
z	-2	3	2	3	
y_1	0	0	0	-1	
y_2	0	0	-1	0	
y_3	1	2	1	1	
y_4	1	1	0	0	
y_5	1/3	1	2/3	1/3	
y_6	2/3	0	1/3	2/3	
x_1	0	-1	0	0	
x_2	2/3	-1	-2/3	-1/3	
x_3	1/3	0	-1/3	-2/3	
s_1	-2/3	0	-1/3	-2/3*	←

↑

Iteration 3. As the new b vector is non-negative, the solution is primal feasible and optimal to the linear programming problem obtained by removing the integer restriction from the

zero-one problem. Since the solution is not integer valued, it is necessary to add an additional constraint. Thus the first two steps in the summary (of the cutting plane method) are completed and the algorithm proceeds to step 3.

Step 3. As more than one component of $Y_o(b)$ is non-negative, the component having the largest fraction f_{uo} is to be selected to form the additional constraint. From the optimal solution tableau, it is evident that y_5, y_6, x_2 and x_3 are non-integers. The fraction f_{io} is obtained from the relation

$$b_i = y_i = \delta_{io} + f_{io}$$

where δ_{io} is the largest integer less than y_i and f_{io} is a positive fraction which if added to δ_{io} equals to b_i. It is clear that b_6 and b_8 have the largost fractions

$$f_{60} = f_{80} = 2/3 .$$

When there is a tie, either one may be selected. Hence by selecting the row 8 for obtaining a new constraint, we obtain the new constraint as

$$s_1 = - f_{80} - f_{81}(-x_1) - f_{82}(-y_2) - f_{83}(-y_1)$$

where the f_{8j} are to be determined from row 8 which is stated below.

$$x_2 = 2/3 - (-x_1) - 2/3\ (-y_2) - 1/3\ (-y_1)$$

Now

$$f_{80} = b_8 - \delta_{80} = 2/3 - 0 = 2/3$$

$$f_{81} = a_{81} - \delta_{81} = -1 - (-1) = 0$$

$$f_{82} = a_{82} - \delta_{82} = -2/3 - (-1) = 1/3$$

$$f_{83} = a_{83} - \delta_{83} = -1/3 - (-1) = 2/3$$

Therefore the new constraint to be added to the last tableau is

$$S_1 = -2/3 - 0 - 1/3\,(-y_2) - 2/3\,(-y_1) .$$

This row is shown on the bottom of the last tableau. Now the variable leaving the basis is S_1 since it is the only variable taking negative value, that is $S_1 = -2/3 = b_{10}$.

Step 4. The column k entering the basis is

$$\frac{c_k}{a_{rk}} = \min \left| \frac{c_j}{a_{rj}} \right| \qquad (j \mid a_{rj} < 0)$$

$$\frac{c_2}{a_{102}} = \frac{2}{-1/3} = -6$$

$$\frac{c_3}{a_{103}} = \frac{3}{-2/3} = -4\,1/2$$

and

$$\left| \frac{c_3}{a_{103}} \right| < \left| \frac{c_2}{a_{102}} \right| .$$

Hence $k = 3$. Y_1 replaces S_1. The pivot element is $a_{103} = -2/3$.

Step 5. The new tableau is shown on the next page.

	$-b_1$	$-x_1$	$-y_2$	$-s_1$
z	-5	3	1/2	9/2
y_1	1	0	1/2	-3/2
y_2	0	0	-1	0
y_3	0	2	1/2	3/2
y_4	1	1	0	0
y_5	0	1	1/2	1/2
y_6	0	0	0	1
x_1	0	-1	0	0
x_2	1	-1	-1/2	-1/2
x_3	1	0	0	-1
s_1	0	0	0	-1

Iteration 4. From the new tableau it is evident that the solution is primal feasible and integer valued. Hence it is the optimal zero-one solution which is stated below.

$x_1 = 0$	$y_1 = 1$	$y_4 = 1$
$x_2 = 1$	$y_2 = 0$	$y_5 = 0$
$x_3 = 1$	$y_3 = 0$	$y_6 = 0$

Minimum value of the objective function = z = 5.

APPENDIX II

UPPERBOUND TECHNIQUE

The upperbound technique is a modified form of the simplex method. This method is utilized to solve any linear programming problem in which some of the variables are upperbounded, that is they cannot exceed a certain value. As explained earlier, the upperbound technique cuts down the computational time considerably when solving these type of problems. The problem to be considered in this section is stated as follows:

$$\text{maximize} \quad z = \sum_{j=1}^{n} C_j x_j \qquad \left.\begin{array}{l} \\ \\ \end{array}\right\} 1$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \left.\begin{array}{l} \\ \\ \end{array}\right\} 1$$

$$x_j = 0 \qquad \left.\begin{array}{l} \\ \\ \end{array}\right\} 2$$

$$x_j = U_j \qquad \left.\begin{array}{l} \\ \\ \end{array}\right\} 2$$

$$i \in M$$

$$j \in N$$

where

U_j = upperbound on the variable x_j

= 1 in the case of zero-one variables

= ∞ if the variable x_j is not upperbounded

$M = \{ 1, 2, 3, \ldots, m \}$

$$N = \{1, 2, 3, \ldots, n\} .$$

Problem (1) is a standard linear programming problem. The simplex method is used to solve this problem and it yields a solution in which

a, at mest m variables (called basic variables) take values greater than zero and

b, the remaining (n-m) variables (called non-basic variables) take values equal to zero.

This solution is termed basic solution. In short, any basic solution to a linear pregramming problem has (n-m) nen-basic variables at lower bounds (equal to zero), while the remaining m basic variables take positive values. However a basic solution to a problem with upperbounded variables, denoted by (2) consists of

a, m variables with $0 \leq x_j \leq U_j$, $j \in N$

b, k variables with $x_j = U_j$, $j \in N$ and

c, n - (m + k) variables with $x_j = 0$, $j \in N$.

It is noted that the initial solution to problem (1) is the same as that of preblem (2), since $k = 0$ for the initial solution. The procedure for solving problem (2) is the same as that of problem (1) except for the selection of i, the variable entering the basis and ii, the variable leaving the basis. Also the optimality criterion for problem (2) is different from that of problem (1). Therefore only these modifications will be summarized in this section. For detailed explanation,

the reader is referred to [4] and [12]. Now consider the solution to problem (2) at p-th iteration (p = 0, for the initial solution), value of the objective function

$$= z = \sum_{i=1}^{m} c_{B_i} x_{B_i} + \sum_{j=m+1}^{m+k} c_j U_j$$

and

$$x_{B_i} = b_i^p = b_i' - \sum_{j=m+1}^{m+k} y_{ij} U_j$$

where

$$b_i' = B^{-1} b_i^o \qquad (p = 0)$$

$$y_j = B^{-1} a_j$$

Now it is necessary to find the variable x_s entering the basis and x_{B_r} be the variable leaving the basis. The optimality criterion for the upper technique is satisfied when,

1, $z_j - c_j \leq 0$, $j = m + 1, m + 2, \ldots, m + k$; (for non-basic variables at upporbounds) and

2, $z_j - c_j \geq 0$, $j = m + k + 1, m + k + 2, \ldots, n$; (for non-basic variables at zero).

If these two conditions are not satisfied, the solution to problem (2) is not optimal and hence the procedure is to be repeated to find a new basis with an improvement in the value of the objective function. For this, it is necessary to find the variable

entering the basis.

Selection of the variable x_s entering the basis: Find

$$z_s - c_s = \min \left\{ -(z_{m+1} - c_{m+1}), -(z_{m+2} - c_{m+2}), \ldots, \right.$$
$$-(z_{m+k} - c_{m+k}), z_{m+k+1} - c_{m+k+1},$$
$$\left. z_{m+k+2} - c_{m+k+2}, \ldots, z_n - c_n \right\}.$$

In otherwords, multiply $z_j - c_j$ by (-1) for the non-basic variables at their upperbounds that is $x_j = U_j$ and select the min $(z_j - c_j)$, $j \in R$ where R is the set of indices corresponding to non-basic variables. For a non-optimal solution, $\min(z_j - c_j)$ must be negative. This insures an improvement in the objective function value. The corresponding variable x_s enters the basis. Thus having found x_s, it remains to find the variable x_{B_r} leaving the basis which depends on the value of x_s.

Selection of x_{B_r}: A, if $x_s = 0$. Find

$$\frac{b_r^{p+1}}{y_{rs}} = \min_i \left\{ \frac{b_i^p}{y_{is}}, \; y_{is} > 0 \text{ and } \frac{b_i^p - U_{B_i}}{y_{is}}, \right.$$
$$\left. y_{is} < 0 \right\},$$

where

U_{B_i} = upperbound on the variable x_{B_i}.

There are two cases that can arise in the determination of $\frac{b_r^{p+1}}{y_{rs}}$.

1. If

$$\frac{b_r^{p+1}}{y_{rs}} = \frac{b_r^p}{y_{rs}}, \qquad y_{rs} > 0,$$

then x_{B_r} leaves the basis and becomes a non-basic variable with zero value at (p+1)th iteration. The solution at (p+1)th iteration consists of

a, m variables with $0 \leq x_{B_i} \leq U_{B_i}$, $i \in M$

b, k variables with $x_j = U_j$, $j \in N$ and

c, n - (m+k) variables with $x_j = 0$, $j \in N$.

2. If

$$\frac{b_r^{p+1}}{y_{rs}} = \frac{b_r^p - U_{B_r}}{y_{rs}}, \qquad y_{rs} < 0,$$

then x_{B_r} leaves the basis and becomes a non-basic variable at upperbound at (p+1)th iteration. The solution at (p+1)th iteration consists of

a, m variables with $0 \leq x_{B_i} \leq U_{B_i}$, $i \in M$

b, k+1 variables with $x_j = U_j$, $j \in N$ and

c, n - (m + k + 1) variables with $x_j = 0$, $j \in N$.

B, if $x_s = U_s$. Find

$$\frac{b_r^{p+1}}{y_{rs}} = \max_i \left\{ \frac{b_i^p + y_{is}U_s}{y_{is}}, \; y_{is} < 0 \text{ and } \frac{b_i^p + y_{is}U_s - U_{B_i}}{y_{is}}, \; y_{is} > 0 \right\}.$$

Here again there are two cases arising in the determination of x_{B_r} .

1. If

$$\frac{b_r^{p+1}}{y_{rs}} = \frac{b_r^p + y_{rs}U_s}{y_{rs}} , \quad y_{rs} < 0 ,$$

then the non-basic variable with $x_s = U_s$ becomes a basic variable with a value $0 \leq x_s \leq U_s$ and the basic variable x_{B_r} leaves the basis to become a non-basic variable with zero value at (p+1)th iteration. The solution at (p+1)th iteration consists of

a, m variables with $0 \leq x_{B_i} \leq U_{B_i}$, $i \in M$

b, (k-1) variables with $x_j = U_j$, $j \in N$ and

c, $n - (m + k - 1)$ variables with $x_j = 0$, $j \in N$.

2. If

$$\frac{b_r^{p+1}}{y_{rs}} = \frac{b_r^p + y_{rs}U_s - U_{B_r}}{y_{rs}} , \quad y_{rs} > 0 ,$$

then the non-basic variables with $x_s = U_s$ becomes a basic variable with $0 \leq x_s \leq U_s$ and the basic variable with $0 \leq x_{B_r} \leq U_{B_r}$ leaves the basis to become a non-basic variable with $x_{B_r} = U_{B_r}$ at (p+1)th iteration. The solution at (p+1)th iteration consists of

a, m variables with $0 \leq x_{B_i} \leq U_{B_i}$, $i \in M$.

b, k variables with $x_j = U_j$, $j \in N$ and

c, n - (m+k) variables with $x_j = 0$, $j \in N$.

Thus having determined the variable leaving the basis and the variable entering the basis, the simplex method is applied to obtain the new tableau. If the optimality criterion is satisfied, the solution is then optimal at (p+1)th iteration. Otherwise, the procedure is repeated until the optimality criterion is satisfied.

APPENDIX III

The example solved in Appendix I is arranged in a simplex tableau as follows:

Tableau 1

			-2	-1	4	0	0	0	-100	
		b	x_1	x_2	x_3	y_1	y_2	y_3	y_4	
0	y_1	0	-1	1*	-2	1	0	0	0	←
-100	y_4	1	-2	2	-1	0	-1	0	1	
0	y_3	2	1	1	1	0	0	1	0	
				↑						

x_2 enters the basis and y_1 leaves the basis. The new tableau is shown below.

Tableau 2

		b	x_1	x_2	x_3	y_1	y_2	y_3	y_4	
-1	x_2	0	-1	1	-2	1	0	0	0	
-100	y_4	1	0	0	3*	-2	-1	0	1	←
0	y_3	2	2	0	3	-1	0	1	0	
					↑					

Here x_3 enters the basis and y_4 leaves the basis. Since y_4 is an artificial variable leaving the basis, the corresponding column y_4 is dropped from the following tableaus.

Tableau 3 (F_o-tableau)

		b	x_1	x_2	x_3	y_1	y_2	y_3	y_4
-1	x_2	2/3	-1	1	0	-1/3	-2/3	0	2/3
-4	x_3	1/3	0	0	1	-2/3	-1/3	0	1/3
0	y_3	1	2	0	0	1	1	1	-1
	$z_j - c_j$	-2	+3			+3	+2		+96

Since all $z_j - c_j$ are positive, the solution is optimal to the linear programming problem but not to the zero-one problem since b is not integer. There are no alternate optima for $z = 2$, since there is no j such that $V_j = 0$, $j \in R$ where $R = \{ x_1, y_1, y_2 \}$.

The D-equation to be annexed to F_o-tableau is

$$- 3x_1 - 3y_1 - 2y_2 + D_1 = -d^1$$

where the coefficients are $-(z_j - c_j) = -V_j$

Now it is necessary to determine d_1, where

$$d^1 = \min_i \left[\min_j \left\{ \frac{b_i V_j}{a_{ij}}, \; a_{ij} > 0 \text{ or } \frac{(b_i - B_o)V_j}{a_{ij}}, \; a_{ij} < 0 \right\} \right]$$

$i = 1$

$$\frac{(b_1 - B_1)V_1}{a_{11}} = \frac{(2/3 - 1)3}{-1} = 1$$

$$\frac{(b_1 - B_1)V_4}{a_{14}} = \frac{(2/3 - 1)3}{-1/3} = 3$$

$$\frac{(b_1 - B_1)V_5}{a_{15}} = \frac{(2/3 - 1)2}{-2/3} = 1$$

$$\min_j \left\{ \frac{(b_1 - B_1)V_j}{a_{ij}} \right\} = \frac{(b_1 - B_1)V_1}{a_{11}} = 1$$

Note that d is minimum for $j = 1$ and 5. Hence it is indifferent to select either one. Selecting the first one that is

$$\min_j \left\{ \frac{(b_1 - B_1)V_1}{a_{11}} \right\} = 1$$

i = 2

$$\frac{(b_2 - B_2)V_4}{a_{24}} = \frac{(1 - 1/3)3}{2/3} = 3$$

$$\frac{(b_2 - B_2)V_5}{a_{24}} = \frac{(1 - 1/3)2}{1/3} = 4$$

i = 3

$$\frac{b_3V_1}{a_{31}} = \frac{1 \times 3}{2} = 1.5$$

$$\frac{b_3 v_4}{a_{34}} = \frac{1 \times 3}{1} = 3$$

$$\frac{b_3 v_5}{a_{35}} = \frac{1 \times 2}{1} = 2$$

Hence

$$\min_i \left[\min_j \left\{ d_{ij}^1 > 0 \right\} \right] = d^{1*} = 1$$

So the D-equation is

$$-3x_1 - 3y_1 - 2y_2 + D_1 = -1$$

The F_1 tableau can be formulated as

	b	x_1	x_2	x_3	y_1	y_2	y_3	D_1	
x_2	2/3	-1	1	0	-1/3	-2/3	0	0	
x_3	1/3	0	0	1	-2/3	-1/3	0	0	
y_3	1	2	0	0	1	1	1	0	
D_1	-1	-3^*	0	0	-3	-2	0	1	←
		↑							

According to step 3 of the algorithm, the pivot row is the D-equation, since $D_1 = -1$ is negative. The pivot column is $k = 1$, since d^* was found from that column. The ordinary simplex is applied for obtaining the new tableau which is as follows

Tableau 4

		b	x_1	x_2	x_3	y_1	y_2	y_3	D_1
-1	x_2	1	0	1	0	2/3	0	0	-1/3
-4	x_3	1/3	0	0	1	-2/3	-1/3	0	0
0	y_3	1/3	0	0	0	-1	-1/3	1	2/3
-2	x_1	1/3	1	0	0	1	2/3	0	-1/3
$z_j - c_j$		-3				0	0		1

Determination of d^{2*}: $i = 1$

$$-V_j \prec 0 \quad \text{for} \quad j = 7 \quad \text{(corresponding to } D_1 \text{ column)}$$

$$\frac{(b_1 - B_1)V_7}{a_{17}} = \frac{(1 - 1)1}{-1/3} = 0$$

Hence $d^2 = 0$

For this tableau $d^2 \leq 0$; if $d \succ d^2$, it is observed from the tableau that x_2 will exceed 1. So it is necessary to remove x_2 from the basis. For this, let us make transformation $x_2 = 1 - x_2'$, where $x_2' = 0$. Since the elements in the x_2 column are zeros in all rows except the first one, we only need to transform that row. The equation corresponding to this row is

$$x_2 + 2/3\ y_1 - 1/3\ D_1 = 1$$

Substituting $x_2 = 1 - x_2'$, we obtain

$$-x_2' + 2/3\ y_1 - 1/3\ D_1 = 0$$

Multiplying this equation throughout by (-1), we get

$$x_2' - 2/3\ y_1 + 1/3\ D_1 = 0$$

So the new transformed tableau is as follows (note that the sign of the cost coefficient of x_2' is opposite to that of x_2)

Transformed tableau 4

			-2	1	-4	0	0	0	0	
		b	x_1	x_2'	x_3	y_1	y_2	y_3	D_1	
1	x_2'	0	0	1	0	-2/3	0	0	1/3*	←
-4	x_3	1/3	0	0	1	-2/3	-1/3	0	0	
0	y_3	1/3	0	0	0	-1	-1/3	1	2/3	
-2	x_1	1/3	1	0	0	1	2/3	0	-1/3	
$z_j - c_j$		-3				0	0		1	
									↑	

Again $d^2 \leq 0$, since $\dfrac{b_1 v_7}{a_{17}} = \dfrac{0 \times 1/3}{1} = 0$

This means that for $d > 0$, x_2' will become negative. To meet this situation, it is necessary to drive x_2' out of basis. Hence the pivot row is the first row. Since there is only one positive element (corresponding to non-basic variable) in the pivot row it is selected as the pivot element. Hence r = 1,

k = 7. The new tableau is:

Tableau 4a

	b	x_1	x_2	x_3	y_1	y_2	y_3	D_1
D_1	0	0	3	0	-2	0	0	1
x_3	1/3	0	0	1	-2/3	-1/3	0	0
y_3	1/3	0	-2	0	1/3	-1/3	1	0
x_1	1/3	1	1	0	1/3	2/3	0	0

A constraint is redundant, when $D \geq 0$. This makes the process simplified. In the above tableau, D_1 is in the basis and its value is zero. Hence the first row corresponding to the basic variable D_1 and D_1 column can be disregarded from the tableau. This is shown in tableau 4b. Note that 4b is an alternate solution to 4.

Tableau 4b

			-2	1	-4			
		b	x_1	x_2'	x_3	y_1	y_2	y_3
-4	x_3	1/3	0	0	1	-2/3	-1/3	0
0	y_3	1/3	0	-2	0	1/3	-1/3	1
-2	x_1	1/3	1	1	0	1/3	2/3*	0 ←
	$z_j - c_j$	-3		-3		2	0	
							↑	

$-V_j < 0$ for $j = 4$

The D-equation is

$$3x_2' - 2y_1 + D_2 = -d_1^2$$

$$\frac{(b_1 - B_1)V_4}{a_{14}} = \frac{(1/3 - 1)}{-2/3} = 2$$

$$\frac{b_2V_4}{a_{24}} = \frac{1/3 \times 2}{1/3} = 2$$

$$\frac{b_3V_4}{a_{34}} = \frac{1/3 \times 2}{1/3} = 2$$

Hence $$d_1^2 = 2 ,$$

for $i = 1, 2, 3$ and $j = 4$. From the tableau 4b, it is observed that $V_j = 0$ for the non-basic variable y_2. This indicates that there is an alternate optimal solution. The pivot element is a_{34}. y_2 enters the basis and x_1 leaves the basis. The alternate solution is shown in tableau 4c.

Tableau 4c

			-2	1	-4	0	0	
		b	x_1	x_2	x_3	y_1	y_2	y_3
-4	x_3	1/2	1/2	1/2	1	-1/2	0	0
0	y_3	1/2	1/2	-3/2	0	1/2	0	1
0	y_2	1/2	3/2	3/2	0	1/2	1	0
	$z_j - c_j$	-2	0	-3		2		

$$-V_j < 0 \quad \text{for} \quad j = 4$$

The D-equation for this iteration is

$$3x_2' - 2y_1 + D_2 = -d_2^2$$

$$\frac{(b_1 - B_1)V_4}{a_{14}} = \frac{(1/2 - 1)2}{-1/2} = 2$$

$$\frac{b_2V_2}{a_{24}} = \frac{1/2 \times 2}{1/2} = 2$$

$$\frac{b_3V_2}{a_{24}} = \frac{1/2 \times 2}{2} = 2$$

Hence $d_2^2 = 2$, for $i = 1, 2, 3$ and $j = 4$.

$$d^{2*} = \min_k \; d_k^2, \quad k = 1, 2$$

Since $d_1^2 = d_2^2$, it makes no difference to select either one. Selecting $d^* = d_1^2$ for forming D-equation, we obtain the pivot row $r = 4$ corresponding to this D-equation and pivot column $k = 4$. The new annexed tableau 4d obtained from tableau 4b is shown below.

Tableau 4d

	b	x_1	x_2	x_3	y_1	y_2	y_3	D_2	
x_3	1/3	0	0	1	-2/3	-1/3	0	0	
y_3	1/3	0	-2	0	1/3	-1/3	1	0	
x_1	1/3	1	1	0	1/3	2/3	0	0	
D_2	-2	0	3	0	-2*	0	0	1	←
					↑				

The simplex procedure is applied to obtain tableau 5 and is shown below.

Tableau 5

			-2	1	-4	0	0	0	
		b	x_1	x_2	x_3	y_1	y_2	y_3	D_2
-4	x_3	1	0	-1	1	0	-1/3	0	-1/3
0	y_3	0	0	-3/2	0	0	-1/3	1	1/6
-2	x_1	0	1	3/2	0	0	2/3	0	1/6
0	y_1	1	0	-3/2	0	1	0	0	-1/2
z_j-c_j	-4	-1=	-5		0		0		1

This is the optimal tableau, since b vector is non-negative and $z_j - c_j \geq 0$ for all j . Hence the optimal integer solution to the example problem is

$$x_2 = 1 - x_2' = 1, \qquad x_3 = 1$$

$$x_1 = 0, \qquad y_2 = y_3 = 0, \qquad y_1 = 1$$

$$z^* = 1 + 4 = 5 .$$

Note that

$$z^* = z_o - d^{1*} - d^{2*} = - 2 - 1 - 2 = -5$$

APPENDIX IV

Previously a small and easy problem is solved. However this time a large problem with 7 constraints and 10 variables is chosen. The problem is to

minimize

$$6x_1 + 5x_2 + x_3 + 7x_4 + 2x_5 + 4x_6 + 3x_7 + x_8 + 5x_9 + 3x_{10} .$$

subject to

$$\begin{aligned}
3x_1 - 8x_2 - 8x_3 + x_4 + 0 + 0 + 0 + x_8 - 6x_9 + 2x_{10} &\leq -2 \\
0 + x_2 - 5x_3 + 0 - 5x_5 + x_6 + x_7 + x_8 + 0 + 0 &\leq -1 \\
2x_1 - 2x_2 - x_3 + 0 + x_5 + 0 + 0 - 2x_8 + 0 + x_{10} &\leq -3 \\
-5x_1 + 3x_2 + x_3 + 0 + 0 + 0 + x_7 + x_8 - x_9 - x_{10} &\leq 1 \\
0 + 0 - 2x_3 - 3x_4 + 0 + 4x_6 + x_7 - 5x_8 - 2x_9 + 0 &\leq -4 \\
0 - 8x_2 + 0 - 6x_4 - 6x_5 + 5x_6 + 0 + 2x_8 - 12x_9 - 4x_{10} &\leq -7 \\
-7x_1 + 4x_2 + 3x_3 - 6x_4 - x_5 + 0 - 5x_7 - x_8 - 8x_9 + 0 &\leq -5
\end{aligned}$$

$$x_j = 0 \text{ or } 1 , \quad j = 1, 2, \ldots, 10$$

This problem is arranged in tableau form on the next page. First we start with an initial solution $U^o = (0,b)$

$$J_o = \emptyset \quad , \quad z_o = 0 \quad , \quad H_o = T_o = Q_o = C^o = \emptyset$$

$$\begin{aligned}
y_1^o = b_1^o &= -2 , & y_4^o = b_4^o &= 1, & y_6^o = b_6^o &= -7 \\
y_2^o = b_2^o &= -1 , & y_5^o = b_5^o &= -4, & y_7^o = b_7^o &= -5 \\
y_3^o = b_3^o &= -3 ,
\end{aligned}$$

6	5	1	7	2	4	3	1	5	3	→ cost coefficients
1	2	3	4	5	6	7	8	9	10	

$\rightarrow j$ $\qquad a_{ij} \qquad$ b°

$\downarrow i$

$$\begin{bmatrix} 3 & -8 & -8 & 1 & 0 & 0 & 0 & 1 & -6 & 2 \\ 0 & 1 & -5 & 0 & -5 & 1 & 1 & 1 & 0 & 0 \\ 2 & -2 & -1 & 0 & 1 & 0 & 0 & -2 & 0 & 1 \\ -5 & 3 & 1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & -3 & 0 & -4 & 1 & -5 & -2 & 0 \\ 0 & -8 & 0 & -6 & -6 & 5 & 0 & 2 & -12 & -4 \\ -7 & 4 & 3 & -6 & -1 & 0 & -5 & -1 & -8 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} \leq \begin{bmatrix} -2 \\ -1 \\ -3 \\ 1 \\ -4 \\ -7 \\ -5 \end{bmatrix}$$

These are shown in the top middle of the tableau 2. The computations are briefly arranged in the tableau. The superscripts on the values of V_j^s represent the order of cancellation at any iteration. It is easy to follow the tableau very easily by the help of the following iterative procedure.

Iteration 1

Step 1. $y_i^o < 0$ for $i = 1, 2, 3, 5, 6, 7$.

So we are in situation 1b. So we pass to step 2.

Step 2. $N_o = N - (C^o \cup D_o \cup E_o)$

$C^o = D_o = \emptyset$,

$E_o = \emptyset$,

since there is no j such that for all $i \mid y_i < 0$, the corresponding $A_j > 0$

Hence $N_o = N = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$

So we are in case 2b, and pass to step 3.

Step 3. Check the relation

$$\sum_{j \in N_o} \bar{a}_{1j} \leq y_i^o \quad \text{for } i = 1, 2, 3, 5, 6, 7 .$$

$$\sum_{j \in N_o} \bar{a}_{1j} = \bar{a}_{12} + \bar{a}_{13} + \bar{a}_{19} = -8 - 8 - 6 = -22 < y_1^o = -2$$

$$\sum_{j \in N_o} a_{2j} = a_{23} + a_{25} = -5 - 5 = -10 < y_2^o = -1$$

Tableau 2

Row No.	s	J_s	z_s	N_s		i								C^s	D_s	E_s	F_s	H_s
						1	2	3	4	5	6	7	V_j^s					
1	0	$\emptyset$	0		$y_i^o \rightarrow$	- 2	- 1	-3	1	- 4	- 7	- 5		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2					$\sum_{j \in N_o} \bar{a}_{ij}$	-20	-10	-5		-16	-36	-28						
3					$+\min(a_{ij} \geq 0)$	-20	-10	-5		-16	-36	-28						
4					$-\max(a_{ij} < 0)$	-14	- 5	-4		-14	-32	-27						
5				1	$y_i^o - a_{i1}$	- 5	- 1	-5		- 4	- 7		-22					
6				2	$y_i^o - a_{i2}$		- 2	-1	-2	- 4		- 9	-18					
7				3	$y_i^o - a_{i3}$			-2		- 2	- 7	- 8	-19^{12}					
8				4	$y_i^o - a_{i4}$	- 3	- 1	-3		- 1	- 1		$- 9^{4}$					
9				5	$y_i^o - a_{i5}$	- 2		-4		- 4	- 1	- 4	-15^{11}					
10				6	$y_i^o - a_{i6}$	- 2	- 2	-3			-12	- 5	-24					
11				7	$y_i^o - a_{i7}$	- 2	- 2	-3		- 5	- 7		-19					
12				8	$y_i^o - a_{i8}$	- 3	- 2	-1			- 9	- 4	-19					
13				9	$y_i^o - a_{i9}$		- 1	-3		- 2			$- 6^{1}$					
14				10	$y_i^o - a_{i10}$	- 4	- 1	-4		- 4	- 3	- 5	-21					

Tableau 2 (cont.)

Row No.	s	J_s	z_s	N_s		i								C^s	D_s	E_s	F_s	H_s
						1	2	3	4	5	6	7	V_j^s					
15	1	9	5		$y_i^1 \rightarrow$	4	- 1	-3	2	- 2	5	3		9	ø	1,7,10	ø	ø
16					$\sum_{j \in N_1} \bar{a}_{ij}$		-10	-5		-14								
17					$+\min(a_{ij} \geq 0)$		-10	-5		-14								
18					$-\max(a_{ij} < 0)$		- 5	-4		-12								
19				2	$y_i^1 - a_{i2}$		- 2	-1	-1	- 2		- 1	$- 7^5$					
20				3	$y_i^1 - a_{i3}$			-2					$- 2^2$					
21				4	$y_i^1 - a_{i4}$		- 1	-3					$- 4^6$					
22				5	$y_i^1 - a_{i5}$			-4	-2				$- 6^7$					
23				6	$y_i^1 - a_{i6}$		- 2	-3					$- 5^8$					
24				8	$y_i^1 - a_{i8}$		- 2	-1					$- 3^{10}$					

Tableau 2 (cont.)

Row No.	s	J_s	z_s	N_s		i								C^s	D_s	E_s	F_s	H_s
						1	2	3	4	5	6	7	V_j^s					
25	2	9,3	6		$y_i^2 \rightarrow$	12	4	-2	1	0	5	0		9,3	∅	1,4,5, 6,7,10	∅	∅
26					$\sum_{j \in N_2} \bar{a}_{ij}$			-4										
27					$+\min(a_{ij} \geq 0)$			—										
28					$-\max(a_{ij} < 0)$			-2										
29				2	$y_i^2 - a_{i2}$			-2				- 4	$- 6^9$					
30				8	$y_i^2 - a_{i8}$								0^3					
31	3	9,3,8	7		$y_i^3 \rightarrow$	11	3	0	0	5	3	1						
32					$\sum_{j \in N_1^3} \bar{a}_{ij}$		0	-2		- 5								
33					$\sum_{j \in N_o^3} \bar{a}_{ij}$	-16	-10	-5		-14	-24	-20						
34					$+\min(a_{ij} \geq 0)$	-16	-10	-5		-14	-24	-20						
35					$-\max(a_{ij} < 0)$	- 8	- 5	-4		-12	-20	-19						

Tableau 2 (concl.)

Row No.	s	J_s	z_s	N_s		i							V_j^s	C^s	D_s	E_s	F_s	H_s
						1	2	3	4	5	6	7						
36	4	5	2		$y_i^4 \rightarrow$	- 2	4	-4	1	- 4	- 1	- 4		4,5,9	1,2	∅	∅	4,9
37					$\sum_{j \in N_4} \bar{a}_{ij}$	- 8		-3		-11	- 4	- 6						
38					$\sum_{j \in N_o^4} \bar{a}_{ij}$	-16	- 5	-5		-11	-12	-13						
39					$+\min(a_{ij} \geq 0)$	-16	- 5	-5		-11	-12	-13						
40					$-\max(a_{ij} < 0)$	- 8	0	-4		- 9	- 8	-12						
41	5	3	1		$y_i^5 \rightarrow$	6	4	-2	0	- 2	- 7	- 8		3,4, 5,9	1	∅		4,5,9
42					$\sum_{j \in N_5} \bar{a}_{ij}$			-4		- 9	-12	- 6						
43					$\sum_{j \in N_o^5} \bar{a}_{ij}$	- 8	0	-4		- 9	-12	-13						

$$\sum_{j \in N_o} \bar{a}_{3j} = \bar{a}_{32} + \bar{a}_{33} + \bar{a}_{38} = -2 - 1 - 2 = -5 < y_3^o = -3$$

$$\sum_{j \in N_o} \bar{a}_{5j} = \bar{a}_{53} + \bar{a}_{54} + \bar{a}_{56} + \bar{a}_{58} + \bar{a}_{59}$$

$$= -2 - 3 - 4 - 5 - 2 = -16 < y_5^o = -4$$

$$\sum_{j \in N_o} \bar{a}_{6j} = \bar{a}_{62} + \bar{a}_{64} + \bar{a}_{65} + \bar{a}_{69} + \bar{a}_{610}$$

$$= -8 - 6 - 6 - 12 - 4 = -36 < y_6^o = -7$$

$$\sum_{j \in N_o} \bar{a}_{7j} = \bar{a}_{71} + \bar{a}_{74} + \bar{a}_{75} + \bar{a}_{77} + \bar{a}_{78} + \bar{a}_{79}$$

$$= -7 - 6 - 1 - 5 - 1 - 8 = -28 < y_7^o = -5$$

As all relations hold, we are in case 3b.

3b.

$$\sum_{j \in N_o} \bar{a}_{1j} + 0 = -22 < y_1^o = -2$$

$$\sum_{j \in N_o} \bar{a}_{2j} + 0 = -10 < y_2^o = -1$$

$$\sum_{j \in N_o} \bar{a}_{3j} + 0 = -5 < y_3^o = -3$$

$$\sum_{j \in N_o} \bar{a}_{5j} + 0 = -16 < y_5^o = -4$$

$$\sum_{j \in N_o} \bar{a}_{6j} + 0 = -36 < y_6^o = -7$$

$$\sum_{j \in N_o} \bar{a}_{7j} + 0 = -28 < y_7^o = -5$$

So we are in case 3c.

3c. Check the relation $\sum_{j \ N_o} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i^o$

$$\sum_{j \in N_o} \bar{a}_{1j} - \bar{a}_{19} = -22 + 6 = -16 < y_1^o = -2$$

$$\sum_{j \in N_o} \bar{a}_{2j} - \bar{a}_{13} = -10 + 5 = -5 < y_2^o = -1$$

$$\sum_{j \in N_o} \bar{a}_{3j} - \bar{a}_{32} = -5 + 1 = -4 < y_3^o = -3$$

$$\sum_{j \in N_o} \bar{a}_{5j} - \bar{a}_{53} = -16 + 2 = -14 < y_5^o = -4$$

$$\sum_{j \in N_o} \bar{a}_{6j} - \bar{a}_{610} = -36 + 4 = -32 < y_6^o = -7$$

$$\sum_{j \in N_o} \bar{a}_{7j} - \bar{a}_{75} = -28 + 1 = -27 < y_7^o = -5$$

So we are in case 3c(i).

3c i, Now we have to calculate V_j^o for all $j \in N_o$. Computed values of V_j^o is shown in rows 5 through 14 of the tableau

2. Computation is shown on the next page.

$$J_{s+1} = J_1 = 9, \text{ that is } V^o_{J_{s+1}} = \max V^o_j = V^o_9$$

Therefore the variable chosen on the arc generated from node 0 is x_9. We assign a value 1 from $s_o = \{ 0, 1 \}$. Hence cancel V^o_9 from N_o and pass to stop 8.

Step 8. $J_1 = J_o \cup \{ 9 \} = \{ 9 \}$ $H_1 = \emptyset$, $T_1 = \{ 9 \}$

$z_1 = z_o + c_9 = 5$

The solution at node 1 is $U^1 = (u^1, X^-, Y^-)$

where $u^1 = (x_9)$

$X^1 = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_{10}) = 0$

$Y^1 = Y^o - A_9$

$y^1_1 = y^o_1 - a_{19} = 4$. $y^1_4 = 1 + 1 = 2$

$y^1_2 = y^o_2 - a_{29} = -1$, $y^1_5 = -4 + 2 = -2$

$y^1_3 = -3 - 0 = -3$, $y^1_6 = -7 + 12 = 5$

$y^1_7 = -5 + 8 = 3$

Iteration 2

Step 1. $y^1_i < 0$ for $i = 2, 3, 5$

So we are in case 1b.

Step 2. $N_1 = N - (O^1 \cup J_1 \cup E_1)$

$$\begin{bmatrix} 3 & -8 & -8 & 1 & 0 & 0 & 0 & 1 & -6 & 2 \\ 0 & 1 & -5 & 0 & -5 & 1 & 1 & 1 & 0 & 0 \\ 2 & -2 & -1 & 0 & 1 & 0 & 0 & -2 & 0 & 1 \\ -5 & 3 & 1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & -3 & 0 & -4 & 1 & -5 & -2 & 0 \\ 0 & -8 & 0 & -6 & -6 & 5 & 0 & 2 & -12 & -4 \\ 7 & 4 & 3 & -6 & -1 & 0 & -5 & -1 & -8 & 0 \end{bmatrix} \begin{bmatrix} -2 \\ -1 \\ -3 \\ 1 \\ -4 \\ -7 \\ -5 \end{bmatrix} = \begin{bmatrix} -5 & 6 & 6 & -3 & -2 & -2 & -2 & -3 & 4 & -4 \\ -1 & -2 & 4 & -1 & 4 & -2 & -2 & -2 & -1 & -1 \\ -5 & -1 & -2 & -3 & -4 & -3 & -3 & -1 & -3 & -4 \\ 6 & -2 & 0 & 1 & 1 & 1 & 0 & 0 & 2 & 2 \\ -4 & -4 & -2 & -1 & -4 & 0 & -5 & 1 & -2 & -4 \\ -7 & 1 & -7 & -1 & -1 & -12 & -7 & -9 & 5 & -3 \\ -12 & -9 & -8 & 1 & -4 & -5 & 0 & -4 & 3 & -5 \end{bmatrix}$$

$$V_j^o = \sum_{i \in M} \bar{b}_i^1 = \sum_{i \in M} (y_i^o - a_{ij})^- = \begin{array}{cccccccccc} -34 & -18 & -19 & -9 & -15 & -24 & -19 & -19 & -6 & -21 \end{array}$$

where $\sum_{i \in M} (y_i^o - a_{ij})^-$ = summation of negative $(y_i^o - a_{ij})$

It is only necessary to calculate the negative values of $(y_i^o - a_{ij})$. But for better understanding, all values are shown explicitly. As we are selecting the variable having max V_j^o, this will converge to a feasible solution very soon.

i, $$C^1 = U_p \mid J_p \subset J_1 \quad C_p^1$$

Since $J_o = \emptyset$, is strictly included in J_1 ; for $p = 0$, $C^1 = C_o^1 = \{9\}$.

ii, $N - C^1 = \{1, 2, 3, 4, 5, 6, 7, 8, 10\}$, $z^* = \infty$, $z_1 = 5$

$$D_1 = \{ j \mid j \in (N-C^1),\ 5 + c_j \geq \infty \}$$

Since there is no j satisfying the above relation, $D_1 = \emptyset$

iii, $N - (C^1 \cup D_1) = \{1, 2, 3, 4, 5, 6, 7, 8, 10\}$

$$E_1 = \{ j \mid j \in [N - (C^1 \cup D_1)],\ y_i < 0 \text{ and } a_{ij} \geq 0 \}$$

considering $i = 2, 3, 5$ (since for these i, $y_i < 0$) the set of those j satisfying the above relation is $E_1 = \{1, 7, 10\}$. Hence the set of improving vectors N_1 is

$$N_1 = N - (C^1 \cup D_1 \cup E_1) = \{2, 3, 4, 5, 6, 8\}$$

So we are in case 2b.

Step 3. Checking the relation

$$\sum_{j \in N_1} \bar{a}_{ij} \leq y_i^1 , \qquad \text{for } i = 2, 3, 5$$

we obtain

$$\sum_{j \in N_1} \bar{a}_{2j} = \bar{a}_{23} + \bar{a}_{25} = -5 - 5 = -10 < y_2^1 = -1$$

$$\sum_{j \in N_1} \bar{a}_{3j} = \bar{a}_{32} + \bar{a}_{33} + \bar{a}_{38} = -2 - 1 - 2 = -5 < y_3^1 = -3$$

$$\sum_{j \in N_1} \bar{a}_{5j} = \bar{a}_{53} + \bar{a}_{54} + \bar{a}_{56} + \bar{a}_{58} = -2 - 3 - 4 - 5$$
$$= -14 < y_5^1 = -2$$

So we are in case 3b.

3b. Check the relation

$$\sum_{j \in N_1} \bar{a}_{1j} + \min (a_{1j} \geq 0) \leq y_1^1$$

Since $\min a_{1j} = 0$, for $i = 2, 3, 5$, the above relation is satisfied. So we are in case 3c.

3c. Check the relation

$$\sum_{j \in N_1} \bar{a}_{1j} - \max (a_{1j} < 0) \leq y_1^1$$

$$\sum_{j \in N_1} \bar{a}_{2j} - \bar{a}_{23} = -10 + 5 = -5 < y_2^1 = -1$$

$$\sum_{j \in N_1} \bar{a}_{3j} - \bar{a}_{33} = -5 + 1 = -4 < y_3^1 = -3$$

$$\sum_{j \in N_1} \bar{a}_{5j} - \bar{a}_{53} = -14 + 2 = -12 < y_5^1 = -2$$

So we are in case 3c i.

3c i, $\quad V^1_{j_{s+1}} = V^1_{j_2} = V^1_3 = -2$

Computed values of V^1_j are shown in rows 19 through 24 of tableau 2. Computations are shown on the next page, so cancel V^1_3 from N_1 and pass to step 8.

Step 8. $\quad J_2 = J_1 \cup \{3\} = \{9,3\}$

$$H_2 = \emptyset\ , \quad T_2 = \{9,3\}\ , \quad z_2 = z_1 + c_3 = 6$$

The solution at node 2 is $U^2 = (u^2, X^2, Y^2)$

$$u^2 = (x_9, x_3) = 1$$

$$X^2 = (x_1, x_2, x_4, x_5, x_6, x_7, x_8, x_{10}) = 0$$

$$Y^2 = Y^1 - A_3$$

$$y^2_1 = y^1_1 - a_{13} = 4 + 8 = 12$$

$$y^2_2 = -1 + 5 = 4\ , \qquad y^2_5 = -2 + 2 = 0$$

$$y^2_3 = -3 + 1 = -2, \qquad y^2_6 = 5 - 0 = 5$$

$$y^2_4 = 2 - 1 = 1\ , \qquad y^2_7 = 3 - 3 = 0$$

Iteration 3

Step 1. $\quad y^2_i < 0 \quad$ for $i = 3$. So we are in 1b.

Step 2. $\quad N_2 = N - (C^2 \cup D_2 \cup E_2)$

i, $\quad C^2 = U_p \mid J_p \subset J_2 \quad C^2_p$

2	3	4	5	6	8		2	3	4	5	6	8
-8	-8	1	0	0	1	4	12	12	3	4	4	3
1	-5	0	-5	1	1	-1	-2	4	-1	4	-2	-2
-2	-1	0	1	0	-2	-3	-1	-2	-3	-4	-3	-1
3	1	0	0	0	1	2	= -1	1	2	2	2	1
0	-2	-3	0	-4	-5	-2	-2	0	1	-2	2	3
-8	0	-6	-6	5	2	5	13	5	11	11	0	3
4	3	-6	-1	0	-1	3	-1	0	9	4	3	4
							—	—	—	—	—	—
							-7	-2	-4	-6	-5	-3

$$v_j^1 = \sum_{i \in M} b_i^2 = \sum_{i \in M} (y_i^1 - a_{ij})^- = -7 \quad -2 \quad -4 \quad -6 \quad -5 \quad -3$$

$$v_{j_{s+1}}^1 = v_{j_2}^1 = \max_{j \in N_1} v_j^1 = v_3^1 = -2$$

Since the nodes 0 and 1 are on the same chain as the chain from 0 and 2 , $J_o \subset J_2$, $J_1 \subset J_2$.

Hence $C^2 = C_o^1 \quad C_1^2 = \{9,3\}$

ii, $N - C^2 = \{1, 2, 4, 5, 6, 7, 8, 10\}$

$z^* = \infty$, $z_2 = 6$

$D_2 = \{ j \mid j \in (N - C^2) , \quad 6 + c_j \geq \infty \}$

Since there is no j satisfying the above relation, $D_2 = \emptyset$

iii, $N - (C^2 \cup D_2) = \{1, 2, 4, 5, 6, 7, 8, 10\}$

$E_2 = \{ j \mid j \in [N - (C^2 \cup D_2)], \ y_i < 0 \text{ and } a_{ij} \geq 0 \}$

Considering $i = 3$, for which $y_i < 0$

$a_{ij} > 0$ for $j = 1, 4, 5, 6, 7, 10$

Therefore $E_2 = \{1, 4, 5, 6, 7, 10\}$

Hence $N_2 = N - (C^2 \cup D_2 \cup E_2) = \{2,8\}$ So we are in 2b.

Step 3. Checking the relation $\sum_{j \in N_2} \bar{a}_{ij} \leq y_i^2$

we get $\sum_{j \in N_2} \bar{a}_{3j} = -2 - 2 = -4 < y_3^2 = -2$

So we are in 3b.

3b. Checking the relation

$$\sum_{j \in N_2} \bar{a}_{ij} + \min (a_{ij} \geq 0) \leq y_i^2$$

Since $a_{ij} < 0$ for $j = 2,8$ the above relation is satisfied. So we are in 3c.

3c. Checking the relation

$$\sum_{j \in N_2} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i^2 \text{ , we obtain}$$

$$\sum_{j \in N_2} \bar{a}_{3j} - \bar{a}_{32} = -4 + 2 = -2 = y_i^2 = -2$$

So we are in case 3c(i).

3c i, Calculate V_j^2 for $i = 2,8$

-8	1	12	20	11
1	1	4	3	3
-2	-2	-2	0	0
3	1	1	-2	0
0	-5	0	0	5
-8	2	5	13	3
4	-1	0	-4	1
			—	—
		V_j^2 =	-6	0

Note that for $j = 8$, $(y_i^2 - a_{i8})$ is non-negative for all $i \in M$. Hence $V_j^2 = 0$. Also $\max_{j \in N_2} V_j^2 = V_8^2 = 0$. So cancel V_8^2 from

N_2 and pass to step 8.

Step 8. $J_3 = J_2 \cup \{8\} = \{9,3,8\}$

$H_3 = \emptyset$, $T_3 = \{9,3,8\}$,

$z_3 = z_2 + c_8 = 6 + 1 = 7$

The solution at node 3 is $U^3 = (u^3, X^3, Y^3)$

where

$u^3 = (x_9, x_3, x_8) = 1$

$X^3 = (x_1, x_2, x_4, x_5, x_6, x_7, x_{10}) = 0$

$Y^3 = Y^2 - A_8$

$y_1^3 = y_1^2 - a_{18} = 12 - 1 = 11$

$y_2^3 = 4 - 1 = 3$, $y_5^3 = 0 + 5 = 5$

$y_3^3 = -2 + 2 = 0$, $y_6^3 = 5 - 2 = 3$

$y_4^3 = 1 - 1 = 0$, $y_7^3 = 0 + 1 = 1$

Iteration 4

Step 1. $y_i^3 \geq 0$ for all $i \in M$. So we are in case 1a.

Set $z_3 = 7 = Z^*$ Hence $N_3 = \emptyset$. Node 3 is closed. Hence it is necessary to form sets D_k^3 for $k = 2, 1, 0$

$$D_k^3 = \{ j \mid j \in (N_k - C_k^3),\ z_k + c_j \geq Z^* \}$$

$i,\ k = 2,$ $z_2 = 6$, $N_2 = \{2,8\}$, $C_2^3 = \{8\}$

$N_2 - C_2^3 = \{2\}$

$$D_2^3 = \{ j \mid j \in (N_2 - C_2^3), \quad 6 + c_j \geq Z^* \}$$

For $j = 2$, $c_2 = 5$, $\quad 6 + 5 = 11 \succ Z^* = 7$

Hence $D_2^3 = 2$

ii, $N_1 = \{ 2, 3, 4, 5, 6, 8 \}$, $C_1^3 = \{ 3 \}$

$N_1 - C_1^3 = 2, 4, 5, 6, 8$, $z_1 = 5$

$$D_1^3 = \{ j \mid j \in (N - C_1^3), 5 + c_j \geq Z^* \}$$

$5 + c_2 = 5 + 5 \succ Z^* = 7$, $\quad 5 + c_6 = 5 + 4 \succ Z^* = 7$

$5 + c_4 = 5 + 7 \succ Z^* = 7$, $\quad 5 + c_8 = 5 + 1 \not\succ Z^* = 7$

$5 + c_5 = 5 + 2 = 7 \ (Z^*)$

Hence the above relation is satisfied for $j = 2, 4, 5, 6$

$$D_1^3 = \{ 2, 4, 5, 6 \}$$

iii, $N_o = \{ 1, 2, 3, \ldots, 10 \}$, $C_o^3 = \{ 9 \}$

$N_o - C_o^3 = \{ 1, 2, 3, 4, 5, 6, 7, 8, 10 \}$

$z_o = 0$, $\quad Z^* = 7$

$$D_o^3 = \{ j \mid j \in (N_o - C_o^2), z_o + c_j = c_j \geq Z^* \}$$

The above relation is satisfied for $j = 4$. Hence $D_o^3 = \{ 4 \}$

Now we cancel all V_o^3, V_1^3, and V_2^3 from D_o^3, D_1^3 and D_2^3, respectively. We pass to step 5.

Step 5. We form the set of improving vectors N_k^3 for the solution U^k (at node k) for $k = 2$

$$N_2^3 = N_2 - (C_2^3 \cup D_2^3) = \emptyset \quad \text{(Since all } V_j^3 \text{ are cancelled)}$$

Now let us find N_k^3 for $k = 1$

$$N_1^3 = N_1 - (C_1^3 \cup D_1^3) = \{8\}$$

So we are in case 5b.

Step 6. We check the relation

$$\sum_{j \in N_1^3} \bar{a}_{ij} \leq y_i^2 \qquad (i \mid y_i < 0) \text{ for } i = 2, 3, 5$$

$$\sum_{j \in N_1^3} \bar{a}_{2j} = 1 \not< y_2^1 = -1$$

$$\sum_{j \in N_1^3} \bar{a}_{3j} = \bar{a}_{38} = -2 \not< y_3^1 = -3$$

$$\sum_{j \in N_1^3} \bar{a}_{5j} = \bar{a}_{58} = -5 < y_5^1 = -2$$

Since the above relation does not hold for $i = 2$ and 3, the node 1 is closed. Hence cancel $V_j^{k_1}$ (V_j^1) from N_1 for $j \in N_1^3$ i.e., cancel V_8^1 from N_1. Now we pass to step 5.

Step 5. $k = 0$. Now we have to check the set of improving vectors N_o^3 for the node 0 or solution U^o.

$$N_o^3 = N_o - (C_o^3 \cup D_o^3) = \{1, 2, 3, 5, 6, 7, 8, 10\}$$

So we are again in case 5b.

Step 6. We check the relations

$$\sum_{j \in N_o^3} \bar{a}_{1j} \leq y_1^o \qquad \text{for } i = 1, 2, 3, 5, 6, 7 .$$

$$N_o^3 = \{1, 2, 3, 5, 6, 7, 8, 10\}$$

$$\sum_{j \in N_o^3} \bar{a}_{1j} = \bar{a}_{12} + \bar{a}_{13} = -8 - 8 = -16 < y_1^o = -2$$

$$\sum_{j \in N_o^3} \bar{a}_{2j} = \bar{a}_{23} + \bar{a}_{25} = -5 - 5 = -10 < y_2^o = -1$$

$$\sum_{j \in N_o^3} \bar{a}_{3j} = \bar{a}_{32} + \bar{a}_{33} + \bar{a}_{38} = -2 - 1 - 2 = -5 < y_3^o = -3$$

$$\sum_{j \in N_o^3} \bar{a}_{5j} = \bar{a}_{53} + \bar{a}_{54} + \bar{a}_{56} + \bar{a}_{58}$$

$$= -2 - 3 - 4 - 5 = -14 < y_5^o = -4$$

$$\sum_{j \in N_o^3} \bar{a}_{6j} = \bar{a}_{62} + \bar{a}_{64} + \bar{a}_{65} + \bar{a}_{610}$$

$$= -8 - 6 - 6 - 4 = -24 < y^o = -7$$

$$\sum_{j \in N_o^3} \bar{a}_{7j} = \bar{a}_{71} + \bar{a}_{74} + \bar{a}_{75} + \bar{a}_{77} + \bar{a}_{78}$$

$$= -7 - 6 - 1 - 5 - 1 = -20 < y_7^o = -5$$

So we are in case 6b.

6b. Check the relation

$$\sum_{j \in N_o^3} \bar{a}_{ij} + \min (a_{ij} \geq 0) \leq y_i^o$$

Since $\min a_{ij} = 0$ for all $i \in M$, the above relation is automatically satisfied. So we are in case 6c.

6c. Check the relation

$$\sum_{j \in N_o^3} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i^o \quad (i \mid y_i < 0)$$

$$\sum_{j \in N_o^3} \bar{a}_{1j} - \bar{a}_{12} = -16 + 8 = -8 < y_1^o = -2$$

$$\sum_{j \in N_o^3} \bar{a}_{2j} - \bar{a}_{23} = -10 + 5 = -5 < y_2^o = -1$$

$$\sum_{j \in N_o^3} \bar{a}_{3j} - \bar{a}_{33} = -5 + 1 = -4 < y_3^o = -3$$

$$\sum_{j \in N_o^3} \bar{a}_{5j} - \bar{a}_{53} = -14 + 2 = -12 < y_5^o = -4$$

$$\sum_{j \in N_o^3} \bar{a}_{6j} - \bar{a}_{610} = -24 + 4 = -20 < y_6^o = -7$$

$$\sum_{j \in N_o^3} \bar{a}_{7j} - \bar{a}_{75} = -20 + 1 = -19 < y_7^o = -5$$

So we are in case 6c(i).

6c 1, See the tableau 2, rows 5 through 14

$$V_{j_{s+1}}^{k_2} = \max_{j \in N_o^3} V_j^{k_2} ; \quad k_2 = 0, \; s = 3$$

$$V_{j_3}^{o} = \max_{j \in N_o^3} V_j^{o} = V_5^{o} = -15$$

Therefore x_5 enters the solution. Hence cancel V_5^o from N_o and pass to step 8.

Step 8. $J_4 = J_o \cup j_{s+1} = \{ 5 \}$

$H_4 = \{ 4, 9 \}$, $T_4 = \{4, 5, 9\}$; $p = k_2 = 0$

$z_4 = z_p + c_{j_{s+1}} = 0 + 2 = 2$,

The solution at node 4 is $U^4 = (u^4, X^4, Y^4)$ where

$u^4 = (x_4 = x_9 = 0, \; x_5 = 1)$

$X^4 = (x_1, x_2, x_3, x_6, x_7, x_8, x_{10}) = 0$

$Y^4 = Y^o - A_5$

$y_1^4 = y_1^o - a_{15} = -2 - 0 = -2$

$y_2^4 = -1 + 5 = 4$, $y_5^4 = -4 - 0 = -4$

$y_3^4 = -3 - 1 = -4$, $y_6^4 = -7 + 6 = -1$

$y_4^4 = 1 - 0 = 1$, $y_7^4 = -5 + 1 = -4$

Iteration 5

Step 1. $y_i^4 < 0$, for $i = 1, 3, 5, 6, 7$

So we are in 1b.

Step 2. $N_4 = N - (C^4 \cup D_4 \cup E_4)$

i, $C^4 = \cup_p \mid J_p \subset J_4 \; C_p^4$, for $p = 3 \quad J_3 \not\subset J_4$

$p = 2 \quad J_2 \not\subset J_4$

$p = 1 \quad J_1 \not\subset J_4$

$p = 0 \quad J_0 \subset J_4$

$C^4 = C_0^4 = \{4, 5, 9\}$

ii, $N - C^4 = \{1, 2, 3, 6, 7, 8, 10\}$; $Z^* = 7$, $z_4 = 2$

$D^4 = \{ j \mid j \in (N - C^4),\ z_4 + c_j = 2 + c_j \geq Z^* = 7\}$

$2 + c_1 = 2 + 6 = 8 > 7$

$2 + c_2 = 2 + 5 = 7 = 7$, $2 + c_7 = 2 + 3 = 5 \not> 7$

$2 + c_3 = 2 + 1 = 3 \not> 7$, $2 + c_8 = 2 + 1 = 3 \not> 7$

$2 + c_6 = 2 + 4 = 6 \not> 7$, $2 + c_{10} = 2 + 3 = 5 \not> 7$

The above relation is satisfied for $j = 1, 2$

Hence $D_4 = \{1,2\}$

iii, $N - (C^4 \cup D_4) = \{3, 6, 7, 8, 10\}$

$E_4 = \{ j \mid j \in [N - (C^4 \cup D_4)],\ y_i < 0 \text{ and } a_{ij} \geq 0\}$

Since there is no j satisfying the above relation (that is, there

is no $A_j \succ 0$, for $i = 1, 3, 5, 6, 7)$, $\quad E_4 = \emptyset$

Hence

$$N_4 = N - (C^4 \cup D_4 \cup E_4) = \{3, 6, 7, 8, 10\}$$

So we are in 2b.

Step 3. We check the relation

$$\sum_{j \in N_4} \bar{a}_{ij} \leq y_i^4 \qquad \text{for } i = 1, 3, 5, 6, 7 .$$

$$\sum_{j \in N_4} \bar{a}_{1j} = \bar{a}_{13} = -8 \prec y_1^4 = -2$$

$$\sum_{j \in N_4} \bar{a}_{3j} = \bar{a}_{33} + \bar{a}_{38} = -1 - 2 = -3 \not\prec y_3^4 = -4$$

$$\sum_{j \in N_4} \bar{a}_{5j} = \bar{a}_{53} + \bar{a}_{56} + \bar{a}_{58} = -2 - 4 - 5 = -11 \prec y_5^4 = -4$$

$$\sum_{j \in N_4} \bar{a}_{6j} = \bar{a}_{610} = -4 \prec y_6^1 = -1$$

$$\sum_{j \in N_4} \bar{a}_{7j} = \bar{a}_{77} + \bar{a}_{78} = -5 - 1 = -6 \prec y_7^4 = -4$$

Since for $i = 3$, the above relation does not hold, we pass to step 5.

Step 5. The node $(4 + 1)$ is closed. So we backtrack to

$$k \prec 4 \qquad \text{and} \qquad J_k \subset J_4$$

Only for $k = 0$, $J_k \subset J_4$. So let us find the set of improving vectors N_o^4

$$N_o^4 = N_o - (C_o^4 \cup D_o^4)$$

$$C_o^4 = \{4, 5, 9\} \quad ; \quad Z^* = 7$$

$$D_o^4 = \{ j \mid j \in (N_o - C_o^4),\ z_o + c_j = c_j \geq Z^* \}$$

Since there is no j satisfying this relation, $D_o^4 = \emptyset$

$$N_o^4 = N_o - C_o^4 = \{1, 2, 3, 6, 7, 8, 10\}$$

Step 6. We check the relation

$$\sum_{j \in N_o^4} \bar{a}_{ij} \leq y_i^o \qquad \text{for } i = 1, 2, 3, 5, 6, 7 .$$

This relation is satisfied for $i = 1, 2, 3, 5, 6, 7$. Hence we are in case 6b.

6b. Check the relation

$$\sum_{j \in N_o^4} \bar{a}_{ij} + \min(a_{ij} \geq 0) \leq y_i^o \qquad (i \mid y_i < 0)$$

Since $\min a_{ij} = 0$, for all $i \mid y_i < 0$, there is no need to check the relation above. Hence we are in case 6c.

6c. Check the relation

$$\sum_{j \in N_o^4} \bar{a}_{ij} - \max (a_{ij} < 0) \leq y_i \qquad (i \mid y_i < 0)$$

$$\sum_{j \in N_o^4} \bar{a}_{1j} - \bar{a}_{12} = -16 + 8 = -8 < y_1^o = -2$$

$$\sum_{j \in N_o^4} \bar{a}_{2j} - \bar{a}_{23} = -5 + 5 = 0 \not< y_2^o = -1$$

Since the above relation is not satisfied for i = 2, we are in case 6c(ii).

Step 7. We form the set F_o^4 as defined by

$$F_o^4 = (j \mid j \in N_o^4, a_{ij} < 0)$$

$$= \{3\} \qquad \text{for } i = 2 .$$

Before introducing x_3 into the solution, we have to check

$$z_{s+1} \geq Z^*. \quad \text{But} \quad z_o + c_3 = 1 \not\geq Z^* .$$

Therefore x_3 enters the solution. Cancel V_o^3 from N_o. Now

$$z_5 = z_o + c_3 = 1$$

$$J_5 = J_o \cup \{3\} = \{3\}, \qquad H_5 = \{4, 5, 9\}$$

$$T_5 = \{3, 4, 5, 9\}$$

The solution at node 5 is $U^5 = (u^5, X^5, Y^5)$

where

$$u^5 = (x_3 = 1,\ x_4 = x_5 = x_9 = 0)$$

$$X^5 = (x_1, x_2, x_6, x_7, x_8, x_{10}) = 0$$

$$Y^5 = Y^o - A_3$$

$$y_1^5 = y_1^o - a_{13} = -2 + 8 = 6$$

$$y_2^5 = -1 + 5 = 4\ , \qquad y_5^5 = -4 + 2 = -2$$

$$y_3^5 = -3 + 1 = -2, \qquad y_6^5 = -7 - 0 = -7$$

$$y_4^5 = 1 - 1 = 0\ , \qquad y_7^5 = -5 - 3 = -8$$

Iteration 6

Step 1. $y_i < 0$, for $i = 3, 5, 6, 7$.

So we are in case 1b.

Step 2. $N_5 = N - (C^5 \cup D_5 \cup E_5)$

i, $C^5 = \cup_p \mid J_p \subset J_5\ C_p^5$; only for $p = 0$, $J_p \subset J_5$

Hence $C^5 = C_o^5 = \{3, 4, 5, 9\}$

ii, $N - C^5 = \{1, 2, 6, 7, 8, 10\}$

$$D_5 = \{ j \mid j \in (N - C^5),\ z_5 + c_j = 1 + c_j \geq Z^* = 7 \}$$

This relation is satisfied for $j = 1$. Hence $D_5 = \{1\}$.

iii, $N - (C^5 \cup D_5) = \{2, 6, 7, 8, 10\}$

$$E_5 = \{ j \mid j \in [N - (C^5 \cup D_5)], y_i < 0 \text{ and } a_{ij} \geq 0 \}$$

Since there is no j satisfying this relation, $E_5 = \emptyset$.

Hence $N_5 = \{ 2, 6, 7, 8, 10 \}$. So we are in case 2b.

Step 3. Check the relation

$$\sum_{j \in N_5} \bar{a}_{ij} \leq y_i^5 \qquad \text{for} \quad i = 3, 5, 6, 7 .$$

$$\sum_{j \in N_5} \bar{a}_{3j} = \bar{a}_{32} + \bar{a}_{38} = -2 - 2 = -4 < y_3^5 = -2$$

$$\sum_{j \in N_5} \bar{a}_{5j} = \bar{a}_{56} + \bar{a}_{58} = -4 - 5 = -9 < y_6^5 = -2$$

$$\sum_{j \in N_5} \bar{a}_{6j} = \bar{a}_{62} + \bar{a}_{610} = -8 - 4 = -12 < y_6^5 = -7$$

$$\sum_{j \in N_5} \bar{a}_{7j} = \bar{a}_{77} + \bar{a}_{78} = -5 - 1 = -6 \not< y_7^5 = -8$$

The above relation is not satisfied for $i = 7$. Hence we pass to step 5.

Step 5. The node $(5 + 1)$ is closed. So we backtrack to another node $k < 5$ and $J_k \subset J_5$. Only for $k = 0$, $J_o \subset J_5$ So we form sets N_o^5 from the relation

$$N_o^5 = N_o - (C_o^5 \cup D_o^5)$$

$$C_o^5 = \{3, 4, 5, 9\} \; ; \qquad N - C_o^5 = \{1, 2, 6, 7, 8, 10\}$$

$$D_o^5 = \{ j \mid j \in (N - C_o^5) \; , \quad z_o + c_j - c_j \geq Z^* \}$$

Since there is no j satisfying this relation, $D_o^5 = \emptyset$.

Hence $N_o^5 = \{1, 2, 6, 7, 8, 10\}$. We are in case 5b.

Step 6. Check the relation

$$\sum_{j \in N_o^5} a_{ij} \leq y_i^o \qquad (i \mid y_i^o < 0)$$

$$\sum_{j \in N_o^5} \bar{a}_{1j} = \bar{a}_{12} = -8 < y_1^o = -2$$

$$\sum_{j \in N_o^5} \bar{a}_{2j} = 0 \not< y_2^o = -1$$

The above relation is not satisfied for i = 2, also since the above relation does not hold for any k such that $N_k^5 = \emptyset$ $(k < 5)$, the algorithm has come to an end.

Hence $z_s = z_3 = Z^* = 7$, for the solution U^3 .

Therefore the optimum solution is U^3 and

$$x_j = \begin{cases} 1 \; , & j = 3, 8, 9 \\ 0 \; , & j = 1, 2, 4, 5, 6, 7, 10 \end{cases}$$

$$y_1^3 = 11 \; , \qquad y_4^3 = 0 \; ,$$

$$y_2^3 = 3\,, \qquad y_5^3 = 5\,,$$

$$y_3^3 = 0\,, \qquad y_6^3 = 3\,, \qquad y_7^3 = 1$$

This is optimal for problem 'P'. The solution tree for this problem is shown in Fig. 4.

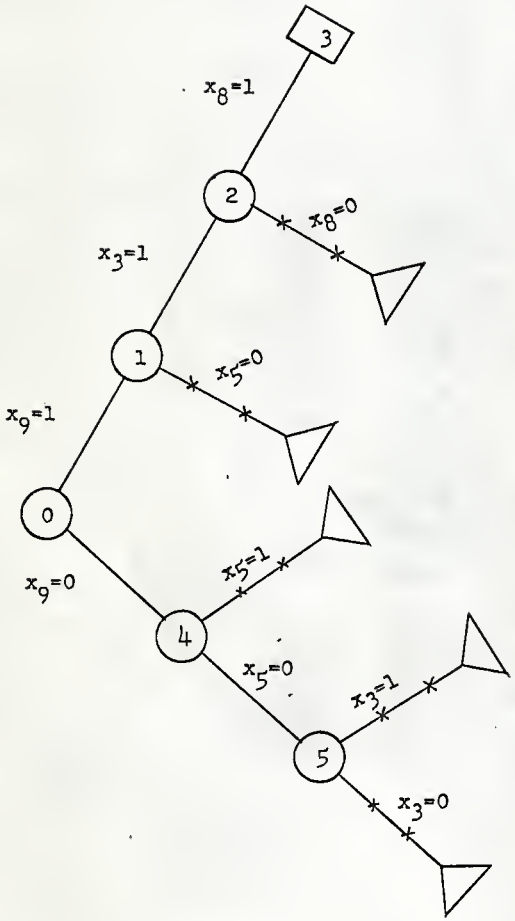

Fig. 4. Solution tree of the example problem.

ZERO-ONE INTEGER PROGRAMMING

by

RATNAM VENKATA CHITTURI

B. S.(Mech. Engg), Andhra University
Waltair, India, 1964

AN ABSTRACT OF A MASTER'S REPORT

submitted in the partial fulfillment of

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1967

Linear programming problems can be solved by using either the standard simplex method or the dual simplex method. However these methods do not yield optimal solutions to problems, where integer solutions are desired. Hence it is necessary to solve these integer programming problems by using different techniques such as Gomory's cutting plane method. There is also a special class of integer programming problems which require zero-one integer solutions. Gomory's cutting plane methods can be used to solve these zero-one problems, but they are inefficient. Other methods for solving these problems utilize the special structure of the zero-one problems. This paper investigates the various approaches that were developed to solve zero-one problems which are divided into three different categories as follows:

1. Cutting plane methods.
2. Parallel shifts of the objective function hyperplane.
3. Combinatorial methods.

A brief survey of each of the three approaches is presented in this paper. Most of the discussion is devoted to the combinatorial methods which the author believes are most efficient. Gomory's cutting plane method is presented along with Elmaghraby's method which falls into the second category. It is interesting to note that both these methods use additional constraints to cut the solution space W in order to exclude as many of the non-integer solutions as possible, but not any of the integer solutions are excluded. The first approach generates the additional

constraint from one of the problem constraint, where as the second approach generates it from the objective function. In Gomory's cutting plane method, the problem size increases as the constraints of the form $x_j \leq 1$, $j \in N$ are added to the original problem and consequently the computational time increases rapidly. However this difficulty is overcome in Elmaghraby's method using the upperbound technique.

A general combinatorial approach is presented in later sections followed by a specific combinatorial algorithm developed by Balas. This approach seems to yield very good results. However its efficiency mainly depends on the tests being applied to exclude the non-feasible solutions. Freeman [6] modified Balas algorithm [1] to include some of the tests developed by Glover [8] and reported very good results when there are less than 30 variables. But it seems to be somewhat less efficient with more than 30 variables. In conclusion, more research is needed on zero-one integer programming algorithms since none are well suited for solving large practical problems.