To the Graduate Council:

I am submitting herewith a thesis written by David F. Hassell entitled "Methods for Solution to the Integer Linear Programming Problem." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Yueh-er Kuo

Yueh-er Kuo, Major Professor

We have read this thesis and recommend its acceptance:

Robert Davenport

Torsti P. Salo

Accepted for the Council:

Hilton A. Smith

Vice Chancellor
Graduate Studies and Research

METHODS FOR SOLUTION TO THE INTEGER LINEAR PROGRAMMING PROBLEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

David F. Hassell

June 1976

1286580

ACKNOWLEDGEMENTS

The author would like to express his appreciation to Dr. Yueh-er Kuo for her assistance, which has made this research project a valuable learning experience.

The author is especially grateful to his wife, Susanne, for her constant encouragement during the preparation of this thesis.

ABSTRACT

The purpose of this thesis is to consider methods for the solution to the integer constrained linear programming problem (ILP). To be considered are the method of reducing the feasible solution set by introduction of cutting plane constraints, and the method of reducing the ILP to a knapsack formulation.

Chapter I introduces the ILP and the theoretical basis for the cutting plane approach, along with a method for choosing the cutting plane. The all-integer algorithm with its special choice of cutting planes is then presented in Chapter II.

The knapsack problem, a special case of the ILP, is discussed in Chapter III. Following in Chapter IV is a proof that any ILP in bounded variables can be reduced to a knapsack problem in bounded variables. Finally, an algorithm is presented for solution of the knapsack problem in bounded variables.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	A CUTTING PLANE METHOD FOR SOLUTION TO THE ILP	1
	The Statement and Interpretation of the ILP	1
	The Cutting Plane Solution to the ILP	4
	A Method for Choosing the Cutting Plane	11
II.	THE ALL-INTEGER ALGORITHM	21
	The Algorithm	21
	Adapting the Algorithm for Machine Computation	32
III.	THE KNAPSACK PROBLEM	35
	An Algorithm Using Recursive Equations	36
	Adapting the Knapsack Algorithm for Machine Computation .	40
IV.	TRANSFORMATION OF AN ILP IN BOUNDED VARIABLES TO A KNAPSACK PROBLEM IN BOUNDED VARIABLES	44
	Derivation of an Equivalent Knapsack Constraint from any Set of ILP Constraints in Bounded Variables	45
	Algorithm for Solving the Knapsack Problem in Bounded Variables	51
BIBLIOGRAPHY		57
APPENDIX		59
VITA		62

LIST OF FIGURES

FIGURE		PAGE
1.1.	Graphical Interpretation Revealing the Solution of the LP (1.5)	5
1.2.	Reduction of the Feasible Solution Set by Introducing Cutting Planes	10
1.3.	Graphical Interpretation Revealing the Solution to the LP Corresponding to (1.18)	16
1.4.	Graphical Interpretation of the Reduction of the Feasible Solution to (1.18), Exposing the Integer Solution . . .	20
2.1.	Flowchart for All-Integer Algorithm	33
3.1.	Flowchart for Knapsack Algorithm	42

CHAPTER I

A CUTTING PLANE METHOD FOR SOLUTION TO THE ILP

Since the advent of the simplex method for solution to the Linear Programming Problem (LP) in 1951 by George Dantzig [4] there has been a mushrooming interest in the mathematical programming problem (i.e. optimization of a function subject to a set of constraints). Applications for the mathematical programming problem often occur in industrial scheduling and production problems; these have led to the study of solution to the problem when special conditions are placed on the optimal function or on the set of constraints. For example, as the name implies, the LP must meet the special condition that both the optimal function and the constraints be linear. To be considered in these pages is the Integer Linear Programming Problem (ILP), which, in addition to the conditions of linearity, is restricted to an all integer solution.

I. THE STATEMENT AND INTERPRETATION OF THE ILP

Before proceeding, it is necessary to establish some notation. The ILP will be stated in the following form:

$$\max z = c_1x_1 + c_2x_2 + \ldots + c_px_p$$

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p &\leq b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2p}x_p &\leq b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mp}x_p &\leq b_m
\end{aligned} \tag{1.1}$$

$$x_i \geq 0,\ x_i \text{ integer for } i = 1,2,3, \ldots, p$$

Or, after the addition of slack variables to change the constraints to equalities, we have the following, where $n = p + m$.

$$\max z = c_1x_1 + c_2x_2 + \ldots + c_px_p + 0x_{p+1} + \ldots + 0x_n$$

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p + x_{p+1} + 0x_{p+2} + \ldots + 0x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2p}x_p + 0x_{p+1} + x_{p+2} + \ldots + 0x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mp}x_p + 0x_{p+1} + 0x_{p+2} + \ldots + x_n &= b_n
\end{aligned} \tag{1.2}$$

$$x_i \geq 0,\ x_i \text{ integer for } i = 1,2, \ldots, n$$

We will usually refer to (1.2) in the more brief notation (1.3).

$$\begin{aligned}
&\max z = c^Tx \\
&Ax = b \\
&x_i \geq 0,\ x_i \text{ integer for } i = 1,2, \ldots, n
\end{aligned} \tag{1.3}$$

In this formulation (1.3), z represents the optimal function, x is an n-dimension column vector, c^T (equal to c transpose) is an n-dimension row vector representing the coefficients of the optimal function, b is an m-dimension column solution vector for the set of constraints and A is an $m \times n$ coefficient matrix with columns a_i and elements a_{ij} (for simplicity $m \leq p$). The optimal function here is represented as maximal, but the minimum and maximum case are interchangeable by the relation $\min(z) = \max(-z)$.

Now, let us take a brief look at how the LP corresponding to the ILP can be represented graphically. The feasible solution to an LP is always a convex set C, since it is composed of the intersection of finitely many convex constraints. The optimal function can be represented graphically as a family of parallel hyperplanes in the same space as C. An optimal value z* occurs with infinitely many solutions, when the hyperplane z is coincident with a face of the feasible set C. If a unique solution x* exists for the problem, it occurs when the hyperplane intersects C in one and only one point (i.e. an extreme point of C). The latter case is of primary consideration.

The simplex method can be described graphically as a technique which evaluates z at the extrema of C until an optimal value for z is determined (the extreme point then is x*). (The theoretical and computational development of the simplex method can be found in Dantzig's paper [4] or in any introductory linear programming text.) Now, with respect to the ILP, an interesting and perplexing complication occurs. It may at first seem that one could solve the corresponding LP by the simplex method and

then round this solution to the "nearest" integer vector, which is an element of C. Let us demonstrate the fallacy of this assumption by an example, which will also serve as an example of the graphical interpretation of the ILP.

II. THE CUTTING PLANE SOLUTION TO THE ILP

As in most of Linear Algebra, a two dimensional example can provide a valid illustration of the n-dimensional case. Consider the following example.

$$\begin{aligned} &\max \ z = x_1 + 2x_2 \\ &5x_1 + 9x_2 \leq 15 \\ &-13x_1 + 9x_2 \leq 9 \\ &x_i \geq 0, \ x_i \text{ integer} \qquad \text{for } i = 1,2 \end{aligned} \tag{1.4}$$

For this example, the solution to the corresponding LP can easily be seen graphically (Figure 1.1). The feasible set of solutions is the shaded region, and the unique solution occurs where $z^* = \frac{89}{27}$ intersects the extreme point of C created where the constraint equalities intersect in the first quadrant.

Let us solve the LP corresponding to (1.4) by the simplex method. We first must write the LP in standard form with slack variables.

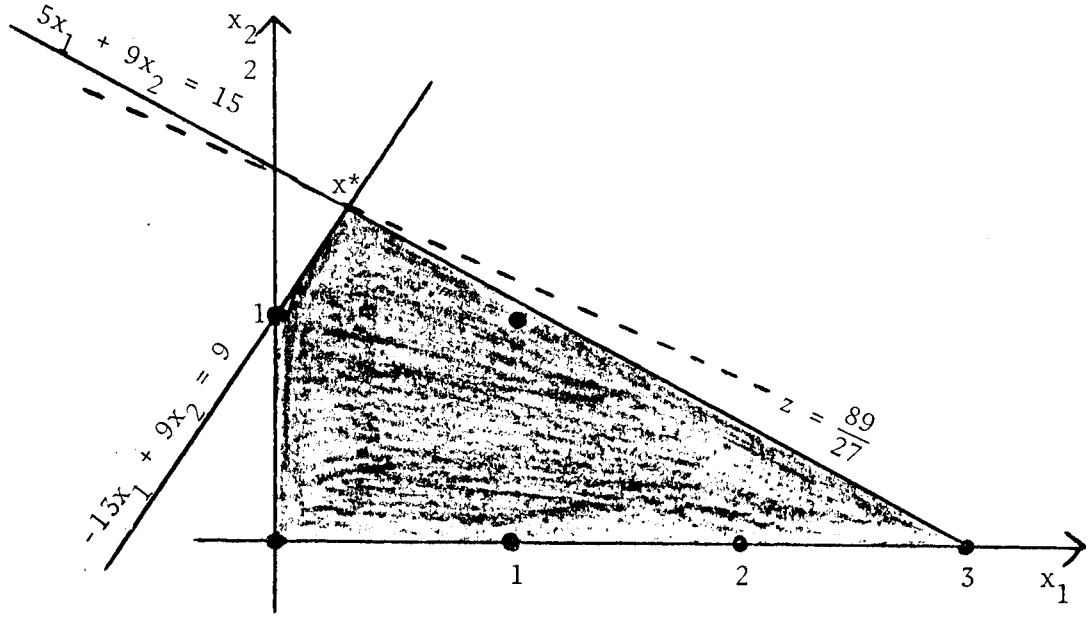


Figure 1.1. Graphical interpretation revealing the solution of the LP (1.5).

$$\max z = x_1 + 2x_2 + 0x_3 + 0x_4$$

$$\begin{aligned} 5x_1 + 9x_2 + x_3 + 0x_4 &= 15 \\ -13x_1 + 9x_2 + 0x_3 + x_4 &= 9 \end{aligned} \qquad (1.5)$$

$$x_i \geq 0 \qquad \text{for } i = 1,2,3,4$$

The initial simplex tableau will take the following general form:

Tableau 1.1

			$c_1 \ . \ . \ . \ c_p$			$c_{p+1} \ . \ . \ . \ c_n$	
i	Basis	c_B	P_o	$P_1 \ . \ . \ . \ P_p$		$P_{p+1} \ . \ . \ . \ P_n$	
1	a_{p+1}	c_{p+1}	x_{10}	x_{11}	x_{1p}	1 .	0
2	a_{p+2}	c_{p+2}	x_{20}	x_{21}	x_{2p}	0 .	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
m	a_n	c_n	x_{m0}	x_{m1}	x_{mp}	0	1
m+1			z_o	$(z_1-c_1) \ . \ . \ . (z_p-c_p)$		0 . . .	0

Initially the $x_{i0} = b_i$ and $x_{ij} = a_{ij}$ for i = 1,2, ..., m and j = 1,2, ..., n, but these values will be transformed at every step of the method. The $z_o = \sum_{i=1}^{m} c_{Bi}x_{i0}$ and $z_j - c_j = \sum_{i=1}^{m} c_{Bi}x_{ij} - c_j$ will also be transformed at every stage. The initial basis will be chosen to be the identity matrix of slack variable coefficients and a new m × n basis will be formed at each stage when one vector is selected to enter the basis and another is selected to leave by the standard simplex process.

Now, for our example we select the initial basis to be the identity matrix composed of a_3 and a_4 and convert this information into the initial tableau.

Tableau 1.2

				1	2	0	0
i	Basis	c_B	P_0	P_1	P_2	P_3	P_4
1	a_3	0	15	5	9	1	0
2	a_4	0	9	-13	9	0	1
3			0	-1	-2	0	0

Then by the standard procedure a_2 is selected to enter the basis and a_4 is selected to leave. Then the simplex transformation is performed using the pivot element, x_{22} (circled in Tableau 1.2), to derive the second tableau, Tableau 1.3.

Tableau 1.3

i	Basis	c_B	P_o	P_1	P_2	P_3	P_4
1	a_3	0	6	18	0	1	-1
2	a_2	2	1	$-\frac{13}{9}$	1	0	$\frac{1}{9}$
3			2	$-\frac{35}{9}$	0	0	$\frac{2}{9}$

Again the new basis is chosen, a_1 enters and a_3 leaves, and the simplex transformation is performed using the pivot element, x_{11}, to derive the third tableau, Tableau 1.4.

Tableau 1.4

i	Basis	c_B	P_0	P_1	P_2	P_3	P_4
1	a_1	1	$\frac{6}{18}$	1	0	$\frac{1}{18}$	$-\frac{1}{18}$
2	a_2	2	$\frac{40}{27}$	0	1	$\frac{13}{162}$	$\frac{5}{162}$
3			$\frac{89}{27}$	0	0	$\frac{35}{162}$	$\frac{1}{162}$

Since all $z_j - c_j \geq 0$ the optimal solution to the LP can be read from column P_0. $x_1^* = \frac{1}{3}$, $x_2^* = \frac{40}{27}$, $x_3^* = 0$, $x_4^* = 0$, $z^* = \frac{89}{27}$. Note that this solution agrees with our earlier solution by graphing, since the intersection of the two constraint equalities in quadrant one has the coordinates $x_1^* = \frac{1}{3}$, $x_2^* = \frac{40}{27}$.

Still it would seem logical that the solution to the ILP could be obtained by rounding the solution x* above to the nearest integer vector, which is an element of C. That is $x^T = (0,1)$. (Even such a process as this would involve many computations to be certain that the nearby integer solution were an element of C if m were large.) But we can see for our example, by evaluating z at each of the six feasible integer solutions (represented by the dots in Figure 1.1, page 5),

for $x^T = (3,0)$; $z = 3$

$x^T = (2,0)$; $z = 2$

$x^T = (1,0)$; $z = 1$

$x^T = (0,0)$; $z = 0$

$x^T = (0,1)$; $z = 2$

$x^T = (1,1)$; $z = 3$

that $z^* = 3$ for the ILP and this value occurs at $x^T = (1,1)$ and at $x^T = (3,0)$. Obviously, the assumption that rounding the solution for the corresponding LP to the nearest feasible integer vector to solve the ILP would not yield a reliable method.

Another look at the feasible solution set C shows that, when we introduce new constraints (cutting planes) a and b, the convex set C is reduced to C' which has both optimal solutions to the ILP as extreme points (see Figure 1.2).

Adding such constraints which do not exclude any feasible integer solutions, but do exclude the solution to the original corresponding LP will enable us to reach the optimal integer solution using the simplex method. Thus the most common approach to solving the ILP:

1. Determine the optimal tableau for the corresponding LP as a starting point.
2. Introduce a new constraint called a cutting plane which meets the following two conditions:
   a. The cutting plane must not exclude any feasible integer solution.

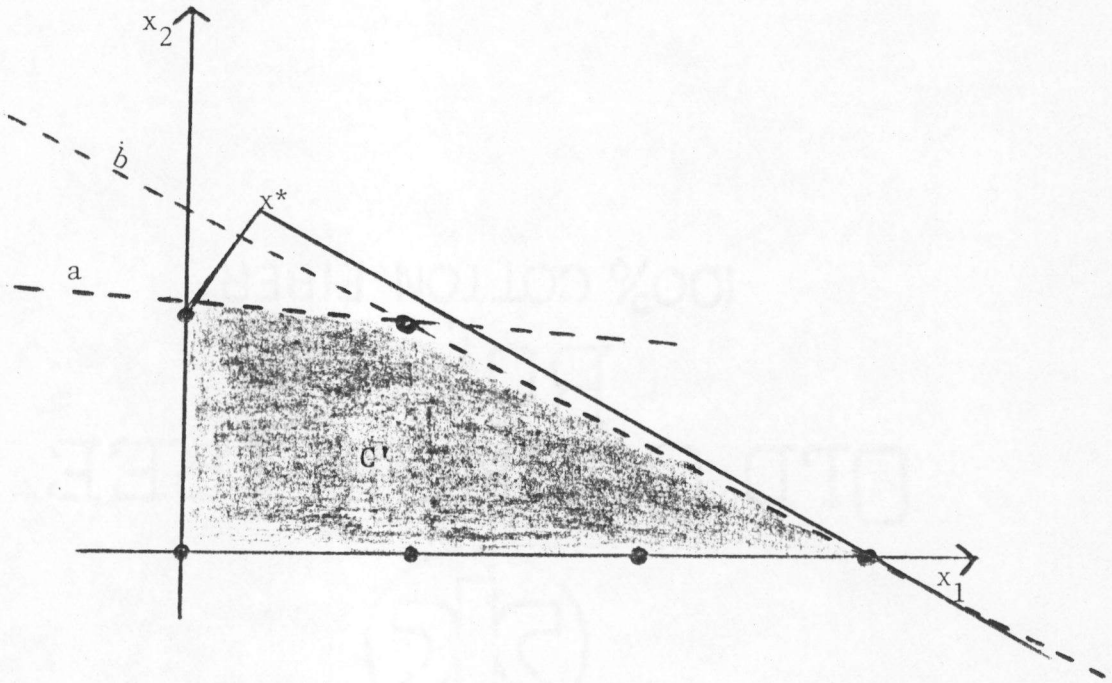


Figure 1.2. Reduction of the feasible solution set by introducing cutting planes.

b. The cutting plane must exclude the solution to the corresponding LP.

3. Solve the new problem using the dual simplex method (see the Appendix).

4. Repeat steps 2 and 3 until an all integer solution has been reached.

III. A METHOD FOR CHOOSING THE CUTTING PLANE

The choice of the cutting plane is crucial to the success of the method. The following development by Gomory [7] gives us one method for choosing the cutting plane at each phase of the problem. Begin with the optimal tableau for the corresponding LP. For simplicity in the development, assume the optimal noninteger solution to the LP employs the first m columns of A. Then A can be partitioned into A = [B,R], where B (the basis matrix) contains the first m columns of A, and R contains the remaining n - m columns of A. The optimal solution x^* is correspondingly partitioned $x^{*T} = [x_B^{*T}, x_R^{*T}]$, where $x_R^* = \emptyset$ (i.e. each component of x_R^* has the value 0). Substituting into the standard problem statement (1.3), we get

$$Ax^* = b$$

$$[B,R]\begin{bmatrix} x_B^* \\ x_R^* \end{bmatrix} = b \qquad (1.6)$$

Upon expanding (1.6) and left multiplication by B^{-1}, we have

$$Bx_B^* + Rx_R^* = b$$

$$B^{-1}Bx_B^* + B^{-1}Rx_R^* = B^{-1}b$$

$$Ix_B^* = B^{-1}b - B^{-1}Rx_R^* \tag{1.7}$$

By the simplex process the columns, P_ℓ, of the tableau can be represented at any stage by a product of the inverse of the basis matrix with each column, a_ℓ, of A. That is

$$B^{-1}a_\ell = P_\ell \qquad \text{for } \ell = 0,1,2, \ldots, n \tag{1.8}$$

Also we know $P_0 = B^{-1}b$ so we can write (1.7) in general as

$$x_B = P_0 - \sum_{j\in N} P_j x_j \tag{1.9}$$

$N = \{j \mid a_j \text{ is a column of } R\}$.

Let x_{Br}^*, representing the r^{th} component of x_B^*, be the component with the largest fractional part. From (1.9) we have

$$x_{Br} = x_{ro} - \sum_{j\in N} x_{rj} x_j \tag{1.10}$$

In the present solution $x_j = 0$ for all j, so x_{ro} is trivially noninteger. Let the notation [g] mean "the greatest integer less than or equal to g." Now, we can express the desired elements of the r^{th} row of the optimal tableau as the sum of an integer, $w_{r\ell} = [x_{r\ell}]$, and a nonnegative fraction, $f_{r\ell}$.

$$x_{r0} = [x_{r0}] + f_{r0} = W_{r0} + f_{r0} \tag{1.11}$$

$$W_{ro} \geq 0$$

$$0 < f_{r0} < 1$$

and for $j \in N$

$$x_{rj} = [x_{rj}] + f_{rj} = W_{rj} + f_{rj} \tag{1.12}$$

$$0 \leq f_{rj} < 1$$

Substitution of (1.11) and (1.12) into (1.10) leads to the following result:

$$\begin{aligned} x_{Br} &= (W_{ro} + f_{ro}) - \sum_{j \in N} (W_{rj} + f_{rj}) x_j \\ &= (W_{ro} - \sum_{j \in N} W_{rj} x_j) + (f_{ro} - \sum_{j \in N} f_{rj} x_j) \end{aligned} \tag{1.13}$$

Now, for the modified solution to be all integer, at least one of the currently nonbasic x_j ($j \in N$) must become a positive integer (because when all $x_j = 0$ we know there is a unique solution x^* noninteger). For this modified solution then, the first quantity in (1.13) is integer, since $W_{r\ell}$ are integer by definition we have all x_j are either zero or positive integer. So that

$$x_{Br} = (\text{integer}) + (f_{r0} - \sum_{j \in N} f_{rj} x_j) \tag{1.14}$$

and in order for x_{Br} to be an integer, we must guarantee that the second quantity in (1.14) can be integer. We know that $f_{rj} \geq 0$ and $x_j \geq 0$ (for $j \in N$) so

$$\sum_{j\in N} f_{rj}x_j \geq 0 \tag{1.15}$$

Also, f_{ro} is positive and less than one. Thus, if $(f_{ro} - \sum_{j\in N} f_{rj}x_j > 0)$, then it must be less than one which is, of necessity, noninteger. Therefore we must make

$$f_{ro} - \sum_{j\in N} f_{rj}x_j \leq 0 \tag{1.16}$$

in order for x_{Br} to possibly be integer.

We have now derived an inequality which meets the two desired conditions for a cutting plane to be added to the set of constraints $A x = b$. That is:

a) <u>The cutting plane must not exclude any feasible integer solutions</u>.

Let x_I be a feasible integer solution, f_{I0} doesn't exist since all x_{Ii} are integer. All $x_j \geq 0$ by the constraint of (1.3), so $\sum_{j\in N} f_{rj}x_j \geq 0$ and $-\sum_{j\in N} f_{rj}x_j \leq 0$. Thus (1.16) satisfies a).

b) <u>The cutting plane must exclude the solution x* from the corresponding LP</u>.

For x* we know $x_j = 0$ for $j\in N$, and $f_{ro} > 0$ by (1.11). Thus $f_{ro} - 0 \not\leq 0$ and (1.16) satisfies b).

In order to add the cutting plane we add a slack variable x_{n+1}, which conveniently has for its coefficient vector e_{m+1}, the natural choice for the $(m + 1)^{st}$ vector in the ammended basis. Quite neatly,

the cutting plane (1.17) has been constructed with nonbasic variables only; therefore, it can be added to the optimal tableau with no additional computations.

$$- \sum_{j \in N} f_{rj} x_j + x_{n+1} = -f_{r0} \tag{1.17}$$

Certainly, there are many ways to construct cutting planes that meet our criteria. A great deal of research has been conducted in this area. According to Burdett [2], computational difficulties are usually encountered when a cut is constructed, which guarantees deepness. Therefore, one should seek cutting planes, which can be efficiently computed. Our choice (1.17) yields a cutting plane which takes a minimal number of computations to generate, and can be added directly to the optimal tableau with no modifications.

We now have the necessary information and a precise method for solving the ILP using cutting planes.

EXAMPLE 1.1

Consider the following example:

$$\max z = x_2 - x_1$$

$$2x_1 + x_2 \leq 9 \tag{1.18}$$

$$2x_1 + 6x_2 \leq 21$$

$$x_i \geq 0, \; x_i \text{ integer} \qquad i = 1,2$$

The graphical interpretation, Figure 1.3, reveals the solution to the corresponding LP to be $x_1 = 0$, $x_2 = \frac{7}{2}$.

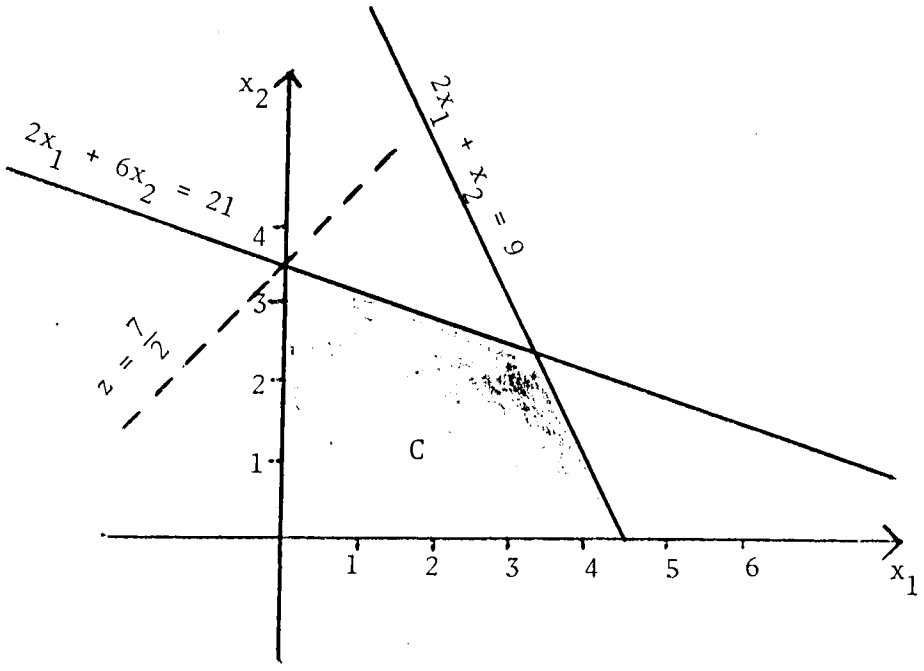


Figure 1.3. Graphical interpretation revealing the solution to the LP corresponding to (1.18).

Placing the problem in standard form (1.19), we can determine the optimal tableau for the corresponding LP to Tableau 1.5.

$$\max z = -x_1 + x_2 + 0x_3 + 0x_4$$

$$2x_1 + x_2 + x_3 + 0x_4 = 9 \qquad (1.19)$$

$$2x_1 + 6x_2 + 0x_3 + x_4 = 21$$

$$x_i \geq 0 \qquad x_i = \text{integer} \qquad i = 1,2,3,4$$

Tableau 1.5

i	Basis	c_B	P_o	P_1	P_2	P_3	P_4
1	a_3	0	$\frac{11}{2}$	$\frac{5}{3}$	0	1	$\frac{1}{6}$
2	a_2	1	$\frac{7}{2}$	$\frac{1}{3}$	1	0	$\frac{1}{6}$
3			$\frac{7}{2}$	$\frac{1}{3}$	0	0	$\frac{1}{6}$

$$x_1^* = 0,\ x_2^* = \frac{7}{2},\ x_3^* = \frac{11}{2},\ x_4^* = 0,\ z^* = \frac{7}{2}$$

In order to determine the cutting plane according to (1.16) and (1.17), we make note of the necessary information from the optimal Tableau 1.5. Setting $x_{r0} = x_{10}$ we have

$$f_{10} = \frac{1}{2}, \qquad f_{11} = \frac{2}{3}, \qquad f_{14} = \frac{1}{6}$$

Now the inequality and cutting plane thus determined are (1.20) and (1.21).

$$\frac{1}{2} - (\frac{2}{3}x_1 + \frac{1}{6}x_4) \leq 0 \tag{1.20}$$

$$\frac{2}{3}x_1 + 0x_2 + 0x_3 - \frac{1}{6}x_4 + x_5 = -\frac{1}{2} \tag{1.21}$$

Adding the new constraint to row three of the optimal tableau, and choosing $a_5 = e_3$ as the third vector for the basis, we get Tableau 1.6, which can be solved by one iteration of the dual simplex method.

Tableau 1.6

i	Basis	c_B	P_0	P_1	P_2	P_3	P_4	P_5
1	a_3	0	$\frac{11}{2}$	$\frac{5}{3}$	0	1	$\frac{1}{6}$	0
2	a_2	1	$\frac{7}{2}$	$\frac{1}{3}$	1	0	$\frac{1}{6}$	0
3	a_5	0	$-\frac{1}{2}$	$-\frac{2}{3}$	0	0	$-\frac{1}{6}$	1
4			$\frac{7}{2}$	$\frac{4}{3}$	0	0	$\frac{1}{6}$	0

a_5 leaves the basis, since $-\frac{1}{2} < 0$.

a_4 enters, since $\max\left\{\frac{4/3}{-2/3}, \frac{1/6}{-1/6}\right\} = -1 = \frac{z_4 - c_4}{x_{34}}$.

By the standard dual simplex method remove a_5 from the basis and insert a_4. Performing the transformation using $x_{34} = -\frac{1}{6}$ as the pivot element, we get Tableau 1.7.

Tableau 1.7

i	Basis	c_B	P_0	P_1	P_2	P_3	P_4	P_5
1	a_3	0	5	1	0	1	0	1
2	a_2	1	3	$-\frac{1}{3}$	1	0	0	1
3	a_4	0	3	4	0	0	1	6
4			3	$\frac{2}{3}$	0	0	0	1

Thus, we have the integer solution. $x_1^* = 0$, $x_2^* = 3$, $x_3^* = 5$, $x_4^* = 3$, $x_5^* = 0$, $z^* = 3$. We can visualize the cutting plane by substituting for x_4 the value derived from (1.19) into (1.20).

$$2x_1 + 6x_2 + 0x_3 + x4 = 21$$

$$x_4 = 21 - 2x_1 - 6x_2 \tag{1.22}$$

$$\frac{1}{2} - (\frac{2}{3}x_1 + \frac{1}{6}(21 - 2x_1 - 6x_2)) \leq 0$$

$$\frac{1}{2} - \frac{2}{3}x_1 - \frac{7}{2} + \frac{1}{3}x_1 + x_2 \leq 0$$

$$-\frac{1}{3}x_1 + x_2 \leq 3 \tag{1.23}$$

Adding (1.23) to the feasible set C of Figure 1.4 graphically confirms that the cutting plane did restrict the feasible solution, eliminating x*, not eliminating any integer points from C, and exposing a true integer solution as an extreme point of the new feasible solution set C'.

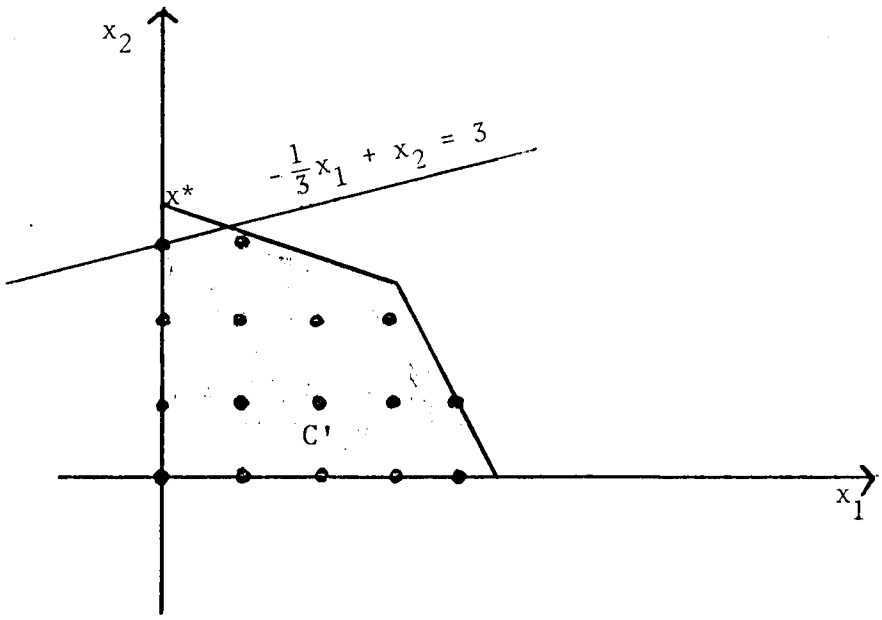


Figure 1.4. Graphical interpretation of the reduction of the feasible solution to (1.18), exposing the integer solution.

CHAPTER II

THE ALL-INTEGER ALGORITHM

The cutting plane method of Chapter I provides a dependable approach to solving the ILP. As mentioned previously, there are a number of ways to select the cutting planes and, in fact, alternate methods of employing the simplex computation. In 1963, Gomory [8] proposed an algorithm which maintains an all-integer tableau at every stage of the calculation; thus, we are guaranteed an integer solution to the problem. We will consider in this chapter this all-integer algorithm and its special choice of cutting planes; also, we will consider the facility with which a computer program can be adapted to this method.

I. THE ALGORITHM

For a starting point, we assume the initial tableau is completely integer valued, and we have a feasible solution for the dual problem but not for the primal (i.e. for the minimization problem, which we shall consider, all $z_j - c_j \leq 0$, but some $x_{i0} < 0$). Also, we add p equations (recall $m + p = n$) of the form $x_j' - x_j = 0$ for $j = 1,2, \ldots, p$, to the original set of m constraints, and set $c_j' = c_j$. This gives us the following initial tableau.

In Tableau 2.1 row zero is defined so that z_0 is the value of z at each iteration, and $x_{0j} = z_j - c_j$ for $j = 1, \ldots, p, (n+s)$. Note that the addition of the $x_j' - x_j = 0$ for $j = 1, \ldots, p$ forces us to have an

Tableau 2.1

				$c_1 \cdots c_p$	0 ... 0	$c_1 \cdots c_p$	
i	Basis	c_B	P_o	$P_1 \cdots P_p$	$P_{p+1} \cdots P_n$	$P'_1 \cdots P'_p$	P_{n+s}
0			z_0	$x_{01} \cdots x_{0p}$	0 ... 0	0 ... 0	
1	a_{p+1}	c_{p+1}	x_{10}	$x_{11} \cdots x_{1p}$	1 ... 0	0 ... 0	
2	a_{p+2}	c_{p+2}	x_{20}	$x_{21} \cdots x_{2p}$	0 ... 0	0 ... 0	
.	.	.	.	. .	. . .	. .	
.	.	.	.	. .	. .	. .	
.	.	.	.	. .	. .	. .	
m	a_n	c_n	x_{m0}	$x_{m1} \cdots x_{mp}$	0 ... 1	0 ... 0	
m+1	a'_1	c_1	0	-1 ... 0	0 ... 0	1 ... 0	
.	.	.	.	. .	. .	. .	
.	.	.	.	. .	. .	. .	
.	.	.	.	. .	. .	. .	
n	a'_p	c_p	0	0 ... -1	0 ... 0	0 ... 1	
n+1							

$n \times n$ basis instead of $m \times m$, thus forcing every iteration to yield a point including the variables of the original convex set. Now, if we can select appropriate cutting planes, the new point will be inside the original convex set. This new cutting plane will be chosen from nonbasic variables, so that it can be directly added to the $(n + 1)^{st}$ row of the tableau. It will also be chosen to be all integer and to provide a unit pivot element. Thus we preserve the all-integer character of our tableau at every iteration.

At the end of each iteration the transformed cutting constraint may be removed from the tableau in order for a new cutting plane to be introduced. When all $x_{i0} \geq 0$ for i = 1,2, ..., n, we are guaranteed to have an all integer solution to the ILP.

Now, let us choose a row from the tableau which has a negative constant term (i.e. some $x_{\ell 0} < 0$) to generate the cutting constraint; for instance the ℓ^{th} row. We know

$$x_{\ell 0} = x_{\ell 1}x_1 + x_{\ell 2}x_2 + \ldots + x_{\ell p}x_p + 1 \cdot x_{\ell+p} \tag{2.1}$$

where $x_1, x_2, \ldots, x_p$ are nonbasic variables. Now, write each coefficient as the sum of the multiple of an integer, $b_{\ell j}$, and a remainder (2.2),

$$x_{\ell j} = b_{\ell j}\lambda + r_{\ell j} \qquad j = 0,1,2, \ldots, p, (\ell + p) \tag{2.2}$$

where $b_{\ell j}$ is a positive integer, λ is a positive constant to be determined, and $r_{\ell j}$ is the remainder term. Then, we can set

$$x_{\ell j} = b_{\ell j}\lambda + r_{\ell j} = \left\lfloor \frac{x_{\ell j}}{\lambda} \right\rfloor \lambda + r_{\ell j} \tag{2.3}$$

$$1 = \left\lfloor \frac{1}{\lambda} \right\rfloor \lambda + r$$

$$0 \leq r_{\ell j} < \lambda, \qquad 0 \leq r \leq \lambda, \qquad \lambda > 0$$

so that the brackets again indicate the "greatest integer less than or equal to." Note that if $\lambda > 1$, then $\left\lfloor \frac{1}{\lambda} \right\rfloor = 0$. If we substitute (2.3) into (2.1) we get the following:

$$\left\lfloor \frac{x_{\ell 0}}{\lambda} \right\rfloor \lambda + r_{\ell 0} = \left(\left\lfloor \frac{x_{\ell 1}}{\lambda} \right\rfloor \lambda + r_{\ell 1} \right) x_1 + \left(\left\lfloor \frac{x_{\ell 2}}{\lambda} \right\rfloor \lambda + r_{\ell 2} \right) x_2$$

$$+ \ldots + \left(\left\lfloor \frac{x_{\ell p}}{\lambda} \right\rfloor \lambda + r_{\ell p} \right) x_p + \ldots$$

$$+ \left(\left\lfloor \frac{1}{\lambda} \right\rfloor \lambda + r \right) x_{\ell+p} \tag{2.4}$$

or

$$r_{\ell 0} + \lambda \left\{ \left\lfloor \frac{x_{\ell 0}}{\lambda} \right\rfloor - \left\lfloor \frac{x_{\ell 1}}{\lambda} \right\rfloor x_1 - \left\lfloor \frac{x_{\ell 2}}{\lambda} \right\rfloor x_2 - \ldots - \left\lfloor \frac{x_{\ell p}}{\lambda} \right\rfloor x_p \right.$$

$$\left. - \left\lfloor \frac{1}{\lambda} \right\rfloor x_{\ell+p} \right\} = r_{\ell 1} x_1 + r_{22} x_2 + \ldots + r_{\ell p} x_p + r x_{\ell+p} \tag{2.5}$$

We note that any nonnegative integer solution to (2.1) must also be a solution to its equivalent form (2.5). Also note that since all $r_{\ell j} \geq 0$ and $x_j \geq 0$, the right hand side of (2.5) is nonegative.

If we denote the quantity in braces as x_{n+s}, we have

$$x_{n+s} = \left\lfloor \frac{x_{\ell 0}}{\lambda} \right\rfloor - \sum_{j \in N} \left\lfloor \frac{x_{\ell j}}{\lambda} \right\rfloor x_j - \left\lfloor \frac{1}{\lambda} \right\rfloor x_{\ell+p} \tag{2.6}$$

for $N = \{j \mid a_j$ is not a basis column$\}$

where x_{n+s} is a nonnegative integer. The integer quality is obvious since all coefficients and variables in the right side of (2.6) are integer. In order to establish the nonegativity of x_{n+s}, observe that $0 \leq r_{\ell 0} < \lambda$. If $x_{n+s} < 0$, we know it must be a negative integer and $|\lambda x_{n+s}| > r_{\ell 0}$, which implies that $r_{\ell 0} + \lambda x_{n+s} = r_{\ell 0} - |\lambda x_{n+s}| < 0$.

But, this contradicts the fact that the right side of (2.5) is nonnegative. Therefore x_{n+s} is nonnegative. We will introduce x_{n+s} as a new variable in the problem.

If we now restrict $\lambda > 1$, which implies $[\frac{1}{\lambda}] = 0$ and substitute for the $b_{\ell j}$ as defined by (2.3), then (2.6) becomes

$$x_{n+s} = \left\lfloor \frac{x_{\ell 0}}{\lambda} \right\rfloor - \sum_{j \in N} \left\lfloor \frac{x_{\ell j}}{\lambda} \right\rfloor x_j - 0 \cdot x_{\ell+p}$$

$$= b_{\ell 0} - \sum b_{\ell j} x_j \qquad (2.7)$$

or

$$b_{\ell 0} = \sum b_{\ell j} x_j + x_{n+s} \qquad (2.8)$$

Now with a suitable selection of λ, (2.8) can be used as our all-integer cutting constraint. We need λ to be large enough to yield at least one $b_{\ell j} = \left\lfloor \frac{x_{\ell j}}{\lambda} \right\rfloor = -1$ to be used as our unit pivot element. But, the smaller the value for λ the better the improvement in the optimal value. Note that for the minimization dual simplex method, we choose P_k, the pivot column, to be such that

$$\frac{z_k - c_k}{x_{\ell k}} = \min_{x_{\ell j} < 0} \frac{z_j - c_j}{x_{\ell j}} \qquad (2.9)$$

By our design $b_{\ell k}$, the pivot element, will have the value negative one, so that

$$\frac{z_k - c_k}{b_{\ell k}} = \frac{z_k - c_k}{-1} \leq \frac{z_j - c_j}{[x_{\ell j}/\lambda]} = \frac{z_j - c_j}{b_{\ell j}} \qquad \text{for } x_{\ell j} < 0$$

or

$$\frac{z_k - c_k}{-1} \leq \frac{z_j - c_j}{b_{\ell j}} \tag{2.10}$$

So, to choose λ let m_j be the largest integer for which

$$\frac{z_k - c_k}{-1} \leq \frac{z_j - c_j}{-m_j} \tag{2.11}$$

so that

$$-b_{\ell j} = -\left\lfloor \frac{x_{\ell j}}{\lambda} \right\rfloor \leq m_j \qquad \text{for } x_{\ell j} < 0 \tag{2.12}$$

If $z_k - c_k = 0$ there is degeneracy in the dual problem, which must be avoided by the usual technique. Now for each j, the smallest positive number that satisfies (2.12) and allows P_k to be the pivot column is $\lambda_j = -\left\lceil \frac{x_{\ell j}}{m_j} \right\rceil$. But since the minimum possible λ must be at least as large as the largest λ_j, we choose

$$\lambda = \max_{x_{\ell j} < 0} \lambda_j \tag{2.13}$$

Note that λ must be greater than one to guarantee that the cutting constraint consists only of nonbasic variables. We now can summarize the method for a minimization problem:

1. Determine a dual feasible solution with optimal tableau.
2. From the rows containing a negative constant term select a row, the ℓ^{th} row, to be used to generate the cutting plane. If all $x_i \geq 0$, we have a minimal all integer solution.
3. The pivot column corresponds to the P_k for which

$$-(z_k - c_k) = \min_{x_{\ell j}<0} -(z_j - c_j) \qquad \text{for } j \geq 1$$

If no $x_{\ell j} < 0$ the problem is not feasible.

4. For each j having $x_{\ell j} < 0$ determine the largest integer, m_j, such that

$$-(z_k - c_k) \leq -\frac{(z_j - c_j)}{m_j}$$

5. Let $\lambda_j = -\left|\frac{x_{\ell j}}{m_j}\right|$ for each $x_{\ell j} < 0$.
6. Let $\lambda = \max_{x_{\ell j}<0} \lambda_j$.
7. Determine the cutting plane from equation (2.8). Add it to the $(n + 1)^{st}$ row of the tableau.
8. Perform the simplex transformation using $b_{\ell k} = -1$ as the pivot element. Condense the new tableau removing any columns which cannot be changed by further transformations, and remove the transformed cutting plane to prepare for the next iteration.

EXAMPLE 2.1

Consider the following problem:

$$\begin{aligned}
&\min\ z = x_1 + 8x_2 \\
&2x_1 + 3x_2 \geq 6 \\
&-3x_1 + x_2 \leq 1 \\
&x_1 + x_2 \leq 7 \\
&x_1 - 3x_2 \leq 1 \\
&x_i \geq 0,\ x_i \text{ integer} \qquad \text{for } i = 1,2
\end{aligned} \tag{2.14}$$

Upon adding slack variables we have

$$\begin{aligned}
&\min\ z = x_1 + 8x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 \\
&-2x_1 - 3x_2 + x_3 = 6 \\
&-3x_1 + x_2 + x_4 = 1 \\
&x_1 - x_2 + x_5 = 7 \\
&x_1 - 3x_2 + x_6 = 1 \\
&x_i \geq 0,\ x_j \text{ integer} \qquad \text{for } i = 1,2, \ldots, 6
\end{aligned} \tag{2.15}$$

Transferring this information to the initial all-integer tableau and adding the additional equations $x_1' = x_1$ and $x_2' = x_2$, we get Tableau 2.2. Now, we determine the cutting plane in accordance with the steps given above.

2) Since $x_{10} < 0$, let $\ell = 1$.

3) $-(z_1 - c_1) = \min(1, 8)$, so $k = 1$.

Tableau 2.2

				1	8	0	0	0	0	1	8	
i	Basis	c_B	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_1'	P_2'	P_7
0			0	-1	-8	0	0	0	0	0	0	0
1	a_3	0	-6	-2	-3	1	0	0	0	0	0	0
2	a_4	0	1	-3	1	0	1	0	0	0	0	0
3	a_5	0	7	1	1	0	0	1	0	0	0	0
4	a_6	0	1	1	-3	0	0	0	1	0	0	0
5	a_1'	1	0	-1	0	0	0	0	0	1	0	0
6	a_2'	8	0	0	-1	0	0	0	0	0	1	0
7	a_7	0	-3	-1	-2	0	0	0	0	0	0	1

4) $1 \leq \frac{1}{m_1}$ so $m_1 = 1$, $1 \leq \frac{8}{m_2}$ so $m_2 = 8$.

5) $\lambda_1 = -\left\{\frac{-2}{1}\right\} = 2$, $\lambda_2 = -\left\{\frac{-3}{8}\right\} = \frac{3}{8}$.

6) $\lambda = \max\left(2, \frac{3}{8}\right) = 2$.

7) $\left\{\frac{-6}{2}\right\} = \left[\frac{-2}{2}\right]x_1 + \left[\frac{-3}{2}\right]x_2 + x_7$

$-3 = -x_1 - 2x_2 + x_7$. Add this to Tableau 2.2.

8) Perform the simplex transformation to get Tableau 2.3.

Tableau 2.3

				1	8	0	0	0	0	1	8	0
i	Basis	c_B	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_1	P_2	P_7
0			3	0	-6	0	0	0	0	0	0	-1
1	a_3	0	0	0	1	1	0	0	0	0	0	-2
2	a_4	0	10	0	7	0	1	0	0	0	0	-3
3	a_5	0	4	0	-1	0	0	1	0	0	0	1
4	a_6	0	-2	0	-5	0	0	0	1	0	0	1
5	a'_1	1	3	0	2	0	0	0	0	1	0	-1
6	a'_2	8	0	0	-1	0	0	0	0	0	1	0
7												

We can condense Tableau 2.3 to the form of Tableau 2.4 by removing P_3, P_4, P_5, P_6, P_1 and placing P_7 in the column vacated by P_1. Also, we have left the $(n + 1)^{st}$ row clear for addition of a new constraint in the next iteration. We determine the new constraint in a similar fashion.

2) $\ell = 4$.

3) Only $x_{42} \leq 0$ so $k = 4$.

4) $6 \leq \frac{6}{m_2}$ so $m_2 = 1$.

5) $\lambda_2 = -\left(\frac{-5}{1}\right) = 5$ so $\lambda = 5$.

7) $\left[-\frac{2}{5}\right] = \left[\frac{-5}{-5}\right]x_2 + \left[\frac{1}{5}\right]x_7 + x_8$.

7) $-1 = -1x_2 + 0 \cdot x_7 + x_8$

Placing this constraint in Tableau 2.4 and performing the transformation we get Tableau 2.5.

Tableau 2.4

i	Basis	P_0	P_7	P_2	P_8
0		3	-1	-6	0
1	a_3	0	-2	1	0
2	a_4	10	-3	7	0
3	a_5	4	1	-1	0
4	a_6	-2	1	-5	0
5	a'_1	3	-1	2	0
6	a'_2	0	0	-1	0
7	a_8	-1	0	-1	1

Tableau 2.5

i	Basis	P_0	P_7	P_8	P_9
0		9	-1	-6	0
1	a_3	-1	-2	1	0
2	a_4	3	-3	7	0
3	a_5	5	1	-1	0
4	a_6	3	1	-5	0
5	a'_1	1	-1	2	0
6	a'_2	1	0	-1	0
7	a_9	-1	-1	0	1

Once again we determine the new constraint and find it to be:

$$-1 = -x_7 + x_9$$

Placing the new cutting plane in Tableau 2.5, we begin to perform the transformation and find $x_{i0} \geq 0$ for i = 1,2, ..., n. Thus, we need not complete Tableau 2.6. The solution, reading from P_0 is $x'_1 = x^*_1 = 2$, $x'_2 = x^*_2 = 1$, $x^*_3 = 1$, $x^*_4 = 6$, $x^*_5 = 4$, $x^*_6 = 2$, $z^* = 10$.

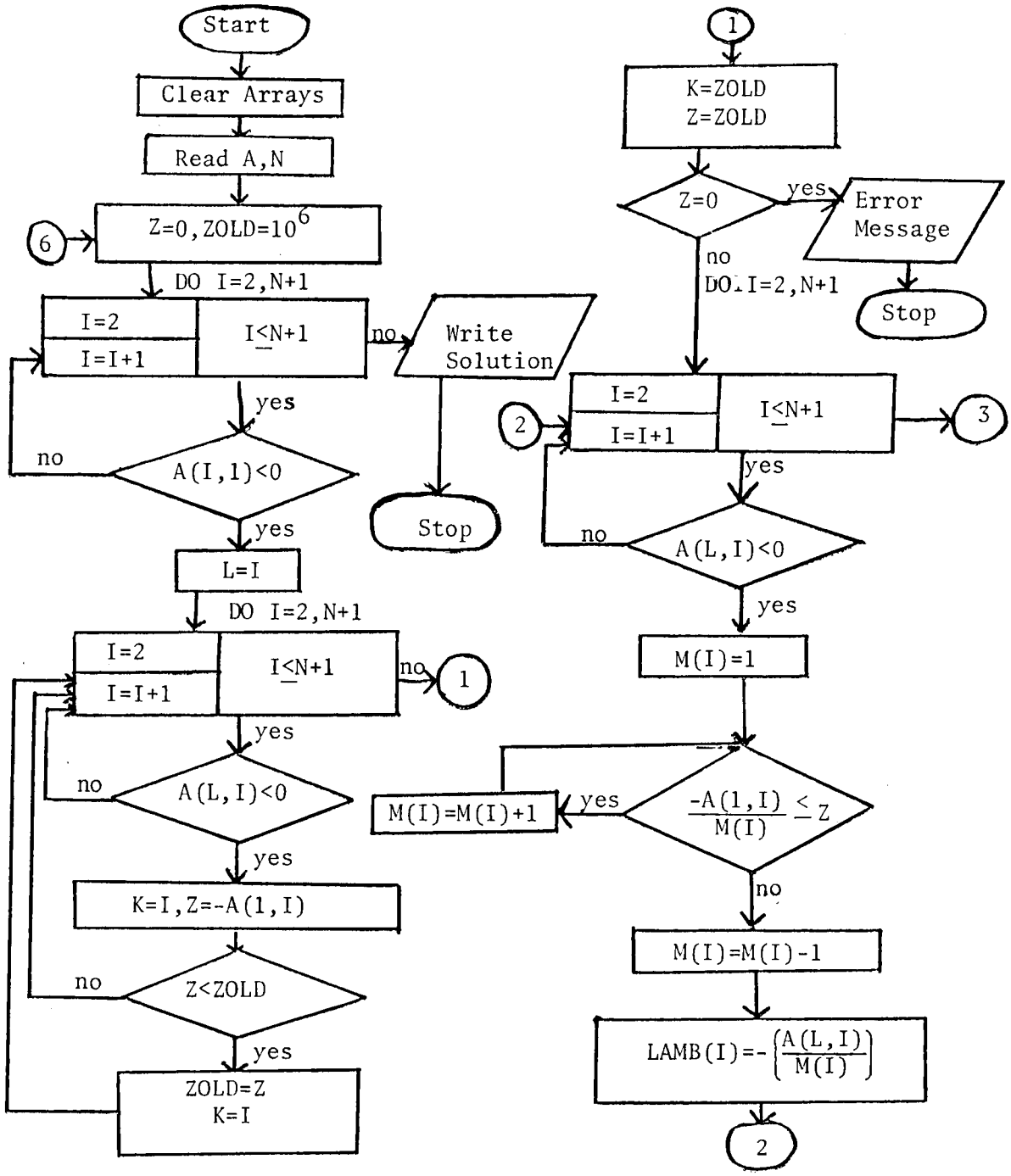
Tableau 2.6

i	Basis	P_0	P_9	P_8	P_{10}
0		10			
1	a_3	1			
2	a_4	6			
3	a_5	4			
4	a_6	2			
5	a'_1	2			
6	a'_2	1			
7					

II. ADAPTING THE ALGORITHM FOR MACHINE COMPUTATION

There are a great number of computations involved in the solution to an ILP using the all-integer algorithm. As demonstrated by the example above, solving even a small problem by hand can become a burdensome task. For problems arising in production and scheduling, one may have a thousand constraints in several hundred unknowns. Thus, it is worthwhile to consider the adaptability of our program to a computer algorithm. As one can see, the tableau format is easily converted to a machine storable matrix form so that a Fortran program can be easily written to execute the algorithm. The flowchart, Figure 2.1, demonstrates pictorially the adaptability to linear logic.

Unfortunately, due to occurrences of cyclic degeneracy and cases of numerical instability the machine computation is fallible, although it has met with overall success. The all-integer algorithm has the advantage that machine representation of integers is exact, whereas, the rounding and truncation error in the standard cutting plane algorithm creates additional numerical instabilities.



2.1. Flowchart for all-integer algorithm.

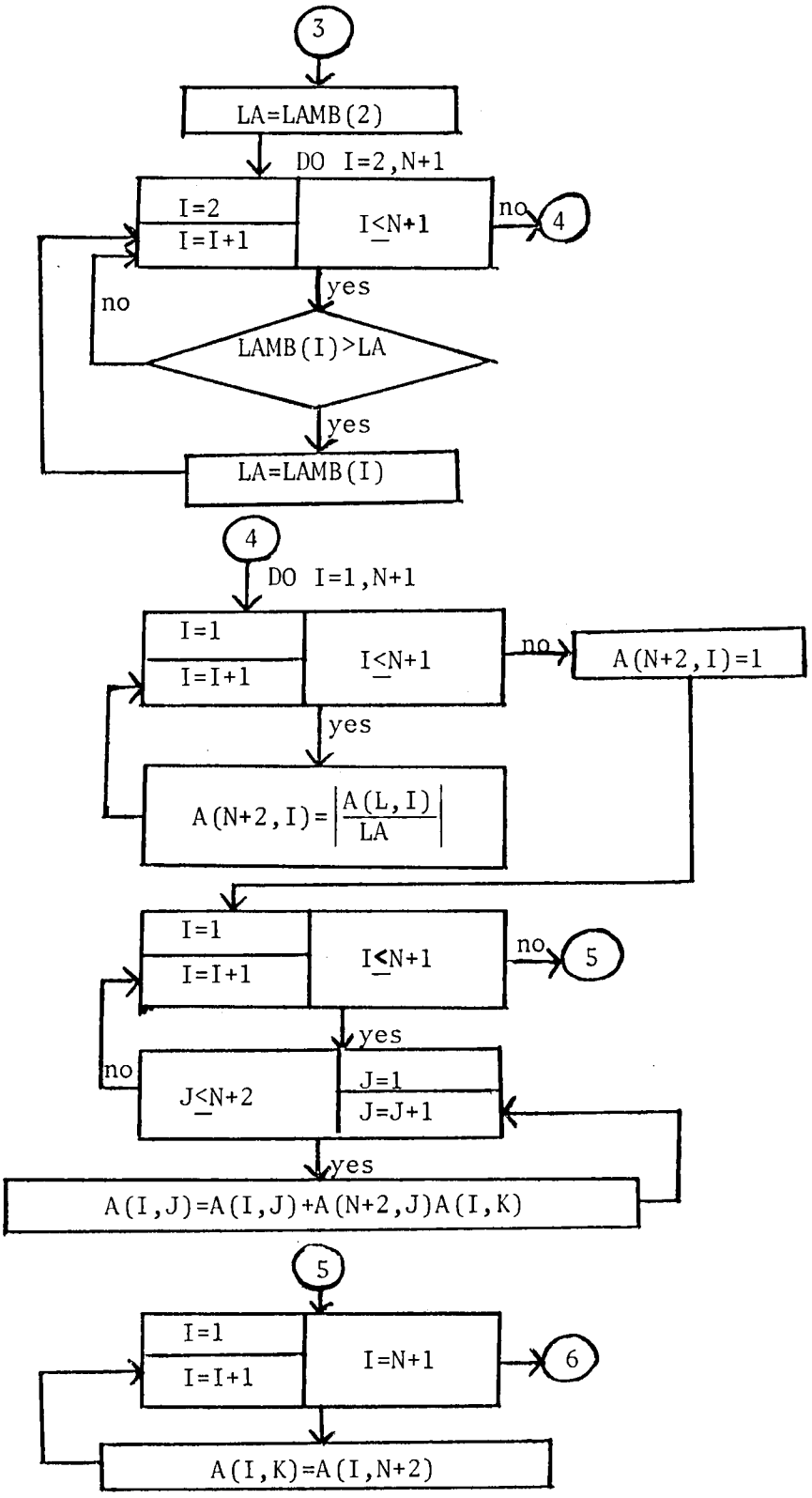


Figure 2.1 (continued)

CHAPTER III

THE KNAPSACK PROBLEM

Suppose that n types of camping equipment are being considered for a backpack hike. For $j = 1,2, \ldots,n$, $c_j > 0$ is the hiking value per unit j, and $a_j > 0$ is the weight limitation b, and our desire is to maximize the value of equipment taken. Certainly, the number of each item included is integer only. We can write the problem formulation as

$$\max z = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_j x_j \leq b \qquad (3.1)$$

$$x_j \geq 0, \; x_j \text{ integer} \qquad \text{for } j = 1,2, \ldots, n$$

Now, all data is integer since the value scale is arbitrarily designed, and a weight unit can be chosen small enough that each type of equipment has a weight which is some multiple of one unit (i.e. microgram, nanogram, etc.). This formulation shows how the name "knapsack problem" came to be the common name for (3.1).

Since the knapsack problem is simply an ILP with only one constraint, any general ILP algorithm can be used to solve it. But, due to its simplicity more efficient algorithms are possible. The value of studying the knapsack problem is found in the fact that some algorithms for more complex problems use it as a subroutine; also, a general

ILP in bounded variables can be transformed into the knapsack formulation (see Chapter IV).

I. AN ALGORITHM USING RECURSIVE EQUATIONS

Garfinkel and Nemhauser [5] have presented this algorithm using recursive equations. Let $f_k(y)$ be the maximum value of (3.1) where the weight limitation for the knapsack is y, and only the first k items are considered for (y = 0,1,2,...,b; k = 1,2,...,n). One can see that $f_n(b)$ would be z* the optimal value. The objective then is to generate a recursive sequence of functions $\{f_k(y)\}$, k = 1,2,...,n. (This approach is often called dynamic programming.) We have then

$$f_k(y) = \max \sum_{j=1}^{k} c_j x_j$$

$$\sum_{j=1}^{k} a_j x_j \leq y \qquad (3.2)$$

$$x_j \geq 0,\ x_j \text{ integer} \qquad j = 1,2,\ldots k$$

We can isolate x_k in (3.2) for k = 2, . . . , n to get formulation (3.3).

3.3
$$f_k(y) = \left\{ \begin{array}{l} \max\ c_k x_k \\ a_k x_k \leq y \\ x_k \geq 0,\ x_k \text{ integer} \end{array} \right\} + \left\{ \begin{array}{l} \max \sum_{j=1}^{k-1} c_j x_j \\ \sum_{j=1}^{k-1} a_j x_j \leq y - a_k x_k \\ x_j \geq 0,\ x_j \text{ integer} \end{array} \right\}$$

But the first term in (3.3) can be written more concisely, since $x_k \leq \frac{y}{a_k}$, as $\max_{x_k=0,1,2,\ldots,[y/a_k]} (c_k x_k)$. The second term is by definition (3.2) equal to $f_{k-1}(y - a_k x_k)$. So we can make these substitutions to get

$$f_k(y) = \max_{x_k=0,1,2,\ldots,[y/a_k]} (c_k x_k + f_{k-1}(y - a_k x_k)) \tag{3.4}$$

Thus, we have an equation to generate the recursive sequence of functions, if $f_1(y)$ can be determined. Since $c_1 > 0$, we know by the original definition of $f_k(y)$

$$\begin{aligned} f_1(y) &= \max_{x=0,1,2,\ldots,[y/a_1]} c_1 x_1 \\ &= c_1[y/a_1] \end{aligned} \tag{3.5}$$

So we need to initialize $f_0(y) = 0$ for $y = 0,1,\ldots,b$, in order for the recursive formulation (3.4) to be valid for $k = 1$ also.

By equation (3.4), we see that when optimizing over the first k types of items (regardless of the value assumed by x_k) the remaining space $(y - a_k x_k)$ must be optimally assigned to the first $(k - 1)$ items. Again, when we reach $f_n(b)$ we indeed have the optimal value.

In order for the formulation to hold for an equality constraint, it has been found that $f_0(y) = -\infty$ for $y = 1,2,\ldots,b$ and $f_0(0) = 0$. That is, no weight capacity can remain in the optimal knapsack. Equation (3.4) gives us a springboard to develop an efficient algorithm for solving (3.1), and will be referred to for the solution of the ILP in bounded variables to be considered in Chapter IV.

Now, if for a given k and y, $x_k = 0$ gives a maximal solution to (3.4) we have

$$x_k = 0 \Longrightarrow f_k(y) = c_k \cdot 0 + f_{k-1}(y - a_k \cdot 0) = f_{k-1}(y) \tag{3.6}$$

But if $x_k > 0$ gives a maximal solution to (3.4), we can design an optimal knapsack of size y over the first k items by combining one item of type k with an optimal knapsack of size $(y - 1 \cdot a_k)$ over the first k - 1 items. That is

$$x_k > 0 \Longrightarrow f_k(y) = c_k \cdot 1 + f_k(y - a_k \cdot 1) = c_k + f_k(y - a_k) \tag{3.7}$$

Now, if $f_0(y)$ is initialized as suggested above, we can combine (3.6) and (3.7) to get

$$f_k(y) = \max(f_{k-1}(y),\ c_k + f_k(y - a_k),\ a_k \leq y) \tag{3.8}$$
$$y=0,1,2,\ldots,b;\ k = 1,2,\ldots,n$$

This gives us a modification of (3.4) which is valid for solving (3.1) but involves a comparison of only two numbers to derive each $f_k(y)$.

To use (3.8) we must calculate $f_k(y)$ in ascending order of y so that $f_k(y - a_k)$ will always be available for $a_k \leq y$. We must also have an indicator $p_k(y)$ for all k and y to facilitate the determination of the optimal value of each x_k after completing the tabular computation of $f_k(y)$. Let

$$p_k(y) = \begin{cases} 0 \text{ if } f_k(y) = f_{k-1}(y) \\ 1 \text{ if } f_k(y) = c_k + f_k(y - a_k) \end{cases} \tag{3.9}$$

Putting the above information together, we have the following algorithm for solution to (3.1) using a recursive equation to determine a sequence of functions $f_k(y)$.

1) Initialize $k = 0$, and $f_0(y) = 0$ for $y = 0,1,2,\ldots,b$.

2) Set $k = k + 1$. For all $y < a_k$ set $f_k(y) = f_{k-1}(y)$ and $p_k(y) = 0$. Let $y = a_k$.

3) Compute $V_k(y) = c_k + f_k(y - a_k)$. If $V_k(y) > f_{k-1}(y)$ set $f_k(y) = V$ and $p_k(y) = 1$. If not set $f_k(y) = f_{k-1}(y)$ and $p_k(y) = 0$.

4) If $y < b$, set $y = y + 1$ and return to 3). If $y = b$ go to 5.

5) If $k < n$ return to step 2. If $k = n$, set $z = 0$ and go to 6. (z is a counter for the value of x_k.)

6) If $p_k(y) = 1$ set $z = z + 1$ and go to step 7. If $p_k(y) = 0$, go to step 8.

7) Set $y = y - a_k$ and go to 6.

8) Set $x_k = z$ and $z = 0$. If $k > 1$, set $k = k - 1$ and return to 6. If $k = 1$, the algorithm is complete.

Observe that the completion of step 5 leaves the optimal value $f_n(b)$ in the $f_k(y)$ position. In the last three steps we are determining the optimal solution for x_k, $k = 1,2,\ldots,n$. Saving the values $p_k(y)$ makes it quite simple to find the values for each x_k.

Application of this program to a Fortran computer routine would be immediate and, in fact, necessary for a problem with k or b large. But, we can demonstrate the use of the algorithm by a hand calculation, if we make one alteration. That is, mark all $f_k(y)$ with a minus sign if $p_k(y) = 1$, thus saving four additional columns in the solution table.

EXAMPLE 3.1

Consider the following knapsack problem:

$$\max z = 11x_1 + 9x_2 + 5x_3 + 3x_4$$

$$6x_1 + 5x_2 + 3x_3 + 2x_4 \leq 45$$

$$x_j \geq 0,\ x_j \text{ integer} \qquad \text{for } j = 1,2,3,4$$

Using the algorithm above, we can compute Table 3.1. Upon completing the table, we have the maximum value $z^* = 82$ in the position $f_4(45)$. Since $f_4(45)$ doesn't have a minus sign [i.e. $p_4(45) = 0$] $x_4 = 0$. Reducing k to k = 3 we have $f_3(45)$ with no minus sign [i.e. $p_3(45) = 0$] so $x_3 = 0$. Then $p_2(45) = 1$ so $x_2 \geq 1$ and $y = 45 - 5 = 40$. We find $p_2(40) = 1$, $p_2(35) = 1$ and $p_2(30) = 0$, so $x_2 = 3$. Similarly, we can determine $x_1 = 5$. So the optimal value $z^* = 82$ is attained when $x_1 = 5$, $x_2 = 3$, $x_3 = 0$, $x_4 = 0$.

II. ADAPTING THE KNAPSACK ALGORITHM FOR MACHINE COMPUTATION

Certainly, from the example we can see that a machine routine is needed for any practical use of the knapsack algorithm. We need only store the $p_k(y)$ and at any one time we need the k^{th} and $(k - 1)^{st}$ columns of $f_k(y)$ values. The $V_k(y)$ should not be stored since they are not recalled after their computation. The operations are all integer in nature and thus lead to no errors in rounding or truncation.

So, with a minimum of storage the algorithm can be completed by the computer. The flowchart (Figure 3.1) gives a visual demonstration

Table 3.1. Hand Computation for Knapsack Algorithm

y	$f_1(y)$	$V_2(y)$	$f_2(y)$	$V_3(y)$	$f_3(y)$	$V_4(y)$	$f_4(y)$
0	0	-	0	-	0	-	0
1	0	-	0	-	0	3 + 0	0
2	0	-	0	-	0	3 + 0	3-
3	0	-	0	5 + 0	5-	3 + 0	5
4	0	-	0	5 + 0	5-	3 + 3	6-
5	0	9 + 0	9-	5 + 0	9	3 + 5	9
6	11-	9 + 0	11	5 + 5	11	3 + 6	11
7	11-	9 + 0	11	5 + 5	11	3 + 9	12-
8	11-	9 + 0	11	5 + 9	14-	3 + 11	14
9	11-	9 + 0	11	5 + 11	16-	3 + 12	16
10	11-	9 + 9	18-	5 + 11	18	3 + 14	18
11	11-	9 + 11	20-	5 + 14	20	3 + 16	20
12	22-	9 + 11	22	5 + 16	22	3 + 18	22
13	22-	9 + 11	22	5 + 18	23-	3 + 20	23
14	22-	9 + 11	22	5 + 20	25-	3 + 22	25
15	22-	9 + 18	27-	5 + 22	27	3 + 23	27
16	22-	9 + 20	29-	5 + 23	29	3 + 25	29
17	22-	9 + 22	31-	5 + 25	31	3 + 27	31
18	33-	9 + 22	33	5 + 27	33	3 + 29	32
19	33-	9 + 22	33	5 + 29	34-	3 + 31	34
20	33-	9 + 27	36-	5 + 31	36	3 + 32	36
21	33-	9 + 29	38-	5 + 33	38	3 + 34	38
22	33-	9 + 31	40-	5 + 34	40	3 + 36	40
23	33-	9 + 33	42-	5 + 36	42	3 + 38	42
24	44-	9 + 33	44	5 + 38	44	3 + 40	44
25	44-	9 + 36	45-	5 + 40	45	3 + 42	45
26	44-	9 + 38	47-	5 + 42	47	3 + 44	47
27	44-	9 + 40	49-	5 + 44	49	3 + 45	49
28	44-	9 + 42	51-	5 + 45	51	3 + 47	51
29	44-	9 + 44	53-	5 + 47	53	3 + 49	53
30	55-	9 + 45	55	5 + 49	55	3 + 51	55
31	55-	9 + 47	56-	5 + 51	56	3 + 53	56
32	55-	9 + 49	58-	5 + 53	58	3 + 55	58
33	55-	9 + 51	60-	5 + 55	60	3 + 56	60
34	55-	9 + 53	62-	5 + 56	62	3 + 58	62
35	55-	9 + 55	64-	5 + 58	64	3 + 60	64
36	66-	9 + 56	66	5 + 60	66	3 + 62	66
37	66-	9 + 58	67-	5 + 62	67	3 + 64	67
38	66-	9 + 60	69-	5 + 64	69	3 + 66	69
39	66-	9 + 62	71-	5 + 66	71	3 + 67	71
40	66-	9 + 64	73-	5 + 67	73	3 + 69	73
41	66-	9 + 66	75-	5 + 69	75	3 + 71	75
42	77-	9 + 67	77	5 + 71	77	3 + 73	77
43	77-	9 + 69	78-	5 + 73	78	3 + 75	78
44	77-	9 + 71	79-	5 + 75	80-	3 + 77	80
45	77-	9 + 73	82-	5 + 77	82	3 + 78	82

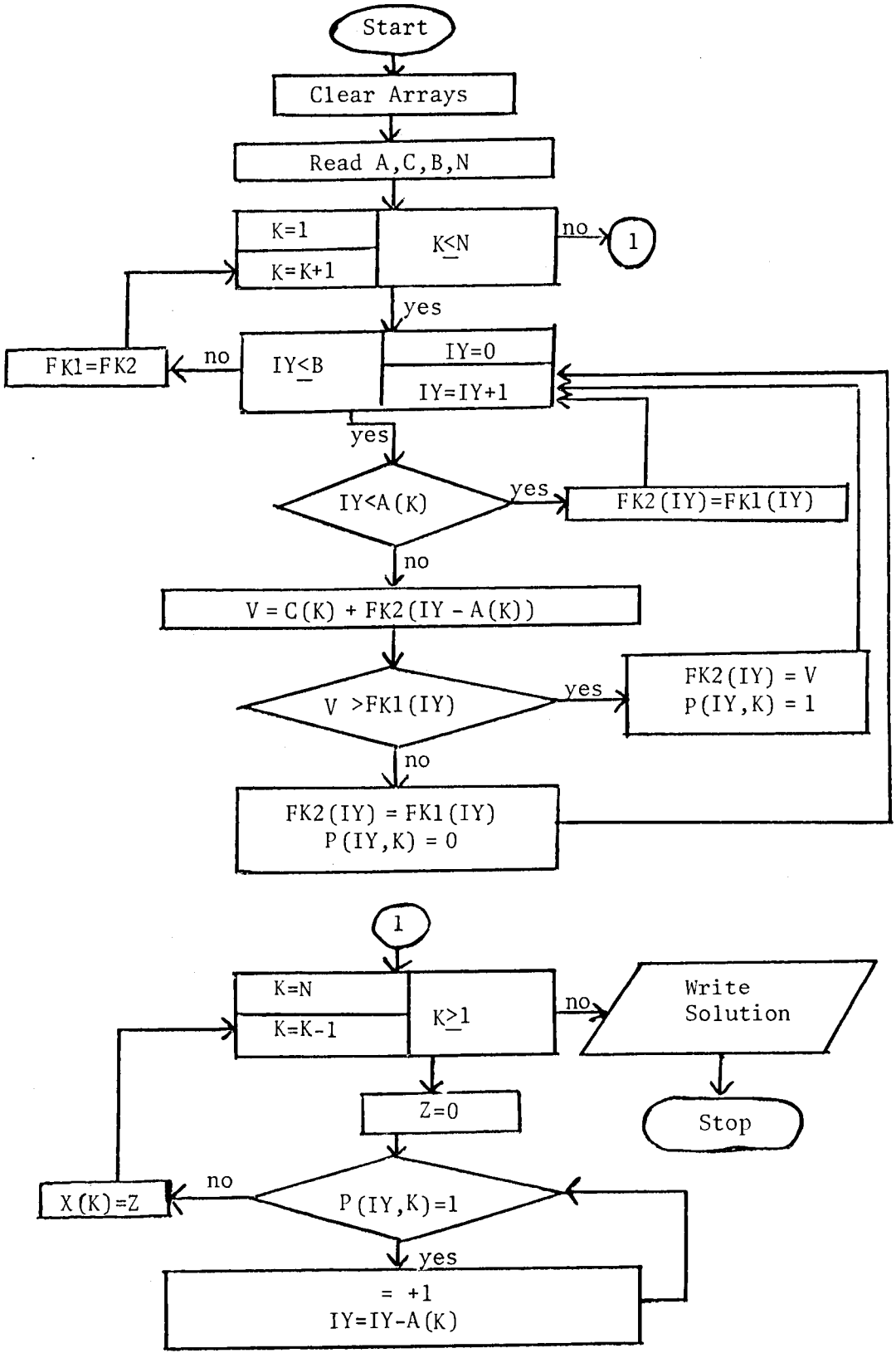


Figure 3.1. Flowchart for knapsack algorithm.

of the simplicity of the algorithm for this special ILP. One need only compare this machine routine to the general ILP routine in Chapter II in order to be convinced that the special properties of the knapsack problem allow for more efficient means of solution.

CHAPTER IV

TRANSFORMATION OF AN ILP IN BOUNDED VARIABLES TO A KNAPSACK PROBLEM IN BOUNDED VARIABLES

In Chapter III, we considered the knapsack problem as a special case of the general ILP. We can now show that for any ILP in bounded variables there is always an equivalent knapsack representation. That is for the constraint set S, where

$$S = \{x \mid Ax = b,\ 0 \leq x \leq d,\ x \text{ integer}\} \tag{4.1}$$

we can always guarantee a formulation T, where

$$T = \{x\ h^T Ax = h^T b,\ 0 \leq x \leq d,\ x \text{ integer}\} = S \tag{4.2}$$

In the above sets, A is an $m \times n$ coefficient matrix, x is an n-dimension variable vector with components x_i, b is an m-dimension solution vector, h is an m-dimension weight vector (or vector of multipliers), and d is an n-dimension boundary vector with components d_i such that $0 \leq x_i \leq d_i$ for $i = 1,2,\ldots,n$. If we ignore the upper bound consideration, T can be written as

$$\sum_{j=1}^{n} \left(\sum_{j=1}^{m} h_i a_{ij} \right) x_j = \sum_{j=1}^{n} a_j x_j = \sum_{i=1}^{m} h_i b_i = b_0 \tag{4.3}$$

which is, if we guarantee the $a_j > 0$ for $j = 1,2,\ldots,n$, a knapsack formulation where the constraint is an equality. For a bounded variable knapsack problem with an equality constraint, we encounter no difficulty

by allowing an objective coefficient to be negative, $c_k < 0$. If we initialize $f_0(0) = 0$ and $f_0(y) = -\infty$ for $y > 0$, we can solve the problem by a variation of the recursive equation (3.4), which allows for the bound condition. That is

$$f_k(y) = \max_{x_k = 0,1,2,\ldots,\min(d_k, [\frac{y}{a_k}])} c_k x_k + f_{k-1}(y - a_k x_k) \qquad (4.4)$$

We need to prove that there exists some weight vector h, which will allow us to derive (4.2) from any ILP constraint set (4.1). (Note that h simply allows us to aggregate the m equations of (4.1) into one equation with a guarantee that the solution sets are equivalent.) We then must establish an algorithm which will give us the solution to the derived bounded variable knapsack problem (4.5), using the recursive equation (4.4).

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

$$\sum_{j=1}^{n} a_j x_j = b_0 \qquad (4.5)$$

$$0 \leq x \leq d, \; x \text{ integer}, \; a_j > 0$$

I. DERIVATION OF AN EQUIVALENT KNAPSACK CONSTRAINT FROM ANY SET OF ILP CONSTRAINTS IN BOUNDED VARIABLES

Establishing the existence of the equivalent knapsack constraint does involve the aggregation of m equations into a single constraint.

In order to prove the existence of the desired vector h, Bradley [1] proposes that we consider the aggregation of the equations two at a time. Theorem 4.1 will enable us to reduce a set of two linear equations (4.6) to a single equivalent linear equation (4.7). In these equations, f and g are n-dimension coefficient vectors and δ and γ are constants, x is an n-dimension variable and C is an arbitrary set, α and β are integer.

$$f^T x = \delta$$

$$g^T x = \gamma \qquad (4.6)$$

$$x \varepsilon C,\ x \text{ integer}$$

$$(\alpha f + \beta g)^T x = \alpha\delta + \beta\gamma \qquad (4.7)$$

$$x \varepsilon C,\ x \text{ integer}$$

We will need these definitions

$$SP1 \equiv \sup(g^T x - \gamma) \qquad (4.8)$$

$$\text{for } x \varepsilon C,\ \text{sgn}(\beta)[f^T x - \delta) \leq -|\beta|,\ x \text{ integer}$$

$$SP2 \equiv \sup(g^T x - \gamma) \qquad (4.9)$$

$$\text{for } x \varepsilon C,\ \text{sgn}(\beta)[f^T x - \delta] \geq |\beta|,\ x \text{ integer}$$

$$SP3 \equiv \sup(f^T x - \delta) \qquad (4.10)$$

$$\text{for } x \varepsilon C,\ \text{sgn}(\alpha)[g^T x - \gamma] \geq |\alpha|,\ x \text{ integer}$$

$$SP4 \equiv \sup(f^T x - \delta) \qquad (4.11)$$

$$\text{for } x \varepsilon C,\ \text{sgn}(\alpha)[g^T x - \gamma] \leq -|\alpha|,\ x \text{ integer}$$

where sgn(β) is -1,0,1 depending on whether β is negative, zero, or positive respectively. If for some SPi there is no solution, or if the sup is negative, set SPi equal to zero. Rather than repeat much notation, we can define IFi by replacing "sup" in the above definitions with "inf" for i = 1,2,3,4. If for IFi there is no solution, or if the solution is positive, set IFi equal to zero.

THEOREM 4.1. If α and β are relatively prime integers, if at least one of these four conditions is satisfied

A1. $\alpha > SP1$

A2. $\alpha < IF1$

A3. $-\beta > SP3$

A4. $-\beta < IF3$

and if at least one of these four conditions is satisfied

B1. $-\alpha > SP2$

B2. $-\alpha < IF2$

B3. $\beta > SP4$

B4. $\beta < IF4$

then the set of solutions to (4.6) is identical to the set of solutions to (4.7).

PROOF. Clearly, if x is a solution to (4.6) then x is a solution to (4.7). We need to show that when x is a solution to (4.7), it is also a solution to (4.6).

Let $\bar{x}$ be an arbitrary solution to (4.7), then we know

$$(\alpha f + \beta g)^T \bar{x} = \alpha\delta + \beta\gamma$$

$$\alpha f^T \bar{x} + \beta g^T \bar{x} = \alpha\delta + \beta\gamma$$

$$\alpha f^T \bar{x} - \alpha\delta = -(\beta g^T \bar{x} - \beta\gamma)$$

and for some integer π

$$\alpha(f^T \bar{x} - \delta) = \pi \tag{4.12}$$

$$\beta(g^T \bar{x} - \gamma) = -\pi \tag{4.13}$$

Since $(f^T \bar{x} - \delta)$ and $(g^T \bar{x} - \gamma)$ are integer, we can say α divides π (i.e. $\frac{\pi}{\alpha}$ is integer), and β divides π. Since α and β are relatively prime and are both factors of π, we know that for some integer θ, $\pi = \alpha\beta\theta$. By substituting this value for π into (4.12) and (4.13), we get

$$f^T \bar{x} - \delta = \beta\theta \tag{4.14}$$

$$g^T \bar{x} - \gamma = -(\alpha\theta) \tag{4.15}$$

Now, if we can show that each condition A1 to A4 implies $\theta \geq 0$, and if each condition B1 to B4 implies $\theta \leq 0$, then the satisfaction of one condition from each group guarantees $\theta = 0$. If $\theta = 0$, (4.14) and (4.15) become equivalent to (4.6), and thus $\bar{x}$ is a solution to (4.6). Let us suffice to show that satisfaction of A1 implies $\theta \geq 0$ and that satisfaction of B4 implies $\theta \leq 0$.

Assume we meet condition A1. Assume $\theta < 0$. Since θ is an integer, we have from (4.14) $f^T\bar{x} - \delta = -\beta$ or -2β or ..., if $\beta > 0$, that is $f^T\bar{x} - \delta \geq -\beta$. If $\beta < 0$, $f^T\bar{x} - \delta \geq -\beta$. We can summarize these conditions by $\text{sgn}(\beta)[f^T\bar{x} - \delta] \leq -|\beta|$. From (4.15) and the definition of sup,

$$g^T\bar{x} - \gamma = -\alpha\theta \leq \sup(g^Tx - \gamma) \equiv SP1 \qquad (4.16)$$
$$\text{for } x \text{ integer}, x\varepsilon C, \text{sgn}(\beta)[f^Tx - \delta] \leq -|\beta|$$

We defined SP1 to be nonnegative so $\alpha > SP1 \geq 0$. If we now divide (4.16) by α, we get $-\theta \leq SP1/\alpha$. Since $0 \leq \frac{SP1}{\alpha} < 1$, we know $\theta > -1$, which is a contradiction. Therefore $\theta \geq 0$.

Assume we meet condition B4. Assume $\theta > 0$. Since θ is integer, we have from (4.15), if $\alpha > 0$, $g^T\bar{x} - \gamma \leq -\alpha$, or if $\alpha < 0$, $g^Tx - \gamma \geq -\alpha$. These conditions can be summarized by $\text{sgn}(\alpha)[g^Tx - \gamma] \leq -|\alpha|$. From (4.14) and the definitions of inf, we know

$$f^T\bar{x} - \delta = \beta\theta \geq \inf f^Tx - \delta \equiv IF4 \qquad (4.17)$$
$$\text{for } \text{sgn}(\alpha)[g^T\bar{x} - \gamma] \leq -|\alpha|, x\varepsilon C, x \text{ integer}$$

We defined IF4 to be nonpositive so we know $\beta < IF4 \leq 0$. If we divide (4.17) by β, we get $\theta \leq \frac{IF4}{\beta}$. Since $0 \leq \frac{IF4}{\beta} < 1$ we know $\theta < 1$. This conclusion is a contradiction. Therefore $\theta \leq 0$.

The proof that the other six conditions produce the desired bound on θ follows analogously.

This general result allows us to aggregate any two equations preserving the solution set by producing the proper relatively prime multipliers α and β. Notice that the existence of α and β is dependent upon the boundedness of C. Since we are concerned with an ILP in bounded variables our C certainly guarantees the existence of SPi and IFi, thus the existence of α and β. The following corollary to Theorem 4.1 guarantees the existence of the desired weight vector h.

COROLLARY 4.1. There exists an integer m-dimension vector h of weights, such that the set of solutions to the system of constraints

$$Ax = b,\ 0 \leq x \leq d,\ x \text{ integer}$$

is identical to the set of solutions to

$$h^T Ax = h^T b,\ 0 \leq x \leq d,\ x \text{ integer} \tag{4.18}$$

PROOF. We can combine the constraints two at a time applying Theorem 4.1 each time. Since the variables are bounded at each stage the SPi and IFi exist for i = 1,2,3,4 and α and β exist for each aggregation. By making (m - 1) such applications of Theorem 4.1 we have established the existence of h.

In fact, we can demonstrate one choice of h if we methodically combine the equations, always combining the first equation with the second, labelling the multipliers guaranteed by Theorem 4.1 as α_i and β_i for the i^{th} aggregation for i = 1,2,...,m-1. We find that h produced in this way has the precise form

$$h = \begin{pmatrix} \alpha_{m-1}\alpha_{m-2} \cdots \alpha_2\alpha_1 \\ \alpha_{m-1}\alpha_{m-2} \cdots \alpha_2\beta_1 \\ \alpha_{m-1}\alpha_{m-2} \cdots \alpha_3\beta_2 \\ \vdots \\ \alpha_{m-1}\alpha_{m-2}\beta_{m-3} \\ \alpha_{m-1}\beta_{m-2} \\ \beta_{m-1} \end{pmatrix} \tag{4.19}$$

We have now proven that for any ILP in bounded variables we can write a knapsack formulation (4.2), which is guaranteed to have the same solution set as (4.1). Moreover we have even produced one choice of h which will allow us to generate (4.2).

II. ALGORITHM FOR SOLVING THE KNAPSACK PROBLEM IN BOUNDED VARIABLES

Let us refer to the problem in the form (4.5). The algorithm must include the special condition of boundedness, and will be based on the recursive equation (4.4). Recall that we must initialize $f_0(0) = 0$ and $f_0(y) = -\infty$ for $y > 0$. Also, we may allow any of the objective coefficients c_k to be negative with no difficulty. But, we must guarantee that all $a_j > 0$ for $j = 1,2, \ldots,n$. If some $a_j < 0$ we can substitute $x_j' = d_j - x_j$ for x_j to guarantee all constraint coefficients to be positive.

Thus we can apply the following algorithm to any knapsack problem of the form (4.5).

1) Set $k = 1$, $f_0(0) = 0$, $f_0(y) = -\infty$ for $y > 0$, then set $y = 1$.

2) Set $p = \min(d_k, [\frac{y}{a_k}])$, set $q = 0$.

3) Compute $f_k(y)_q = c_k q + f_{k-1}(y - a_k q)$.

4) If $q = p$, set $f_k(y) = \max_{q=1,2,\ldots,p} f_k(y)_q$ and save the maximizing value of q as $q_k(y)$ then go to 5). If $q < p$, set $q = q + 1$ and return to 3).

5) If $y < b$, set $y = y + 1$ and go to 2). If $y = b$, go to 6).

6) If $k < n$ set $k = k + 1$ and $y = 1$ return to 2). If $k = n$ go to 7).

7) Set $x_k = q_k(y)$. If $k = 1$, terminate. If $k > 1$, go to 8).

8) Set $y = y - a_k x_k$, set $k = k - 1$ and go to 7).

Note that the value in $f_k(y)$ as we enter 7) is the optimal value $f_n(b)$. The last two steps determine what value of x_j for $j = 1,2,\ldots,n$, contributes to the optimal value $f_n(b)$. Thus, we have determined the optimal solution and optimal value of the knapsack problem in bounded variables. Certainly, many of the values $f_k(y)_q$ will be $-\infty$, and it is meaningless to consider $-\infty$ in a group of terms for which we seek a maximum. Thus, when $-\infty$ occurs in our table we will simply record a dashed line.

EXAMPLE 4.1

Let us look at an example, which will serve as an illustration of converting a problem of the form (4.1) to one of the form (4.5). Then we will solve our knapsack problem using the algorithm above.

$$\max z = 2x_1 + 6x_2$$

$$-x_1 + x_2 = 1 \qquad (4.20)$$

$$x_1 + x_2 = 3$$

$$0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 2,\ x_1,\ x_2 \text{ integer}$$

To fit the notation of Theorem 4.1 we can set $f^T = (-1,1)$ and $g^T = (1,1)$. Choosing $\alpha = 2$ and $\beta = 5$, we need to verify that at least one condition from each group in Theorem 4.1 is satisfied. Only one choice for x meets the restrictions of (4.8), that is $x^T = (4,0)$. So we have $SP1 = (1,1)\begin{pmatrix}4\\0\end{pmatrix} - 3 = 1$. Now $\alpha > SP1$ so we do satisfy condition A1. Three choices for x meet the restrictions of (4.11); these are $x^T = (0,0)$, $x^T = (0,1)$, and $x^T = (1,0)$. We can compute

$$\sup[(-1,1)\begin{pmatrix}x_1\\x_2\end{pmatrix} - 1] = \sup(-1,0,-2) = 0 = SP4\ .$$

Now, our choice of $\beta = 5$ certainly meets condition B3. Thus by Theorem 4.1, we are guaranteed that (4.21) has exactly the same solution set as (4.20).

$$\max z = 2x_1 + 6x_2$$

$$3x_1 + 7x_2 = 17 \qquad (4.21)$$

$$0 \leq x_1 \leq 4,\ 0 \leq x_2 \leq 2, \qquad x_1,\ x_2 \text{ integer}$$

We can solve (4.21) by the algorithm derived above. We will use the convention mentioned, and place a dash in any position where the

value $-\infty$ would appear. Tables 4.1 and 4.2 give the values for $f_k(y)_q$ derived from the algorithm; the values for $f_k(y)$ appear in the right-hand column. The values for $q_k(y)$ are not recorded separately, since in a visual table they can easily be read from the column heading.

Reading from the tables we find the optimal value $f_2(17) = 14 = z^*$. For $f_2(17)$ we see that $q_2(17) = 2 = x_2^*$, then y becomes $17 - a_k \cdot x_k = 17 - 14 = 3$. We find then that $f_{k-1}(y) = f_1(3)$ and $q_1(3) = 1 = x_1^*$. We thus have the optimal value and solution to 4.21 which we are guaranteed are equivalent to those of 4.20, $z^* = 14$, $x_1^* = 1$, and $x_2^* = 2$.

Table 4.1. Hand Computation of $f_k(y)_q$ by the Algorithm (A)

y	p	$f_1(y)_0$	$f_1(y)_1$	$f_1(y)_2$	$f_1(y)_3$	$f_1(y)_4$	$f_1(y)$
0	0	0					0
1	0	-					-
2	0	-					-
3	1	-	(2)				(2)
4	1	-	-				-
5	1	-	-				-
6	2	-	-	4			4
7	2	-	-	-			-
8	2	-	-	-			-
9	3	-	-	-	6		6
10	3	-	-	-	-		-
11	3	-	-	-	-		-
12	4	-	-	-	-	8	8
13	4	-	-	-	-	-	-
14	4	-	-	-	-	-	-
15	4	-	-	-	-	-	-
16	4	-	-	-	-	-	-
17	4	-	-	-	-	-	-

Table 4.2. Hand Computation of $f_k(y)_q$ by the Algorithm (B)

y	p	$f_2(y)_0$	$f_2(y)_1$	$f_2(y)_2$	$f_2(y)$
0	0	-			-
1	0	-			-
2	0	-			-
3	0	2			2
4	0	-			-
5	0	-			-
6	0	4			4
7	1	-	6		6
8	1	-	-		-
9	1	6	-		6
10	1	-	8		8
11	1	-	-		-
12	1	8	-		8
13	1	-	10		10
14	2	-	-	12	12
15	2	-	-	-	-
16	2	-	12	-	12
17	2	-	-	14	14

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Bradley, Gordon H., "Transformation of Integer Programs to Knapsack Problems," Discrete Mathematics, Vol. 1, No. 1, pp. 29-45, 1971.

2. Burdett, Claude, "On Cutting Planes," Faculty of Management Sciences, University of Ottawa, Working Paper 73-22.

3. Cooper, Leon and Steinberg, David, Linear Programming. Philadelphia: W. B. Saunders Company, 1974.

4. Dantzig, G., "Maximization of a Linear Functional of Variables Subject to Linear Inequalities," Activity Analysis of Production and Allocation, edited by T. C. Koopmans. New York: John Wiley and Sons, 1951.

5. Garfinkel, R. S., and Nemhauser, G. L., Integer Programming. New York: John Wiley and Sons, 1972.

6. Gass, Saul I., Linear Programming. New York: McGraw-Hill Book Company, 1969.

7. Gomory, R. E., "Outline of an Algorithm for Integer Solutions to Linear Programs," Bulletin of the American Mathematical Society, Vol. 64, pp. 275-278, 1958.

8. Gomory, R. E., "All-Integer Programming Algorithm," Industrial Scheduling, edited by J. F. Muth and J. L. Thompson. Englewood Cliffs, New Jersey: Prentice Hall, 1963.

APPENDIX

THE DUAL SIMPLEX METHOD

The dual simplex method is often used to solve the LP, when no feasible basis can be found for the primal problem. For instance, if the optimal function is to be maximized and all $(z_j - c_j) \geq 0$, the dual problem can often be solved. This is the case when the cutting plane method is used to solve the ILP. Since the starting point is an optimal tableau for the LP, this means all $(z_j - c_j)$ must be nonnegative.

If the primal problem is stated as (A.1), then its associated dual is stated as (A.2). Associated with these two forms

$$\begin{aligned} &\max\ f(x) = c^T x \\ &Ax = b \\ &x_i \geq 0 \qquad \text{for } i = 1,2,\ldots,n \end{aligned} \tag{A.1}$$

$$\begin{aligned} &\min\ g(v) = b^T v \\ &A^T v \geq c \end{aligned} \tag{A.2}$$

is an important theorem.

DUALITY THEOREM. If either the primal or dual problem has a finite optimal solution, then the other problem has a finite optimal solution, and the optima of the linear functionals are equal. That is, $\max f(x) = \min g(v)$.

Now, using this principle of duality, the following algorithm has been established for solution to the dual problem.

1. Determine an initial basic solution to the primal problem for which $z_j - c_j \geq 0$ for all j, and not necessarily feasible in that some x_{i0} may be less than zero.

2. The vector to be removed from the basis will be the a_i associated with x_{r0} such that:

$$x_{r0} = \min\{x_{i0} | x_{i0} < 0\}$$

3. The vector to enter the basis is determined to be the a_k such that:

$$\frac{z_k - c_k}{x_{rk}} = \max\{\frac{z_j - c_j}{x_{rj}} | x_{rj} < 0\}$$

4. The simplex transformation is then performed using x_{rk} as the pivot element.

5. Continue until a feasible solution is reached. That is:

$x_{i0} \geq 0$ for all i.

VITA

David Frank Hassell was born in Florence, Alabama, on April 15, 1951. He attended elementary school in that town and graduated from Bradshaw High School of Florence in May, 1969. He received a Bachelor of Arts degree in Mathematics from The University of Tennessee in August, 1974.

In September, 1974, he accepted a graduate teaching assistantship in Mathematics at The University of Tennessee, where he was selected as a member to the American Mathematical Society. He received the Master of Science degree from that school in June, 1976.