To the Graduate Council:

I am submitting herewith a thesis written by Kerwin Everson, entitled "Solving 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:

Julius Smith

Steven M. Serbin

Accepted for the Council:

L. Evans Peck

Vice Chancellor
Graduate Studies and Research

# SOLVING THE INTEGER LINEAR PROGRAMMING PROBLEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Kerwin Everson

August 1979

## ABSTRACT

The purpose of this thesis is to consider both the development and the workings of some selected methods to solve the integer linear programming problem.

Chapter I introduces the general theory of linear programming. Both the simplex and the dual simplex method are introduced along with a summary of the general steps of the algorithms.

Chapter II is an introduction to the integer linear programming problem. The characteristics of the integer problem are discussed as well as an overview of the methods used to solve them.

Chapter III is concerned with two specific methods used to solve integer problems. They are cutting plane methods and apply to the pure and the mixed integer problem respectively. Both are developed in detail and an example of each is given.

Chapter IV is also concerned with a specific method. The method is a combination of a primal method with an enumerative method. The theory behind the method is discussed and an example is given.

TABLE OF CONTENTS

LIST OF FIGURES

# CHAPTER I

## INTRODUCTION TO LINEAR PROGRAMMING

### I. THE LINEAR PROGRAMMING PROBLEM

The general linear programming problem is concerned with finding a solution to a system of equations (or inequalities), which are linear, that maximizes (or minimizes) some given function which is also linear. The general linear programming problem is as follows:

Maximize (or minimize)

$$f(x) = c_1x_1 + c_2x_2 + \ldots + c_px_p$$

Subject to:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p \{\leq, =, \text{ or } \geq\} b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2p}x_p \{\leq, =, \text{ or } \geq\} b_2$$

$$\vdots \qquad\qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mp}x_p \{\leq, =, \text{ or } \geq\} b_m$$

$$x_j \geq 0, \; j = 1,2,\ldots,p$$

where $a_{ij}$, $b_j$, $c_j$ are all real numbers.

For each constraint $\sum_{j=1}^{p} a_{ij}x_j \{\leq, =, \text{ or } \geq\} b_i$, $i = 1,2,\ldots,m$ only one of the relations $\{\leq, =, \text{ or } \geq\}$ will occur at any given time.

A geometric representation of the general linear programming problem can be instructive in that it helps to exhibit the nature of the solutions to the problem. Consider for example the following problem:

Maximize: $z = 3x_1 + 4x_2$

Subject to: $x_1 + x_2 \leq 10$

$-x_1 + 6x_2 \leq 18$

$x_1, x_2 \geq 0$

The shaded area in Figure 1-1 represents the set of all possible feasible solutions. A feasible solution is simply a vector $\bar{x} = (x_1, x_2, \ldots, x_p)$ which satisfies all the constraints of the linear programming problem. A basic feasible solution is a feasible solution with no more than m positive $x_j$ and such that the vectors $\bar{a}_j$ associated with these latter $x_j$ are a linearly independent set of vectors.

The set of all possible feasible solutions is always a convex set. In fact the optimal solution will occur at a corner (extreme point) of the convex set of all possible feasible solutions. In this case, the optimal value of $z^* = 34$ occurs at the optimal point $\bar{x}^* = (x_1^*, x_2^*) = (6,4)$.

The following theorems will also give further insight into the nature of the solutions to linear programming

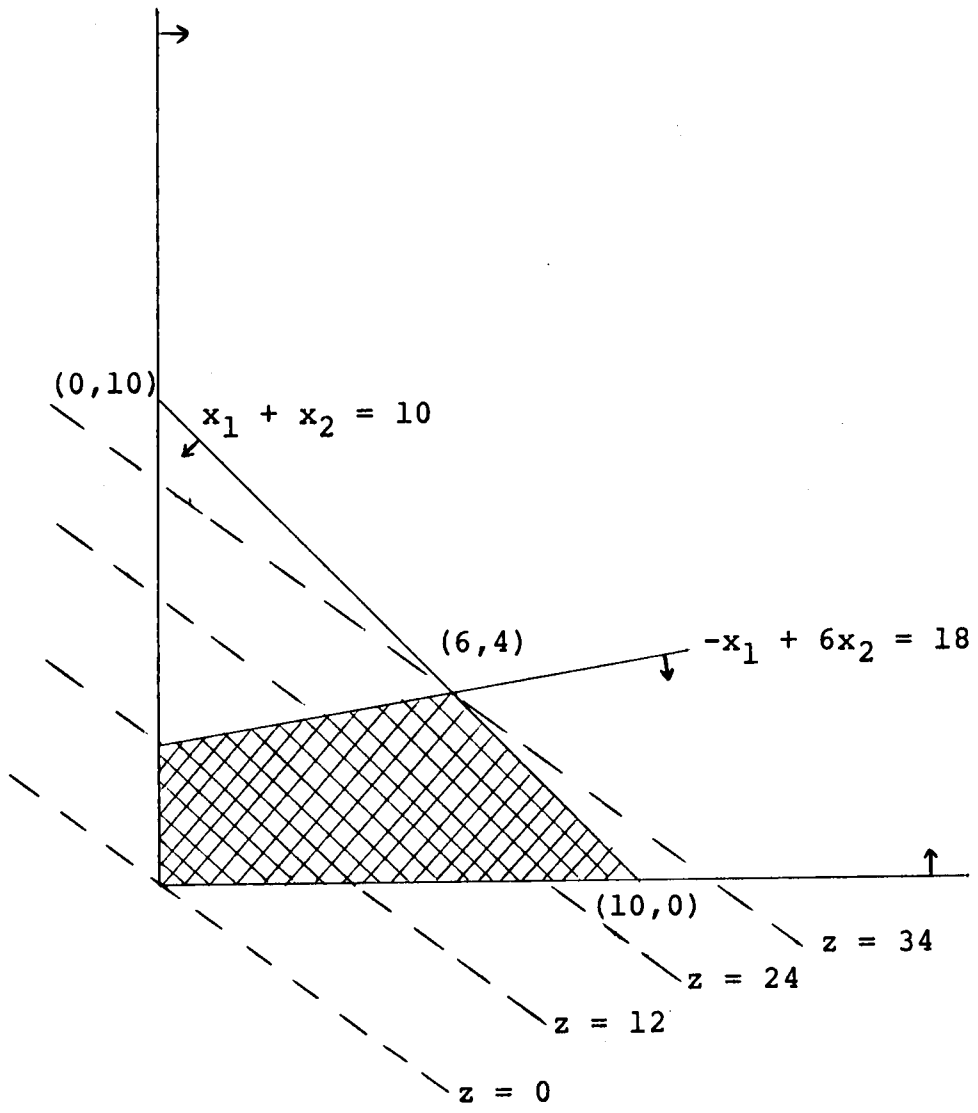


Figure 1-1. Geographical Interpretation of Feasible Region and Optimal Value z.

problems. The proofs are found in most books containing introductory material on linear programming. (See [3])

First, a linear programming problem written in "standard form" is

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

$$\text{Subject to: } \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1,2,\ldots,m$$

$$x_j \geqslant 0 \quad j = 1,2,\ldots,n$$

or in equivalent matrix notation, where $\bar{x}$ is a row vector ($\bar{x} = [x_1, x_2,\ldots,x_n]$) and $\bar{x}'$ is a column vector

$$(\bar{x}' = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix})$$

is

$$\text{Maximize } z = \bar{c}'\bar{x}$$

$$\text{Subject to: } A\bar{x} = \bar{b}$$

$$\bar{x} \geqslant 0$$

The constraints above, in both the summation notation and the matrix notation, are strict equalities because appropriate "slack" or "surplus" variables have been added

to any existing inequalities. These forms will make some of the notation in the theorems which follow more clear.

Theorem 1

Given a set of constraints of the form

$$D\bar{x}\{\leq, =, \text{ or } \geq\}\bar{d}$$

$$\bar{x} \geqslant 0 \qquad \text{(a.)}$$

(where D is an m x p matrix, $\bar{d}$ is an m-vector, and $\bar{x}$ is a p-vector) and also the constraint set obtained by adding n-p slack or surplus variables

$$A\bar{y} = \bar{d}$$

$$\bar{y} \geqslant 0 \qquad \text{(b.)}$$

where A is an m x n matrix and $\bar{y}$ is an n-vector. Then every solution to (a.) corresponds to a solution to (b.) and conversely.

Theorem 2

The set of points $\bar{x}$ satisfying $A\bar{x} = \bar{b}$, $\bar{x} \geqslant 0$ forms a convex set.

Theorem 3

The objective function of a linear programming problem assumes its maximum at an extreme point of the convex set of feasible solutions.

Theorem 4

If there exists a set of $k \leq m$ linearly independent columns of A, namely, $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_k$, such that

$$x_1\bar{a}_1 + x_2\bar{a}_2 + \ldots + x_k\bar{a}_k = b, \text{ with all } x_j \geq 0,$$

then the point $\bar{x} = (x_1, x_2, \ldots, x_k, 0, \ldots, 0)$ is an extreme point of $X_c$, the convex set of feasible solutions.

Theorem 5

Let $\bar{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ be an extreme point of $X_c$. Then the columns of A which are associated with positive $x_j^*$ are linearly independent, with at most m of the $x_j^*$ being positive.

## II. THE SIMPLEX ALGORITHM

The simplex method is an iterative process in which at each stage we have a basic feasible solution to the standard form of the linear programming problem on page 3. After a determination of optimality, one is finished if the current solution is optimal and must proceed if not. The simplex method then searches for a "better" basic feasible solution, i.e. one that yields a greater value of z. This process is necessarily finite since there are only finitely many basic feasible solutions. The simplex method moves from one extreme point (if it is not optimal) to an adjacent extreme point.

Both the theoretical and computational development of the simplex method are given by Cooper and Steinberg [3]. However, included here will be a brief outline of the development and the basic algorithm itself.

Beginning with a basic feasible solution to $A\bar{x} = \bar{b}$ and assuming for simplicity that the first m columns of A are in the basis matrix B (where B consists of m linearly independent columns of A and hence is a basis for $E^m$), then

$$\sum_{i=1}^{m} x_{Bi}\bar{a}_i = \bar{b}.$$

To find another basic feasible solution, vectors currently in B must be replaced by vectors in N (the remaining n-m columns of A). Geometrically this will mean that the method, upon replacement of one vector by another, is moving from one extreme point to an adjacent one.

The decisive criteria in choosing both the vector to leave the basis B, in order to find an improved basic feasible solution, are the retention of a basic feasible solution at each step and the insurance of a larger value of z. These criteria are taken into consideration in the detailed theoretical development of the simplex method, and they are included in the outline of the algorithm which will follow a brief look at the simplex tableau.

There are many ways in which the simplex tableau may be put together. Cooper and Steinberg [3] use the following form (see Figure 1-2):

| | | | $c_1$ | $c_2 \cdots$ | $c_k \cdots$ | $c_n$ |
|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2 \cdots$ | $\bar{y}_k \cdots$ | $\bar{y}_n$ |
| $c_{B1}$ | $\bar{b}_1$ | $x_{B1}$ | $y_{11}$ | $y_{12}\cdots$ | $y_{1k}\cdots$ | $y_{1n}$ |
| $c_{B2}$ | $\bar{b}_2$ | $x_{B2}$ | $y_{21}$ | $y_{22}\cdots$ | $y_{2k}\cdots$ | $y_{2n}$ |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| $c_{Bm}$ | $\bar{b}_m$ | $x_{Bm}$ | $y_{m1}$ | $y_{m2}\cdots$ | $y_{mk}\cdots$ | |
| | | $z$ | $z_1-c_1$ | $z_2-c_2$ | $z_k-c_k$ | $z_n-c_n$ |

Figure 1-2. Simplex Tableau.

Note in Figure 1-2 that under the column headed "Basis" are the vectors $\bar{a}_j$ which are currently in the basis. For example, $\bar{b}_1$ could be $\bar{a}_6$, $\bar{b}_2$ could be $\bar{a}_3$, etc. The values of $x_{Bi}$ in the next column are then the values of $x_6$, $x_3$, etc., that are in the basic feasible solution, $\bar{x}_B \geqslant 0$. Also, the values of $c_{Bi}$ in the very first column are the corresponding prices of the basic variables. The remaining columns give the components $y_{ij}$ of the vectors $\bar{y}_j = B^{-1}\bar{a}_j$. Under the $\bar{x}_B$ column is listed the current value of the objective function. Finally, the entries in the last row, $z_j - c_j$,

which determine optimality are found by

$$z_j = \sum_{i=1}^{m} y_{ij} c_{Bi} = c_B' \bar{y}_j$$

and $c_j$ being the jth cost coefficient.

## Summary of the Simplex Algorithm

1. Determine an initial basic feasible solution, $\bar{x}_0$ by some method. (Several methods are given in [3].)

2. For those vectors not in the basis, $j \in J_N$, calculate $z_j - c_j$. If the $z_j - c_j \geq 0$, for all j, then the current solution is optimal.

3. If one or more $z_j - c_j < 0$, $j \in J_N$, select a vector to enter the basis, $\bar{a}_K$, using the criterion

$$z_k - c_k = \min_{j \in J_N} \{z_j - c_j \mid z_j - c_j < 0\}$$

4. If all $y_{ik} \leq 0$, an unbounded maximum exists. If at least one $y_{ik} > 0$, select the vector to be removed from the basis, $\bar{a}_r$, by the criterion

$$\frac{x_{Br}}{y_{rk}} = \min_i \left\{ \frac{x_{Bi}}{y_{ik}} \middle| y_{ik} > 0 \right\}$$

5. With the new basis $B_1$ formed from $B_0$ by removing $\bar{a}_r$ and replacing it with $\bar{a}_k$, compute the new solution $\bar{x}_1$, new values of $y_{ij}$, $z_j - c_j$, and z. (For transformation formulae, see [3].)

6. Return to step 3.

An example is now given simply to show how the process works and progresses toward the optimal tableau, and hence the optimal solution. The transformation from one tableau to the next is not included, and again the formulae to do so are found in [3]. Also an initial basic feasible solution is obtained to begin the process by a two-phase method.

Example 1.1

$$\begin{aligned} \text{Maximize} \quad z = {} & x_1 + 2x_2 + 4x_3 + 6x_4 \\ & 2x_1 + 3x_2 + 4x_3 + x_4 = 26 \\ & x_1 + 5x_2 + x_3 + 4x_4 \leqslant 20 \\ & 3x_1 + x_2 + 3x_3 + 3x_4 \leqslant 30 \\ & x_1, x_2, x_3, x_4 \geqslant 0 \end{aligned}$$

In standard form obtain,

$$\begin{aligned} \text{Maximize} \quad z = {} & x_1 + 2x_2 + 4x_3 + 6x_4 \\ & 2x_1 + 3x_2 + 4x_3 + x_4 = 26 \\ & x_1 + 5x_2 + x_3 + 4x_4 + x_5 = 20 \\ & 3x_1 + x_2 + 3x_3 + 3x_4 + x_6 = 30 \\ & x_j \geqslant 0, \; j = 1,2,\ldots,6 \end{aligned}$$

Tableau I

| | | | 1 | 2 | 4 | 6 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ | $\bar{y}_5$ | $\bar{y}_6$ |
| 4 | $\bar{a}_3$ | 26/4 | 2/4 | 3/4 | 1 | 1/4 | 0 | 0 |
| 0 | $\bar{a}_5$ | 54/4 | 2/4 | 17/4 | 0 | (15/4) | 1 | 0 |
| 0 | $\bar{a}_6$ | 42/4 | 6/4 | -5/4 | 0 | 9/4 | 0 | 1 |
| | | 26 | 1 | 1 | 0 | -5 | 0 | 0 |

To determine the vector to enter the basis, check (in Tableau I):

$$z_k - c_k = \min\{z_j - c_j \mid z_j - c_j < 0\} = \min\{-5\} = -5 = z_4 - c_4.$$

Therefore, $\bar{a}_4$ enters the basis. Similarly, to determine the vector to leave the basis, check:

$$\frac{x_{Br}}{y_{r4}} = \min\left(\frac{26/4}{1/4}, \frac{54/4}{15/4}, \frac{42/4}{9/4}\right) = \frac{54}{4} = \frac{x_{B2}}{y_{24}}.$$

So $\bar{a}_5$ leaves the basis, (see Tableau I). After calculating the new values for Tableau II, obtain:

Tableau II

| | | | 1 | 2 | 4 | 6 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ | $\bar{y}_5$ | $\bar{y}_6$ |
| 4 | $\bar{a}_3$ | 84/15 | 7/15 | 7/15 | 1 | 0 | -1/15 | 0 |
| 6 | $\bar{a}_4$ | 54/15 | 2/15 | 17/15 | 0 | 1 | 4/15 | 0 |
| 0 | $\bar{a}_6$ | 36/15 | 18/15 | 57/15 | 0 | 0 | -9/15 | 1 |
| | | 44 | 25/15 | 100/15 | 0 | 0 | 20/15 | 0 |

Since $z_j - c_j > 0$ for all j (see bottom row of Tableau II), we have an optimal solution (see step 2 of the algorithm). The optimal solution is $x_1^* = 0$, $x_2^* = 0$, $x_3^* = 84/15$, $x_4^* = 54/15$, $x_5^* = 0$, and $x_6^* = 36/15$. To check, $z^* = x_1^* + 2x_2^* + 4x_3^* + 6x_4^* = 44$, substitute the values into z, or notice $z^* = 44$ under the $\bar{x}_B$ column.

Note: In this example problem $x_6 > 0$ and $x_6$ is a slack variable. This implies that one of the resources is not being fully allocated in order to achieve an optimal solution.

The final topic to be discussed in this overview of linear programming in the general case, will be that of duality. Duality theory in linear programming is a very important topic, and is discussed in detail in most linear programming books.

## III. THE DUAL SIMPLEX ALGORITHM

Along with all linear programming problems there exists a related linear programming problem which is called the "dual" of the original linear programming problem. The original problem is then called the "primal" problem.

These two problems are different in that one is a maximization problem and the other is a minimization problem. Actually, these two problems are simply rearrangements of the same basic data. If one has solved either of the two problems, then the optimal tableau of the simplex method contains all

of the information required to calculate the solution of the unsolved problem. Additional detailed information concerning the dual problem and how to formulate it from the primal can be found in most any linear programming book.

Also, there is a dual simplex method which can be used to solve the linear programming problem. It is basically just the reverse of the simplex method, which is generally called the primal simplex method when compared with the dual simplex method. The dual simplex method is not used to solve the general linear programming problem. Many times though, the situation will arise in which the dual simplex method will be superior to or required over the primal simplex algorithm. A summary of the algorithm will now be given.

## Summary of the Dual Simplex Algorithm

1. Determine an initial basic solution to the primal problem such that $z_j - c_j \geq 0$ for all j and one that is not necessarily feasible in the sense that one or more of the $x_{Bi}$ may be less than 0.

2. The vector to be removed from the basis is determined first. The rule is somewhat arbitrary but most commonly is given by

$$x_{Br} = \min_i \{x_{Bi} \mid x_{Bi} < 0\}$$

Thus $\bar{b}_r$, i.e., column r is removed from B and $x_{Br}$ becomes zero.

3. The vector to enter the basis is determined from

$$\theta = \frac{z_k - c_k}{y_{rk}} = \max_j \left\{ \frac{z_j - c_j}{y_{rj}} \middle| y_{rj} < 0 \right\}$$

4. All quantities of interest are transformed by the usual simplex transformation formulae.

5. When a basic solution is obtained that is also feasible, i.e., $\bar{x}_B \geqslant \bar{0}$, then the solution obtained is the optimal basic feasible solution.

# CHAPTER II

# INTEGER LINEAR PROGRAMMING

## I. INTRODUCTION

In the general linear programming problem the variables are allowed to be continuous. Geometrically, they can take on any value in the convex set of all feasible solutions. However, this does not make sense in many practical problems. Many situations arise for which solutions other than integer values would not be feasible. For example, one would not want to purchase a fraction of a machine or ship a fraction of an automobile.

A linear programming problem in which some (or all) of the variables are required to take on integer values is called an integer linear programming problem. If all of the variables are required to take on integer values, then the problem is called a pure integer programming problem. If some of the variables can take on values other than integer values, then the problem is called a mixed integer programming problem.

In some problems an integer solution may be obtained automatically, but this is usually not the case. The extreme points of the convex set of all possible feasible solutions to a general linear programming problem are not restricted

to (and usually do not take on) only integer values. Further, the following example points out that the fractional values of an optimal solution of a linear programming problem cannot simply be rounded to produce the optimal solution of integer values.

Example 2.1

Maximize $z = 9x_1 + 14x_2$

Subject to: $-9x_1 + 33x_2 \leqslant 33$

$11x_1 + 33x_2 \leqslant 198$

The optimal solution turns out to be $z^* = 119.75$ and occurs at $x_1^* = 8.25$, $x_2^* = 3.25$. Now consider the same example only suppose we want the optimal integer solution. If we try to round off the values (up or down) of $x_1^*$ and $x_2^*$, and look at the set of all possible rounded integer solutions, namely {(8,3), (8,4), (9,3), (9,4)}, observe that every element of this set is infeasible. That is, all points in the set above lie outside the convex set of all possible feasible solutions (see Figure 2.1). The nearest feasible integer solution to (8.25, 3.25) is (7,3) which yields $z = 105$. However, this is not the optimal integer solution either. In fact the optimal solution is (6,4) which yields $z^* = 110$.

Thus, this example illustrates that simply rounding off the optimal solution in the unrestricted case will not suffice to serve as the optimal solution in the restricted

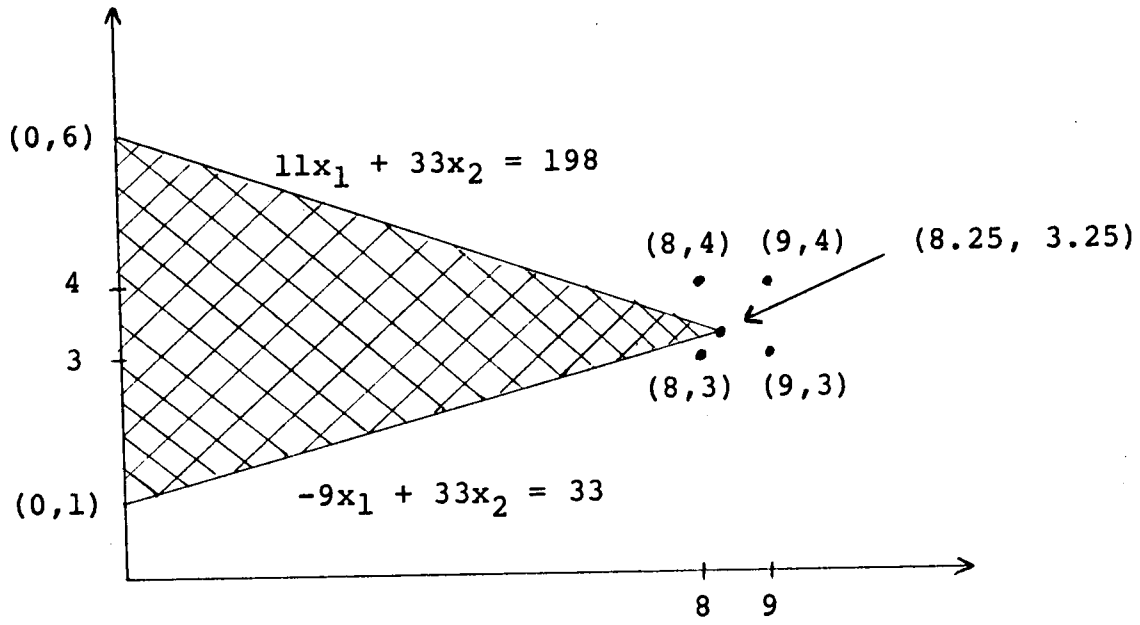


Figure 2-1. Geometrical Interpretation of Example 2.1.

case of integer values only. In fact rounding the solution may not even yield a feasible solution. Thus we cannot rely on the simplex method to solve the integer linear programming problem.

A variety of algorithms have been developed to solve integer linear programming problems. Unlike the situation discussed in Chapter I, where there is one basic algorithm (the simplex method) which is used more often than any others to solve the problems, there are a great many algorithms that are quite different from one another to solve the integer linear programming problem. However, no one algorithm has proven to be superior to the others, nor has any one algorithm proven to be capable of being consistently able to solve large integer linear programming problems. Generally speaking, problems with more than several hundred integer variables cannot be solved, except in some very special cases.

## II. OVERVIEW OF INTEGER ALGORITHMS

Approaches for solving the integer linear programming problem are basically divided into 2 areas:

1. Cutting plane methods, and
2. Enumerative methods.

The basic idea of the cutting plane method is to derive some new constraints. These constraints are called "cutting planes" or simply "cuts." They are derived so that the

resulting new feasible region for the related linear programming problem (the same problem without the integer restrictions) has an optimal extreme point with the desired integer properties. Some of the methods (within the cutting plane framework) work as follows. One method is to construct the initial simplex tableau so that it contains only integer values, and then maintain this property throughout the process by using the cutting plane constraints. This method only applies to pure integer linear programming problems. Other methods do not require the elements in the simplex tableau to be integers. These methods are the most widely used and two such methods are to be discussed in this paper. They are both methods developed by Gomory. Many persons have revised Gomory's algorithms. Most of the revisions require a lot of extra work which must be weighted against the anticipated decrease in the number of iterations. In fact the Primal Algorithm given by Ryan [7] presents the idea that the extra effort of computing these "fancier cuts" is not warranted.

While cutting plane methods have not been able to achieve the results for computer calculations obtained by methods of direct search (for which widespread commercial programs now exist), using the concept of Gomory's cut as an exclusion criterion in direct methods has proven itself helpful. The performance of these cutting plane algorithms

is completely unpredictable, which leads to the next type of approach.

The other basic approach to integer linear programming problems is the enumerative method. Enumerative methods can be divided into two types:

1. Branch and bound methods, and
2. Search or implicit enumeration methods.

Most of the more recently developed commercial computer programs for solving integer linear programming problems are based on an enumerative method by Dakin.

Enumeration methods consist of a partial search for all possible solutions, many of which can be eliminated by certain bounds on the variables. Basically, all possible integer solutions are enumerated either explicitly or implicitly. Throughout this process of enumeration, integer solutions which are feasible occur, and these are saved for future comparisons with other integer feasible solutions generated later. A set is said to have been <u>implicitly enumerated</u> once the entire set has been shown to contain no better solutions than those which are currently known. Solutions in a set which has been implicitly enumerated, then do not need to be evaluated individually or <u>explicitly</u>. The implicit enumeration is accomplished when one shows that all solutions contained in a particular set violate certain relations which have been derived by constraints that are necessary for an improved solution to exist.

The effectiveness of enumeration methods depends a great deal on the ability of the method to eliminate, in the above described way, a large percentage of solutions without having to explicitly evaluate them. The main advantage of enumeration methods is that a feasible approximation to the solution is available at any stage prior to completion of the algorithm. This is not the case with a cutting plane method.

# CHAPTER III

## GOMORY'S CUTTING PLANE ALGORITHMS

### I. INTRODUCTION

In this chapter, two of Gomory's cutting plane algorithms will be presented. The first is applicable only to pure integer linear programming problems, while the second one is for mixed linear programming problems but can be made applicable to pure integer linear programming problems as well. Not only were these methods among the first algorithms devised for solving integer linear programming problems (1959) and therefore have some historical interest, but these methods are also still the most effective algorithms for solving large scale problems. However, as mentioned in the last chapter, the performance of these algorithms is unpredictable.

To begin with, we consider first the pure integer linear programming problem:

Maximize $z = \bar{c}' \bar{x}$

Subject to: $A\bar{x} = \bar{b}$

$\bar{x} \geq \bar{0}$

$x_j$ integer-valued for $j = 1,2,\ldots,n$.

Further, the above problem without the integer restrictions shall be referred to as the related linear programming problem. The basic idea behind the cutting

plane approach is: given an integer linear programming problem as above, first solve the related linear programming problem. If the optimal solution to that problem is integer-valued then clearly that must be the optimal integer solution to the integer linear programming problem given originally. If the optimal solution contains at least one value that is not an integer, a new constraint (the nature of which will be discussed in detail below) is added to the problem, and then the new related linear programming problem is solved. As before, the optimal solution is checked for integer values. If the optimal solution contains only integer values, then again this solution is the optimal integer-valued solution. If this is not the case, then the process is repeated until one of the related linear programming problems has an optimal solution which contains only integer values.

In order for the above described process to be valid, the new constraints mentioned must possess the following two properties:

1. Every feasible integer solution to the given problem must also be a feasible solution to the problem developed after the addition of the new constraint.

2. The optimal solution to the related linear programming problem that must be solved at each step, must become infeasible after the new constraint has been added.

Property (1) simply says that when the new constraint is added it must not eliminate any of the integer solutions

from the set of all possible feasible solutions. Therefore, this will not cause the optimal integer solution to be made infeasible because of the addition of a new constraint. Property (2) is necessary because if it were not true (i.e. if the addition of the new constraint did not eliminate the optimal linear programming solution), then the old optimal solution would also be the solution to the new related linear programming problem. Since the old solution is not an all integer-valued solution, then nothing would be gained by the addition of the constraint. Thus property (2) must hold.

## II. PURE INTEGER ALGORITHM

Now Gomory's cutting plane method for pure integer linear programming problems will be developed in detail. For simplicity in this development, we shall assume that the optimal solution to the related linear programming problem is such that the first m columns of the matrix A are contained in the basis. Therefore, the matrix A may be partitioned so that it contains two submatrices as A = [B,R]. Now B contains the first m columns of A while R contains the remaining (n-m) columns of A. Therefore, the corresponding optimal solution could be written as $\bar{x}^* = [\bar{x}_B^*, \bar{x}_R^*]$, where $\bar{x}_R^* = \bar{0}$. Also since $A\bar{x} = \bar{b}$ we could write

$$[B,R] \begin{bmatrix} \bar{x}_B^* \\ \bar{x}_R^* \end{bmatrix} = \bar{b} \qquad (1)$$

That is,

$$B\bar{x}_B^* + R\bar{x}_R^* = \bar{b} \qquad (2)$$

Premultiply this by $B^{-1}$, (i.e., on the left) obtain

$$\bar{x}_B^* + (B^{-1}R)\ \bar{x}_R^* = B^{-1}\bar{b}$$

or

$$\bar{x}_B^* = B^{-1}\bar{b} - (B^{-1}R)\bar{x}_R^* \qquad (3)$$

Since we have already said that R consists of the last (n-m) columns of the matrix A, then equation (3) can be expressed in terms of the columns of the simplex tableau. Let $\bar{y}_j = B^{-1}\bar{a}_j$ and $\bar{y}_0 = B^{-1}\bar{b}$, to obtain

$$\bar{x}_B^* = \bar{y}_0 - \sum_{j \in N} \bar{y}_j x_j \qquad (4)$$

where

$$N = \{j \mid \bar{a}_j \text{ is a column of } R\}$$

$$= \{j \mid x_j \text{ is a nonbasic element}\}$$

Now suppose that the $r^{th}$ basic variable of $\bar{x}_B^*$ is not an integer. From (4) we get the following relationship:

$$x_{Br} = y_{r0} - \sum_{j \in N} y_{rj}\ x_j \qquad (5)$$

In the current solution, all of the $x_j$, $j \in N$, are zero which implies that $y_{r0}$ is not an integer (since $x_{Br}$ is not an integer). The convention now is to write $y_{r0}$ as the sum of its integer part and its fractional part. We let $w_{r0}$ represent the integer part (stands for "whole" part of $y_{r0}$), or the greatest integer which is less than or equal to $y_{r0}$. Also, let $f_{r0}$ stand for the fractional part of $y_{r0}$ and then obtain

$$y_{r0} = w_{r0} + f_{r0} \tag{6}$$

$$w_{r0} \geqslant 0 \tag{7}$$

$$0 < f_{r0} < 1 \tag{8}$$

Similarly, all of the $y_{rj}$, $j \in N$ can be represented in this manner.

$$y_{rj} = w_{rj} + f_{rj} \tag{9}$$

$$0 \leqslant f_{rj} < 1 \tag{10}$$

Notice, however, that $f_{r0} > 0$ while $f_{rj} \geqslant 0$ is because many of the other basic variables may presently be integers, (i.e., they would contain no fractional part).

Now substitute equations (6) and (9) into equation (5) to obtain:

$$x_{Br} = w_{r0} + f_{r0} - \sum_{j \in N} (w_{rj} + f_{rj})\, x_j$$

$$= (w_{r0} - \sum_{j \in N} w_{rj}x_j) + (f_{r0} - \sum_{j \in N} f_{rj}x_j) \quad (11)$$

Now, the objective of the whole process is to modify the current solution so that the resulting solution will be an all integer solution. In order to do this, at least one of the variables which is currently nonbasic must become positive (i.e., one of the $x_j$, $j \in N$). This is because at present there is a unique solution in which all of the $x_j = 0$ and it is $\bar{x}_B^*$. In the desired all-integer solution the quantity in the first set of parentheses in equation (11) will be an integer. This is because each $x_j$ must be an integer in such a solution, and each $w_{rj}$ is an integer by definition. Therefore the new value of $x_{Br}$ can be expressed as

$$x_{Br} = \text{integer} + (f_{r0} - \sum_{j \in N} f_{rj}x_j) \quad (12)$$

Further, since it is desired to make $x_{Br}$ an integer, it is necessary that the quantity $(f_{r0} - \sum_{j \in N} f_{rj}x_j)$ must be an integer. Since by definition each $f_{rj} \geq 0$ and each $x_j \geq 0$, then it is clear that $\sum_{j \in N} f_{rj}x_j \geq 0$. Thus, the quantity $(f_{r0} - \sum_{j \in N} f_{rj}x_j)$ consists of one non-negative quantity, $\sum_{j \in N} f_{rj}x_j$, subtracted from another positive quantity $f_{r0}$.

However, $f_{r0} < 1$ by definition (see equation (8)). Thus the entire quantity $(f_{r0} - \sum_{j \in N} f_{rj}x_j)$ cannot be a positive integer quantity. Thus if it is to be an integer, then it has to be either a negative integer or zero.

$$\text{i.e., } f_{r0} - \sum_{j \in N} f_{rj}x_j \leq 0 \qquad (13)$$

The inequality (13) is the cutting plane constraint derived by Gomory. Notice that it does satisfy the two properties mentioned at the outset of this chapter. That is, the current solution violates equation (13) since all $x_j = 0$, $j \in N$, and $f_{r0} > 0$. Thus this makes the quantity $f_{r0} - \sum_{j \in N} f_{rj}x_j > 0$, rather than less than or equal to zero. Also, by the way the constraint (13) was derived, every feasible integer solution of the original integer linear programming problem will satisfy (13).

In the above derivation it was not discussed how to determine the cutting plane constraint if more than one of the basic variables happens not to be an integer valued variable. Gomory has shown that the proper choice of the cutting plane constraint at each step of the problem will lead to convergence in a finite number of steps. The most frequent technique which is used is to select the $x_{Br}$ whose fractional part happens to be the largest to obtain the cutting plane constraint in equation (13). There are many

other ways to select the variable by which the cutting plane constraint will be derived, but this one seems to be the most efficient. Also, it can be shown that it is never necessary to have more than n + 1 constraints at one time in the related linear programming problem. That is, it is possible to drop certain constraints which were previously added so that the total number of cutting plane constraints never exceeds n + 1. For a proof of this see Hadley [4].

When the cutting plane constraint equation (13) is added to the optimal tableau of the related linear programming problem, then the problem which results must be solved by the dual simplex algorithm mentioned in Chapter I. This is due to the fact that the current solution is infeasible (not an integer solution) but is still "optimal" (i.e., all $z_j - c_j \geq 0$). This means that the current solution is primal infeasible, but will be dual feasible, implying that we should use the dual simplex method.

An example will now be given. It will be solved using the cutting plane method just described.

Example 3.1

Maximize $z = -x_1 + x_2$

$2x_1 + x_2 \leq 9$

$2x_1 + 6x_2 \leq 21$

$x_1, x_2 \geq 0$

$x_1, x_2$ integer

Adding slack variables obtain:

$$\text{Maximize} \quad z = -x_1 + x_2$$

$$2x_1 + x_2 + x_3 = 9 \qquad (14)$$

$$2x_1 + 6x_2 + x_4 = 21 \qquad (15)$$

$$x_1, x_2 \geq 0, \text{ integer}$$

The optimal tableau from the simplex method is given at the top of the next page. It illustrates the optimal solution to the related linear programming problem to be (0, 7/2). Since this is not an integer solution we must derive a cutting plane constraint, add it to the other constraints, and then solve the related linear programming problem by the dual simplex method.

| | | | -1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ |
| 0 | $\bar{a}_3$ | 33/6 = 11/2 | 10/6 | 0 | 1 | -1/6 |
| 1 | $\bar{a}_2$ | 21/6 = 7/2 | 2/6 | 1 | 0 | 1/6 |
| | | 21/6 = 7/2 | 8/6 | 0 | 0 | 1/6 |

The first cutting plane constraint is derived as follows:

$$y_{10} = 33/6 = 5 + 1/2$$

$$y_{11} = 10/6 = 1 + 2/3$$

$$y_{14} = -1/6 = -1 + 5/6$$

Thus $f_{r0} - \sum_{j \in N} f_{rj}x_j \leq 0$ becomes

$$1/2 - [(2/3)x_1 + (5/6)] \leq 0$$

$$\text{or} \quad 4x_1 + 5x_4 \geq 3 \tag{16}$$

Since $x_4$ is the slack variable in equation (15), then $x_4 = 21 - 2x_1 - 6x_2$. Substituting this into (16) obtain

$$4x_1 + 5(21 - 2x_1 - 6x_2) \geq 3$$

$$\text{or} \quad x_1 + 5x_2 \leq 17 \tag{17}$$

Inequality (17) above is the cutting plane constraint which is now added to the original problem, and then the new related linear programming problem is solved using the dual simplex method. The optimal tableau of that problem is:

| | | | -1 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ | $\bar{y}_5$ |
| 0 | $\bar{a}_3$ | 28/5 | 9/5 | 0 | 1 | 0 | -1/5 |
| 1 | $\bar{a}_2$ | 17/5 | 1/5 | 1 | 0 | 0 | 1/5 |
| 0 | $\bar{a}_4$ | 3/5 | 4/5 | 0 | 0 | 1 | -6/5 |
| | | 17/5 | 6/5 | 0 | 0 | 0 | 1/5 |

The optimal solution is (0, 17/5) which again is not an integer solution so another cutting plane constraint must be added, and then the new related linear programming problem

must be solved. The new cutting plane constraint is derived as follows:

$$y_{20} = 17/5 = 3 + 2/5$$

$$y_{21} = 1/5$$

$$y_{25} = 1/5$$

The cutting plane constraint is $f_{r0} - \sum_{j \in N} f_{rj}x_j \leq 0$ which becomes,

$$2/5 - [(1/5)x_1 + (1/5)x_5] \leq 0$$

$$\text{or} \quad 2 - x_1 - x_5 \leq 0$$

But as before, $x_5$ is a slack variable originating from the first cutting plane constraint and the following equality holds: $x_5 = 17 - x_1 - 5x_2$. Thus the cutting plane constraint becomes,

$$2 - x_1 - (17 - x_1 - 5x_2) \leq 0$$

$$\text{or} \quad x_2 \leq 3.$$

By adding the constraint $x_2 \leq 3$ to the others and then solving the related linear programming problem, it is clearly seen that of the five extreme points, the optimal solution (0,3) is also the optimal integer solution.

The following figures will help illustrate graphically the addition of the two cutting plane constraints derived for the problem of example 3.1.

The first figure (Figure 3-1) is simply the feasible region of the first linear programming problem, and it shows the first optimal point which is (0, 7/2).

The second figure is given to show how the addition of the first cutting plane constraint affects the problem. The first cutting plane constraint which was added is shown in Figure 3-2 as a dotted line.

The addition of the first cutting plane constraint creates a feasible region with five extreme points. These extreme points are shown in Figure 3-2, and as was pointed out in the related optimal tableau, the optimal solution is at (0,117/5). Also, it was mentioned that since this is not an integer solution the addition of a second cutting plane constraint was necessary. The new constraint was found to be $x_2 \leq 3$ and it is illustrated in Figure 3-3, again as a dotted line.

After this cutting plane constraint is added, it is clear to see that of the extreme points illustrated in Figure 3-3 (namely, (0, 0), (0, 3), (3/2, 3), (33/10, 12/5), and (9/2, 0)) that the optimal solution occurs at the integer solution (0, 3).

## III. MIXED INTEGER ALGORITHM

A modification, in the form of a different cutting plane constraint, to the above described process will now be

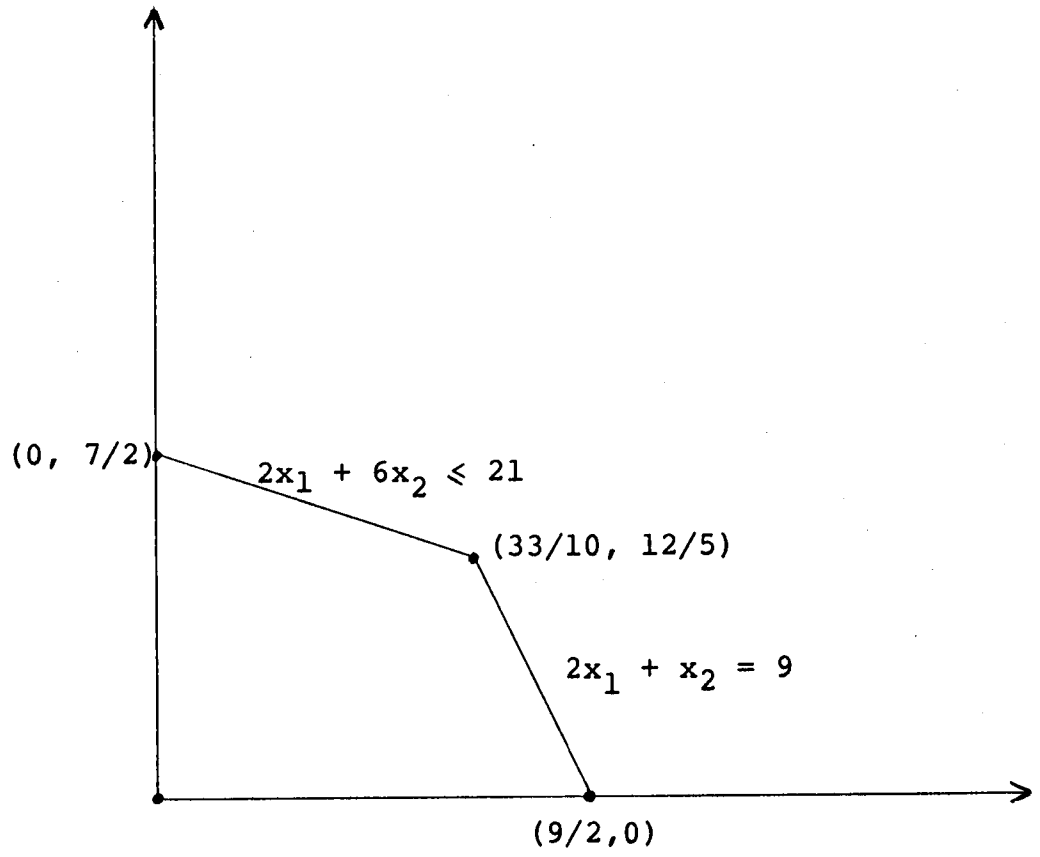


Figure 3-1. Geometrical Interpretation of Example 3.1.

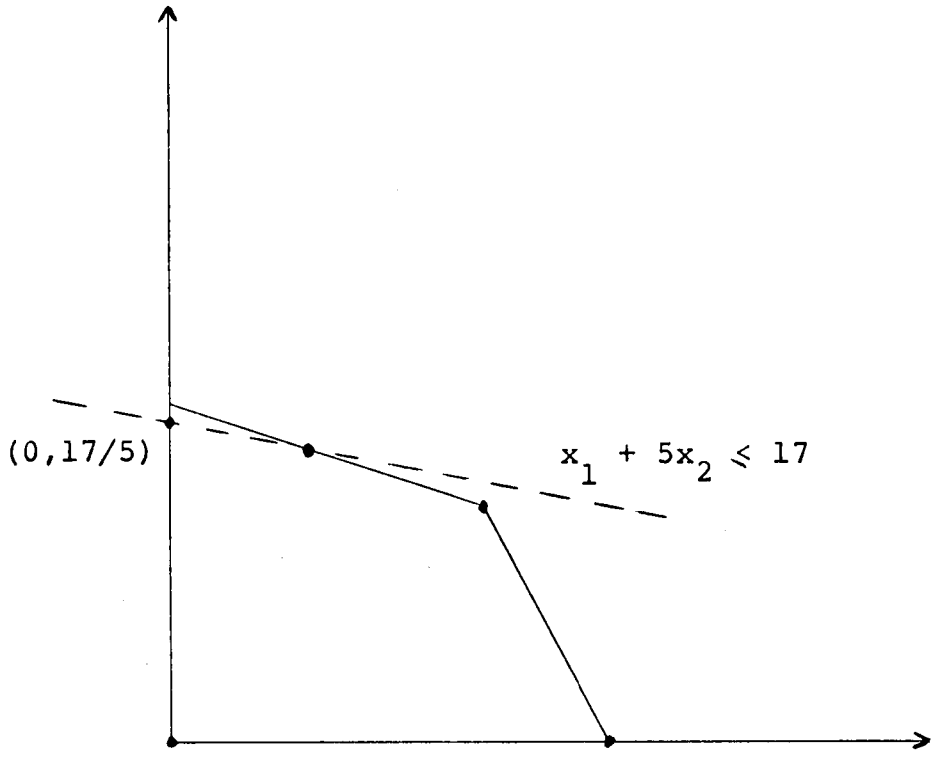


Figure 3-2. Geometrical Interpretation of Example 3.1 after the First Cut.

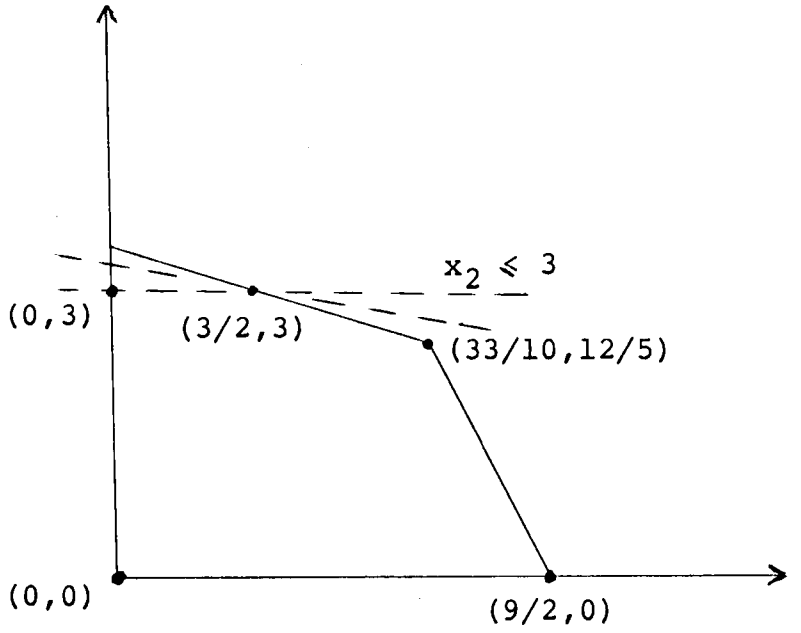


Figure 3-3. Geometrical Interpretation of Example 3.1 after the Second Cut.

made so that it will be possible to solve mixed integer programming problems.

Consider the following problem,

$$\text{Maximize} \quad z = \bar{c}'\bar{x}$$

$$\text{Subject to:} \quad A\bar{x} = \bar{b}$$

$$\bar{x} \geq 0$$

$$x_j \text{ integer}, \; j \in I$$

where the set I consists of all the subscripts corresponding to the variables that must be integer-valued. Note: If I = (1,2,...,n), then the above problem is a pure integer linear programming problem.

Suppose that the related linear programming problem has been solved and the $r^{th}$ basic variable (required to be an integer) is not integer-valued. Then as before (see equation (5)),

$$x_{Br} = y_{r0} - \sum_{j \in N} y_{rj}x_j \tag{18}$$

where $N = \{j | x_j \text{ is nonbasic}\}$. Again as before, let $y_{r0} = w_{r0} + f_{r0}$ as defined in equations (6) through (8), and substituting this equality into (18) obtain,

$$x_{Br} = w_{r0} + f_{r0} - \sum_{j \in N} y_{rj}x_j$$

Since $x_{Br}$ is required to be an integer,

$$f_{r0} - \sum_{j \in N} y_{rj}x_j = x_{Br} - w_{r0} = \text{integer} \tag{19}$$

The convention now is to partition N into the following two sets:

$$N^+ = \{j | y_{rj} \geq 0,\ j \in N\}$$

$$N^- = \{j | y_{rj} < 0,\ j \in N\}$$

Here, N is simply divided into the non-negative entries of the simplex tableau ($N^+$) and the negative entries of the simplex tableau ($N^-$). Then (19) becomes

$$f_{r0} - \sum_{j \in N^+} y_{rj}x_j - \sum_{j \in N^-} y_{rj}x_j = \text{integer} \qquad (20)$$

Consider now the following two cases:

1. $\sum_{j \in N} y_{rj}x_j \geq 0$
2. $\sum_{j \in N} y_{rj}x_j < 0$

Case 1

$\sum_{j \in N} y_{rj}x_j \geq 0$, then from (19)

$$\sum_{j \in N} y_{rj}x_j = f_{r0} + P$$

where P = 0,1,2,...(i.e., P is some non-negative integer). Thus any feasible solution to the original problem must satisfy $\sum_{j \in N} y_{rj}x_j \geq f_{r0}$.

Further, since $\sum_{j \in N^-} y_{rj}x_j \leq 0$ for any feasible solution, then any feasible solution to the original problem must

satisfy

$$\sum_{j \in N^+} y_{rj}x_j \geq f_{r0} \tag{21}$$

Case 2

$\sum_{j \in N} y_{rj}x_j < 0$, then from (19),

$$\sum_{j \in N} y_{rj}x_j = f_{r0} - P$$

where P = 1,2,3,...(i.e., some strictly positive integer). Thus any feasible solution to the original problem must satisfy

$$\sum_{j \in N} y_{rj}x_j \leq f_{r0} - 1,$$

since $f_{r0} - 1$ is the largest value possible for $\sum_{j \in N} y_{rj}x_j$. This can be written as

$$\sum_{j \in N^+} y_{rj}x_j - \sum_{j \in N^-} y_{rj}x_j \leq f_{r0} - 1.$$

Further, since $\sum_{j \in N^+} y_{rj}x_j \geq 0$ then it must be true that

$$\sum_{j \in N^-} y_{rj}x_j \leq f_{r0} - 1 \tag{22}$$

Now multiply (22) by the strictly negative number $f_{r0}/(f_{r0}-1)$, and obtain

$$\sum_{j \in N^-} \left(\frac{f_{r0}}{1-f_{r0}}\right) \left|y_{rj}\right| x_j \geq f_{r0} \tag{23}$$

Since $(f_{r0}/f_{r0}-1)$ is a strictly negative number and $y_{rj} < 0$, for all j, then the negative sign from each $y_{rj}$ has been applied above to $(f_{r0}/f_{r0}-1)$ making it $(f_{r0}/1-f_{r0})$. This is why each $y_{rj} > 0$ and the reason for the notation $|y_{rj}|$ in line (23).

Finally, combining the inequalities (21) and (23) obtained for cases 1 and 2 respectively, obtain

$$\sum_{j \in N^+} y_{rj}x_j + \sum_{j \in N^-} \left(\frac{f_{r0}}{1-f_{r0}}\right)|y_{rj}|x_j \geq f_{r0} \qquad (24)$$

This inequality (24) must hold regardless of whether case 1 or case 2 holds. This is clear because one of the two values of the left-hand side is always greater than or equal to $f_{r0}$, and both are positive. Also, the inequality (24) does not hold for the current solution since all $x_j$, $j \in N^+$ or $j \in N^-$, are zero and $f_{r0} > 0$ by definition. Thus inequality (24) can be used as the cutting plane constraint for the mixed integer programming problem cited at the outset.

Following is a flow chart for the mixed integer programming algorithm just discussed (Figure 3-4). Also an example solved by using constraint (24) is included to show the mixed integer algorithm in use.

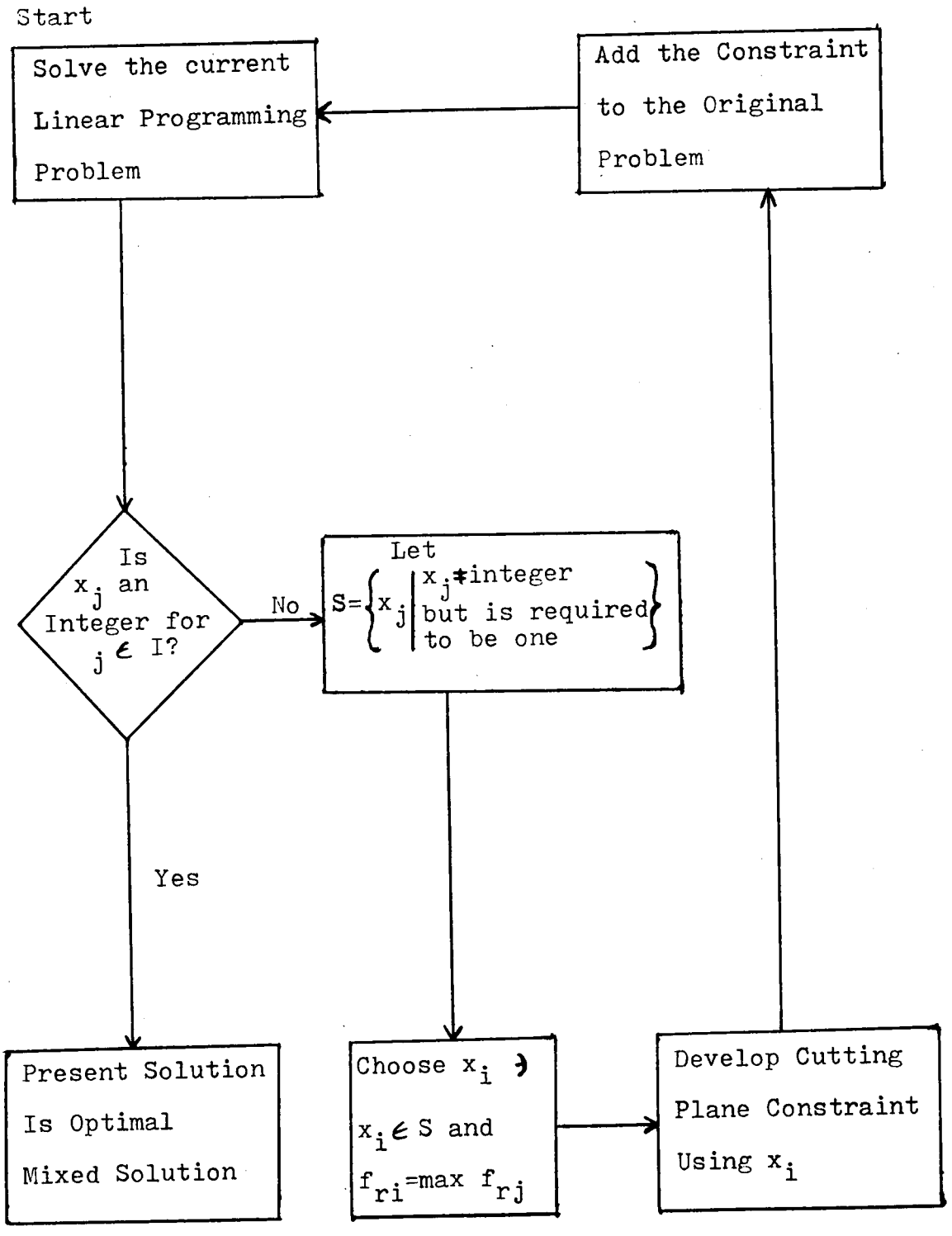


Figure 3-4. Flowchart for Mixed Integer Cutting Plane Method.

Example 3.2

$$\text{Maximize} \quad z = 2x_1 + x_2$$

$$\text{Subject to:} \quad x_1 + x_2 \leq 8$$

$$18x_1 - 8x_2 \leq 27$$

$$x_1, x_2 \geq 0, \text{ integer}$$

In standard form we have

$$\text{Maximize} \quad z = 2x_1 + x_2$$

$$\text{Subject to:} \quad x_1 + x_2 + x_3 = 8$$

$$18x_1 - 8x_2 + x_4 = 27$$

$$x_1, x_2 \geq 0, \text{ integer}$$

The optimal tableau for the related linear programming problem is,

| | | | 2 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ |
| 1 | $\bar{a}_2$ | 9/2 | 0 | 1 | 9/13 | -1/26 |
| 2 | $\bar{a}_1$ | 7/2 | 1 | 0 | 4/13 | 1/26 |
| | | 8 | 0 | 0 | 7/13 | 1/26 |

The optimal solution is (7/2, 9/2) which is not an integer solution. Thus, we must add a cutting plane constraint and this time we use the second algorithm to develop it.

$$y_{10} = 9/2 = 4 + 1/2, \text{ so } f_{r0} = 1/2, \text{ thus } \frac{f_{r0}}{1-f_{r0}} = 1.$$

Therefore, the cutting plane constraint

$$\sum_{j \in N^+} y_{rj}x_j + \sum_{j \in N^-} \left(\frac{f_{r0}}{1-f_{r0}}\right)\left|y_{rj}\right| x_j \geq f_{r0},$$

becomes

$$(9/13)x_3 + (1/26)x_4 \geq 1/2$$

or

$$18x_3 + x_4 \geq 13 \qquad (1)$$

Since $x_3$ and $x_4$ are slack variables, we can substitute the resulting equalities into (1) above.

$$18(8 - x_1 - x_2) + (27 - 18x_1 + 8x_2) \geq 13$$

or

$$18x_1 + 5x_2 \leq 79.$$

Now we add $18x_1 + 5x_2 + x_5 = 79$ to the optimal tableau and solve the resulting problem by the dual simplex method. The optimal tableau of that problem is,

| | | | 2 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| $\bar{c}_B$ | Basis | $\bar{x}_B$ | $\bar{y}_1$ | $\bar{y}_2$ | $\bar{y}_3$ | $\bar{y}_4$ | $\bar{y}_5$ |
| 1 | $\bar{a}_2$ | 5 | 0 | 1 | 18/13 | 0 | -1/13 |
| 2 | $\bar{a}_1$ | 3 | 1 | 0 | -10/26 | 0 | 1/13 |
| 0 | $\bar{a}_4$ | 13 | 0 | 0 | 18 | 1 | -2 |
| | | 11 | 0 | 0 | 8/13 | 0 | 1/13 |

Thus the optimal solution (3, 5) is also the optimal integer solution. Figure 3-5 illustrates the addition of

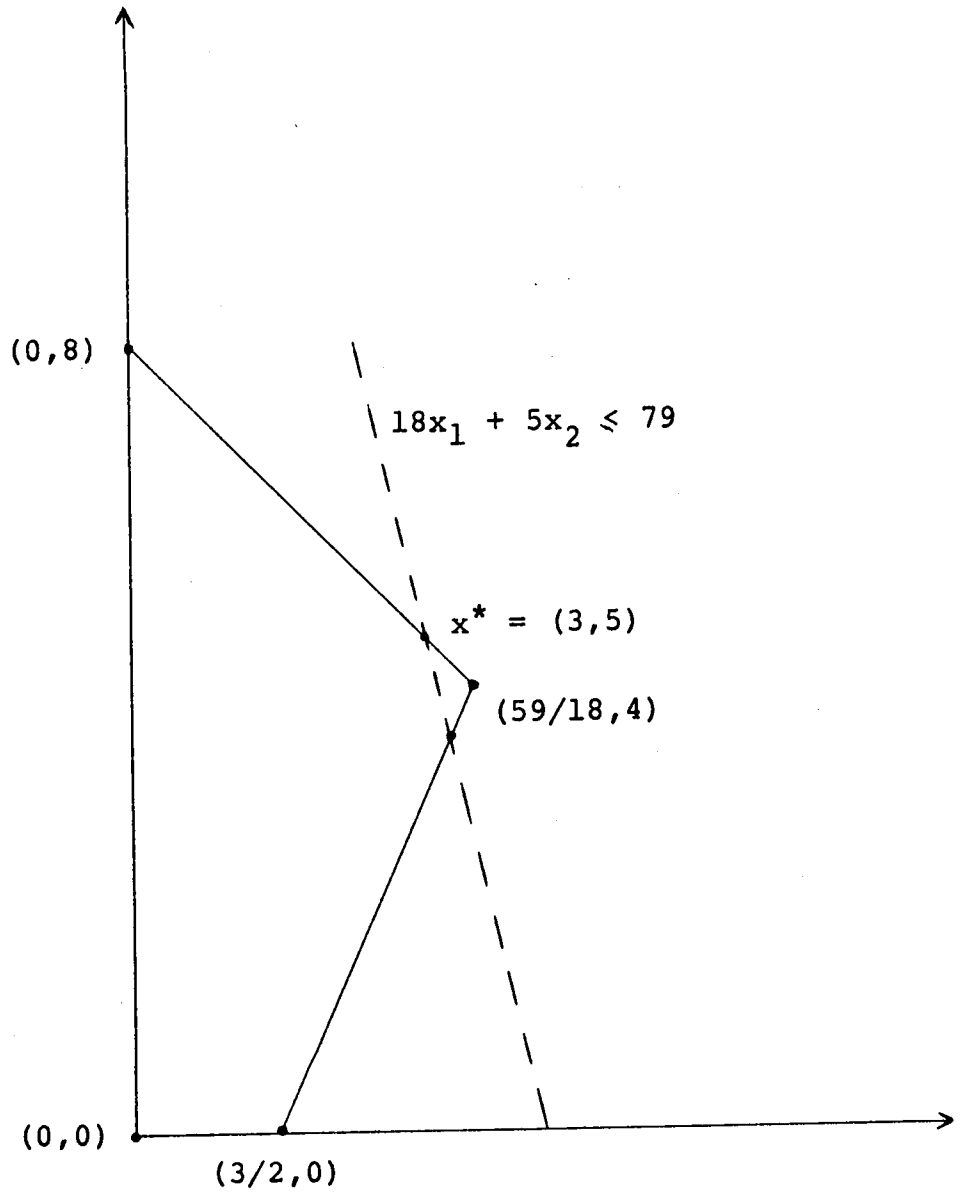


Figure 3-5. Geometrical Interpretation of Example 3.2.

the new cutting plane constraint, the resulting feasible region, and the optimal solution which is (3, 5). Again, the cutting plane is illustrated as a dotted line in Figure 3-5.

# CHAPTER IV

## A COMBINED PRIMAL-ENUMERATIVE ALGORITHM

### I. INTRODUCTION

The obvious disadvantage of the cutting plane methods described in Chapter III is that no useful approximation to the actual optimal solution is available at any stage prior to the completion of the program. Thus the method to be discussed here is one in which a given feasible integer solution is improved upon in an iterative method until an optimal solution is attained. The advantage to such a method is that the algorithm can be terminated prior to completion with at least a feasible integer solution, and perhaps one that is a good approximation to the actual solution.

The method is a combination of an efficient primal algorithm with an enumerative technique. The reason for the addition of the enumerative method is that the primal method used is one which does not guarantee convergence. (The combined algorithm does insure convergence, which will be shown later.) The primal method is combined with the enumerative method when it is "stalled" or in a "stationary cycle," a concept to be discussed shortly. Then the enumerative method either demonstrates that the present

solution is optimal, or it will produce an improved solution which can then be used as a new starting value for the primal method.

Again, the integer linear programming problem under consideration is

$$\begin{aligned} &\text{Maximize} \quad z = \bar{c}'\bar{x} \\ &\text{Subject to:} \quad A\bar{x} = \bar{b} \\ &\qquad\qquad \bar{x} \geqslant \bar{0}, \text{ integer} \end{aligned} \tag{1}$$

where we assume that the problem has a bounded solution. The problem is then transformed via the Tucker formulation. The Tucker formulation will be introduced in order to keep track of all m + n variables (m slack variables and n real variables).

The problem from (1) above can be written as

$$\begin{aligned} &\text{Maximize} \quad z = \bar{c}'\bar{x} \\ &\qquad\qquad I\bar{s} = \bar{b} - A\bar{x} \\ &\qquad\qquad \bar{s}, \bar{x} \geqslant \bar{0} \end{aligned}$$

When we remove a variable from the basis, its value would be lost if not for the addition of the n equations $y_i = x_i$. Then,

$$\begin{aligned} z &= \bar{c}'\bar{x} \\ I\bar{s} &= \bar{b} - A\bar{x} \\ \bar{y} &= \bar{x} \geqslant \bar{0} \end{aligned}$$

is rewritten

$$z = \bar{c}'\bar{y}$$

$$I\bar{s} = \bar{b} - A\bar{y}$$

$$\bar{x} = \bar{0} + \bar{y}$$

$$\bar{s},\ \bar{x},\ \bar{y} \geqslant \bar{0}$$

In tableau form this becomes (see Figure 4-1).

## II. THE PRIMAL ALGORITHM

The primal algorithm to be used (sometimes called the Rudimentary Primal Algorithm) consists of the following iterative steps:

1. Selection of the pivotal column.
   Scan the cost row c and set s = arg min $c_j$; terminate if $c_j \geqslant 0$, for all j. That is, s is simply the smallest argument (index) of the minimum $c_j$; Then designate s as the pivotal column. All $c_j \geqslant 0$ implies the optimal solution has been obtained.
2. Selection of the source row.
   Scan the pivotal column s and set r = arg min $\{b_i/a_{is}$, for $a_{is} > 0\}$ terminate if $a_{is} \leqslant 0$, for all i. Then designate row r as the source row from which the actual pivot row will be determined. ($a_{is} \leqslant 0$ implies an unbounded solution, but we have assumed a bounded one.)

| | 1 | $-y_1$ | $-y_2$ | . | . | . | . | $-y_n$ |
|---|---|---|---|---|---|---|---|---|
| $z$ = | 0 | $-c_1$ | $-c_2$ | . | . | . | . | $-c_n$ |
| $s_1$ = | $b_1$ | $a_{11}$ | $a_{12}$ | . | . | . | . | $a_{1n}$ |
| $s_2$ = | $b_2$ | $a_{21}$ | $a_{22}$ | . | . | . | . | $a_{2n}$ |
| . | . | . | . | | | | | . |
| . | . | . | . | | | | | . |
| $s_m$ = | $b_m$ | $a_{m1}$ | $a_{m2}$ | . | . | . | . | $a_{mn}$ |
| $x_1$ = | 0 | -1 | 0 | . | . | . | . | 0 |
| $x_2$ = | 0 | 0 | -1 | . | . | . | . | 0 |
| . | | | | | | | | |
| . | | | | | | | | |
| $x_n$ = | 0 | 0 | 0 | . | . | . | . | -1 |

Figure 4-1. Tucker Formulation Tableau.

3. Construction of the pivot row.

   Generate the pivot row I by dividing row r by a factor $\lambda$ so that after rounding down to the nearest integer, the pivot element is unity. Also be sure that no feasible integer points are removed by the cut constructed. (This is insured through the choice of $\lambda$, which will be discussed later.)

   Note: $[a_{rj}/a_{rs}]$ will be used to represent the value of $(a_{rj}/a_{rs})$ after the rounding has been performed.

4. Transformation of the tableau.

   Update the data by adding row I and pivoting on $a_{Is}$ according to the general simplex rules.

$$b_i' = b_i - a_{is}b_I$$

$$a_{ij}' = a_{ij} - a_{is}a_{Iy},\ j \neq s \qquad i = 0,1,\ldots,m.$$

$$a_{is}' = -a_{is}$$

A simplified summary of the rules is then,

1. Selection of pivotal column:

$$s = \arg\min_{c_j < 0} c_j,\ \text{(if s is undefined, terminate)}$$

2. Selection of source row:

$$r = \arg\min_{a_{is} > 0} b_i/a_{is},\ \text{(if r undefined, terminate)}$$

3. Construction of pivot row:

$$y_I = [b_r/\lambda] - \{[a_{r1}/\lambda]y_1 + \ldots + y_s + \ldots + [a_{rn}/\lambda]y_n\} \geq 0$$

$$= b_I - (a_{I1}y_1 + \ldots + a_{In}y_n) \geq 0$$

Again, the selection of $\lambda$ is still to be determined.

4. Transformation of tableau:

$$b_i' = b_i - a_{is}b_I$$

$$a_{ij}' = a_{ij} - a_{is}a_{Iy}, \quad j \neq s \qquad i = 0,1,\ldots,m.$$

$$a_{is}' = -a_{is}$$

Something must be said here about how to deal with infeasibilities. Suppose the $i^{th}$ constraint is

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i, \text{ with } b_i > 0.$$

After translating this to Tucker formulation we have

$$-b_i \geq -a_{i1}y_1 - a_{i2}y_2 - \ldots - a_{in}y_n$$

leading to this row in the tableau,

$$s_i \quad -b_i \quad -a_{i1} \quad -a_{i2} \quad \ldots \quad -a_{in}.$$

Therefore we have a tableau which is infeasible (i.e., $b_i < 0$). The method to be used to remove such an

infeasibility will be to check the constant column ($b_i$'s) for negative entries as would occur in row i above. Treat this row as an objective function. It will be called the control row, and we search it (as we do the objective function) for the most negative element and denote its column as the pivotal column. Then find the minimum ratio of the non-negative constants $b_i$ over the positive $a_{is}$ for the pivotal source row, and proceed as usual.

Now the question of choosing the $\lambda$ in step (3) of the primal algorithm is discussed. Some primal algorithms use $\lambda = a_{is}$ which directly makes the pivot element unity. This will very obviously simplify the algorithm, but could do so at the expense of losing a more effective or faster working algorithm. Further, for some of the "improved cuts" there are many additional operations involved in choosing the best $\lambda$. Also, the use of the $\lambda$'s derived from these fancier cuts will invalidate a proof of finiteness given later. In fact, the advantages of a simpler $\lambda$ seem to outweigh those of the so called "improved" cuts, thus we will use $\lambda = a_{rs}$ in the algorithm.

Next we consider the row and column selections of the primal algorithm. The selection rules which were given in the summary of the algorithm are more restrictive than necessary. Any column containing a negative $c_j$ when selected will cause the objective value to increase upon pivoting.

Also, because of the rule for generating the pivot row, any row i in which $b_i/a_{is} \leq b_r/a_{rs}$, $a_{is} > 0$, and r is chosen by rule (2), can be used as a source row. It is very rare that r is the only eligible source row. The above points out that this is only the case if $b_i/a_{is} > b_r/a_{rs}$ for all $i \neq r$.

This freedom of choice (for the pivotal row and column) leads to the resolution of a basic difficulty that arises in primal integer programming. This problem is that of "stationary cycles." Following will be an example of a stationary cycle, and then we will discuss how to handle them. Consider the problem

$$\text{Maximize} \quad z = 8x_1 + 9x_2$$

$$\begin{aligned} \text{Subject to:} \quad & 2x_1 - 3x_2 \leq 6 \\ & 2x_1 - x_2 \leq 8 \\ & 2x_1 + x_2 \leq 14 \\ & x_1 + 2x_2 \leq 14 \\ & -x_1 + 2x_2 \leq 10 \\ & x_1 + 3x_2 \geq 5 \\ & x_1, x_2 \geq 0, \text{ integers.} \end{aligned}$$

This problem results in the following sequence of tableaus:

Tableau I

| | b | $-y_1$ | $-y_2$ | |
|---|---|---|---|---|
| z | 0 | -8 | -9 | |
| $s_1$ | 6 | 2 | -3 | |
| $s_2$ | 8 | 2 | -1 | |
| $s_3$ | 14 | 2 | 1 | |
| $s_4$ | 14 | 1 | 2 | |
| $s_5$ | 10 | -1 | (2) | |
| $s_6$ | -5 | -1 | -3 | (Control row— |
| $x_1$ | 0 | -1 | 0 | infeasible $b_6$) |
| $x_2$ | 0 | 0 | -1 | |
| I | 5 | -1 | 1 | |

Tableau II

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 45 | -17 | 9 |
| $s_1$ | 21 | -1 | 3 |
| $s_2$ | 13 | 1 | 1 |
| $s_3$ | 9 | 3 | -1 |
| $s_4$ | 4 | 3 | -2 |
| $s_5$ | 0 | (1) | -2 |
| $s_6$ | 10 | -4 | 3 |
| $x_1$ | 0 | -1 | 0 |
| $x_2$ | 5 | -1 | 1 |
| I | 0 | 1 | -2 |

Tableau III

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 45 | 17 | -25 |
| $s_1$ | 21 | 1 | 1 |
| $s_2$ | 13 | -1 | 3 |
| $s_3$ | 9 | -3 | 5 |
| $s_4$ | 4 | -3 | (4) |
| $s_5$ | 0 | -1 | 0 |
| $s_6$ | 10 | 4 | -5 |
| $x_1$ | 0 | 1 | -2 |
| $x_2$ | 5 | 1 | -1 |
| I | 1 | -1 | 1 |

Proceeding from Tableau II to Tableau III is in fact a stationary cycle. None of the elements in the constant column ($b_i$'s) were changed. Most importantly, the value of the objective function ($z = 45$) was not changed.

As long as $b_I \geq 1$, the constant column of the tableau will be changed upon pivoting and the point $x = (x_1, x_2, \ldots, x_n)$ moves to a "better" point say $x' = (x_1', x_2', \ldots, x_n')$ where $z = cx'$ is greater than $cx$ by at least an integer amount. This kind of move is referred to as a "transition cycle." A finite number of these transition cycles must result in optimality (since we assumed a bounded solution exists). Now, the problem arises in the case $b_I = 0$. Then the constant column remains unchanged and no improvement is made. This is what is referred to as a stationary cycle. Unless it can be guaranteed that the number of stationary cycles occurring between transition cycles is finite, there exists the possibility that the algorithm may not terminate.

Finiteness can be assured if the coefficients of each row of the tableau are reduced until (if necessary) $a_{ij} \leq b_i$, $j = 1,2,\ldots,n$. This decreasing of the coefficients in the constraint rows means that the cost coefficients, at the top of each column, must increase and eventually become positive. This would mean that the optimal solution has been reached. The following theorem shows that this will be the case with the right choice of $\lambda$ (i.e., $\lambda = a_{rs}$).

Theorem 4.1

If on some iteration row r is selected as the source row and column s as the pivotal column, then the pivot equation becomes

$$y_I = [b_r/a_{rs}] - ([a_{r1}/a_{rs}]y_1 + \ldots + y_s + \ldots + [a_{rn}/a_{rs}]y_n)$$

and the coefficients of row r after the pivot satisfy:

1. $a'_{rs} = -a_{rs}$
2. $a_{rs} > a'_{rj} \geq 0$, for all $j \neq s$.

Proof:

1. Is clearly true by the pivotal rules.
2. By the pivotal rules we have $a'_{rj} = a_{rj} - [a_{rj}/a_{rs}]a_{rs}$. Since $[a_{rj}/a_{rs}] > (a_{rj}/a_{rs}) - 1$, then

$$a'_{rj} < a_{rj} - \{(a_{rj}/a_{rs}) - 1\}a_{rs} = a_{rs}.$$

Also, since $(a_{rj}/a_{rs}) \geq [a_{rj}/a_{rs}]$

$$a'_{rj} \geq a_{rj} - (a_{rj}/a_{rs})a_{rs} = 0. \qquad \text{Q.E.D.}$$

This theorem then assures that repeated pivoting on the same source row (with different column s) will eventually result in all of the elements in that row being reduced to the size of the constant column. Also, now it will be shown that the finiteness is violated for $\lambda \neq a_{rs}$. This will simply be shown by counter-example.

If the source row is

$$4 \quad 8 \quad 5 \quad -3 \quad -2$$

and the pivotal column is s = 3, then with $\lambda = a_{rs} = 5$, the cut is

$$0 \quad 1 \quad 1 \quad -1 \quad -1$$

and following pivoting row r becomes

$$4 \quad 3 \quad -5 \quad 2 \quad 3 \quad (a_{rs} = 5 \text{ becomes } -5).$$

However, with $\lambda = 3$ or 4, the cut would be

$$1 \quad 2 \quad 1 \quad -1 \quad -1$$

and following pivoting row r becomes

$$-1 \quad -2 \quad -5 \quad 2 \quad 3.$$

Thus, both $a'_{r1}$ and $a'_{r2}$ are less than 0 which is contrary to the theorem. Therefore, $\lambda \neq a_{rs}$ will contradict or violate finiteness.

Now, returning to the discussion of stationary cycles, we consider a method of how to possibly avoid them. Clearly, in order to avoid the problem of stationary cycles, it is desirable rather to make a transition cycle if one exists. This brings about the idea of searching for any column in which an integer change is possible. To accomplish this we search "all" negative columns to find the minimum ratios

$$\theta_j = \min_{a_{ij} > 0} b_i/a_{ij}, \text{ for all } j \rightarrow c_j < 0.$$

If any of the $\theta_j > 0$, then this will provide a transition cycle. Consider, however, the case of a sequence of stationary cycles which means the algorithm is stalled at a particular integer solution. Simply moving the basis to another point, better or not, in hopes of moving from there to an improved position is not a good approach. This is because there is a possibility of returning to an earlier tableau which could be considered another kind of stationary cycle as well.

When the above column selection rule yields $\theta_j = 0$ for all $j \rightarrow c_j < 0$, then a stationary cycle must be performed. Thus the problem arises of determining a pivotal column and the source row. Use then the following criteria referred to as the max-min ratio rule.

$$s = \arg \min \theta_j, \text{ for all } j \rightarrow c_j < 0$$

where

$$\theta_j = \min_{a_{ij} > 0} b_i/a_{ij}$$

$$r = \arg \min_{a_{is} > 0} b_i/a_{is}.$$

r is determined by $r = \arg \max_{b_i = 0} a_{is}$ in the case where some $b_i = 0$.

The justification for the above stated max-min rule is that attempt is being made to remove the most restrictive constraint in the column in which we hope to have the best chance of bringing an improvement.

## III. THE ENUMERATIVE METHOD

Because it cannot necessarily be guaranteed that the number of stationary cycles between transition cycles is finite, it is necessary to combine the primal procedure just discussed with an enumerative method. The enumerative method will be used when the primal algorithm is stalled in a stationary cycle.

The procedure will be to use the primal algorithm until a stationary sequence occurs, and then switch to the enumerative method. The enumerative method will provide an improved feasible solution (if the present solution is not optimal), and then return to the primal algorithm starting with the new solution. From here, either the primal algorithm will continue to an optimal solution or the enumerative method will be called on again for another improved feasible solution. Eventually optimality must be reached by one of the two methods.

The method of implementation of the enumerative technique will determine the specific algorithm chosen. Since it is primarily an "escape" tool from stationary cycles, it should

be a method which will quickly provide a feasible solution rather than one which is designed to efficiently provide an optimal one. Since it is hoped that it will only be needed occasionally, the enumerative technique should be one which requires as small a set-up procedure as possible.

The preceeding ideas point to a branch and bound technique, and the one to be used is an algorithm developed by Bradley and Wahi. It is a primal single-branch method. It is able to take advantage of prior information during successive iterations. It requires setting up part of the problem in Hermite normal form which still involves only a reasonable amount of effort, and then applying Fourier-Motzkin elimination. The Fourier-Motzkin elimination will obtain bounds for all the integer variables, and it will provide an improving sequence of integer solutions, if they exist.

## IV. HERMITE NORMAL FORM

The reason for placing the integer programming problem into Hermite normal form is because it provides an advantage in applying Fourier-Motzkin elimination. This will be shown later. Before we discuss which part of the problem is to be transformed, we give a definition of Hermite normal form.

Given an nxn integer matrix C of full rank, there exists an nxn unimodular matrix K (not necessarily unique)

such that CK is lower triangular with positive diagonal elements, and each off-diagonal element of CK is nonpositive and strictly less in absolute value than the diagnoal element in its row. CK is then in Hermite normal form. (Post multiplication of C by K is equivalent to a series of elementary column operations.) Note: A square matrix is unimodular if its determinant has value ±1.

Assuming again that the associated linear programming problem is feasible and bounded, then there is an optimal solution

$$\bar{x} = \bar{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$$

$$z = z^* = c_1 x_1^* + c_2 x_2^* + \ldots + c_n x_n^* \qquad (1)$$

Associated with the above solution $\bar{x}$, there is an optimal basic set of variables $\bar{x}_B$ and also a set of nonbasic variables $\bar{x}_N$. Again the original problem in matrix notation is

$$\begin{aligned} &\text{Maximize} \quad z = \bar{c}'\bar{x} \\ &\text{Subject to:} \quad A\bar{x} \leqslant \bar{b} \\ &\qquad\qquad I\bar{x} \geqslant \bar{0}, \ \bar{x} \text{ is integer} \end{aligned} \qquad (2)$$

and the associated Tucker formulation is

$$\begin{aligned} &\text{Maximize} \quad \bar{c}'\bar{y} \\ &\text{Subject to:} \quad \bar{s} = \bar{b} - A\bar{y} \geqslant \bar{0} \\ &\qquad\qquad \bar{x} = \bar{0} + I\bar{y} \leqslant \bar{0} \end{aligned} \qquad (3)$$

where $\bar{y}$ is integer.

This set of m + n inequalities from (3) can be reordered corresponding to the reordering of $(\bar{s}, \bar{x})$ into $(\bar{x}_B, \bar{x}_N)$, determined by the optimal linear programming solution (1). Then group the reordered columns into renamed matrices $A_1$, $A_2$ corresponding to $\bar{x}_N$ and $\bar{x}_B$ to obtain,

Maximize $\bar{c}'\bar{y}$

Subject to: $\bar{x}_N = \bar{b}_1 + A_1\bar{y} \geqslant \bar{0}$

$\bar{x}_B = \bar{b}_2 + A_2\bar{y} \geqslant \bar{0}$

where $\bar{y}$ is integer. Here $\bar{x}_B$ is the set of basic variables and $A_1$, $A_2$, $\bar{b}_1$, $\bar{b}_2$ are merely the original coefficients and constants of the system (2) rearranged.

The constraints $\bar{x}_N = \bar{b}_1 + A_1\bar{y} \geqslant \bar{0}$ form a cone in $\bar{y}$ with its vertex corresponding to the linear programming optimum $\bar{x}_N = \bar{0}$. When an integer restriction is added along with the objective function, the result is what is referred to as the integer program over the cone, namely

Maximize $\bar{c}'\bar{y}$

Subject to: $\bar{x}_N = \bar{b}_1 + A_1\bar{y} \geqslant \bar{0}$

$\bar{y}$ integer (4)

The n x n matrix $A_1$ is the object of the Hermite normal transformation. The cone problem (4) is considered because it is much easier to solve than the original problem (2). Feasible integer solutions to (4) will yield integer solutions to (2), but not necessarily ones which are feasible.

Note: There appears to be a high probability in practice of the optimal solution to (4) being feasible in (2); and

if it is then it is also an optimal one for that original problem. [7]

It is proven by Bradley that the original problem (2) is equivalent to an infinite number of other programs in its class. The 1-1 correspondence is $\bar{y} = \bar{h} + K\bar{z}$ where K is an n x n unimodular matrix and $\bar{h}$ represents a translation by an integer vector. Thus the original problem (2) is equivalent to,

find z so as to

maximize $\bar{c}'(\bar{h} + K\bar{z})$

subject to: $\bar{s} = \bar{b} - A(\bar{h} + K\bar{z}) \geq \bar{0}$

$\bar{x} = \bar{h} + K\bar{z} \geq \bar{0}$

where $\bar{h}$, $\bar{b}$, A, and K have integer components and K is unimodular. Rearranging we can obtain

maximize $\bar{c}'\bar{h} + \bar{c}'K\bar{z}$

subject to: $\bar{s} = \bar{b} - A\bar{h} - AK\bar{z} \geq \bar{0}$

$\bar{x} = \bar{h} + K\bar{z} \geq \bar{0}$

$\bar{z}$ integer.

K will be chosen so that AK becomes the Hermite normal form of $A_1$ in (4). The integer problem over the cone (4), becomes

maximize $\bar{c}'\bar{h} + \bar{c}'K\bar{z}$

sujject to: $\bar{x}_N = (\bar{b}_1 + A_1\bar{h}) + A_1K\bar{z} \geq \bar{0}$

$\bar{z}$ integer, (5)

where $A_1K$ is in Hermite normal form and each element of $(\bar{b}_1 + A_1K)$ is non-negative and strictly less than the element

on the diagonal of $A_1K$ which corresponds to it. Thus, problem (5) is the problem upon which the Fourier-Motzkin elimination will be performed.

## V. FOURIER-MOTZKIN ELIMINATION

This approach is a means of finding solutions to a set of m linear inequalities in n variables. The technique sequentially eliminates variables, and each elimination generates a new set of inequalities in fewer variables. Finally, when only one variable is left, the inequalities which are present will impose certain bounds on that variable. Then by back-tracking, the other variables can sequentially be bounded also. The inherent disadvantage of the method is the possible rapid increase in the number of inequalities that must be considered. However, the success of the Fourier-Motzkin elimination method within the Bradley-Wahi algorithm relies on the following fact (to be proven shortly):

The transformation to the Hermite normal form of the integer cone problem results in the actual decrease by one of the number of inequalities, rather than an increase during the elimination.

Initiate the elimination, assuming we have a known objective function value $\bar{z}$. Then

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \geq \bar{z} \qquad (6)$$

From the Hermite normal form of the integer program over the cone (5), transform the problem and relabel the coefficients to obtain with (6),

$$\begin{aligned} c_1z_1 + c_2z_2 + \ldots + c_nz_n &\geq b_0 \\ a_{11}z_1 &\geq b_1 \\ a_{21}z_1 + a_{22}z_2 &\geq b_2 \\ \vdots & \quad \vdots \\ a_{n1}z_1 + a_{n2}z_2 + \ldots + a_{nn}z_n &\geq b_n \qquad (7) \\ z_1, z_2, \ldots, z_n \text{ integer} & \end{aligned}$$

where the coefficients $c_j$, $a_{ij}$, and $b_i$ have been obtained by the substitutions:

$$\begin{aligned} c_j &= cK_j \quad j = 1,\ldots,n \\ a_{ij} &\text{ is the } i,j^{th} \text{ entry of } A_1K \\ b_0 &= \bar{z} - \bar{c}\bar{h} \\ b_i &= -(b_{1i} + A_{1i}\bar{h}) \quad i = 1,\ldots,n \end{aligned}$$

for $K_j$ being the $j^{th}$ column of K and $A_{1i}$ the $i^{th}$ row of $A_1$. Further, by previous Hermite construction

$$\begin{aligned} &a_{ii} > 0 \quad i = 1,\ldots,n \\ &0 < -a_{ij} < a_{ii} \quad i = 1,\ldots,n;\ j = 1,\ldots,\ i - 1 \\ &0 < -b_i < a_{ii} \quad i = 1,\ldots,n. \end{aligned}$$

Then the problem consists of n + 1 linear inequalities in n variables.

Theorem 4.2

If the linear problem associated with the original integer problem has a bounded optimal solution (as we have assumed), then $c_n \leqslant 0$ in the Hermite cone problem, (7).

Proof:

(By contradiction) Assume $c_n > 0$. Then a solution with $z_n$ arbitrarily large and $z_j = 0$ for $j = 1,2,\ldots,n-1$ will satisfy the constraints of (7). But the value of $b_0$ in (7) can then be made arbitrarily large and this is linearly related to $\bar{z}$ in (6). Thus $\bar{z}$ can also be made arbitrarily large which is a contradiction to the fact that we assumed at the outset, that the linear programming problem has a bounded optimal solution.

Theorem 4.3

The application of the Fourier-Motzkin elimination to the system of inequalities (7) results in a system of inequalities with one less inequality at each step of the elimination.

Proof:

We will eliminate $z_n$ first. From Theorem 4.2, $c_n \leqslant 0$. If $c_n = 0$, then the generated system of inequalities is the original system with variable $z_n$ removed and the last inequality deleted.

Since $z_n$ appears in only two inequalities (the first and last) it is eliminated by multiplying each inequality by the coefficient of $z_n$ in the other and then subtracting. For $c_n < 0$ and with $a_{nn} > 0$, we generate the following inequality

$$\sum_{j=1}^{n-1} (a_{nn}c_j - c_n a_{nj}) z_j \geq a_{nn}b_0 - b_n c_n$$

Let $c'_j = a_{nn}c_j - c_n a_{nj}$, $j = 1,\ldots,n-1$

$$b'_0 = a_{nn}b_o - b_n c_n,$$

then dropping the two inequalities in $z_n$, we have generated the following system of inequalities:

$$\begin{aligned}
c'_1 z_1 + c'_2 z_2 + \ldots + c'_{n-1} z_{n-1} &\geq b'_0 \\
a_{11} z_1 &\geq b_1 \\
a_{21} z_1 + a_{22} z_2 &\geq b_2 \\
&\vdots \\
a_{n-1,1} z_1 + a_{n-1,2} z_2 + \ldots + a_{n-1,n-1} z_{n-1} &\geq b_{n-1}
\end{aligned}$$

$$z_1, z_2, \ldots, z_{n-1} \text{ integer}$$

Thus the generated set of inequalities has one less (n) inequality and one less (n-1) variable. This new set of inequalities has a bounded solution since (7) does, and so by Theorem 4.2, $c'_{n-1} \leq 0$. Then the above arguments are simply repeated for the elimination of $z_{n-1}, z_{n-2}, \ldots, z_2$

in that order. At each step the generated set of inequalities will consist of one less inequality and one less variable.

So now it has been proven that Hermite normal form allows a decrease rather than an increase in the number of inequalities while narrowing down to the first bounds of $\bar{z} = (z_1, z_2, \ldots, z_n)$. Only the inequality in the cost factors ($c_j$'s) is modified. It is modified by the following transformation for the $k^{th}$ elimination (which removes the $n - k + 1^{st}$ variable),

$$c_j^{(n-k+1)} = a_{kk}c_j^{(n-k)} - c_k^{(n-k)}a_{kj},\ j = 1, \ldots, k-1$$

$$b_0^{(n-k+1)} = a_{kk}b_0^{(n-k)} - c_k^{(n-k)}b_k.$$

Also it has been shown that the rightmost cost coefficient ($c_k$) will be nonpositive.

After eliminating $z_n, z_{n-1}, \ldots, z_2$, the following two inequalities remain,

$$c_1^{(n-1)}z_1 \geq b_0^{(n-1)}$$

$$a_{11}z_1 \geq b_1$$

where $c_1^{(n-1)} \leq 0$, $a_{11} > 0$, and $b_1 \leq 0$. Assume for the moment that $c_1^{(n-1)} < 0$. Then after dividing by $c_1^{(n-1)}$ and $a_{11}$ obtain

$$z_1 \leq b_0^{(n-1)}/c_1^{(n-1)} \qquad (8)$$

$$z_1 \geq b_1/a_{11}.$$

Thus the cost constraint provides us with an upper bound on $z_1$, and the other constraint provides a lower bound. Also notice now, that if $c_1^{(n-1)}$ had not been negative then it would not be possible to put an upper bound on $z_1$ which would contradict our boundedness assumption. From (8) by rounding up and down obtain

$$\text{(upper bound)} \quad BU(1) = [b_0^{(n-1)}/c_1^{(n-1)}]$$

$$\text{(lower bound)} \quad BL(1) = \left\langle b_1/a_{11} \right\rangle$$

Now the process for working back through the inequalities will be as follows. Start by setting $z_1$ at its lower integer bound and substituting this value back into the system of inequalities. Using the lower bound will then provide lower bounds for other variables since the off-diagonal elements are negative, i.e.,

$$a_{21}BL(1) + a_{22}z_2 \geq b_2$$

$$a_{22}z_2 \geq b_2 - a_{21}BL(1)$$

$$= b_2 + |a_{21}|BL(1)$$

$$\text{and } a_{22}z_2 \leq b_2 + |a_{21}|BU(1).$$

We progress in this manner through the variables setting them at their lower bounds and thus determining new bounds. Having set the $k - 1^{st}$ variable, we determine the

bounds for $z_k$ by extracting the $k^{th}$ constraint and the $n - k^{th}$ form of the cost inequality

$$a_{k1}z_1 + a_{k2}z_2 + \ldots + a_{kk}z_k \geqslant b_k$$

$$c_1^{(n-k)}z_1 + c_2^{(n-k)}z_2 + \ldots + c_k^{(n-k)}z_k \geqslant b_0^{(n-k)}.$$

Rearranging and dividing by $a_{kk} > 0$ and $c_k^{(n-k)} \leqslant 0$ (as before, $c_k^{(n-k)}$ is negative due to the boundedness assumption), obtain the bounds

$$z_k \leqslant \frac{1}{c_k^{(n-k)}} \{b_0^{(n-k)} - c_1^{(n-k)}z_1^* - c_2^{(n-k)}z_2^* - \ldots - c_k^{(n-k)}z_{k-1}^*\}$$

$$z_k \geqslant \frac{1}{a_{kk}} \{b_k - a_{k1}z_1^* - a_{k2}z_2^* - \ldots - a_{k,k-1}z_{k-1}^*\}$$

where $z_1^*, z_2^*, \ldots, z_{k-1}^*$ are the assigned values, and we are able to calculate the actual bounds

$$BU(k) = \frac{1}{c_k^{(n-k)}} \{b_0^{(n-k)} - c_1^{(n-k)}z_1^* - c_2^{(n-k)}z_2^* - \ldots - c_{k-1}^{(n-k)}z_{k-1}^*\}$$

$$BL(k) = \frac{1}{a_{kk}} \{b_k - a_{k1}z_1^* - a_{k2}z_2^* - \ldots - a_{k,k-1}z_{k-1}^*\}$$

The assigned values of $z_1, z_2, \ldots, z_{k-1}$ will be denoted as $z_1^*, z_2^*, \ldots, z_{k-1}^*$ rather than BL(1), BL(2), ..., BL(k-1), since $z_j$ = BL(j) by assignment only until on some iteration

k we find that BU(k) < BL(k) which is impossible. At this point we must backtrack and make the following assignment

$$z_{k-1} = BL(k - 1) + 1$$

assuming that $BL(k - 1) + 1 \leq BU(k - 1)$. That is, if during a backwards step we find that the assigned value $z_j^*$ exceeds BU(j) (the upper bound), we must backtrack to the previous variable, making the above assignment.

An assignment up to the $k^{th}$ variable

$$(z_1, z_2, \ldots, z_k) = (z_1^*, z_2^*, \ldots, z_k^*)$$

is called a partial solution. Our goal is to move forward to a feasible solution for all variables

$$(z_1, z_2, \ldots, z_n) = (z_1^*, z_2^*, \ldots, z_n^*)$$

which is referred to as a completion. Through each iteration the objective function is increased until, on some iteration, no feasible completion can be found. This will be demonstrated by $z_1^* > BU(1)$ on some backtracking step.

For the purpose of this method, the goal is simply to find a completion. This is because we are seeking an improvement to the solution provided by the primal algorithm. If there is no improvement (i.e., if the solution from the primal algorithm is optimal), then the Bradley-Wahi algorithm will indicate this in its inability to find a feasible

completion. That is, having found the Hermite normal form and performed the elimination, we reach $z_1^* > BU(1)$ and would then conclude that the present solution is optimal.

A summary of the elimination and bounding procedure is:

1. Eliminate the right-most variable $z_k$ from the cost inequality by the substitution

$$c_j^{(n-k+1)} = a_{kk}c_j^{(n-k)} - c_k^{(n-k)}a_{kj},\ j = 1,\ldots,\ k-1$$

$$b_0^{(n-k+1)} = a_{kk}b_0^{(n-k)} - b_k c_k^{(n-k)}$$

2. Drop the $k^{th}$ constraint and go to 1, if $k > 1$.

3. Determine upper and lower bounds

$$BU(k) = \frac{1}{c_k^{(n-k)}}\{b_0^{(n-k)} - c_1^{(n-k)}z_1^* - c_2^{(n-k)}z_2^* - \ldots - c_{k-1}^{(n-k)}z_{k-1}^*\}$$

$$BL(k) = \frac{1}{a_{kk}}\{b_k - a_{k1}z_1^* - a_{k2}z_2^* - \ldots - a_{k,k-1}z_{k-1}^*\}$$

4. If $BL(k) \leq BU(k)$, set $z_k^* = BL(k)$.

5. If $k = 1$ and $BL(1) > BU(1)$ or $z_1^* > BU(1)$, terminate.

6. If $k > 1$ and $BL(k) > BU(k)$, set $z_{k-1}^* = z_{k-1}^* + 1$.

7. IF $z_{k-1}^* > BU(k-1)$, set $k = k - 1$ and $z_{k-1}^* = z_{k-1}^* + 1$ and go to 5.

8. If $k < n$, go to 3.

When the Bradley-Wahi algorithm has provided a solution $z^*$ which has proved to be a feasible one, then we return the

associated $\bar{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ to the beginning of the primal algorithm as a starting feasible solution. As noted before, this new feasible solution $\bar{x}^*$ provides the algorithm with a stop to possible infinite cycling and therefore finiteness has been assured by the algorithm.

Examples will now be given to show how these algorithms work. The first one illustrates the primal algorithm.

Example 4.1

$$\text{Maximize } z = x_1 + x_2$$

$$x_1 + 2x_2 + x_3 = 12$$

$$5x_1 + 2x_2 + x_4 = 30$$

$$2x_1 - x_2 + x_5 = 9$$

$$x_1, x_2 \geq 0, \text{ integer.}$$

The sequence of tableaus derived for the solution of this problem is given, each one being in the Tucker formulation. The pivotal $\lambda = a_{rs}$ is circled in each tableau and the optimal solution is obtained in tableau 7. ($c_j \geq 0$, for all j.)

The second example will show how the Fourier-Motzkin elimination is performed.

Example 4.2

$$\text{Maximize } 2x_1 + 4x_2$$

$$\text{Subject to: } -2x_1 + 3x_2 + x_3 = 12$$

$$6x_1 + 9x_2 + x_4 = 54$$

$$x_1, x_2 \geq 0, \text{ integer.}$$

Tableau I

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 0 | -1 | -1 |
| $s_1$ | 12 | 1 | 2 |
| $s_2$ | 30 | 5 | 2 |
| $s_3$ | 9 | (2) | -1 |
| $x_1$ | 0 | -1 | 0 |
| $x_2$ | 0 | 0 | -1 |
| I | 4 | 1 | -1 |

Tableau II

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 4 | 1 | -2 |
| $s_1$ | 8 | -1 | 3 |
| $s_2$ | 10 | -5 | 7 |
| $s_3$ | 1 | -2 | (1) |
| $x_1$ | 4 | 1 | -1 |
| $x_2$ | 0 | 0 | -1 |
| I | 1 | -2 | 1 |

Tableau III

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 6 | -3 | 2 |
| $s_1$ | 5 | 5 | -3 |
| $s_2$ | 3 | (9) | -7 |
| $s_3$ | 0 | 0 | -1 |
| $x_1$ | 5 | -1 | 1 |
| $x_2$ | 1 | -2 | 1 |
| I | 0 | 1 | -1 |

Tableau IV

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 6 | 3 | -1 |
| $s_1$ | 5 | -5 | -2 |
| $s_2$ | 3 | -9 | (2) |
| $s_3$ | 0 | 0 | -1 |
| $x_1$ | 5 | 1 | 0 |
| $x_2$ | 1 | 2 | -1 |
| I | 1 | -5 | 1 |

Tableau V

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 7 | -2 | 1 |
| $s_1$ | 3 | (5) | -2 |
| $s_2$ | 1 | 1 | -2 |
| $s_3$ | 1 | -5 | 1 |
| $x_1$ | 5 | 1 | 0 |
| $x_2$ | 2 | -3 | 1 |
| I | 0 | 1 | -1 |

Tableau VI

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 7 | 2 | -1 |
| $s_1$ | 3 | -5 | (3) |
| $s_2$ | 1 | -1 | -1 |
| $s_3$ | 1 | 5 | -4 |
| $x_1$ | 5 | -1 | 1 |
| $x_2$ | 2 | 3 | -2 |
| I | 1 | -2 | 1 |

Tableau VII

| | b | $-y_1$ | $-y_2$ |
|---|---|---|---|
| z | 8 | 0 | 1 |
| $s_1$ | 0 | 1 | -3 |
| $s_2$ | 2 | -3 | 1 |
| $s_3$ | 5 | -3 | 4 |
| $x_1$ | 4 | 1 | -1 |
| $x_2$ | 4 | -1 | 2 |
| I | | | |

Here we can see that the optimal solution is

$x^* = (4, 4)$ yielding

$z^* = 8$.

After using the simplex algorithm, we arrive at the optimal linear programming solution which is (3/2, 5). This is not the optimal integer solution of course, so we use the enumerative method discussed in Chapter IV. Since both constraints are active we can obtain

$$b_1 = \begin{bmatrix} 12 \\ 54 \end{bmatrix} \text{ and } A_1 = \begin{bmatrix} 2 & -3 \\ -6 & -9 \end{bmatrix}$$

Now, we proceed to perform the column operations to get $A_1$ into Hermite normal form.

$$\left[\begin{array}{c|cc} 12 & 2 & -3 \\ 54 & -6 & -9 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

Add column 3 to column 2 and then change the sign of new column 2.

$$\left[\begin{array}{c|cc} 12 & 1 & -3 \\ 54 & 15 & -9 \\ \hline 0 & -1 & 0 \\ 0 & -1 & 1 \end{array}\right]$$

Add 3 × column 2 to column 3.

$$\left[\begin{array}{c|cc} 12 & 1 & 0 \\ 54 & 15 & 36 \\ \hline 0 & -1 & -3 \\ 0 & -1 & -2 \end{array}\right]$$

Add -1 × column 3 to column 2.

$$\left[\begin{array}{c|cc} 12 & 1 & 0 \\ 54 & -21 & 36 \\ 0 & 2 & -3 \\ 0 & 1 & -2 \end{array}\right] \qquad \text{Add } -12 \times \text{column 2 to column 1.}$$

$$\left[\begin{array}{c|cc} 0 & 1 & 0 \\ 306 & -21 & 36 \\ -24 & 2 & -3 \\ -12 & 1 & -2 \end{array}\right] \qquad \text{Add } -8 \times \text{column 3 to column 1.}$$

$$\left[\begin{array}{c|cc} 0 & 1 & 0 \\ 18 & -21 & 36 \\ 0 & 2 & -3 \\ 4 & 1 & -2 \end{array}\right] = \left[\begin{array}{c|c} b_1 + A_1h & A_1k \\ h & K \end{array}\right]$$

Thus $cK_z = [2, 4] \begin{bmatrix} 2 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = 8z_1 - 14z_2$

Using the Hermite normal form, initiate with $b_0 = 1$. (Since $b_0 = 0$ is trivially satisfied by $z_i = 0$, $i = 1, 2$.) Arrange the new problem as

$$A_1Kz \geq -(b_1 + A_1h)$$

$$cKz \geq b_0$$

and obtain

$$z_1 \geq 0 \quad (1)$$

$$-21z_1 + 36z_2 \geq -18 \quad (2)$$

$$8z_1 - 14z_2 \geq 1 \quad (3)$$

Eliminating $z_2$ from equation (3), obtain

$$-6z_1 \geqslant -216 \quad (3')$$

and the bounds are found to be

BL(1) = 0 from (1)

BU(1) = 36 from (3').

So we set $z_1 = 0$ and find the other bounds.

BL(2) = 0 from (2)

BU(2) = -1 from (3).

However, BU(2) < BL(2) implies there is no completion so we backtrack and set $z_1 = 1$, obtaining these bounds

BL(2) = 1 from (2)

BU(2) = 0 from (3).

Again we find no completion so once again we must backtrack. Set $z_1 = 2$ and obtain the following bounds,

BL(2) = 1 from (2)

BU(2) = 1 from (3).

Having found a completion, check

$$\tilde{b}_0 = \max(b_0 + 1,\ cz + 1) = (2,\ 9) = 9.$$

Repeat the elimination by replacing (3) with

$$8z_1 - 14z_2 \geqslant 9 \quad (3).$$

Recompute $b_0'$ to get

$$-6z_1 \geqslant 72 \quad (3')$$

and obtain the bounds

$BL(1) = 0$ from (1)

$BU(1) = -12$ from (3').

This means that no completion will be possible and that the present solution is optimal. (See step 5 of algorithm.) The present solution is $z_1 = 2$ and $z_2 = 1$. To find the optimal $\bar{x}^* = (x_1^*, x_2^*)$ we do the following.

$$\bar{x}^* = Kz + h = \begin{bmatrix} 2 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 2z_1 - 3z_2 \\ z_1 - 2z_2 + 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

and

$$z^* = ch + cKz = [2, 4] \begin{bmatrix} 0 \\ 4 \end{bmatrix} + [2, 4] \begin{bmatrix} 2 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$= 16 + 8z_1 - 14z_2 = 18.$$

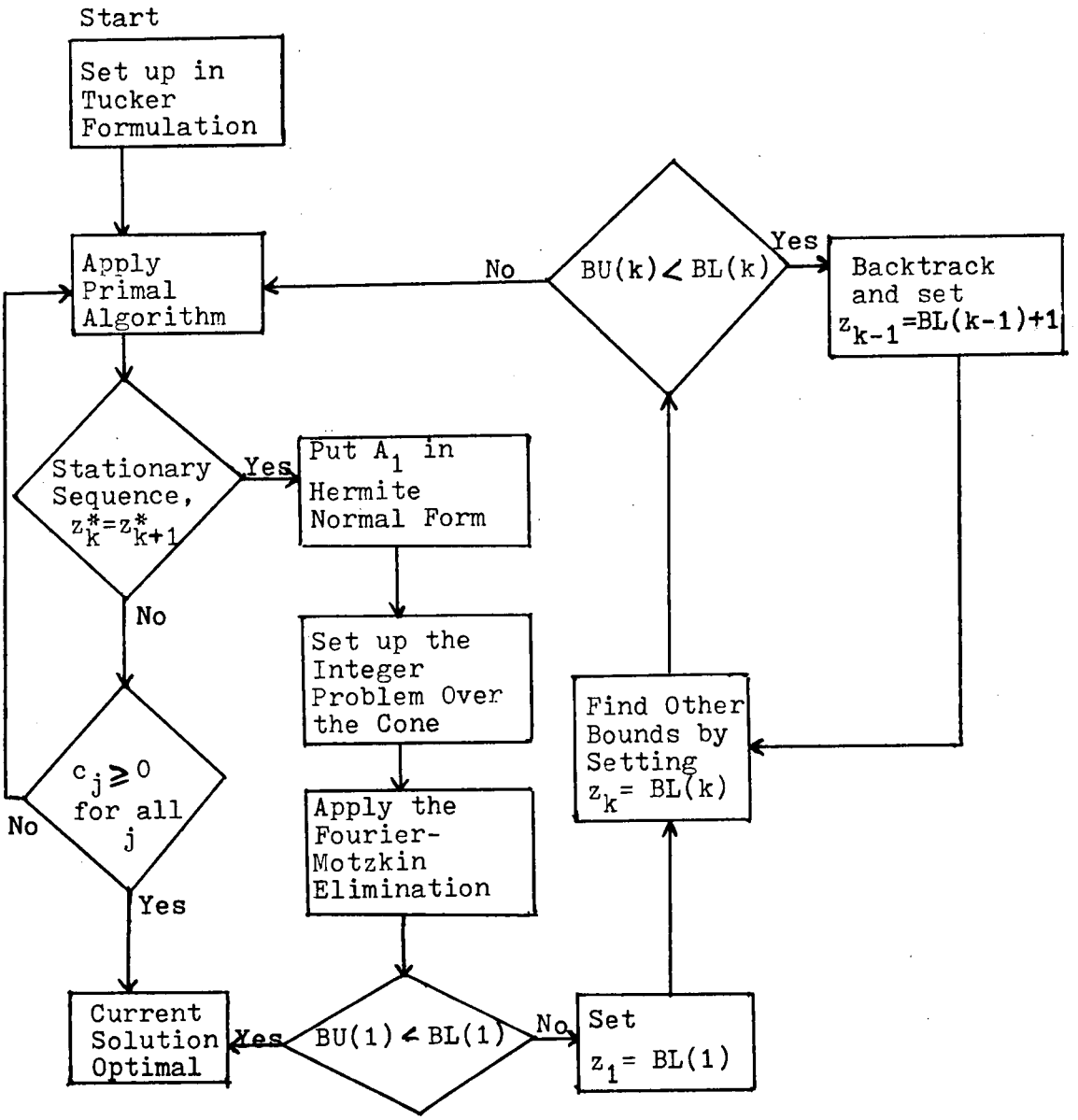


Figure 4-2. Flowchart for Combined Primal-Enumerative Method.

# CHAPTER V

## RECENT DEVELOPMENTS

Most recently, there are four main areas concerning approaches to solving the integer linear programming problem. Two of these have been discussed in this paper, namely algebraic (i.e., cutting plane methods) and enumerative methods. The other two areas of research are combinatorial and approximative or heuristic methods. Briefly, the recent developments in these methods are as follows.

### I. ALGEBRAIC APPROACHES

Little can be added to the discussion already included in this paper concerning cutting plane methods. New research to develop so called "better" cuts is being conducted, however a better cut does not necessarily mean that the resulting algorithm will actually be more effective in terms of helping solve the problem.

### II. ENUMERATIVE APPROACHES

Again, these methods are the basis of the commercial computer programs used today to solve the integer linear programming problem. The most current research is involved in trying to limit the extent of search in finding an optimal solution from the set of "all" possible feasible

solutions. The research is most intense in this area primarily due to the fact that the only way these methods can be evaluated is in the manner by which they eliminate a large percentage of the possible solution set. However, they can only be evaluated by how long it takes to solve a specific method on a specific machine.

Perhaps the best conclusion to be made is that when faced with an optimization problem to be solved by an enumerative method, it is best to devise some "custom made" rules adapted to the structure of the problem in the attempt to find the optimal solution.

## III. COMBINATORIAL APPROACHES

For the general integer linear programming problem very few combinatorial approaches exist. These approaches are designed and used primarily on problems which are not simply linear programming problems. Examples of these are assignment and transportation problems, which are integer problems but are not necessarily linear programming problems.

## IV. APPROXIMATIVE OR HEURISTIC METHODS

These methods involve either solution of one or a sequence of derived problems for finding an optimal solution. They rely on the use of some heuristics or reasonable rules for finding the solutions to these problems, which are

hopefully accompanied with bounds on the true optimal value, though they may not be. Increasing interest is being given to these methods for solving specially structured problems.

# REFERENCES

## REFERENCES


1. Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Proceedings of the Princeton Symposium on Mathematical Programming, edited by Harold W. Kuhn, Princeton University Press, Princeton, N. J., 1970.

2. Bradley, Gordon H., "Equivalent Integer Programs and Canonical Problems," Management Science 16 (1970).

3. Cooper, L. and Steinberg, D., Methods and Applications of Linear Programming, W. B. Saunders Company, Philadelphia, 1974.

4. Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley Publishing Company, Reading, Massachusetts, 1964.

5. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley Publishing Company, Reading, Massachusetts, 1969.

6. Kaufmann, A. and Henry-Labordére, A., "Integer and Mixed Programming: Theory and Applications," Mathematics in Science and Engineering, Volume 137, Academic Press, New York, 1977.

7. Ryan, P. J., "A Composite Primal-Enumerative Integer Programming Algorithm," Operations Research House Technical Report No. 72-4, Stanford University, Stanford, California, March 1972.

# VITA

Kerwin Edwin Everson was born in Flint, Michigan on April 24, 1956. He graduated from Davison High School in Davison, Michigan in June of 1974. Then he attended David Lipscomb College in Nashville, Tennessee and received a B.A. degree in Mathematics in June of 1977. After accepting a Graduate Teaching Assistantship at The University of Tennessee, he entered the Graduate School there in September of 1977 in order to pursue the Master of Science degree in Mathematics. He graduated from The University of Tennessee with a Master of Science degree in Mathematics in August of 1979.