THE DEVELOPMENT OF ALGORITHMS IN MATHEMATICAL PROGRAMMING

by

GHOLAMREZA JAHANSHAHLOU

A Thesis submitted to the Brunel University for the Degree of Doctor of Philosophy

Stats & O.R.

April 1976

Dedicated to

my late father, my mother and my wife

ACKNOWLEDGEMENT

I would like to thank Dr. G. Mitra, my supervisor, for his generous encouragement and valuable discussion throughout the period of my research at Brunel University.

I would also like to express my appreciation to Professor P. Macdonald, whom I could always approach for any advice or guidance.

Mrs. M. Demmar and Mrs. P. Dudley have taken meticulous care in typing the present thesis, and for this help I am greatly indebted.

Finally, I must thank my mother and my wife; but for their moral support this thesis would never have been completed.

G.R. Jahanshahlou
March 1976

ABSTRACT

In this thesis some problems in mathematical programming have been studied. Chapter 1 contains a brief review of the problems studied and the motivation for choosing these problems for further investigation.

The development of two algorithms for finding all the vertices of a convex polyhedron and their applications are reported in Chapter 2.

The linear complementary problem is studied in Chapter 3 and an algorithm to solve this problem is outlined.

Chapter 4 contains a description of the plant location problem (uncapacited). This problem has been studied in some depth and an algorithm to solve this problem is presented.

By using the Chinese representation of integers a new algorithm has been developed for transforming a nonsingular integer matrix into its Smith Normal Form; this work is discussed in Chapter 5.

A hybrid algorithm involving the gradient method and the simplex method has also been developed to solve the linear programming problem. Chapter 6 contains a description of this method.

The computer programs written in FORTRAN IV for these algorithms are set out in Appendices R1 to R5. A report on study of the group theory and its application in mathematical programming is presented as supplementary material.

The algorithms in Chapter 2 are new. Part one of Chapter 3 is a collection of published material on the solution of the linear complementary problem; however the algorithm in Part two of this Chapter is original.

The formulation of the plant location problem (uncapacited) together with some simplifications are claimed to be original. The use of Chinese representation of integers to transform an integer matrix into its Smith Normal Form is a new technique.

The algorithm in Chapter 6 illustrates a new approach to solve the linear programming problem by a mixture of gradient and simplex method.

CONTENTS

CHAPTER ONE

An Introduction to the Problems Investigated in This Thesis

1.1 General

The role of mathematics as an aid to the processes of scientific problem solving has been established for a long time. The rapid development of the digital computer over the last twenty-five years has greatly extended the applicability of mathematics, and it has become increasingly possible to obtain numerical solution to the mathematical models, and even to add to the refinement or the complexity of the models which can be solved.

From a theoretical point of view building a mathematical model is a process of writing a set of relations which connects the variables in the model. An algorithm is a set of rules for computation which must be followed to obtain a numerical solution to a problem or a class of problems. In this thesis the author is mainly concerned with developing algorithms (and the theory where appropriate) for the solution of a few well known problems in mathematical programming.

1.2 The general mathematical programming problem

The general mathematical programming problem may be defined as that of finding a vector $x \in R^n$ which maximizes or minimizes the function f(x) commonly known as the "objective function", subject to $x \in S$, where S is a subset of R^n.

The real impetus for the growth of interest in and the practical applications of programming problems came in 1947, when George Dantzig devised the simplex algorithm [1.1] for solving the general linear programming problem, which is a special case of the problem mentioned above where f(x) takes the form

$$f(x) = \sum_{j=1}^{n} c_j x_j \ , \tag{1}$$

and S is defined by a set of inequalities

$$\begin{cases} Ax \leq b \ , \\ x \geq 0 \ , \end{cases} \tag{2}$$

where A and b are two given matrices of order mxn and mx1 respectively; and c_j (j=1,...,n) are known constants.

If f(x) is in the form

$$f(x) = cx + x^T Dx \ , \tag{3}$$

where D is an nxn matrix, and T denotes the transpose of x, and set S is the same as defined in (2); then the problem is a quadratic programming problem [1.3].

If f(x) is not linear or some of the relations used to define S are nonlinear, then such a problem is commonly known as a non-linear programming problem.

The function f(x) and the set S may be classified from the point of convexity [1.2], and non-convexity. This classification also defines two categories of problems, called convex programming and non-convex programming.

An integer linear programming problem is a non-linear and non-convex problem which would be linear if it were not for the fact that some or all variables are restricted to integral values.

Therefore, the nature of f(x) and S define different problems. In this thesis the author has considered some well known problems of this type. In developing the theory and algorithms for their solution, the author has concerned himself mainly with the constraint set , and the methods of exploring these. In the following sections, 1.3 to 1.6, these problems are considered briefly.

1.3 Convex polyhedron and its vertices

A convex polyhedron is a convex set, S, which is defined by (2) see [1.2]. A vertex of this set is a point corresponding to a vector, not having more than m non-negative components different from zero. These points are specially important in the study of the classes of problems which are set out below.

(i) The fixed charge problem.

Consider a non-linear programming problem of the form

$$\text{Min } f(x) = \sum_{j=1}^{n} (c_j x_j + f_j y_j) \ , \tag{4}$$

subject to

$$\begin{cases} Ax \leq b \\ \ \ x \geq 0 \ , \end{cases} \tag{5}$$

$$y_i = \begin{cases} 0 & \text{if } \ x_j = 0 \ , \\ 1 & \text{if } \ x_j > 0 \ , \end{cases} \tag{5a}$$

$$\text{and} \quad f_j > 0 \ .$$

This is a concave objective function which is minimized over a linear constraint set. It can be shown [1.3] that the local optima of this function takes place at vertices of S. Therefore the local optima as also the global optimum is a basic solution of (5).

(ii) Alternative optimal solutions for linear programming problem.

It is well known that the simplex method [1.1] provides a solution to a linear programming problem, or it shows that no solution exists.

For problems which are dual degenerate the optimum solution is not unique and the alternative optima takes place at more than one vertex of the constraint set. In this situation one may be interested in finding all such alternative optimum (basic) solutions. One may note that these are vertices of the polyhedron in which in addition to the original constraints the objective function is constrained to be exactly equal to the optimal value.

(iii) Game theory.

Two person zero-sum games can be related to linear programming problems [1.2]. When mixed strategies are admitted, these take place at the vertices of the linear constraint set.

These are a few examples in which the vertices of S play an important role.

In chapter 2 the algorithms (two) for finding all the vertices of such a set, S, is described in detail.

1.4 Fundamental problem

Given the square matrix M of order NXN and the vector q of order N it is required to find the two non-negative vectors w,z each of order N such that they solve the system

$$\begin{cases} w = q + Mz \,, \\ z, w \geq 0 \\ w^T z = 0 \,. \end{cases} \tag{6}$$

This problem plays an important role in mathematical programming inasmuch as the special cases of this problem are linear programming problem, quadratic programming problem, and finding equilibrium points in bimatrix games which are stated as follows:

(i) Linear programming

Consider the linear programming problem

$$\text{Max } f(x) = cx \tag{7}$$

$$\text{subject to } Ax \leq b \quad x \geq 0 ,$$

and its dual [1.2]

$$\text{Min } f'(v) = bv \tag{8}$$

$$\text{subject to } A^T v \geq c , \quad v \geq 0 ,$$

where A, b, x are defined as earlier and v is a vector of order m. Introducing a vector y of slack variables of order m, and a vector u of surplus variables of order n these problems may be re-expressed as

$$\text{Max } f(x) = cx \tag{9}$$
$$\text{subject to } Ax + Iy = b; \quad x, y \geq 0 ,$$

and

$$\text{Min } f'(v) = bv \tag{10}$$
$$\text{subject to } A^T v - Iu = c; \quad u, v \geq 0 .$$

From duality theory [1.2] of linear programming it follows that for the optimum feasible solution to this problem pair the following relationships must hold,

$$\begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} -c \\ b \end{pmatrix} + \begin{pmatrix} 0 & A^T \\ -A & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} ; \quad \begin{matrix} x, y, u, v \geq 0 . \\ x.u, y.v = 0 \end{matrix} \tag{11}$$

By substituting $w = \begin{pmatrix} u \\ y \end{pmatrix}$ $q = \begin{pmatrix} -c \\ b \end{pmatrix}$, $z = \begin{pmatrix} x \\ v \end{pmatrix}$ $M = \begin{pmatrix} 0 & A^T \\ -A & 0 \end{pmatrix}$

(11) becomes equivalent to the Fundamental problem;
where $N = n + m$.

(ii) Quadratic programming problem

Consider the quadratic programming problem stated as:

$$\text{Min } z = cx + \tfrac{1}{2}x^T Dx \tag{12}$$

$$\text{subject to} \quad Ax \geq b \ , \quad x \geq 0 \ , \quad (D \text{ is symmetric})$$

and for this quadratic programming problem define u, v as:

$$u = Dx - A^T y + c \ , \quad v = Ax - b \tag{13}$$

A vector x^0 yields minimum $\bar{z}$ only if there exists a vector y^0 and vector u^0, v^0 given by (13) satisfying

$$\begin{aligned} & x^0 \geq 0 \ , \quad u^0 \geq 0 \ , \quad y^0 \geq 0 \ , \quad u^0 \geq 0 \ , \\ & x^0 u^0 = 0 \ , \quad y^0 v^0 = 0 \ . \end{aligned} \tag{14}$$

See [3.5]. Thus the problem of solving a quadratic programming problem leads to a search for the solution of the system

$$\begin{aligned} & u = Dx - A^T y + c \ , \quad x \geq 0 \ , \quad y \geq 0 \ , \\ & v = Ax - b \ , \quad u \geq 0 \ , \quad v \geq 0 \ , \\ & xu + yu = 0 \ . \end{aligned} \tag{15}$$

Again by substituting

$$w = \begin{pmatrix} u \\ v \end{pmatrix} \quad q = \begin{pmatrix} c \\ -b \end{pmatrix} \quad M = \begin{pmatrix} D & -A^T \\ A & 0 \end{pmatrix} \quad z = \begin{pmatrix} x \\ y \end{pmatrix} \ , \tag{16}$$

the problem becomes the Fundamental Problem.

(iii) Bimatrix Game

Consider the bimatrix game defined by two pay-off matrices A, B [1.4] each of order mxn such that m + n = N. It follows from the necessary condition for an equilibrium point that,

$$\begin{aligned} y &= Ax - e_m \qquad y, x \geq 0 \\ u &= B^T x - e_n \qquad u, v \geq 0 \\ xu &+ yv = 0 \;. \end{aligned} \tag{17}$$

This is once again in the form of the Fundamental Problem, where

$$M = \begin{pmatrix} 0 & A \\ B^T & 0 \end{pmatrix}, \; w = \begin{pmatrix} y \\ u \end{pmatrix}, \; q = \begin{pmatrix} -e_m \\ -e_n \end{pmatrix}, \; z = \begin{pmatrix} x \\ v \end{pmatrix} .$$

In chapter 3, the work done to date for solving the Fundamental Problem is reviewed, in addition an algorithm developed by the author is described. This method is particularly powerful since no assumption concerning the nature of the matrix M is made.

1.5 Integer programming and related problems.

Formulation of certain classes of combinatorial problems, and problems of other types as integer or mixed integer linear programming,is well known in literature and adequately dealt with in text books. The two prominent methods cutting plane and branch and bound are the most commonly used methods for the solution of these problems. However, for certain problems the methods seem to require unusually large computing effort; this despite a formal proof for their convergence. In trying to visualize the solutions in which such difficulties arise, and, if possible to counter these, it is desirable to take advantage of the structure of these problems. Equally another approach may be to transform these problems into equivalent problems which may be handled by more efficient algorithms.

The following two types of problems have been handled in this way in the present investigation.

a) Group knapsack problem

Mathematically knapsack problem may be stated as:

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

subject to (20)

$$\sum_{j=1}^{n} a_j x_j \leq b \ ,$$

$$x_j \geq 0 \text{ , and integer for } j = 1,\dots,n \ ,$$

where a_j, c_j for $(j = 1,\dots,n)$ are given integers.

There exist a number of special algorithms which solve this problem efficiently [5.4].

Application of group theory to integer programming problem makes it possible to transform a given problem into its group knapsack problem (see G.R. Jahanshahlou & G. Mitra [5.3]) which can be solved very efficiently (as a knapsack problem). Under certain conditions the solution to the corresponding group knapsack problem provides the desired solution to the given problem.

Let B be the optimal basis of the linear programming problem corresponding to the given integer program, which is obtained by relaxation of integrality condition on the variables. Transforming B into its Smith Normal Form $\Delta = [\delta_i]$, (see [5.4]) where δ_i divides $\delta_i + 1$ for all i $(i = 1,\dots,m-1)$ is one of the major steps in re-expressing the problem into its corresponding group knapsack form. It is proven that δ_i in the ith step of the procedure of the transforming B into Δ is the greatest common factor of the elements of the matrix which is of order $(m - i + 1)(m - i + 1)$.

The chinese representation of integers seems to be an efficient method of finding the greatest common factor of a set of elements of the matrix in the above mentioned transformation.

This idea is exploited in the algorithm developed by the author whereby the matrix B is transformed to its Smith Normal Form Δ. This work is fully described in chapter 5.

b) Plant location problem.

Given m plants with unlimited capacity and handling cost functions which are concave, it is required to find an optimum subset of the plants to supply the demand centres in the system.

In this simple form, plant location can be posed as a transportation problem with no constraint on the amount shipped from any source. However, there is a cost associated with each source (plant). This cost (called a fixed cost or a fixed charge) is zero if nothing is shipped from the plant, i.e. plant is 'closed'. It is positive and independent of the amount shipped if any shipment from the plants takes place, i.e. the plant is 'open'. Because the fixed charge associated with each plant does not vary linearly with the amount shipped from the plant (there is discontinuity at zero shipment) this problem cannot be handled using standard linear programming method. Balinski [4.3] has formulated this problem as a mixed-integer program.

In practical problems this approach leads to, say five to twenty thousand rows and about the same number of columns [4.1]. From the practical standpoint therefore a standard solution technique for mixed integer program cannot be applied directly unless particularly efficient ways can be found to solve the associated linear programming subproblems generated by such a technique. Because of the assumption of unlimited capacity of the plants, S, the region in which the objective function is minimized has got a special structure.

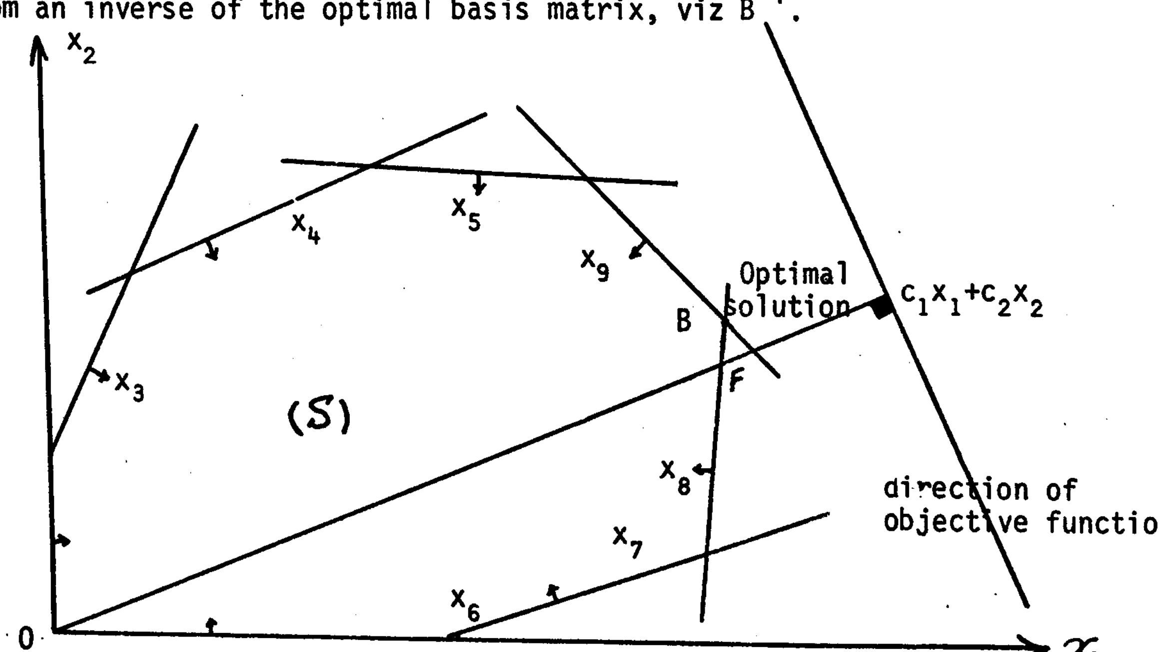
The associated linear programs obtained by relaxing the integer condition on the fixed charge variables assume minima at some vertices of S. It is proven [4.1] that such vertices of S are generated directly without recourse to the simplex method.

In chapter 4 this problem is discussed in some depth.

1.6 Hybrid gradient and simplex method.

To date the simplex method is the most attractive method for solving linear programming problems. This is an iterative method which converges to an optimal solution in a finite number of steps, or alternatively shows that there is no solution to the given problem.

In the final step of the simplex method the information concerning the optimal solution and the dual solution values can be obtained from an inverse of the optimal basis matrix, viz B^{-1}.



Fig(1)

The ability to obtain this final basis matrix rapidly, therefore constitutes the foundation of any accelerated method for solving the linear programming problem. The hybrid gradient method developed by the author is set out to achieve exactly this. Consider the problem illustrated in Fig(1). The region S is a convex set which is defined by the set of inequalities $x_i \geq 0$ (i + 1,...,9). The objective function $c_1x_1 + c_2x_2$ is to be maximized over this region. Starting from the origin and moving in the direction perpendicular to the objective function one exits from the region S at the point F, which is a feasible point (but not basic). Then at this point the values of eight variables (in general more than m variables) are positive. In the proposed method some of these variables may be reduced to zero without recourse to pivotal transformation, and a basic feasible solution with improved objective function value is obtained. If the basic feasible solution so obtained is not the optimal solution then the whole procedure may be applied repeatedly until an optimum solution is obtained. It seems plausible that such an algorithm which starts from F instead of 0 (as in the usual simplex method) might reduce some intermediate steps in arriving at the optimal solution.

This investigation is fully described in chapter 6.

References 1

1.1 Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

1.2 Hadley, G., Linear Programming, Addison-Wesley Publishing Company, 1962.

1.3 ____________ Nonlinear and Dynamic Programming, Addison-Wesley Publishing Company, 1964.

1.4 G. Mitra Lecture note, on linear programming, Brunel University.

CHAPTER TWO

Two Algorithms for Finding All the Vertices of a Convex Polyhedron.

2.0 Summary

In this chapter the problem of finding all the vertices of a convex polyhedron

$$S = \{x \mid Ax \leq b, x \geq 0\} \tag{1}$$

defined by a set of linear inequalities, and non-negativity condition on the variables is considered. Two algorithms for its solution are presented. The first employs a tree construction scheme, and in the second the convex set S is partitioned into two mutually exclusive sets $S \cap H, S \cap \Gamma H$ such that $S \cap H \cup S \cap \Gamma H = S$ By finding extreme points lying in each of these sets, and to do this further separating one such set into mutually exclusive subsets, all the feasible extreme points of S are obtained.

2.1 Introduction

In this introductory section the work of the other authors in solving this problem and the contexts in which the problem arises have been briefly reviewed. In the next section the notation and the representation of the tableau are explained. In Section 2.3 first the theory underlying the tree development algorithm : ALGORITHM I is presented, the algorithm is then described. Another algorithm, "Branch and Exclude", and labelled as ALGORITHM II is developed in section 2.5. Two worked examples solved by the application of each of these algorithms are set out in section 2.4 and section 2.6 respectively. In section 2.7 the computational results are discussed.

The following problems may be cited as possible areas of application of the algorithms discussed in this chapter

- In a two-person zero sum game [2.6] if there exists more than one optimal mixed strategy then the problem of finding all such optimal strategies may be investigated by these algorithms.

- If the problem of post optimal analysis [2.7] is posed as that of finding all the basic feasible solutions within a given percentage of the optimum solution, then this can be clearly investigated by the proposed methods. The limiting case of the above problem viz: all the optimal solutions must be within zero percent i.e. find all the basic optimal solutions of a dual degenerate problem can therefore be investigated in the same way.

- The plant location or the fixed charge problem involves minimisation of a concave function subject to linear constraints. A local optimum solution and hence the global optimum of this problem is an extreme point [2.3] hence for this problem vertices may be investigated for local and global optimality.

- Kirby et al [2.4] have considered a nonlinear programming problem which requires an algorithm like that proposed here for its computational solution.

For the solution of this problem Van-de-Panne [2.7] employs a method which he calls the 'Reverse Simplex' method. In this a linear form is first maximized over the linear constraint set. Starting from this basic feasible solution variables are introduced into the basis and the value of the objective function is decreased; by continuing this procedure all the extreme points are generated. Charnes [2.2] has discussed a method based on the simplex algorithm and Tary's solution to the labyrinth problem of the theory of graphs . Manas and Nedoma [2.5] have developed an algorithm which involves exploring the graph $\Gamma(V,U)$ adjoined to the polyhedron S; where V denotes the vertices and U the edges of the graph. This method is similar to the ALGORITHM I considered here. Motzkin, et al [2.6] provide a method based directly on the Fourier-Motzkin scheme for linear inequalities whereby the convex polyhedron is built up progressively introducing linear inequalities/half-spaces one at a time. Balinski's approach for solving this problem has somewhat motivated the second algorithm : ALGORITHM II considered in this paper. His method is further discussed in Section 2.5.

2.2. Notation and Tableau Representation.

Let A be an m x n matrix, b an m-vector, and x an n-vector of n unknowns, and S be the convex polyhedron defined by the inequalities

$$A\,x \le b, x \ge 0. \qquad (2)$$

This may be written out in full as

$$\begin{array}{l} a_{11}(-x_1) + a_{12}(-x_2) + \ldots + a_{1n}(-x_n) + b_1 = x_{n+1} \\ a_{21}(-x_1) + a_{22}(-x_2) + \ldots + a_{2n}(-x_n) + b_2 = x_{n+2} \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ a_{m1}(-x_1) + a_{m2}(-x_2) + \ldots + a_{mn}(-x_n) + b_m = x_{n+m} \\ x_1, x_2, \ldots, x_n \ge 0, \\ x_{n+1}, x_{n+2}, \ldots, x_{n+m} \ge 0, \end{array} \qquad (3)$$

where $x^s = (x_{n+1}, x_{n+2}, \ldots, x_{n+m})$ is a vector of the slack variables.

In this chapter the condensed form of the simplex tableau due to Tucker, has been used, the initial tableau has the form set out in Tableau 0. Basic variables appear in the left hand column, and non basic variables in the top row. A basic feasible solution corresponds to a vertex of S.

$$
\begin{array}{c|ccccc|}
 & (-x_1) & (-x_2) & \dots & (-x_n) & 1 \\
\hline
x_{n+1} & a_{11} & a_{12} & \dots & a_{1n} & b_1 \\
x_{n+2} & a_{21} & a_{22} & \dots & a_{2n} & b_2 \\
 & \dots & & \vdots & \dots & \\
x_{n+m} & a_{m1} & a_{m2} & \dots & a_{mn} & b_m \\
\hline
\end{array}
$$

Tableau 0.

2.3. Tree Development Algorithm : ALGORITHM I.

The algorithm described in this section and also that in section 2.5 use an essential simplex step to go from one vertex to another. Consider the vertex X^i defined by the intersection of n hyperplanes $x_{r_1} = 0$, $x_{r_2} = 0$, ... $x_{r_q} = 0$, ... $x_{r_n} = 0$, and the vertex X^j where all the hyperplanes are the same as that of X^i except $x_{r_q} = 0$, is replaced by $x_{r_{n+p}} = 0$. The vertices are contained in Tableau 3.1 and Tableau 3.2, and the pivotal transformation on $\bar{a}_{pq}$ generates X^j from X^i ; for the feasibility of X^j the following identity must hold,

$$\frac{\bar{b}^i_p}{\bar{a}_{pq}} = \underset{1\le t\le m}{\text{Min}} \left\{ \frac{\bar{b}_t}{\bar{a}_{tq}} \,\middle|\, \bar{a}_{tq} > 0 \right\}. \tag{4}$$

X^j, X^i are called 'adjacent basic feasible' solutions or 'adjacent vertices'. During a typical simplex step the unbounded condition may be detected i.e., for a given column q, $\bar{a}_{tq} \leq 0$ for all t. In this case an auxiliary bounded problem may be proposed : this is discussed later on in the present section.

	$-x_{r_1}$	$-x_{r_2}$	..	$-x_{r_q}$	...	1
$x_{r_{n+1}}$				$\bar{a}_{1q}$		$\bar{b}_1^i$
.						
.						
$x_{r_{n+p}}$				$\bar{a}_{pq}$		$\bar{b}_p^i$
$x_{r_{n+m}}$				$\bar{a}_{mq}$		$\bar{b}_m^i$

Tableau 3.1

	$-x_{r_1}$	$-x_{r_2}$	..	$-x_{r_{n+p}}$	...	1
$x_{r_{n+1}}$						$\bar{b}_1^j$
.						
.						
x_{r_q}						
$x_{r_{n+m}}$						$\bar{b}_m^j$

Tableau 3.2

The steps of the algorithm are now stated and are accompanied by explanatory notes to outline both the abstract idea of the graphical representation of the process and the practical implementation.
In the following algorithmic steps a basis and the corresponding solution is said to be 'marked' if it is already generated and the indices of the variables in the basis are stored in a table.

Step 1. Choose X^o, Tableau 0 (it may be chosen arbitrarily from any of the vertices) and call this the node zero of the tree to be constructed. Set the counters $N = 0$, $K = 0$, 'mark' X^o as being generated.

Step 2. Take the tableau N associated with the vertex X^N; from this generate all the adjacent solutions $X^{K+1}, X^{K+2}, \ldots, X^{K+k_N}$ which were not 'marked'. Now 'mark' these solutions as being generated. The vertex X^N in S corresponds to the node N in the tree, and $(N,K+1)$, $(N,K+2)$ ... $(N,K+k_N)$ are edges connecting node N to the nodes corresponding to the k_N vertices generated in this step.
Set $K = K+k_N$

Step 3. Set $N = N+1$, if $N \leq K$ go to step 2.

Step 4. All feasible vertices of S are obtained.

The procedure constructs a tree to the polyhedron S with the following properties.

a) There is a one-to-one correspondence between the nodes of the tree and the vertices of S i.e., for each vertex, say X^i we have a node i in the tree and vice versa.

b) The above mentioned tree is a spanning tree for the graph formed by the edges and the vertices of the polyhedral set S.

To confirm the property (a) stated above the following theorem is stated and proved.

Theorem : The algorithm above generates all the vertices of S.

Proof : Let X^i be an arbitrary vertex of S ; it is required to prove that there exists a node in the tree corresponding to X^i. The graph formed by the edges and vertices of S is connected, hence, there exists a path η that connects X^o to X^i. Let (X^o,X^{i_1}), $(X^{i_1},X^{i_2})\ldots(X^{i_r},X^i)$ be all the edges of η in order. X^{i_1} is an adjacent vertex of X^o since all the adjacent vertices of X^o are generated so X^{i_1} must be generated, and corresponds to a node in the tree.

Repeating this argument it follows that i_2, i_3 etc., are nodes in the tree hence i must be a node in the tree.

The number of vertices of the polyhedral set is finite: a ready bound is ${}^{n+m}C_m = \frac{(n+m)!}{m!n!}$, but there are stronger bounds [2.5]. Hence the above algorithm terminates in a finite number of steps since a vertex once generated is never revisited.

If S is unbounded, a condition that may be detected at the simplex step when there is no positive pivot cf.p.16, then the following procedure is suggested. Introduce the closed half space

$$H_i = a_{m+1,1}(-x_1) + \ldots\ldots + a_{m+1,n}(-x_n) + b_{m+1} = 0 \qquad (5)$$

and $H_i \geqslant 0$.

such that all the vertices of S lie on the feasible side of H_i.

Define the polyhedron S_1:

$$\begin{aligned} Ax &\le b \\ H_i &\ge 0 \\ x &\ge 0, \end{aligned} \tag{6}$$

such that S_1 is bounded. The algorithm can now be applied to find all the vertices of S_1 and if out of these the vertices for which $H_i = 0$ are dropped all the vertices of S are obtained.

2.4.A worked example by ALGORITHM I.

An example due to Balinski [2.1] is solved by this algorithm. The polyhedron and also the tree constructed in the process of solution are illustrated in Figure 1 and Figure 2. Table 1 illustrates the steps of the ALGORITHM 1 as related to this problem.

Find all the vertices of a convex polyhedron defined by

$$\begin{aligned} x_4 &= 3(-x_1) + 2(-x_2) - 1(-x_3) + 6 \ge 0 \\ x_5 &= 3(-x_1) + 2(-x_2) + 4(-x_3) + 16 \ge 0 \\ x_6 &= 3(-x_1) + 0(-x_2) - 4(-x_3) + 3 \ge 0 \\ x_7 &= \tfrac{9}{4}(-x_1) + 4(-x_2) + 3(-x_3) + 17 \ge 0 \\ x_8 &= (-x_1) + 2(-x_2) + 1(-x_3) + 10 \ge 0 \\ & x_1, x_2, x_3, \ldots\ldots, x_8 \ge 0 . \end{aligned} \tag{7}$$

The starting tableau is as follows :

	$(-x_1)$	$(-x_2)$	$(-x_3)$	1
x_4	3	2	-1	6
x_5	3	2	4	16
x_6	3	0	-4	3
x_7	$\frac{9}{4}$	4	3	17
x_8	1	2	1	10

Tableau 4.0

So $X^0 = (0,0,0)$ is the starting node

	$(-x_6)$	$(-x_2)$	$(-x_3)$	1
x_4	-1	2	3	3
x_5	-1	2	8	13
x_1	$\frac{1}{3}$	0	$-\frac{4}{3}$	1
x_7	$-\frac{3}{4}$	4	6	$\frac{59}{4}$
x_8	$-\frac{1}{3}$	2	$\frac{7}{3}$	9

$X^1 = (1,0,0)$

Tableau 4.1

	$(-x_1)$	$(-x_4)$	$(-x_3)$	1
x_2	$\frac{3}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	3
x_5	0	-1	5	10
x_6	3	0	-4	3
x_7	$-\frac{15}{4}$	-2	5	5
x_8	-2	-1	2	4

Tableau 4.2

$X^2 = (0,3,0)$

	$(-x_1)$	$(-x_2)$	$(-x_5)$	1
x_4	$\frac{15}{4}$	$\frac{5}{2}$	$\frac{1}{4}$	10
x_3	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	4
x_6	6	2	1	19
x_7	0	$\frac{5}{2}$	$-\frac{3}{4}$	5
x_8	$\frac{1}{4}$	$\frac{3}{2}$	$-\frac{1}{4}$	6

Tableau 4.3

$X^3 = (0,0,4)$

	$(-x_6)$	$(-x_4)$	$(-x_3)$	1
x_2	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{3}{2}$
x_5	0	-1	5	10
x_1	$\frac{1}{3}$	0	$-\frac{4}{3}$	1
x_7	$\frac{5}{4}$	-2	0	$\frac{35}{4}$
x_8	$\frac{2}{3}$	-1	$-\frac{2}{3}$	6

Tableau 4.4

$X^4 = (1,\frac{3}{2},0)$

	$(-x_6)$	$(-x_2)$	$(-x_4)$
x_3	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$
x_5	$\frac{5}{3}$	$\frac{10}{3}$	$-\frac{8}{3}$
x_1	$-\frac{1}{9}$	$\frac{8}{9}$	$\frac{4}{9}$
x_7	$\frac{5}{4}$	0	-2
x_8	$\frac{4}{9}$	$\frac{4}{9}$	$-\frac{7}{9}$

Tableau 4.5

$X^5 = (\frac{7}{3},0,1)$

	$(-x_1)$	$(-x_4)$	$(-x_7)$	1
x_2	$\frac{9}{8}$	$\frac{3}{10}$	$\frac{1}{10}$	$\frac{7}{2}$
x_5	$\frac{15}{4}$	1	-1	5
x_6	0	$-\frac{8}{5}$	$\frac{4}{5}$	7
x_3	$-\frac{3}{4}$	$-\frac{2}{5}$	$\frac{1}{5}$	1
x_8	$-\frac{1}{2}$	$-\frac{1}{5}$	$-\frac{2}{5}$	2

Tableau 4.6

$X^6 = (0,\frac{7}{2},1)$

	$(-x_4)$	$(-x_2)$	$(-x_5)$
x_1	$\frac{4}{15}$	$\frac{2}{3}$	$\frac{1}{5}$
x_3	$-\frac{1}{5}$	0	$\frac{1}{5}$
x_6	$-\frac{24}{15}$	-2	$\frac{3}{15}$
x_7	0	$\frac{5}{2}$	$-\frac{3}{4}$
x_8	$-\frac{1}{15}$	$\frac{4}{3}$	$-\frac{4}{15}$

Tableau 4.7

$X^7 = (\frac{8}{3},0,2)$

	$(-x_1)$	$(-x_7)$	$(-x_5)$	1
x_4	$\frac{15}{4}$	-1	1	5
x_3	$\frac{3}{4}$	$-\frac{1}{5}$	$\frac{2}{5}$	3
x_6	6	$-\frac{4}{5}$	$\frac{8}{5}$	15
x_2	0	$\frac{2}{5}$	$-\frac{3}{10}$	2
x_8	$\frac{1}{4}$	$-\frac{3}{5}$	$\frac{1}{5}$	3

Tableau 4.8

$X^8 = (0,2,3)$

	$(-x_4)$	$(-x_7)$	$(-x_5)$
x_1	$\frac{4}{15}$	$\frac{4}{15}$	$-\frac{4}{15}$
x_3	$-\frac{1}{5}$	$\frac{1}{5}$	0
x_6	$-\frac{8}{5}$	0	$\frac{4}{5}$
x_2	0	$-\frac{3}{10}$	$\frac{2}{5}$
x_8	$-\frac{1}{15}$	$\frac{2}{15}$	$-\frac{8}{15}$

Tableau 4.9

$X^9 = (\frac{4}{3},2,2)$

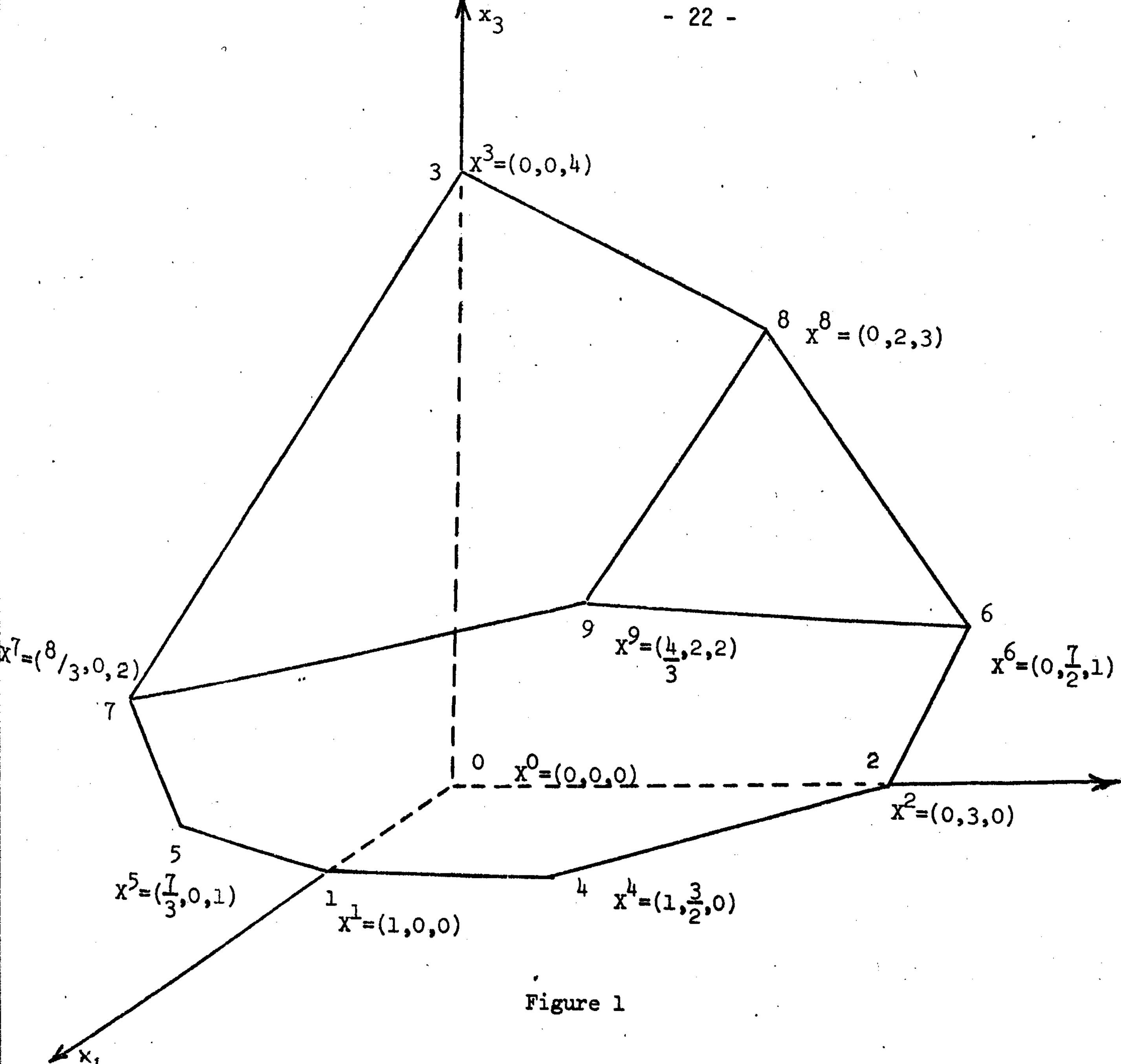


Figure 1

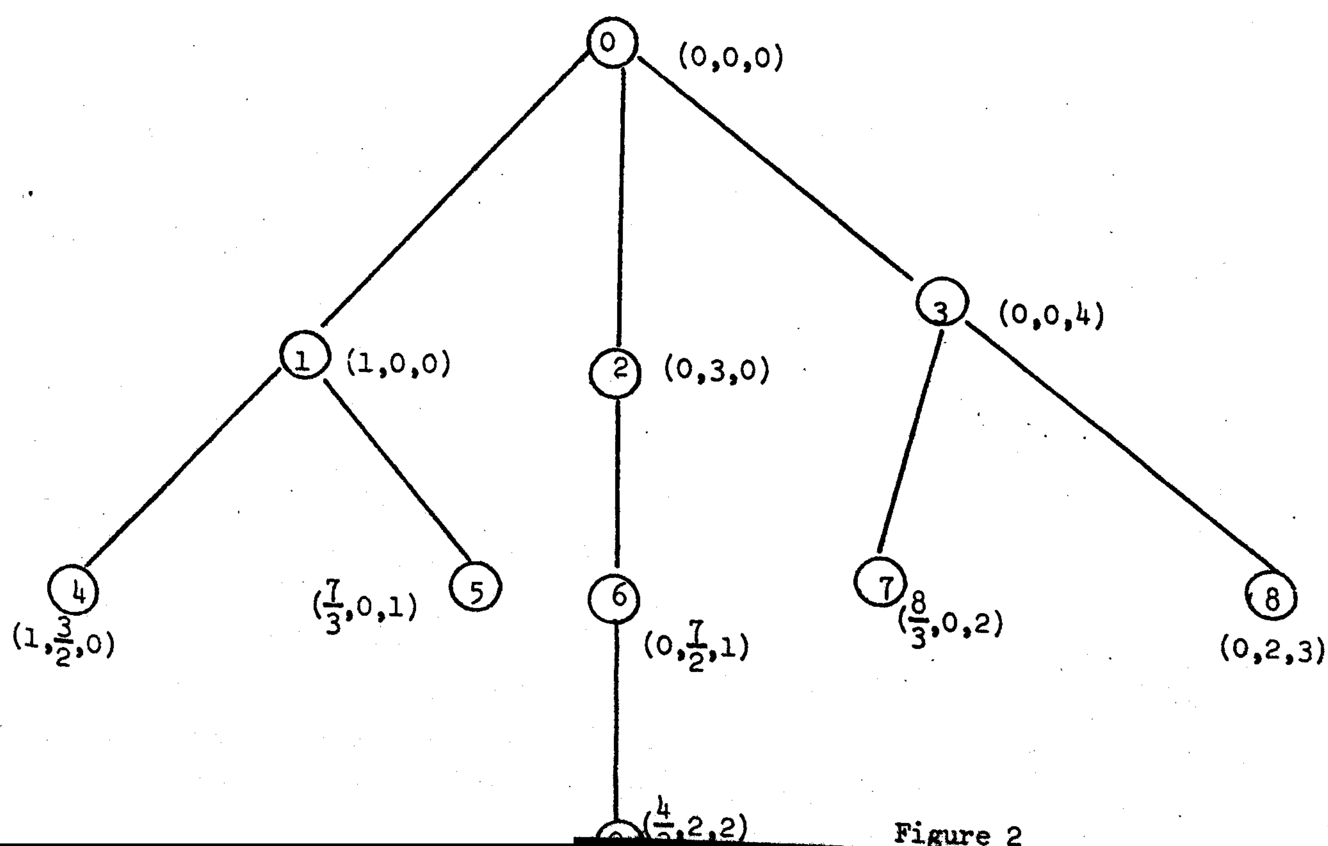


Figure 2

ITERATION NO	VERTICES GENERATED IN THE N^{TH} ITERATION	N	K
1	X^1, X^2, X^3	0	3
2	X^4, X^5	1	5
3	X^6	2	6
4	X^7, X^8	3	8
5	No vertex generated	4	8
6	" " "	5	8
7	X^9	6	9
8	No vertex generated	7	9
9	" " "	8	9
10	" " "	9	9

Table 1

2.5. Branch and Exclude Algorithm: ALGORITHM II.

The algorithm developed in this section has been motivated by Balinski's approach towards solving this problem. A summary of his method is set out below. Let H_i corresponding to $x_{n+i}=0$, $i=1,2,\ldots,m$, be one of the m constraint hyperplanes of the system of inequalities defining the convex set S.

Step 1 Pick a hyperplane H_i.

Step 2 Find all the vertices of S which lie on H_i.

Step 3 Drop the inequality or half space requirement $x_{n+i} \geq 0$ where $x_{n+i}=0$ defines H_i.

Step 4 Pick some other Hyper-plane H_j not already dealt with and continue as in Step 2.

The process terminates in a finite number of steps when m of the m+n half spaces are dropped. Note that because the constraints are relaxed (Step 3) it is possible to generate the extreme points on H_j which may not be extreme points of S. This implies that in order to find all the extreme points of S some infeasible intermediate bases are generated.

The algorithm developed by the author is based on the following property (assuming primal degeneracy does not occur) of the convex polyhedron: all the vertices of S are contined in

(a) all the vertices of $S \cap H_i$, (8)

and

(b) all the vertices of $S \cap \Gamma H_i$. (9)

In the tableau representation of the feasible bases i.e. vertices of S note that all the tableaus for which $x_{n+i}=0$ (non basic) represent vertices on $S \cap H_i$ (8) and the tableaus in which x_{n+i} is basic represent the rest of the vertices corresponding to $S \cap \Gamma H_i$ (9).

In the statement of the algorithm which follows, the condensed tableau due to Tucker is used. As in the ALGORITHM 1 one starts from the Tableau 0 and then constructs a tree of subproblems in the following way. Let x_{r_q} be a variable that is chosen to enter the basis and let $x_{r_{n+p}}$ be a variable in the pth row that is chosen to go out. The rule for finding the pivot element $\bar{a}_{pq}$ is set out later under the heading of 'PIVOT RULE'. However, carrying out the corresponding pivotal operation leads to two subproblems P_1 and P_2 emanating from P_0

P_0: Find all the extreme points of S

P_1: Find all the extreme points of S in which x_{r_q} is non basic i.e. these vertices belong to $S \cap H_{r_q}$ (10)

P_2: Find all the extreme points of S in which x_{r_q} is basic i.e., these vertices belong to $S \cap \Gamma H_{r_q}$

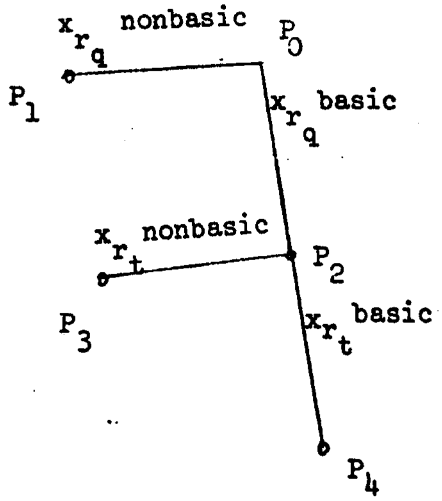


Figure 3

This therefore connects P_1, P_2 to P_0 by the two branches of a bifurcating tree. Out of P_2 one may propose two further subproblems P_3, P_4 by pivoting on a variable x_{r_t}.

P_3: Find all the extreme points of S in which x_{r_t} is non basic (x_{r_q} is forced to be basic) i.e. these vertices belong to $S \cap \Gamma H_{r_q} \cap H_{r_t}$.

P_4: Find all the extreme points of S in which x_{r_t} is basic (x_{r_q} is forced to be basic) i.e. these vertices belong to $S \cap \Gamma H_{r_q} \cap \Gamma H_{r_t}$

The process is illustrated in Figure 3 and may be continued until no further branchingis possible on any of the subproblems in the tree, at which stage all the vertices are generated.

'PIVOT RULE' Before stating the rule the following needs to be defined. In a tableau a variable that has already been chosen for branching is called a 'starred' variable. Similarly a row in which a 'starred' variable is pivoted in one branching step is called a 'flagged' row.

- Column Choice

Choose out of the variables not 'starred' in a tableau a variable which admits a row (out of the rows not 'flagged' in the tableau) with a positive entry. Let this be column q and variable x_{r_q}.

- Row Choice

Out of the rows not 'flagged' in the tableau find a row p such that

$$\frac{\bar{b}_p}{\bar{a}_{pq}} = \min_{t \notin F} \left\{ \frac{\bar{b}_t}{\bar{a}_{tq}} \;\middle|\; \bar{a}_{tq} > 0 \right\} \tag{11}$$

where F denotes the set of row indices which are flagged.

Having chosen this row p x_{r_q} is 'starred' and one branch of the tree is generated and the other branch is obtained by a pivotal transformation and the row p is 'flagged'.

The steps of the branch and exclude algorithm may now be stated.

Step 1 Start from Tableau 0 of section 2.2 as the first basic feasible solution of the set of constraints. Set N=0, K=0.

Step 2. Pick Tableau N from the stack of tableaus, go to Auxiliary step. If a pivotal transformation is carried out set K=K+1 Label new Tableau K, add it to the stack of tableaus and go to Step 3. If no pivotal transformation takes place go to Step 4.

Step 3 Pick Tableau number K out of the stack and go to Auxiliary Step. If a pivotal transformation has taken place put K=K+1, label new tableau K add to stack and go to Step 3. If no pivotal transformation has taken place go to Step 2.

Step 4 Set N = N+1 if N $>$ K go to Exit, otherwise go to Step 2.

Exit All the vertices of the polyhedron are contained in all the basic solutions so far generated. Some of the basic solutions may not be feasible.

Auxiliary Step

Choose the column with the smallest index number q for which a pivot $\bar{a}_{pq}$ may be found by the 'PIVOT RULE' stated earlier. Carry out a pivotal transformation and return to the calling step. If no such column q and variable x_{r_q} can be found no pivotal transformation can be carried out. Return to the calling step.

In this section no formal proof of the finiteness of the steps of the algorithm is supplied. However, ignoring the case where the polyhedral set S is unbounded, the finiteness of the algorithm simply follows from the exclusivity properties of (8),(9) and the adjacency property discussed in Section 2.3.

2.6. A Worked Exampled by ALGORITHM II

The problem due to Balinski [2.1] is again solved in this section this time by ALGORITHM II. The steps of the algorithm as related to this problem are illustrated in TABLE 2. The tree developed by this method is illustrated in Figure 4, and the sequence of tableaus which are generated are set out in Tableau 6.0 until Tableau 6.22.

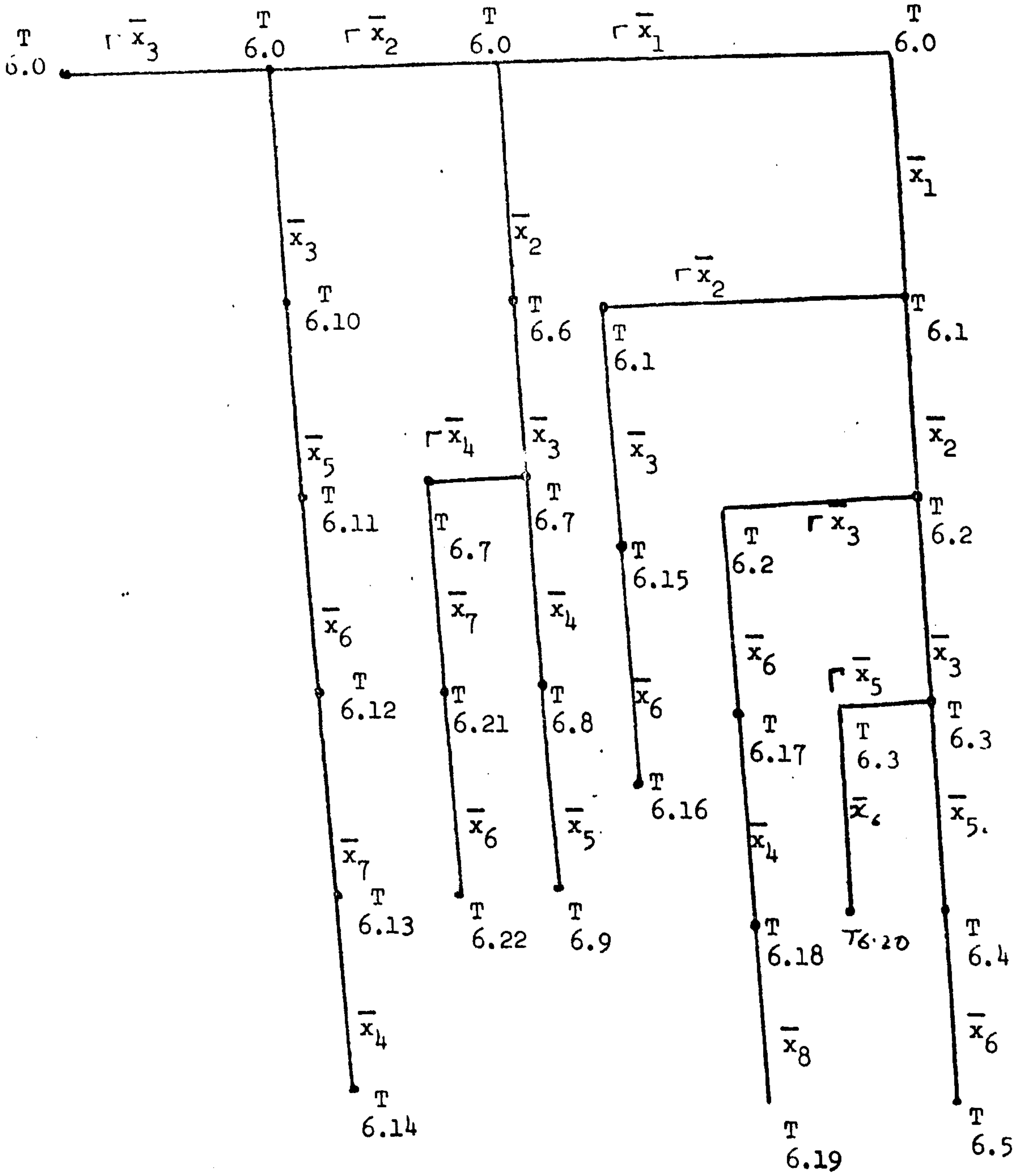


Figure 4

$\bar{x}_j$: indicates variable pivoted in basis and row 'flagged'

Γ $\bar{x}_j$: indicates variable 'starred' and forced to remain non basic

T6.i stands for Tableau 6.i corresponding to node i of the tree.

Iteration No.	The feasibility of the Tableau F=feasible N=infeasible	Row number that is flagged in iteration K	Variable that is starred in iteration K	N	K
0	F	No row flagged	No variable starred	0	0
1	F	Row 3 in Tableau 6.1	x_1 in Tableau 6.0	0	1
2	F	" 1 " 6.2	x_2 " 6.1	0	2
3	N	" 2 " 6.3	x_3 " 6.2	0	3
4	N	" 5 " 6.4	x_5 " 6.3	0	4
5	N	" 4 " 6.5	x_6 " 6.4	0	5
6	F	" 1 " 6.6	x_2 " 6.0	0	6
7	F	" 4 " 6.7	x_3 " 6.6	0	7
8	F	" 2 " 6.8	x_4 " 6.7	0	8
9	N	" 3 " 6.9	x_5 " 6.8	0	9
10	F	" 2 " 6.10	x_3 " 6.0	0	10
11	N	" 3 " 6.11	x_5 " 6.11	0	11
12	N	" 4 " 6.12	x_6 " 6.11	0	12
13	N	" 1 " 6.13	x_7 " 6.12	0	13
14	N	" 5 " 6.14	x_4 " 6.13	0	14
15	F	" 1 " 6.15	x_3 " 6.1	1	15
16	F	" 2 " 6.16	x_6 " 6.15	1	16
17	N	" 4 " 6.17	x_6 " 6.2	2	17
18	N	" 5 " 6.18	x_4 " 6.17	2	18
19	N	" 2 " 6.19	x_8 " 6.18	2	19
20	F	" 4 " 6.20	x_6 " 6.3	3	20
21	-	No pivotal	No pivotal	4	20
22	-	transformation	transformation	5	20
23	-	carried out	carried out	6	20
24	N	row 3 in Tableau6.21	x_7 in Tableau 6.7	7	21
25	N	" 2 " 6.22	x_6 " 6.21	7	22

After iteration 25 N increases and K remains fixed until N=22 when the search is complete.

Table 2

	$(-x_1^*)$	$(-x_2^*)$	$(-x_3^*)$	1
x_4	3.0	2.0	-1.0	6.0
x_5	3.0	2.0	4.0	16.0
x_6	3.0	0.0	-4.0	3.0
x_7	2.25	4.0	3.0	17.0
x_8	1.0	2.0	1.0	10.0

Tableau 6.0

$X^o = (0,0,0)$

	$(-x_6)$	$(-x_2^*)$	$(-x_3^*)$	1
x_4	-1.0	2.0	3.0	3.0
x_5	-1.0	2.0	8.0	13.0
$\bar{x}_1$	0.33	0.0	-1.33	1.0
x_7	-.75	4.0	6.0	14.75
x_8	-.33	2.0	2.33	9.0

Tableau 6.1

$X^1 = (1.0,0.0,0.0)$

	$(-x_6^*)$	$(-x_4)$	$(-x_3^*)$	1
$\bar{x}_2$	-0.5	0.5	1.5	1.5
x_5	0.0	-1.0	5.0	10.0
$\bar{x}_1$	0.33	0.0	-1.33	1.0
x_7	1.25	-2.0	0.0	8.75
x_8	0.67	-1.0	-.67	6.0

Tableau 6.2

$X^2 = (1,1.5,0)$

	$(-x_6^*)$	$(-x_4)$	$(-x_5^*)$	1
$\bar{x}_2$	-.30	0.8	-0.5	-1.5
$\bar{x}_3$	0.20	-.2	0.0	2.0
$\bar{x}_1$	0.27	-0.27	0.33	3.67
x_7	0.0	-2.0	1.25	8.75
x_8	0.67	-1.13	0.13	7.33

Tableau 6.3

$X^3 = (3.67,-1.5,2.0)$

	$(-x_6^*)$	$(-x_4)$	$(-x_8)$	1
$\bar{x}_2$	1.0	-1.75	2.25	15.0
$\bar{x}_3$	-1.0	1.5	-1.5	-9.0
$\bar{x}_1$	-1.0	2.0	-2.0	-11.
x_7	1.25	-2.0	0.0	8.75
$\bar{x}_5$	5.00	-8.5	7.5	55.0

Tableau 6.4

$X^4 = (-11.0,15.0,-9.0)$

	$(-x_7)$	$(-x_4)$	$(-x_8)$	1
$\bar{x}_2$	-.8	-.15	2.25	8.0
$\bar{x}_3$	0.8	-.10	-1.5	-2.0
$\bar{x}_1$	0.8	0.4	-2.0	-4.0
$\bar{x}_6$	0.8	-1.6	0.0	7.0
$\bar{x}_5$	-4.0	-.5	7.5	20.0

Tableau 6.5

$X^5 = (-4.0,8.0,-2.0)$

	$(-x_1^*)$	$(-x_4)$	$(-x_3^*)$	1
$\bar{x}_2$	1.5	-0.5	0.5	3.0
x_5	0.0	5.0	-1.0	10.0
x_6	3.0	-4.0	0.0	3.0
x_7	-3.75	5.0	-2.0	5.0
x_8	-2.0	-1.0	2.0	4.0

Tableau 6.6

$X^6 = (0.0,3.0,0.0)$

	$(-x_1^*)$	$(-x_4^*)$	$(-x_7^*)$	1
$\bar{x}_2$	1.12	0.3	0.1	3.5
x_5	3.75	1.0	-1.0	5.0
x_6	0.0	-1.6	0.8	7.0
$\bar{x}_3$	0.75	-0.4	0.2	1.0
x_8	-.5	-0.2	-0.4	2.0

Tableau 6.7

$X^7 = (0.0,3.5,1.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_5)$	$(-x_7)$	1
$\bar{x}_2$	0.0	-.3	0.4	2.0
$\bar{x}_4$	3.75	1.0	-1.0	5.0
x_6	6.0	1.6	-.8	15.0
$\bar{x}_3$	0.75	0.4	-.2	3.0
x_8	0.25	0.2	-.6	3.0

Tableau 6.8

$X^8 = (0.0,2.0,3.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_2)$	$(-\overset{*}{x}_7)$	1
x_4	3.75	3.33	0.33	11.67
$\bar{x}_3$	0.75	1.33	0.33	5.67
$\bar{x}_5$	0.0	-3.33	-1.33	-6.67
$\bar{x}_6$	6.0	5.33	1.33	25.67
x_8	0.25	0.67	-.33	4.33

Tableau 6.12

$X^{12} = (0.0,0.0,5.67)$

	$(-\overset{*}{x}_1)$	$(-x_6)$	$(-x_7)$	1
$\bar{x}_2$	1.12	0.19	0.25	4.81
$\bar{x}_4$	0.0	-.63	-.5	-4.37
$\bar{x}_5$	3.75	0.63	-.5	9.37
$\bar{x}_3$	-.75	-.25	0.0	-.75
x_8	-.50	-.12	-0.5	1.13

Tableau 6.9

$X^9 = (0.0,4.81,-.75)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_2)$	$(-\overset{*}{x}_4)$	1
$\bar{x}_7$	11.25	10.0	3.0	35.0
$\bar{x}_3$	-3.0	-2.0	-1.0	-6.0
$\bar{x}_5$	15.0	10.0	4.0	40.0
$\bar{x}_6$	-9.0	-8.0	-4.0	-21.0
x_8	4.0	4.0	1.0	16.0

Tableau 6.13

$X^{13} = (0.0,0.0,-6.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_2)$	$(-\overset{*}{x}_5)$	1
x_4	3.75	2.5	0.25	10.0
$\bar{x}_3$	0.75	0.50	0.25	4.0
x_6	6.0	2.0	1.0	19.0
x_7	0.0	2.5	-0.75	5.0
x_8	0.25	1.5	-.25	6.0

Tableau 6.10

$X^{10} = (0.0,0.0,4.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_2)$	$(-x_8)$	1
$\bar{x}_7$	-.75	-2.0	-3.0	-13.0
$\bar{x}_3$	1.0	2.0	1.0	10.0
$\bar{x}_5$	-1.0	-6.0	-4.0	-24.0
$\bar{x}_6$	7.0	8.0	4.0	43.0
$\bar{x}_4$	4.0	4.0	1.0	16.0

Tableau 6.14

$X^{14} = (0.0,0.0,10.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_2)$	$(-\overset{*}{x}_6)$	1
x_4	2.25	2.0	-.25	5.25
$\bar{x}_3$	-0.75	0.0	-.25	-.75
$\bar{x}_5$	6.00	2.0	1.0	19.0
x_7	4.5	4.0	0.75	19.25
x_8	1.75	2.0	0.25	10.75

Tableau 6.11

[redacted]

	$(-\overset{*}{x}_6)$	$(-\overset{*}{x}_2)$	$(-x_4)$	1
$\bar{x}_3$	-0.33	.67	0.33	1.0
x_5	1.67	-3.33	-2.67	5.0
$\bar{x}_1$	-.11	.89	0.44	2.33
x_7	1.25	0.0	-2.0	8.75
x_8	.44	0.44	-.78	6.67

Tableau 6.15

[redacted]

	$(-x_5)$	$(-\overset{*}{x}_2)$	$(-x_4)$	1
$\bar{x}_3$	0.20	0.0	-0.2	2.0
$\bar{x}_6$	0.60	-2.0	-1.60	3.0
$\bar{x}_1$	0.07	0.67	0.27	2.67
x_7	-.75	2.5	0.0	5.0
x_8	-.27	1.33	-0.07	5.33

Tableau 6.16

$X^{16} = (2.66,0.0,2.0)$

	$(-x_7)$	$(-\overset{*}{x}_4)$	$(-\overset{*}{x}_3)$	1
$\bar{x}_2$	0.40	-.30	1.5	5.0
x_5	0.00	-1.0	5.0	10.0
$\bar{x}_1$	-.27	0.53	-1.33	-1.33
$\bar{x}_6$	0.80	-1.60	0.0	7.0
x_8	-.53	0.07	-.67	1.33

Tableau 6.17

$X^{17} = (-1.33,5.0,0.0)$

	$(-x_7)$	$(-\overset{*}{x}_8)$	$(-\overset{*}{x}_3)$	1
$\bar{x}_2$	-2.0	4.50	-1.5	11.0
x_5	-8.0	15.0	-5.0	30.0
$\bar{x}_1$	4.0	-8.0	4.0	-12.0
$\bar{x}_6$	-12.0	24.0	-16.0	39.0
$\bar{x}_4$	-8.0	15.0	-10.0	20.0

Tableau 6.18

$X^{18} = (-12.0,11.0,0.0)$

	$(-x_7)$	$(-x_5)$	$(-\overset{*}{x}_3)$	1
$\bar{x}_2$	0.40	-.30	0.0	2.0
$\bar{x}_8$	-.53	0.07	-.33	2.0
$\bar{x}_1$	-.27	0.53	1.33	4.0
$\bar{x}_6$	0.80	-1.60	-8.00	-9.0
$\bar{x}_4$	0.0	-1.00	-5.0	-10.0

Tableau 6.19

$X^{19} = (4.0,2.0,0.0)$

	$(-x_7)$	$(-x_4)$	$(-\overset{*}{x}_5)$	1
$\bar{x}_2$	0.40	0.0	-.30	2.0
$\bar{x}_3$	0.0	-.20	0.20	2.0
$\bar{x}_1$	-.27	0.27	0.27	1.33
$\bar{x}_6$	0.80	-1.60	0.0	7.0
x_8	-.53	-.07	0.13	2.67

Tableau 6.20

$X^{20} = (1.33),2.0,2.0)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_4)$	$(-\overset{*}{x}_6)$	1
$\bar{x}_2$	1.12	0.5	-.13	2.62
x_5	3.75	-1.0	1.25	13.75
$\bar{x}_7$	0.0	-2.0	1.25	8.75
$\bar{x}_3$	-.75	0.0	-.25	-.75
x_8	-.50	-1.0	0.50	5.50

Tableau 6.21

$X^{21} = (0.0,2.62,-.75)$

	$(-\overset{*}{x}_1)$	$(-\overset{*}{x}_4)$	(x_5)	1
$\bar{x}_2$	1.5	0.40	0.1	4.0
$\bar{x}_6$	3.0	-.80	0.8	11.0
$\bar{x}_7$	-3.75	-1.0	-1.0	-5.0
$\bar{x}_3$	0.0	-.20	0.2	2.0
x_8	-2.0	-.60	-.4	0.0

Tableau 6.22

$X^{22} = (0.0,4.0,2.0)$

2.7. Discussion

Both these algorithms have been programmed by the author in FORTRAN IV; the programs have been run to solve small problems using the ICL 1903A computer at the University. The times taken to solve the test problems depend on the core partition used and are not quoted here. However, one pertinent comparison between the two algorithms should be mentioned. Two of the problems solved by these algorithms may be quoted; these are of dimension 15 x 11 and 30 x 25. For the smaller problem the ALGORITHM I is faster than ALGORITHM II and vice versa. Finally it should be mentioned that these algorithms have been developed to use them as tools in investigating other Mathematical Programming problems.

References 2

2.1 Balinski, M.L., An Algorithm for finding all vertices of convex polyhedral sets. J.Soc.Indust.Appl.Math.9, 72-88(1961).

2.2 Charnes,A., Cooper, W.W., and Henderson,A., An Introduction to Linear Programming, Wiley, New York, 1953.

2.3. Hadley,G., Nonlinear and Dynamic Programming, Addison Wesley, p.91, 1964.

2.4 Kirby, M.J.L., Love,H.R., Swarup, K., Extreme Point Programming with Nonlinear Constraints, Research Report Dalhousie University (Canada) 1970.

2.5 Manas, M., and Nedoma, J., Finding all Vertices of a Convex Polyhedron, Numerische Mathematik 12. 226-229(1968).

2.6 Motzkin, T.S., Raiffa, H., Thompson, G.L. and Thrall, R.M., The double description method. In Contribution to the Theory of games, vol.II. Princeton: Princeton University Press 1953

2.7 Van de Panne, C., Post-Optimality Analysis via the Reverse Simplex Method and the Tary Method, July 1966, presented at TIMS (Warsaw, 1966).

CHAPTER THREE

The Linear Complementarity Problem

3.0 Summary

This chapter contains a brief review of the work done to solve the problem $w = q + Mz$, $w \geq 0$, $z \geq 0$, and $z^T w = 0$. An algorithm developed by the author to solve this problem is also described in this chapter. Unlike any other known algorithm this algorithm makes no assumption concerning the nature of the matrix M and finds all the solutions of the problem if these exist. If no solution exists to the problem then this can also be established by this method.

3.1 Introduction

Consider the linear complementarity problem,

$$w = q + Mz \qquad (1)$$

$$w, z \geq 0 \qquad (2)$$

$$z^T w = 0 \qquad (3)$$

w and z are vectors of n variables, q is a given n element vector, M is a given nxn matrix, and the superscript T denotes transposition. The above problem involves 2n variables, restricted to be non-negative, where (w_i, z_i), $i = 1,\ldots,n$, is a complementary pair; and w_i and z_i are complement of one another.

The special cases of linear complementarity problem are linear and quadratic programming problems, the problem of finding equilibrium points in bimatrix games and some engineering problems; for these and other applications see [3.6].

The two prominent methods of solution for the problem (1), (2), (3) are the principal pivoting method and Lemke's method. The method proposed by Lemke can be considered to be a generalization of Dantzig's self-dual parametric method (see [3.7], and its generalization for convex quadratic programming. This motivated S.R. McCammon [3.15] to develop his parametric pivoting method. Lemke has proven that his method finds a solution to the problem or else the solution comes to an unbounded ray and there is no solution to the given problem if M belongs to a class of matrices called copositive plus.

The principal pivoting method was developed by Cottle and Dantzig [3.6]. This method is applicable to the matrices, which have positive principal minors (in particular to positive definite matrices). The modified form of principal pivoting method can be applied to positive semi-definite matrices.

The method of solution of the problem (1), (2), (3) is generally dependent on the matrix M. In section 3.2, therefore, after introducing the relevant notation, some properties of pivotal transformation and different types of M matrices are considered. Section 3.3 contains brief descriptions of Lemke's method, principal pivoting algorithm, and some remarks on other proposals based on Lemke's method.

Lemke's method and the principal pivoting method may not produce the solution to the problem (1), (2), (3) even if such a solution exists; section 3.4 illustrates such a situation.

The method proposed by the author is then put forward in this section. Section 3.5 contains some remarks on the computational experiences of the author .

3.2 Some Preliminary Notation and Mathematical Background

3.2.1 Notation

Let R^{nxn} denote the set of nxn matrices with real coefficients, let $M \in R^{nxn}$, $M_{i\cdot}$ and $M_{\cdot i}$ denote the ith row and the ith column of M and m_{ij} denote the element of M in row i and column j. Further, let e denote the sum vector $(1,\ldots 1)^T \in R^{nx1}$ and e_i denote the unit vector whose ith component is unity and the other components are zero. The bar above a variable (say $\bar{z}_j$ or $\bar{w}_j$) denotes the explicit value of the variable.

3.2.2 Tableau Representation and Pivotal Transformation

In (1), the components of z are nonbasic variables, while the elements of w comprise the basic variables. A solution of problem (1) is any pair $(\bar{w},\bar{z})$ satisfying (1).

If for some $\bar{z} \geq 0$, $\bar{w} = q + M\bar{z} \geq 0$, then the pair $(\bar{w},\bar{z})$ provides a feasible solution to the problem (1), i.e., a solution which satisfies (1) and (2). A solution of (1) satisfying (3) is a complementary solution.

If every solution $(\bar{w},\bar{z})$ of the problem (1) contains not more than n zero components among the 2n variables (w,z), then the problem (1) is nondegenerate. In the present discussion only such nondegenerate problems are considered.

Assume that the element

$$m_{ij} \neq 0 \quad ;$$

then using the element $m_{ij} \neq 0$ a "pivotal transformation" may be carried out on the form

$$w = q + Mz \quad . \tag{4}$$

This transformation consists of

(a) solving the ith equation of (4) for the variable z_j, this requires dividing by the pivot element m_{ij} ,

(b) replacing z_j by the resulting expression in each of the remaining (n-1) equations.

Upon completion of a pivotal transformation, z_j becomes basic, while w_i becomes nonbasic. (w_i, z_j) is the pivot pair, and by specifying that this pair must be exchanged, the pivot is completely determined. The result of a sequence of pivotal transformations after t steps may be expressed as

$$w^t = q^t + M^t z^t \quad , \tag{5}$$

where w^t denotes the set of basic variables, while z^t denotes the set of nonbasic variables.

For completeness of notation the result of carrying out one pivotal transformation is summarized below. Given the tableau -0

	1	$-z_1^t$	$-z_2^t$	...	$-z_n^t$
w_1^t	q_1^t	$-m_{11}^t$	$-m_{12}^t$	...	$-m_{1n}^t$
w_2^t	q_2^t	$-m_{21}^t$	$-m_{22}^t$	...	$-m_{2n}^t$
				...	
w_n^t	q_n^t	$-m_{n1}^t$	$-m_{n2}^t$		$-m_{nn}^t$

Tableau -0

The next tableau is constructed by the following relationship:

1) $(-m_{ij}^{t+1}) = (1)/(-m_{ij}^t)$

2) $(-m_{ik}^{t+1}) = (-m_{ik}^t)/(-m_{ij}^t) \quad 1 \le k \le n \qquad k \ne j$

3) $(-m_{\ell j}^{t+1}) = -(-m_{\ell j}^t)/(-m_{ij}^t) \quad 1 \le \ell \le n \qquad \ell \ne i$.

4) $(-m_{\ell k}^{t+1}) = (-m_{\ell k}^t) - (-m_{\ell j}^t)(-m_{ik}^t)/(-m_{ij}^t)$

5) Replace the variables such that w_i^{t+1} is the variable in the ith row (z_j^t is renamed) and z_j^{t+1} is the variable in the jth column. (w_i^t is renamed).

3.2.3 Some Different Types of M Matrix [3.1,3.11,3.15]

Definition: A positive matrix M is a matrix, such that: $m_{ij} > 0$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Similarly non-negative and negative matrices can be defined.

Matrix M is said to be 'skew-symmetric' if

$$M = -M^T .$$

Lemma: A necessary and sufficient condition that a $n \times n$ matrix M be skew-symmetric is that $x^T M x = 0$, for all values of the vector $x \in R^n$.

Definition: A $n \times n$ matrix M is positive definite (positive semi-definite) if and only if, for all vector $x \in R^n$ $x \neq 0$ the relation $xMx^T > 0 (xMx^T \geq 0)$ holds.

Lemma: Let M be a $n \times n$ positive semi-definite matrix, then $m_{ii} \geq 0$ for all i $(i = 1, \ldots, n)$; if $m_{ii} = 0$, then $m_{ij} = -m_{ij}$ for all $j, (j = 1, \ldots, n)$

From the above mentioned Lemma it is deduced that, if M is positive definite matrix; then $m_{ii} > 0$ for all $i, (i = 1, \ldots, n)$.

It is well known that a matrix M can be written as:

$$M = \tfrac{1}{2}(M + M^T) + \tfrac{1}{2}(M - M^T) = B + C,$$

where $B = \frac{1}{2}(M + M^T)$, $C = \frac{1}{2}(M - M^T)$, in which B is symmetric and C is skew-symmetric. Now consider

$$xMx^T = x(B + C)x^T = xBx^T + xCx^T = xBx^T,$$

since $xCx^T = 0$, therefore a matrix M is positive definite (positive semi-definite), if and only if its symmetric part is positive definite (positive semi-definite).

Definition: A square matrix M is said to be a co-positive matrix if and only if $x \geq 0$ implies that $xMx^T \geq 0$.

Definition: Co-positive plus matrices are co-positive matrices such that:

$$x \geq 0, \text{ and } xMx^T = 0, \text{ implies that } (M + M^T)x = 0.$$

It is obvious that the class of co-positive matrices includes the class of positive semi-definite matrices, and the class of strictly co-positive matrices includes the class of positive matrices. If M is co-positive plus and S is any skew-symmetric matrix of the same order, then (M + S) is co-positive plus. Block matrices

$$M = \begin{bmatrix} M_1 & -A^T \\ A & M_2 \end{bmatrix}$$

are co-positive plus if and only if M_1 and M_2 are co-positive plus.

Definition: A P-matrix is a matrix M having the property that; each of its principal minor is positive.

When M is a square matrix, say n × n and $I \subset \{1, 2, \ldots, n\}$, then M_{II} is called a principal submatrix of M, and its determinant is called a principal minor of M.

Definition: The n × n matrix M is said to be adequate if

(i) $\det(M_{II}) \geq 0$ for all $I \subset \{1, 2, \ldots, n\}$;

(ii) if $\det(M_{II}) = 0$ for some $I \subset \{1, \ldots, n\}$,
then the columns of $M_{.I}$ are linearly dependent;

(iii) if $\det(M_{II}) = 0$ for some $I \subset \{1, 2, \ldots, n\}$, then the rows of $M_{I.}$ are linearly dependent.

Theorem: if $M = NBN^T$, and B is positive definite then M is adequate.

Theorem: A non-singular matrix M is adequate if and only if, it has positive principal minors.

Definition: The diagonal Matrix E, of order n, is a sign-changing matrix, if $e_{ii} = \pm 1$ for each i = 1, 2, ..., n.

It therefore follows that if E is a sign-changing matrix E^2 is an identity matrix.
Theorem. If M is adequate, and E is a sign-changing matrix of the same order, then EME is an adequate matrix.

3.3. A Brief Review of Lemke's Algorithm and Principal Pivoting Algorithm

3.3.1 Lemke's Method [3.11]

Consider the Fundamental Problem (1), (2), (3) and let L denote the set of solutions, K the set of feasible solutions, and C the set of complementary feasible solutions. It is clear that $C \subseteq K \subseteq L$. (5) is a basic form which is unique once the basic set w^t is specified. A pivot on (5) yields an adjacent basic form; i.e. a basic form whose basic set differs only by a single variable. These basic sets are said to be adjacent. The basic point associated with (5) is the unique point $(\bar{w}^t, \bar{z}^t) = (q^t, 0)$ which has exactly n zeroes, since L is assumed to be nondegenerate. Any solution of (1) containing exactly n zero components is a basic solution. Two basic solutions are adjacent if their basic sets are adjacent.

A 'basic line' through the basic point associated with (4) is the set of solutions to

$$w^t = q^t + z_j^t M_{.j}^t \qquad (6)$$

for some fixed j. Points on a basic line have either n or (n-1) zero components. If in (6) some value of z_j^t makes a component of w^t zero, the corresponding solution has exactly n zero components and hence is a basic solution and in fact an adjacent solution to the basic solution $(\overline{w}^t,\overline{z}^t) = (q^t,0)$. (6) can be written as

$$\begin{pmatrix} w^t \\ z^t \end{pmatrix} = \begin{pmatrix} q^t \\ 0 \end{pmatrix} + \theta \begin{pmatrix} M_{.j}^t \\ e_j \end{pmatrix} \tag{7}$$

and permuting variable to their original order (7) becomes

$$\begin{pmatrix} w \\ z \end{pmatrix} = \begin{pmatrix} \overline{w} \\ \overline{z} \end{pmatrix} + \theta \begin{pmatrix} \overline{v} \\ \overline{u} \end{pmatrix} .$$

In order that the resulting solution should satisfy (5) for all θ, the following relation must hold

$$\overline{v} = M\overline{u} , \tag{8}$$

and $(\overline{v},\overline{u})$ has at least (n-1) zero values.

If $q^t > 0$ in (5), then (5) is a basic form having a basic feasible point. If in one pivot step it is possible to move from one basic feasible point (where $z^t = 0$) to an adjacent basic feasible point where $z_j^t = \overline{z}_j^t$ then the solutions to (6) are feasible on the interval $0 \le z_j^t \le \overline{z}_j^t$, and this set of solutions forms a bounded edge of K, such a pivot step is called a 'feasible pivot'. The end points of this interval are adjacent extreme points. If no feasible pivot is possible from a feasible basic form (5) for z_j^t, then $-M_{.j}^t \le 0$ and the set of solutions to (6) for $z_j^t > 0$ forms an unbounded edge of K.

A 'feasible pivot algorithm' is a succession of 'feasible pivots', which defines an adjacent extreme point path in K. A'proper pivot algorithm'is a pivot algorithm for which no basic set appears twice and hence must terminate in a finite number of pivots; the corresponding basic forms are called 'proper feasible forms'.

Complementary pivot schemes

Two of the three possible pivotal schemes are considered under this heading.

Scheme I Let z_0 be a scalar variable, and $e' > 0$ a column with positive components, and let L' be the set of solutions to

$$w = q + z_0 e' + Mz = q + A\underline{z} , \tag{9}$$

where $A = (e',M)$, and $\underline{z} = \begin{pmatrix} z_0 \\ z \end{pmatrix}$. Therefore, nonbasic sets in (9) have (n+1) components.

Let K' be the set of feasible solutions to (9) and C_0 be the subset of K', such that if $(w,\underline{z}) \in C_0$, then

$$w^T z = 0 .$$

The algorithm creates a succession of a proper feasible basic forms contained in C_0, whose basic points consequently satisfy (3).

If $q > 0$, then the solution to the Fundamental Problem is trivial, therefore assume that some components of q are negative.

Consider the problem (9) on the first pivot z_0 is increased until for the first time $w = q + z_0 e' \geq 0$, and

$$w_r = \min\{q_i + z_0 e'_i , \quad i = 1, \ldots, n\}, \tag{9a}$$

becomes zero. The first pivot is defined by the pivot pair

$$(w_r, z_0) , \tag{10}$$

this leads to the basic form

$$w^t = q^t + A^t \underline{z}^t \quad q^t > 0, \tag{11}$$

for $t = 1$, a single nonbasic complementary pair (w_r, z_r), and the basic feasible points satisfies (3). If z_r the complement of w_r is increased in (11), the condition (3) continues to be satisfied. If a pivot step which makes z_r basic can be made this becomes the second step and leads to the feasible form (11), for t = 2. If such a pivot step cannot be made, the sequence is terminated. In general suppose for $t \geq 1$ pivot steps have led to the feasible form (11), and suppose that (3) is satisfied for all the basic feasible points generated. If z_0 is nonbasic, a complementary solution has been found and the sequence terminates. If z_0 is still basic, suppose that the variable that has become nonbasic on the t^{th} pivot is one of the complementary pair (w_s, z_s), further condition (3) holds, therefore both components of this pair are nonbasic. The complement of the variable which has become nonbasic needs to be increased. Either a unique $(t+1)^{st}$ pivot step is thus specified, or the sequence is terminated. This completes a description of the scheme I.

It can be easily shown that the scheme generates a sequence of proper basic feasible solutions, that is no basic set occurs twice.

Scheme II. In this case it is assumed that M has a positive column, and as before some components of q are negative. For convenience, let the first column of M be positive. Then increasing z_1 defines a unique first pivot determined by the pivot pair (w_r, z_1) for some r, leading to the basic form:

$$w^t = q^t + M^t z^t, \quad q^t > 0, \tag{12}$$

where t = 1. This has a basic solution in which the relation

$$\sum_{i=1}^{n} w_i z_i = w_1 z_1 , \tag{13}$$

holds. Now let C_1 be the set of points of K satisfying (13) and known as 'almost complementary points'.

Entirely analogous to scheme I, scheme II involves, pivot steps in which condition (13) is satisfied. This defines a proper sequence of pivots. In a way similar to scheme I it can be shown [3.11] that, the sequence terminates either in a complementary solution or in an unbounded edge distinct from E_0, where E_0 is an unbounded edge generated by increasing z_0 in scheme I and z_1 in scheme II. For a a discussion of third possible scheme see [3.11].

Theorem. Let M be co-positive plus, then Lemke's method terminates either in a complementary basic feasible solution or leading to the conclusion that for the given q no feasible solution exists.

Theorem. If M is strictly copositive, Lemke's method terminates in a complementary feasible solution for any q.

Theorem. If M is a P-matrix, Lemke's method terminates in a complementary feasible solution for any q.

B.C. Eaves in [3.9] has shown that Lemke's method processes (*) linear complementarity problems for $M \in \mathcal{L}$ where $\mathcal{L}$ is a class of matrices which properly includes

(i) co-positive plus,
(ii) adequate matrices,
(iii) bimatrix game matrices.

He also has shown that

1) If $M \in \mathcal{L}$, and the system $w = Mz + q$, $w,z \geq 0$ is feasible and nondegenerate, then the corresponding linear complementarity problem has an odd number of solutions besides; if $M \in \mathcal{L}$ and $q > 0$ then the solution is unique.

2) If for some M and every $q > 0$, the linear complementarity problem has a unique solution, then $M \in \mathcal{L}$ and the problem with M and every q has a solution.

3) If M has non-negative principal minors, and if the linear complementarity problem with M and q has a non-degenerate complementary solution, then the solution is unique.

4) If $z^TMz + z^Tq$ is bounded from below on the set $z \geq 0$, then Lemke's method leads to a solution to the linear complementarity problem with M and q. If, in addition, the problem is non-degenerate, then it has an odd number of solutions.

(*) solves or shows no solution exists.

5) By using Lemke's method it is possible to find the saddle point for general quadratic programming or to demonstrate that the objective function is bound from below in the feasible region.
6) If a quadratic program has an optimal solution and if a certain nondegeneracy condition holds, then a quadratic program has an odd number of saddle points.

K.G. Murty in [3.12] has shown that, the number of solutions to the linear complementarity problem is finite for all $q \in R^n$ if and only if all the principal minors of M are non-zero. The necessary and sufficient condition for this solution to be unique for each $q \in R^n$ is that all the principal minors of M are strictly positive. When $M \geq 0$, there is at least one complementary feasible solution for each $q \in R^n$ if and only if all the diagonal elements of M are strictly positive, and in this case, the number of these solutions is an odd number whenever q is nondegenerate. If all the principal minors of M are non-zero, then the number of complementary feasible solutions has the same parity (*) for all $q \in R^n$ which are nondegenerate. Also if the number of complementary feasible solutions is constant for each $q \in R^n$, then the constant is equal to one and M is a P-matrix.

3.2 Principal Pivoting Method (**) [3.5]

To describe the method it is first necessary to introduce the concept of an almost complementary path and that of a blocking variable.

(*) If r is any integer, its parity is said to be odd if r is an odd integer or even if r is an even integer. A set of integers is said to be of constant parity if all the numbers in the set have the same parity.

(**) This method is applicable to matrices, M, that have positive principal minors, and after modification to positive semi-definite matrices.

The former is defined as any sequence of solutions through 'almost complementary points' (see (13) page 44). In a tableau a basic variable is said to be a blocking variable for a nonbasic variable which is being increased to a positive value and the former, i.e. the blocking variable, happens to be the first variable to become zero.

In principal pivoting only variables of the original problem are used, but these can take on initially negative as well as non-negative values.

A major cycle of the algorithm is initiated with the complementary basic solution $(w,z) = (q,0)$. If $q \geq 0$ the procedure is immediately terminated. If $q \ngeq 0$, it can be assumed that $w_1 = q_1 < 0$. An almost complementary path is generated by increasing z_1, the complement of the selected negative basic variable.

For points along the path $w_i z_i = 0$, for $i \neq 1$.

Step I. Increase z_1, until it is blocked by a positive basic variable decreasing to zero or by the negative w_1 increasing to zero.

Step II. Make the blocking variable nonbasic by pivoting its complement into the basic set. The major cycle is terminated if w_1 drops out of the basic set of variables, otherwise return to step I.

It can be shown [3.11] that during a major cycle w_1 increases to zero. At this point, a new complementary basic solution is obtained. However, the number of basic variables with negative value is at least one less than at the beginning of the major cycle. Since there are at most n negative basic variables, no more than n major cycles are required to obtain a complementary feasible solution.

3.3.3 Some other Methods which are Equivalent to Lemke's Method

MacCammon's Parameteric Method [3.15]

In Lemke's method, z_0 is introduced as a new variable. In the complementary terminal solutions (original and final) it is an independent variable i.e., nonbasic, which in the intermediate tableau it is a dependent basic variable. Associated with z_0 is the vector e'. This column is associated with a parameter which in Lemke's method determines the path of the solution. In this method z_0 is replaced by a scalar parameter θ. Consider the system

$$w = q + \theta e' + MZ , \tag{14}$$

where $e_i' \geq 0$ if $q_i > 0$ and $e_i > 0$ if $q_i < 0$ for $i = 1, \ldots, n$. A pivot algorithm is now described which is dependent upon the parameter θ.

Let $\overline{\theta}^0 = \min \{\theta | q + \theta e' \geq 0, \theta \geq 0\}$. If $\overline{\theta}^0 = 0$, then $q > 0$ and the basic point $w = q + \overline{\theta}^{\cdot} e$ associated with basic form (14) provides a solution to the Fundamental Problem. If $\overline{\theta}^0 > 0$, then $q_r + \overline{\theta}^0 e_r' = 0$ for some r, $1 \leq r \leq n$. Assuming nondegeneracy, this value of r is unique. If $m_{rr} \neq 0$, then w_r is made nonbasic and z_r becomes basic by one pivotal transformation in which $-m_{rr}$ is the pivotal element. If $m_{rr} = 0$, the first pivot is given by the pair (w_s, z_r), where $-m_{sr} > 0$ and

$$(q_s + \overline{\theta}^0 e_s')/(-m_{sr}) = \min\{(q_i + \overline{\theta}^0 e_i')/(-m_{ir}) | -m_{ir} > 0 \ , \ 1 \leq i \leq n\} \ ;$$

however, for $m_{rr} = 0$ if $-m_{ir} \leq 0$ for all i, then the algorithm terminates.

In the general step consider a basic form

$$w^t = q^t + \theta e'^t + M^t z^t \,, \qquad (15)$$

then either (15) satisfies complementary condition, or it is not complementary. In the former case, consider the set $\{\theta \mid q^t + \theta e^t \geq 0 \,,\ \theta \geq 0\}$, and if this set is empty, then terminate the procedure. If this set is non-empty, then θ has a minimum and a maximum in this set, call these $I(t)$ and $S(t)$, respectively, and it follows that $0 \leq I(t) \leq S(t)$. If $I(t) = 0$ again terminate this procedure. The basic point corresponding to (15) then provides a solution to the fundamental problem. Assuming that this is not the case, then if $S(t) = \infty$, the variable which leaves the basis is the unique basic variable which is zero in (15) and $\theta = \bar{\theta}^t = I(t)$, and the variable which is its complement is made to enter the basis. In either case suppose w_r^t is the variable which leaves the basis, and z_s^t is the variable which enters the basis. If $(-m_{rs}^t) \neq 0$, it is the pivot element and the pair (w_r^t, z_s^t) specifies the exchange. If $(-m_{rs}^t) = 0$ and $(-M_{.s}^t) \leq 0$, terminate the procedure. If $(-m_{rs}^t) = 0$ and $(-m_{is}^t) > 0$ for some $1 \leq i \leq n$, the pivot element is $(-m_{ps}^t)$, where $(-m_{ps}^t) > 0$ and

$$(q_p^t + \bar{\theta}^t e_p'^t)/(-m_p^t) = \min\{q_i + \bar{\theta}^t e_i'^t)/(-m_{is}^t) \mid -m_{is}^t > 0,\ 1 \leq i \leq n\} \,.$$

This completes a brief description of the algorithm.

Complementary Variant of Lemke's Method [3.16]

In this algorithm proposed by Van de Panne [3.16] z_0 (the artificial variable is introduced in Lemke's method) and certain nonbasic variables are varied as parameters. This results in a method which is equivalent to Lemke's method. This method has complementary tableaux and uses principal (single or block) pivots and therefore is called complementary variant of Lemke's method. In contrast to Lemke's method, this method explains, in certain sense, the variation of z_0 and the other variables. Furthermore, since complementary tableaux are used throughout, a better insight is gained by this method and the various possibilities of termination.

The main advantage of this variant is thought to be in infeasibility test, which may be performed on each row. A particular instance of such a test is shown to be the 'plus' condition of the co-positive plus matrices.

3.4 Branching Procedure For Solving the Linear Complementarity Problem

It has been pointed out earlier principal pivoting algorithm can be applied to solve the Fundamental Problem only if M is positive definite, or more generally when M is a P-matrix. Further a modified form of this method can be used to obtain a solution to the Fundamental Problem if the system

$$w = q + Mz ,$$

$$z \geq 0, w \geq 0 ,$$

has a solution and M is positive semi-definite.

If Lemke's method is applied to solve the Fundamental Problem, and the procedure terminates in an unbounded ray and M does not belong to the class of £ matrices, in this case the method does not provide any information concerning the solvability of the problem. For an illustration consider the problem, stated below:

Example 1.

$$\begin{cases} w_1 = 10 - 2z_1 + 3z_2 - z_3 \\ w_2 = -1 + z_1 - 2z_2 + z_3 \\ w_3 = \ 3 - z_1 + 2z_2 + 3z_3 \\ w_1, w_2, w_3, z_1, z_2, z_3 \geq 0 \\ w_i z_i = 0 \quad (i=1,2,3) \end{cases}$$

Note that in this case the matrix

$$M = \begin{pmatrix} -2 & 3 & -1 \\ 1 & -2 & 1 \\ -1 & 2 & 3 \end{pmatrix} ,$$

is not a P-matrix, therefore principal pivoting method cannot be applied to solve this problem.

Now Lemke's method is applied as follows:

$$\begin{cases} w_1 = 10 + z_0 - 2z_1 + 3z_2 - z_3 \\ w_2 = -1 + z_0 + z_1 - 2z_2 + z_3 \\ w_3 = \ 3 + z_0 - z_1 + 2z_2 - 3z_3 \end{cases}$$

In tableau representation this can be set out in Tableau 4-1, in this tableau z_0 is set to 1 following the ratio test of (9a).

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$
w_1	11	-1	2	-3	1
w_2	0	-1	-1	2	-1
w_3	4	-1	1	-2	-3

Tableau 4-1

	1	$-w_2$	$-z_1$	$-z_2$	$-z_3$
w_1	11	-1	3	-5	2
z_0	1	-1	1	-2	1
w_3	4	-1	2	-4	-2

Tableau 4-2

In Tableau 4-2 z_2 is the complement of variable w_2, and cannot be made a basic variable taking non-negative value. Therefore the procedure terminates in an unbounded ray. This means neither Lemke's method nor principal pivoting method can be used to establish the solvability of the problem.

It is shown later on that this problem has the following feasible solutions.

$$z' = \begin{pmatrix} 17 \\ 8 \\ 0 \end{pmatrix} \quad w' = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \quad \text{and} \quad z'' = \begin{pmatrix} \frac{33}{7} \\ 0 \\ \frac{4}{7} \end{pmatrix} \quad w'' = \begin{pmatrix} 0 \\ \frac{30}{7} \\ 0 \end{pmatrix}$$

In Lemke's method the components of the artificial vector do not necessarily have to be unity. Therefore by suitable choice of these components different paths of complementary solution may be followed, this idea is illustrated later on in this section by means of two examples. One may then naturally ask if by following such possible paths a complementary feasible solution may be obtained to the problem if it exists. By means of the following examples it is shown that this assumption is invalid in the general case.

Consider the problem in the following example

Example 2

$$\begin{cases} w_1 = 2 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 + 10z_1 + z_2 - z_3 + z_4 \\ w_3 = 3 - z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = -6 + 20z_1 + 3z_2 - z_3 - 3z_4 \\ w_i \geq 0,\ z_i \geq 0 \quad (i=1, \ldots, 4) \\ w_i z_i = 0 \quad i = 1, \ldots, 4 \end{cases}$$

By introducing z_0 as an artificial variable where $e'^T = (1,1,1,1)$ the problem becomes:

$$\begin{cases} w_1 = 2 + z_0 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 + z_0 + 10z_1 + z_2 - z_3 + z_4 \\ w_3 = 3 + z_0 - z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = -6 + z_0 + 20z_1 + 3z_2 - z_3 - 3z_4 \\ w_i \geq 0,\ z_i \geq 0, \text{ and } z_i w_i = 0 \quad i=1,2,3,4 \end{cases}$$

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	8	-1	-2	1	3	-4
w_2	2	-1	-10	-1	1	-1
w_3	9	-1	1	2	-1	2
w_4	0	-1	-20	-3	1	3

Tableau 4-3

In tableau 4-3 z_0 is set to 6 to make $q_3 + e'_3 z_0 = 0$ (see 9a).

Tableau 4-4

		$-w_4$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	8	-1				-7
w_2	2	-1				-4
w_3	9	-1	not	up	dated	-1
z_0	6	-1				-3

In tableau 4-4, z_4 cannot be made basic variable, therefore the procedure terminates in an unbounded ray.

Now choose $e'^T = (1,\frac{1}{2},1,2)$, the correspondi-ng representation tableau is shown in Tableau 4-5:

Tableau 4-5

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	10	-1	-2	1	3	-4
w_2	0	$-\frac{1}{2}$	-10	-1	1	-1
w_3	11	-1	1	2	-1	2
w_4	10	-2	-20	-3	1	3

The value of z_0 in tableau 4-5 is 8

Tableau 4-6

	1	$-w_2$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	10	-2	18	3	1	-2
z_0	8	-2	20	2	-2	2
w_3	11	-2	21	4	-3	4
w_4	10	-4	20	1	-3	7

z_2 is the complement of the variable w_2, which is made basic in this step. The procedure is then followed until Tableau 4-9.

	1	$-w_2$	$-z_1$	$-w_3$	$-z_3$	$-z_4$
w_1	$\frac{7}{4}$	$-\frac{1}{2}$	$\frac{9}{4}$	$-\frac{3}{4}$	$\frac{13}{4}$	-5
z_0	$\frac{10}{4}$	-1	$\frac{38}{4}$	$-\frac{2}{4}$	$-\frac{1}{2}$	0
z_2	$\frac{11}{4}$	$-\frac{1}{2}$	$\frac{21}{4}$	$\frac{1}{4}$	$-\frac{3}{4}$	1
z_4	$\frac{29}{4}$	$-\frac{7}{2}$	$\frac{59}{4}$	$-\frac{1}{4}$	$-\frac{9}{4}$	6

Tableau 4-7

	1	$-w_2$	$-z_1$	$-w_3$	$-w_1$	$-z_4$
z_3	$\frac{7}{13}$	not	$\frac{9}{13}$	not	$\frac{13}{4}$	not
z_0	$\frac{36}{13}$	up	$\frac{128}{13}$	up	$-\frac{1}{2}$	up
z_2	$\frac{411}{13}$	dated	$\frac{75}{13}$	dated	$-\frac{3}{4}$	dated
w_4	$\frac{110}{13}$		$\frac{212}{13}$		$-\frac{9}{4}$	

Tableau 4-8

	1	$-w_2$	$-z_0$	$-w_3$	$-w_1$	$-z_4$
z_3	$\frac{11}{32}$					
z_1	$\frac{9}{32}$	not	up	dated		
z_2	$\frac{49}{32}$					
w_4	$\frac{124}{32}$					

Tableau 4-9

So the solution is

$$w = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{124}{32} \end{pmatrix} \qquad z = \begin{pmatrix} \frac{9}{32} \\ \frac{49}{32} \\ \frac{11}{32} \\ 0 \end{pmatrix} .$$

In another Example (see below) it is shown that all the possible paths lead to unbounded rays.

Example 3

$$\begin{cases} w_1 = 2 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 - z_1 + 2z_2 - z_3 + z_4 \\ w_3 = 3 + 2z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = -6 + 4z_1 + 3z_2 - z_3 - 3z_4 \\ w_i \geq 0,\ z_i \geq 0,\ w_i z_i = 0 \text{ for all } (i = 1, \ldots, 4). \end{cases}$$

First z_0 is introduced with corresponding column $e'^T = (1,1,1,1)$, so the problem can be written as:

$$\begin{cases} w_1 = 2 + z_0 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 + z_0 - z_1 + 2z_2 - z_3 + z_4 \\ w_3 = 3 + z_0 + 2z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = -6 + z_0 + 4z_1 + 3z_2 - z_3 - 3z_4 \\ w_i \geq 0,\ z_i \geq 0,\ w_i z_i = 0 \ (i = 1, \ldots, 4) \end{cases}$$

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	8	-1	-2	1	3	-4
w_2	2	-1	1	-2	1	-1
w_3	9	-1	-2	2	-1	2
w_4	0	-1	-4	-3	1	3

Tableau 4-10

In tableau (4-10) z_0 is set to 6.

		$-w_4$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	8	-1	0	4	2	-7
w_2	2	-1	3	2	0	-4
w_3	9	-1	0	5	-2	-1
z_0	6	-1	4	3	-1	-3

Tableau 4-11

z_4 is the complement of the variable w_4, and cannot be made basic variable, therefore the procedure terminates in unbounded ray.

Now if the column associated with the artificial variable is introduced as $e'^T = (1,\frac{1}{2},1,2)$, and Lemke's method is applied, the following tableaux are obtained

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	10	-1	-2	1	3	-4
w_2	0	$-\frac{1}{2}$	1	-1	1	-1
w_3	11	-1	-2	2	-1	2
w_4	10	-2	-4	-3	1	3

Tableau 4-12

In tableau 4-12 the value of z_0 is 8.

Tableau 4-13

	1	$-w_2$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	10	-2	-4	3	1	-2
z_0	8	-2	-2	2	-2	2
w_3	11	-2	-4	4	-3	4
w_4	10	-4	-8	1	-3	7

Tableau 4-14

	1	$-w_2$	$-z_1$	$-w_3$	$-z_3$	$-z_4$
w_1			-1	$-\frac{3}{4}$	$\frac{13}{4}$	not
z_0	not	up	0	$-\frac{2}{4}$	$-\frac{2}{4}$	up
z_2	dat	ed	-1	$\frac{1}{4}$	$-\frac{3}{4}$	dated
w_4			-7	$-\frac{1}{4}$	$-\frac{9}{4}$	

Tableau 4-15

	1	$-w_2$	$-z_1$	$-w_3$	$-w_1$	$-z_4$
z_3			$-\frac{4}{13}$			$\frac{4}{13}$
z_0	not	up	$-\frac{2}{13}$	not	up	$\frac{2}{13}$
z_2	dat	ed	$-\frac{16}{13}$	d	ated	$\frac{3}{13}$
w_4			$-\frac{100}{13}$			$\frac{9}{13}$

z_1, the complement of the variable w_1 cannot be made a basic variable. Again the procedure terminates in unbounded ray. It can be seen that this problem has a complementary feasible solution and it is

$$w = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{40}{7} \end{pmatrix} \qquad z = \begin{pmatrix} 1 \\ \frac{19}{7} \\ \frac{3}{7} \\ 0 \end{pmatrix} .$$

Since these two well-known methods and their variants may fail to provide a solution to the linear complementarity problem in the general case an alternative algorithm is suggested. This algorithm is based on an algorithm proposed by the author for finding all the vertices of a convex polyhedron (see 3.11 G.R. Jahanshahlou and G. Mitra). This algorithm generates only a small subset of all the vertices of the problem defined by (1), (2) and further this subset contains all the solutions of the linear complementarity problem. The generality of the procedure is attractive in as much as it makes no assumption about the problem matrix M.

Before stating the algorithm the following terms are defined.

"Kilter number", K, is the number of complementary pairs of variables which are in the basis.

A variable is said to be "starred" if in all the subsequent tableaux it is forced to remain non-basic, similarly a variable and its associated row is said to be "flagged" if in all the subsequent tableaux the variable is forced to stay in the basis.

The steps of the algorithm may be stated as follows:

Step 1. Apply the phase I of the simplex method to the system

$$w = q + M \ , \ z \geq 0, \ w \geq 0 \ , \qquad (16)$$

to find a basic feasible solution. If there is no basic feasible solution to (16), then there is no solution to the Fundamental Problem, and go to step 5, otherwise number the tableau associated with the basic feasible solution Tableau -0, set N = 0, L = 0, K_L = K (K_L is the kilter number in the current tableau).

Step 2. Pick tableau N from the stack of the tableaux, go to auxiliary sequence. If a pivotal transformation is carried out set L = L+1, number the new tableau as tableau L, and add it to the stack of the tableaux and go to step 3. If the auxiliary sequence ends in the terminal step, i.e., step f, go to step 4.

Step 3. Pick tableau L, out of the stack, go to auxiliary sequence. If a pivotal transformation has taken place put L = L+1, number the new tableau as tableau L, and add it to the stack of the tableaux, go to step 3. If the auxiliary sequence ends in the terminal step, i.e. step f, go to step 2.

Step 4. Set N = N+1, if N > L go to step 5. If N ≤ L and the tableau N is marked, go to step 4, and if the tableau N is not marked go to step 2.

Step 5. Tree search is completed.

Auxiliary sequence

In this sequence if possible a pivotal transformation is carried out on the given tableau.
The first three steps are for column choice.

Initial step. If the kilter number of the tableau is zero goto step a, otherwise goto step b.

Step a. Out of the "nonstarred" nonbasic variables choose a column q with variable z_r or w_r which admits a row i ($i \notin F$ where F is the set of row indices which are flagged) with a positive coefficient, i.e. $-m_{iq} > 0$. Go to step c, otherwise no pivotal transformation can be carried out and go to terminal step f.

Step b. If in the given tableau there exists a pair of nonbasic complementary variables, which are starred no column should be chosen and go to terminal step f. If this is not the case define two sets of column indices

$$Q_1 = \{\ell \mid \ell \text{ with one unstarred variable } z_r \text{ or } w_r \text{ and } z_r, w_r \text{ both nonbasic}\}$$

$$Q_2 = \{\ell \mid \ell \text{ with unstarred variables } z_r, w_r \text{ and both nonbasic}\}$$

i) choose $q \in Q_1$ such that the associated variable z_r or w_r can be made basic, i.e. it admits a row i, $i \notin F$ and $-m_{iq} > 0$, and go to step c. else,

ii) choose $q \in Q_2$, such that the associated variable z_r or w_r can be made basic as in (i) above. If such a q does not exist go to step a.

Step c. (Row choice). Out of the rows not "flagged" find a row index p such that

$$\beta_p / -m_{pq} = \min_{i \notin F} \left\{ \beta_i / (-m_{iq}) \;\middle|\; -m_{iq} > 0 \right\}$$

Step d. Pivotal Transformation and flagging and starring. There may be four possible cases in each of which pivotal transformation is carried out on the element $-m_{pq} > 0$.

Case (i)

Let z_r be the basic variable in row $p, p \notin F$ and w_r be the nonbasic variable in column q. [See Tableau a-1 , Tableau a-2 and Tableau a-3]. In this case in the original tableau z_r, and row p are "flagged" and w_r is starred w_r^*, tableau a-2, and in the new tableau L, z_r is starred z_r^* and w_r is flagged $\overline{w}_r$ and the row p is also flagged, tableau a-3.

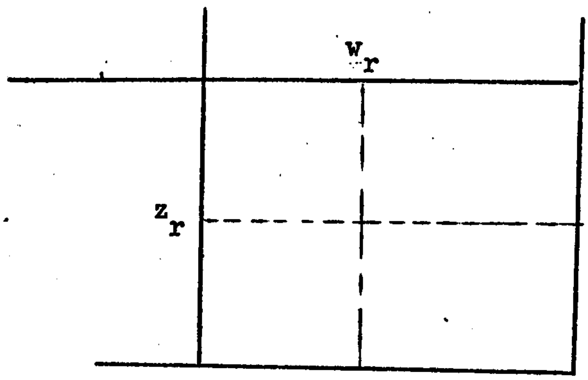


Tableau a-1

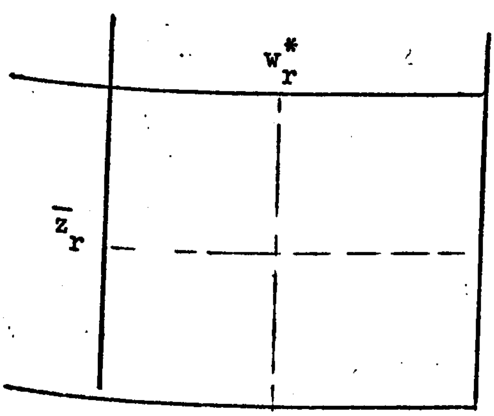


Tableau a-2

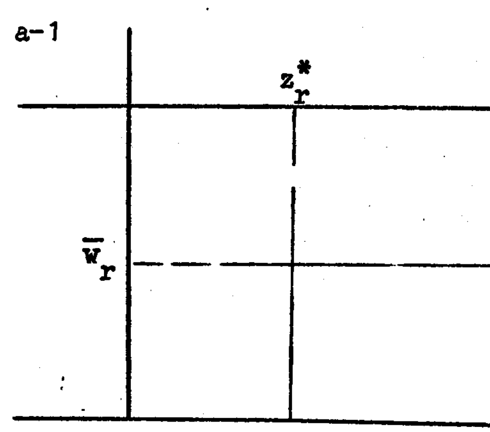


Tableau a-3

Case (ii)

In this case let w_r be the nonbasic variable which is in column q (z_r is also nonbasic) and z_s is the variable in row p [See Tableau a-4] then in the original tableau w_r is starred w_r^*, tableau a-5 ,and in the new tableau L, w_r and row p are "flagged"; and z_r is starred if it has not been already "starred" (tableau a-6).

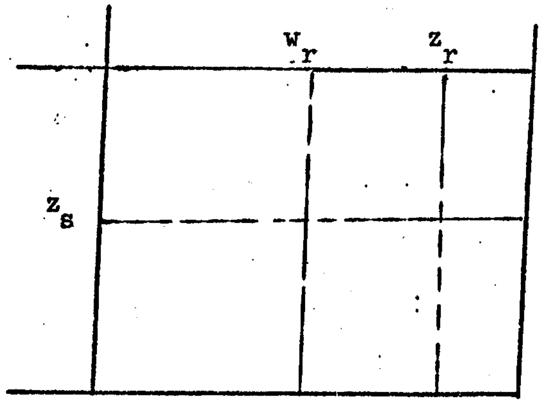


Tableau a-4

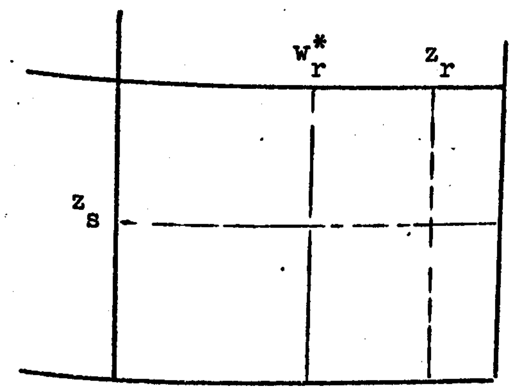


Tableau a-5

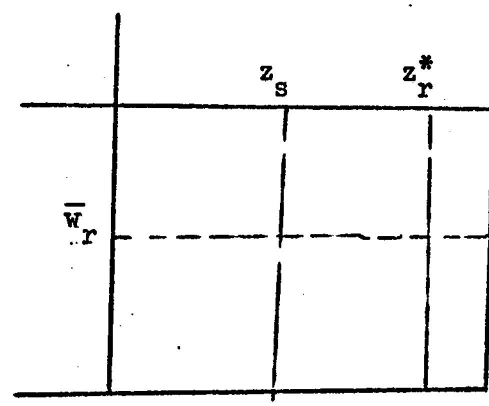


Tableau a-6

Case (iii)

In this case let w_r in column q be the nonbasic variable, whereas z_r is basic. Let z_s be the variable in row p (tableau a-7). Then in the

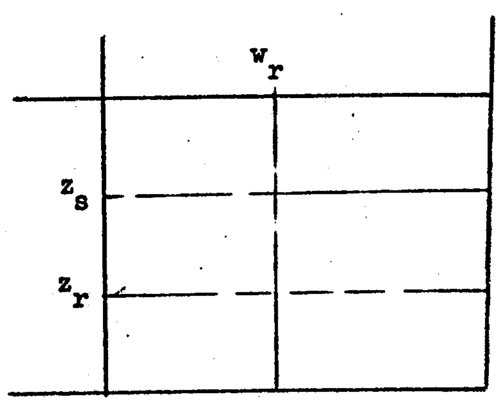


Tableau a-7

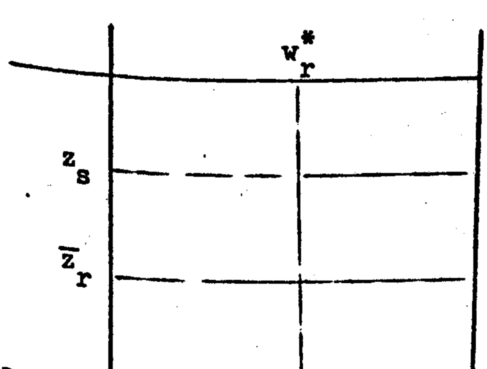


Tableau a-8

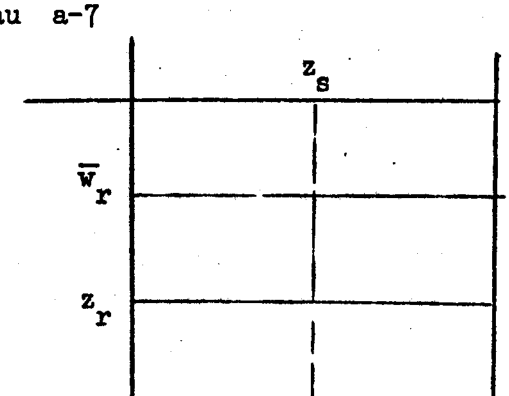


Tableau a-9

original Tableau w_r is starred and z_r and the associated row are "flagged" (tableau a-8), and in the new tableau L, w_r and row p, are flagged (Tableau a-9).

Case (iv)

In this case let w_r in column q be the nonbasic variable, whereas z_s in row p and w_s are both basic, further w_s and the associated row are "flagged" [see Tableau (a-10)].

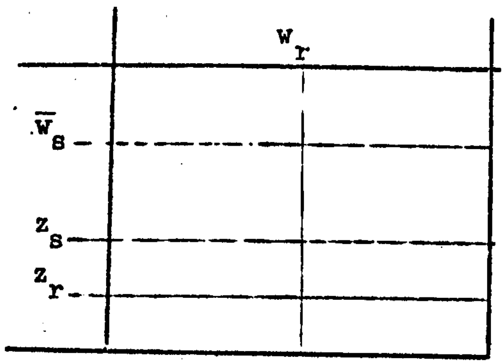


Tableau (a-10)

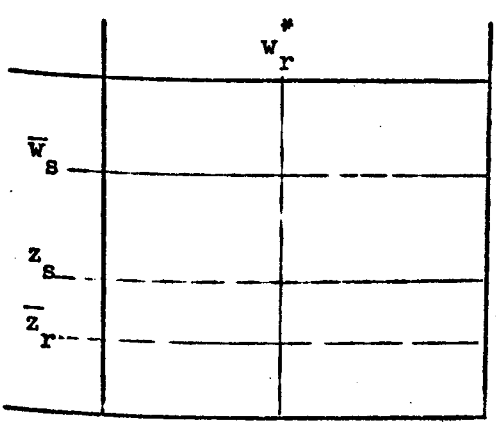


Tableau (a-11)

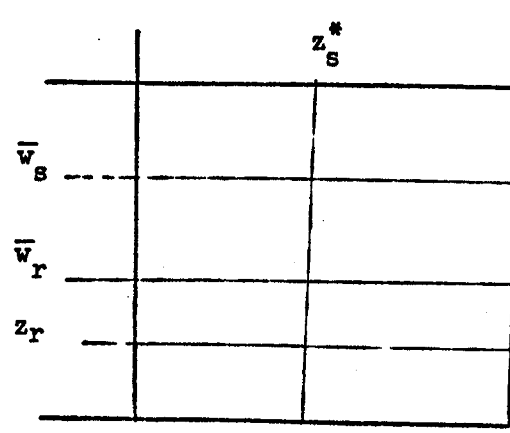


Tableau (a-12)

Then in the original Tableau z_r and the associated row are flagged and w_r is starred w_r^* (tableau a-11). In the new tableau L, z_s is starred z_s^* and w_r and row p are flagged, (tableau a-12).

Step e. If in a tableau its kilter number is zero and the values of the basic variables are non-negative, this tableau represents a feasible complementary solution. Return to the calling step.

Step f. (Terminal) No pivotal transformation is carried out, mark the tableau and return to the calling step.

A set description is now introduced to explain the applicability of the algorithm and the theorem which follows. Let

S_P be the set of all the possible bases of (1) and (2),

S_T be the set of all the bases generated by the algorithm in [3.10],

S_F be the set of all feasible bases (i.e. vertices) of (1) and (2),

S_C be the set of all complementary bases of (1) and (2),

S_{CF} be the set of all complementary feasible bases of (1) and (2).

These are illustrated in Fig(3)

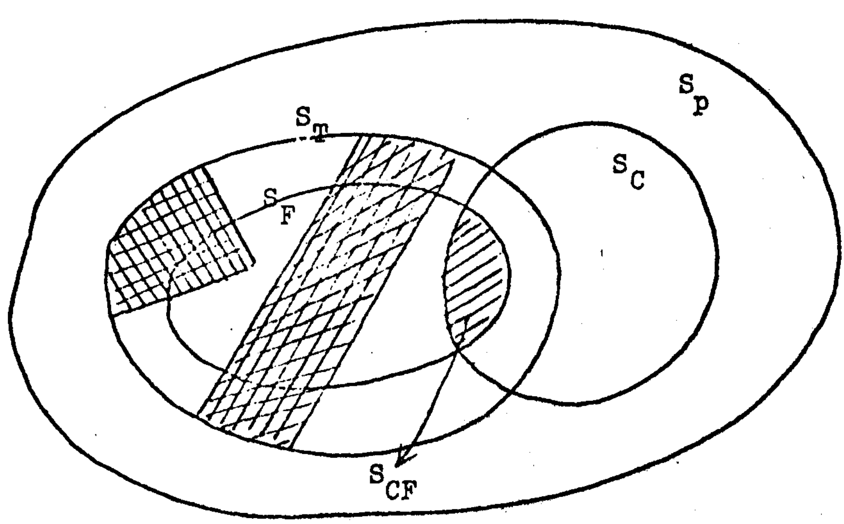


Figure 3

The double shaded areas represent those subsets of S_T which are not generated by the present algorithm. Later on in the theorem it is shown that these must be subsets of the set $(S_T - S_C)$.

Theorem The above mentioned algorithm generates all the complementary feasible bases of the set defined by (1) and (2) provided such vertices exist i.e. the set $(S_F \cap S_C)$ is non empty.

Proof: To prove this theorem, it is first noted that the algorithm in [3.10] generates the set S_T, which contains the set S_F i.e. all the feasible bases of (1) and (2). The modification introduced in the present algorithm leaves out certain subsets of S_T. It is now shown that these subsets are not contained in S_C. The possible cases are considered in turn:

Case a. In the tableau (a-13) w_q is a potential variable to become a basic variable and to generate some bases of (1) and (2). The kilter number of this and all the subsequent

	z_r^*	w_q	w_r^*

Tableau a-13.

tableaux which follow are at least one, because z_r and w_r are forced to remain nonbasic. Therefore no complementary vertices are lost, if this tableau is marked, and the associated branch is terminated in the tree search.

Case b. Consider the possible cases mentioned in step d of the auxiliary sequence. Starring the variables and flagging the rows and their corresponding variables exclude some possible enumerations. In the following it is shown that by these actions no complementary vertices are lost. These are considered in turn:

In Case (i), z_r and row p are flagged; this is not done in the algorithm in [3.10]. If this variable and the row p are not flagged, it might become a nonbasic variable in a subsequent step, and as w_r is forced to remain nonbasic (w_r is a starred variable), therefore in all the subsequent tableaux which might be obtained from this tableau, the kilter number must be at least one. Similarly if z_r is not starred in the new tableau L, and if it could be pivoted into the basis, as w_r is forced to be the basic variable, so in all the subsequent tableaux obtained from this tableau, their kilter number must be one or more. Therefore no complementary vertices are lost in this case [see tableau (a-2) and tableau (a-3)].

By similar argument it can be shown that no complementary vertices are lost as a result of additional starring of nonbasic variables or flagging of basic variables and their associated rows in the other cases. Since all other possible bases which may be generated by the algorithm in [3.10] are considered the bases excluded by this algorithm belong to the set $(S_T - S_C)$. Therefore the set of bases generated by this algorithm contains the subset $S_C \cap S_F$ if this is nonempty.

Example 4.

Here the Example 1 is solved by the proposed algorithm. It has been shown that Lemke's method as well as the principal pivoting method failed to produce a CFS(*) solution to the problem.

The problem is restated here

$$\begin{cases} 2z_1 - 3z_2 + z_3 + w_1 = 10 \\ z_1 - 2z_2 + z_3 - w_2 = 1 \\ z_1 - 2z_2 - 3z_3 + w_3 = 3 \\ w_i \geq 0 \quad z_i \geq 0 \quad (i = 1,2,3) \end{cases} \tag{17}$$

(*) complementary feasible solution

By introducing an artificial variable corresponding to the second equation of the system (17), and applying the Phase I of the simplex method the tableau 4-16 containing the basic feasible solution is obtained. In this tableau kilter number is 1. The rest of the steps of the algorithm as related to this problem are illustrated beow

	1	$-z_2$	$-z_3$	$-w_2$
w_1	8	1	-1	2
z_1	1	-2	1	-1
w_3	2	0	-4	1

(pivot element is circled)

Tableau 4-16

	1	$-z_2^*$	$-z_3$	$-w_2$
w_1	8	1	-1	2
z_1	1	-2	1	-1
w_3	2	0	-4	1

Tableau 4-16a

	1	$-w_1$	$-z_3$	$-w_2^*$
$\bar{z}_2$	8	1	-1	2
z_1	17	2	-1	3
w_3	2	0	-4	1

Tableau 4-17

The new position of the original tableau 4-16 is shown in Tableau 4-16a . In this tableau z_2 is starred. In the tableaus 4-17 which has been obtained by pivotal transformation z_2 and row 1 are flagged and w_2 is starred, which contains a feasible complementary solution.

	1	$-w_1^*$	$-z_3$	$-w_2^*$
$\bar{z}_2$	8	1	-1	2
$\bar{z}_1$	17	2	-1	3
w	2	0	-4	1

Tableau 4-17a

	1	$-z_1^*$	$-z_3$	$-w_2^*$
$\bar{z}_2$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
$\bar{w}_1$	$\frac{17}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{3}{2}$
w	2	0	-4	1

Tableau 4-18

Tableau 4-17a is the new position of the tableau 4-17 in which z_1 and row 2 are flagged and w_1 is starred. By carrying out a pivotal transformation on the tableau 4-17 , tableau 4-18 is obtained, in which w_1 and row 2 are flagged and z_1 is starred.

Tableau 4-18 is marked, because no column can be chosen. Now tableau 4-16a is picked up, w_2 the complement of z_2, which is a starred variable is chosen to become a basic variable and a pivot step is carried out as shown below.

	1	$-z_2^*$	$-z_3$	$-w_2^*$
w_1	8	1	-1	2
z_1	1	-2	1	-1
w_3	2	0	-4	1

Tableau 4-16b

	1	$-z_2^*$	$-z_3$	$-w_3$
w_1	4	1	7	-2
z_1	3	-2	-3	1
$\bar{w}_2$	2	0	-4	1

Tableau 4-19

Tableau 4-16b is the new position of the tableau 4-16a in which w_2 is starred. Tableau 4-19 is obtained by carrying out a pivotal transformation from tableau 4-16a . In Tableau 4-19

w_2 and row 3 are flagged. It should be noted that all three tableaux 4-16, 4-16a and 4-16b are associated with N = 0, i.e. these three tableaux have been considered as one tableau, but for the purpose of illustration they have been considered separately.

Now pick tableau 4-19 and carry out a pivotal transformation on the pivot element 7. Having done this operation the following tableaux is obtained.

	1	$-z_2^*$	$-z_3^*$	$-w_3$
w_1	4	1	7	-2
z_1	3	-2	-3	1
$\bar{w}_2$	2	0	-4	1

Tableau 4-19a

	1	$-z_2^*$	$-w_1$	$-w_3$
$\bar{z}_3$	$\frac{4}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$-\frac{2}{7}$
z_1	$\frac{33}{7}$	$-\frac{11}{7}$	$\frac{3}{7}$	$\frac{1}{7}$
$\bar{w}_2$	$\frac{30}{7}$	$\frac{4}{7}$	$\frac{4}{7}$	$-\frac{1}{7}$

Tableau 4-20

In tableau 4-19a which is the new position of the tableau 4-19 z_3 is starred and in tableau 4-20 which is obtained from tableau 4-19 z_3 and row 1 are flagged, this tableau also contains a complementary feasible solution.

From tableau 4-20 the following are obtained:

	1	$-z_2^*$	$-w_1^*$	$-w_3$
$\bar{z}_3$	$\frac{4}{7}$	$\frac{1}{7}$	$\frac{1}{7}$	$-\frac{2}{7}$
$\bar{z}_2$	$\frac{33}{7}$	$-\frac{11}{7}$	$\frac{3}{7}$	$\frac{1}{7}$
$\bar{w}_2$	$\frac{30}{7}$	$\frac{4}{7}$	$\frac{4}{7}$	$-\frac{1}{7}$

Tableau 4-20a

	1	$-z_2^*$	$-z_1^*$	$-w_3$
$\bar{z}$	-1	$\frac{2}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$
$\bar{w}_1$	11	$-\frac{11}{3}$	$\frac{7}{3}$	$\frac{1}{3}$
$\bar{w}_2$	-2	$\frac{8}{3}$	$-\frac{4}{3}$	$-\frac{1}{3}$

Tableau 4-21

Tableau 4-20a is the new position of tableau 4-20. In this tableau z_1 and row 2 are flagged and w_1 is starred. In tableau 4-21 w_1 and row 2 are flagged and z_1 is starred.

No pivotal transformation can be carried out on tableau 4-21, therefore it is marked. Both w_2 and z_2 are starred in tableau 4-16b, so this tableau is also marked. No column can be chosen from tableaux 4-17a and 4-18, therefore they are also marked. The tableau 4-19a is picked up, and the following tableau obtained from this tableau:

	1	$-z_2^*$	$-z_3^*$	$-w_3^*$
w_1	4	1	7	-2
z_1	3	-2	-3	1
$\bar{w}_2$	2	0	-4	1

Tableau 4-19b

	1	$-z_2^*$	$-z_3^*$	$-z_1$
w_1	10	-3	1	2
$\bar{w}_3$	3	-2	-3	1
$\bar{w}_2$	-1	2	-1	-1

Tableau 4-22

In the new position of tableau 4-19a, i.e. in tableau 4-19b w_3 is starred, and in tableau 4-22 which is obtained from tableau 4-19a w_3 and row 2 are flagged. The tableaux 4-20a, 4-21 and 4-19b are marked, since no pivotal transformation can be carried out. The tableau 4-22 is chosen. In this tableau z_1 is chosen to pivot against w_1. Carrying out a pivot on the pivot element leads to 2 the following two tableaux.

	1	$-z_2^*$	$-z_3^*$	$-z_1^*$
$\bar{w}_1$	10	-3	1	2
$\bar{w}_3$	3	-2	-3	1
$\bar{w}_2$	-1	2	-1	-1

Tableau 4-22a

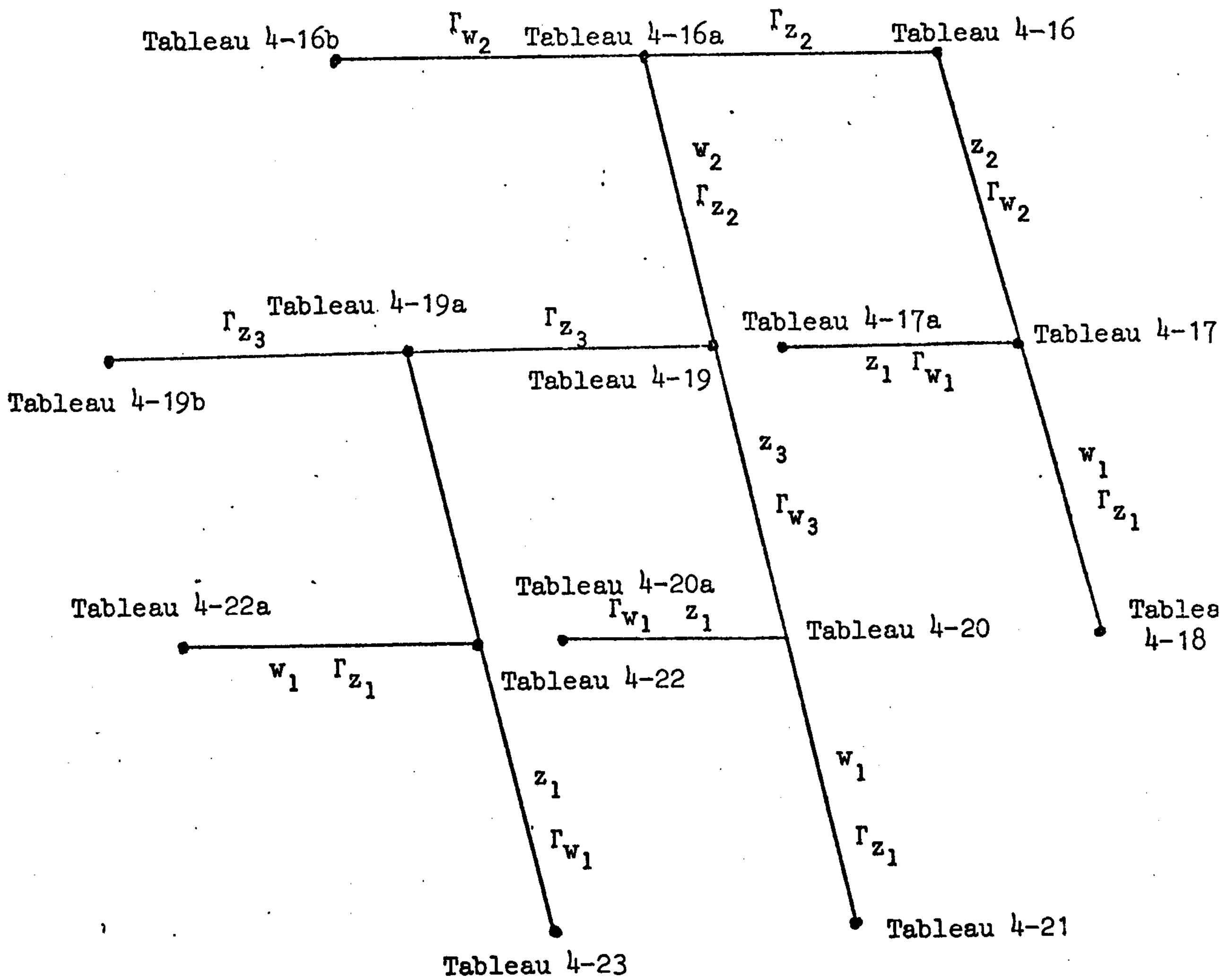
	1	$-z_2^*$	$-z_3^*$	$-w_1^*$
$\bar{z}_1$	5	$-\frac{3}{2}$	$\frac{1}{2}$	1
$\bar{w}_3$	-2	$-\frac{1}{2}$	$-\frac{7}{2}$	$-\frac{1}{2}$
$\bar{w}_2$	4	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$

Tableau 4-23

As no pivotal transformation can be carried out on the tableau 4-22a and 4-23 they are marked, so the search is complete. Tableau 4-24 shows a summary of the steps of the algorithm as related to this problem.

Iteration Number	The feasibility F = feasible N = not feasible	Complementarity C=complementary NC=not complementary	Tableaux generated	L	N
0	F	NC	4-16	0	0
1	F	C	4-16a , 4-17	1	0
2	N	C	4-17a , 4-18	2	0
3	F	NC	4-16b , 4-19	3	1
4	F	C	4-19a , 4-20	4	1
5	N	C	4-20a , 4-21	5	1
6	N	C	4-19b , 4-22	6	2
7	N	C	4-22a , 4-23	7	2
After iteration 7 N increases and L remains fixed until N=7 when the search is complete					

Tableau 4-24



Figure(1)

The tree developed by this method is shown in Fig(1), and the sequence of tableaux which are generated are set out in tableau 4-16 up to tableau 4-23.

In Fig(1) Γz_i, Γw_i indicates that the corresponding variable is not in the basis, and z_i or w_i indicate that the corresponding variable is in the basis.

3.5 Discussion and Some Remarks on the Computational Experience

Application of the phase I of the simplex method to the problem, leads to the conclusion that either there is a basic feasible solution to the problem or otherwise. If there is a basic feasible solution, then the branching starts from the node associated with this solution. It is interesting that in each iteration the size of the problem in the branch is reduced at least by one row and one column or two columns and one row. In the case of principal pivoting or the case of the step e in the auxiliary sequence one row and one column are flagged and starred, i.e. the size of the problem is reduced by one row and one column in each branch.

While studying Lemke's method another problem suggested itself, namely, what other vectors associated with the artificial variable z_0 may be introduced instead of a vector e' with all non-negative components as considered in this paper. The motivation for finding such a vector is that one may be able to follow n different paths starting from the initial basic solution. The author's ideas are described in Appendix 1.

Both Lemke's method, and the algorithm proposed by the author have been programmed in FORTRAN IV. These programs have been used to solve ten problems. The results are set out in table 3.

Problem No	Order of M	$N(S_P)$ $\leqslant$	$N(S_T)$	$N(S_F)$	Lemke's Method F=failed S=succeed	$N(S_A)$	$N(S_{CF})$
1	3	20	14	7	F	8	2
2	4	70	17	4	F	7	2
3	4	70	36	12	F	13	4
4	4	70	31	12	F	13	4
5	5	252	106	25	F	26	2
6	5	252	103	26	F	24	3
7	5	252	129	31	F	25	3
8	6	924	264	21	F	50	1
9	6	924	248	21	F	49	1
10	6	924	312	34	F	43	3

Table 3

S_A the set of all bases generated by the present algorithm

$N(S)$ = Cardinality of the set S.

Appendix 3.1

It has been shown in Example 3, that two possible paths followed by Lemke's method ended in unbounded rays. Another two paths can be followed from the initial basic point. This can be achieved by introducing some negative components of e'. The procedure is explained as follows:

First e'^T is introduced by the vector

$$(-2,8,8,8) .$$

so the problem in Example 3 may be written

$$\begin{cases} w_1 = 2 - 2z_0 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 + 8z_0 - z_1 + 2z_2 - z_3 + z_4 \\ w_3 = 3 + 8z_0 + 2z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = -6 + 8z_0 + 4z_1 + 3z_2 - z_3 - 3z_4 \end{cases}$$

or in tableau form

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	0	2	-2	1	3	-4
w_2	4	-8	1	-2	1	-1
w_3	11	-8	-2	2	-1	2
w_4	2	-8	-4	-3	1	3

Tableau A-1

The value of z_0 in the tableau A-1 is 1.

		$-w_1$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
z_0	1	$\frac{1}{2}$	-1	$\frac{1}{2}$	$\frac{3}{2}$	-2
w_2	4	4	-7	not	up	dated
w_3	11	4	-10	not	up	dated
w_4	2	4	-12	not	up	dated

Tableau A-2

In Tableau A-2, z_1 is the complement of w_1, as this variable cannot be made basic variable therefore Tableau A-2 represents an unbounded ray.

Now e'^T is introduced as:

$$(1,6,-3,8)$$

and the problem is written

$$\begin{cases} w_1 = 2 + z_0 + 2z_1 - z_2 - 3z_3 + 4z_4 \\ w_2 = -4 + 6z_0 - z_1 + 2z_2 - z_3 + z_4 \\ w_3 = 3 - 3z_0 + 2z_1 - 2z_2 + z_3 - 2z_4 \\ w_4 = 6 + 8z_0 + 4z_1 + 3z_2 - z_3 - 3z_4 \end{cases} ;$$

or

	1	$-z_0$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	3	-1	-2	1	3	-4
w_2	2	-6	1	-1	1	-1
w_3	0	3	-2	2	-1	2
w_4	2	-8	-4	-3	1	3

Tableau A-3

The value of z_0 in Tableau A-3 is 1.

	1	$-w_3$	$-z_1$	$-z_2$	$-z_3$	$-z_4$
w_1	3	$\frac{1}{3}$	$-\frac{8}{3}$	not	$\frac{8}{3}$	not
w_2	2	2	-3	up	-1	up
z_0	1	$\frac{1}{3}$	$-\frac{2}{3}$	dated	$-\frac{1}{3}$	dated
w_4	2	$\frac{8}{3}$	$-\frac{28}{3}$		$-\frac{5}{3}$	

Tableau A-4

	1	$-w_3$	$-z_1$	$-z_2$	$-w_1$	$-z_4$
z_3			-1	not	$\frac{3}{8}$	not
w_2	not	up	-4	up	$\frac{3}{8}$	up
z_0	dated		-1	dated	$\frac{1}{8}$	dated
w_4			-11		$\frac{5}{8}$	

Tableau A-5

As in the Tableau A-5 z_1 cannot be made basic variable, the procedure terminates in the unbounded ray.

From the above discussion it is deduced that all possible enumerations of the paths (exactly n paths) does not always guarantee to produce a solution to the fundamental problem.

References 3

3.1. Cottle, R.W., On a problem in Linear Inequalities, Journal of the London Mathematical Society, 43, pp 378-384 (1968).

3.2. ———————— Monotone solution of the Parametric Linear Complementary Problem. Journal of Mathematical Programming 3, pp 210-224 (1972).

3.3. ———————— Solution Rays for a Class of Complementary Problem. Journal of Mathematical Programming Study 1, pp 59-70 (1974).

3.4. ———————— The Principal Pivoting Method of Quadratic Programming in [3.5] pp 144-162.

3.5. Dantzig, G.B. and Veinott, A.F., ed. Mathematics of the Decision Sciences. Published by the American Mathematical Society, 1968.

3.6. Dantzig, G.B. and Cottle, R.W., On the Complementary Pivot Theory, in [3.5] pp 115-136.

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

3.8. Dantzig, G.B. and Cottle, R.W., Positive (Semi) Definite Matrices, and Mathematical Programming. Report of the Operational Research Center. University of California, Berkeley, ORC 63-18 (RR) May 1963.

3.9. Eaves, B.C., The Linear Complementary Problem. Management Science volume 17 No.9, May 1971.

3.10. Jahanshahlou, G.R. and Mitra, G., Two Algorithms for Finding all the Vertices of a Convex Polyhedron, Colloquia Mathematica Societatis János Bolyai 12. Progress in Operations Research, Eger (Hungary), 1974.

3.11. Lemke, C.E., On the Complementary Pivot Theory, in 5, pp 95-114.

3.12. ———————— Bimatrix Equilibrium Points and Mathematical Programming. Management Science volume 11, No 7, May 1965.

3.13. Murty, K.G., On the Number of Solutions to the Complementarity Problem and Spanning Properties of the Complementary Cones. Linear Algebra and its Application 5, pp 65-108 (1972).

3.14 Mitra, G., On the Fundamental Problem, Lecture Note, Brunel University.

3.15 MacCammon, S.R., On the Complementary Pivoting, Ph.D. Thesis, Rensselaer Polytechnic Institute. Tory, N.S. (1970).

3.16 Van de Panne, C.A., A Complementary Variant of Lemke's Method for the Linear Complementary Problem. Journal of Mathematical Programming 7, pp 283-310 (1974).

CHAPTER FOUR

Plant Location Problem

4.0 Summary

In this note plants are considered to have unlimited capacity and concave handling cost functions. This problem is formulated mathematically and some useful simplifications for computational purposes are given.

4.1 Introduction

In [4.1] the uncapacitated plant location problem with m plants and n customers, has been formulated as a mixed integer programming problem in the form

$$\text{Minimize } z = \sum_{i,j} c_{ij}x_{ij} + \sum_{i} f_i y_i \ ,$$

subject to

$$\sum_{i \in N_j} x_{ij} = 1 \quad j = 1,\ldots,n$$

$$0 \le \sum_{j \in P_i} x_{ij} \le n_i y_i \ , \quad i = 1,\ldots,m \qquad (1)$$

$$y_i = 0 \text{ or } 1 \quad (i = 1,\ldots,m) \ ,$$

where, $c_{ij} = D_j t_{ij}$

t_{ij} = the unit transportation cost from plant i to customer j ,

D_j = the demand at customer j ,

x_{ij} = the portion of D_j supplied from plant i ,

y_i = 0 if plant i is not opened
1 if plant i is opened

f_i = the fixed cost associated with the plant i, and $f_i > 0$,

N_j = the set of plants which can supply customer j ,

p_i = the set of those customers, that can be supplied by plant i ,

n_i = the number of elements in p_i .

The main difficulty in this problem is in choosing plants which are to be opened in an optimum solution.

Effroyson and Ray [4.1] suggested using a branch and bound method to find an optimal solution to the problem. Khumawala [4.2] has given some useful simplifications which reduce the computational effort. In section 4.2 the branch and bound method with Khumawala's simplifications is summarized. Section 4.3 describes the formulation of the problems in the general case. Some useful simplifications suggested by the author are put forward in this section. Section 4.4 contains some concluding remarks and computational experience.

4.2 A Branch and Bound Algorithm

Problem (1) is first solved as an LP (linear program) (replacing $y_i = 0$ or 1 by $0 \leq y_i \leq 1$) giving an optimal value z_0. If all the y's are integer then the problem is solved. Is som y_j are fractional, then one such is chosen and first fixed at zero, and the linear program again solved producing z_1, and then fixed at one and the linear program solved producing z_2. it is clear that

$$\bar{z} = \min (z_1, z_2) \tag{2}$$

is a new lower bound on z. This procedure if carried out iteratively will result in the construction of a tree whose nodes are represented by the z's and the corresponding value of the fixed y's. If a node is reached where all the y's are integer in the LP solution then the z value at this node gives an upper bound on z. A node where all the y's are integer will be called a terminal node, as opposed to a non-terminal node, where at least one y is fractional. The LP solution at a terminal node will be referred to as a terminal solution. Branching continues from any nonterminal node, whose optimal LP objective value is less than the current upper bound. The algorithm stops when there are no nonterminal nodes whose LP solution are less than the current upper bound. The current upper bound is then the optimal solution.

If, at some node K_1, K_0 are the set of indices of y's that are fixed at one and zero respectively, and K_2 are the indices of the remaining y's, then because of the assumption of unlimited plant capacity, the optimal solutions to the LP at this node is

$$x_{ij} = \begin{cases} 1 & \text{if } c_{ij} + {}^{g_i}/_{n_i} = \underset{k \epsilon K_1 \cup K_2}{\text{Min}} (c_{kj} + \frac{g_k}{n_k}) , \\ 0 & \text{otherwise} , \end{cases}$$

$$y_i = \begin{cases} 0 & \text{if } \quad i \epsilon K_0 , \\ 1 & \text{if } \quad i \epsilon K_1 , \\ \sum_{j \epsilon P_i} x_{ij}/n_i & \text{if } \quad i \epsilon K_2 ; \end{cases} \qquad (3)$$

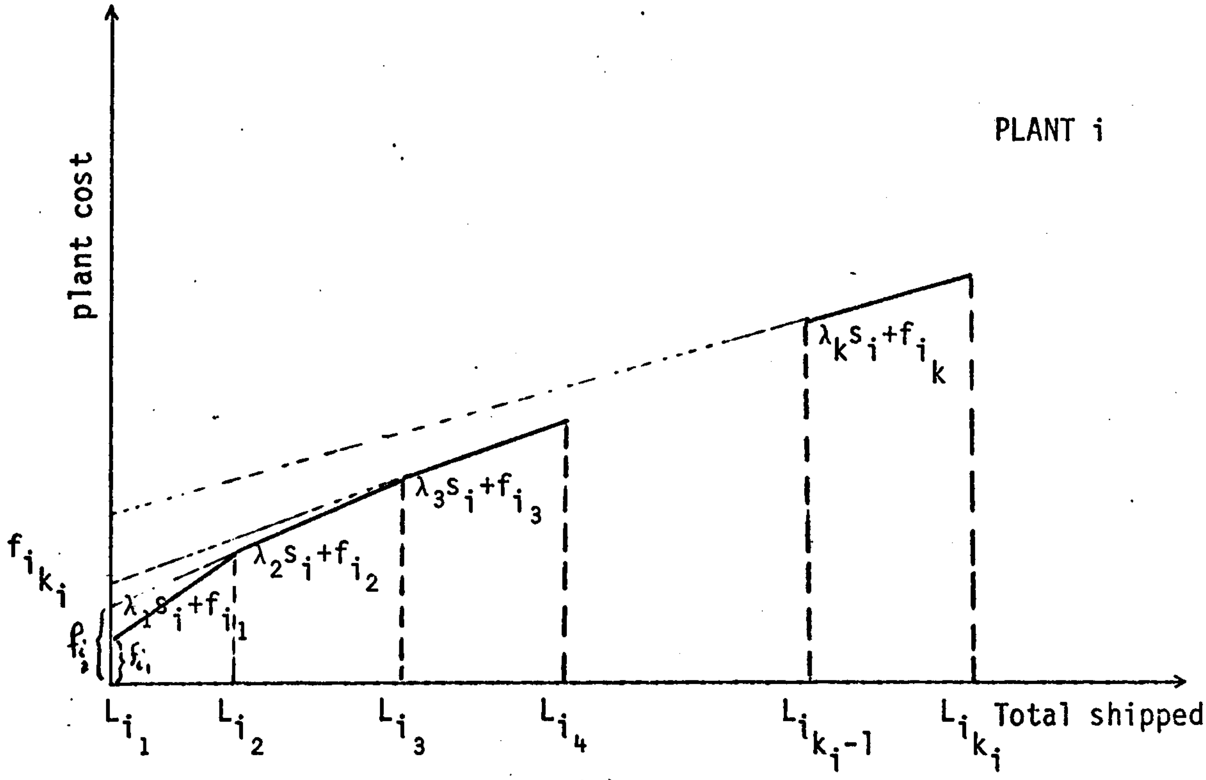
where,

$$g_k = \begin{cases} f_k & \text{if } \quad k \epsilon K_2 \\ 0 & \text{if } \quad k \epsilon K_1 . \end{cases}$$

The use of certain simplifications, which reduce the number of branches are given in [4.1,4.2]. As their modified forms are mentioned in section 4.3, they are not discussed here.

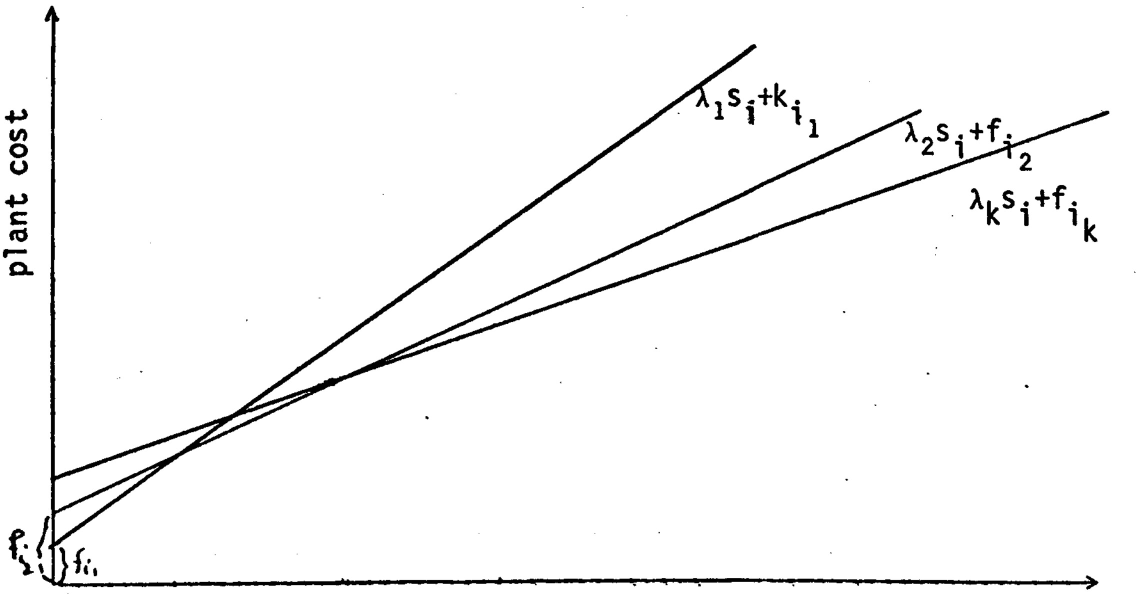
4.3 Formulation of the Problem in the General Case

In this case the function used to describe the plant cost is a piecewise linear concave function as shown in Fig(1)



Fig(1)

This case is of particular importance because it is often encountered in real-life problems. The concave cost function shown in Fig(1) can be represented by k_i separate linear cost functions as shown in Fig(2).



Fig(2)

Note that, the lower envelope of the k_i cost functionsis the original cost function. Replacing the variable x_{ij} with k_i variables $x_{ij1}, x_{ij2}, \ldots, x_{ijk_i}$, allows the concave cost function to be replaced by k_i linear functions, each having a different associated fixed cost $f_{i1}, f_{i2}, \ldots, f_{ik_i}$. Thus the problem has been expanded to have

$$nk_1 + nk_2 + \ldots + nk_m \qquad (4)$$

non-integer variables, and $k_1 + k_2 + \ldots + k_m$ fixed charge variables $y_{11}, y_{12}, \ldots, y_{1k_1}, y_{2_1}, \ldots, y_{2k_2}, \ldots, y_{m_1}, \ldots, y_{mk_m}$.

The objective is to formulate the problem in such a way that a formula like (3) can be used to solve the LP's associated with the problem at each node.

Let $\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{ik_i}$ be the slope of the lines in the Fig(2), $\lambda_{i_1} \geq \lambda_{i_2} \geq \ldots \geq \lambda_{ik_i} \geq 0$, as the original cost function was concave.

Define

$$C_{ijk} = (t_{ij} + \lambda_{ik})D_j \text{ for all } \begin{cases} i \epsilon N_j \ , \ (j = 1, \ldots, n) \\ k \epsilon M_i \end{cases} \qquad (5)$$

where

$$M_i = \{1, 2, \ldots, k_i\}, \ (i = 1, 2, \ldots, m) \ . \qquad (6)$$

Now the problem can be formulated as:

$$\text{Minimize } z = \sum_{\substack{i \epsilon N_j \\ j \epsilon P_i \\ k \epsilon M_i}} C_{ijk} x_{ijk} + \sum_{\substack{i \epsilon P_i \\ k \epsilon M_i}} f_{ik} y_{ik}$$

subject to

$$\begin{cases} \sum_{\substack{i\in N_j \\ k\in M_i}} x_{ijk} = 1 \ , \quad (j = 1,\ldots,n) \\ \sum_{k\in M_i} y_{ik} \le 1 \ , \quad (i = 1,\ldots,m) \\ 0 \le \sum_{j\in P_i} x_{ijk} \le n_i y_{ik} \ , \quad \left({i = 1,\ldots,m \atop k\in M_i}\right) \\ y_{ik} = 0 \text{ or } 1 \end{cases} \tag{7}$$

If at a particular node $K'_1 = \{(i,j) | y_{ij}$ is fixed at 1$\}$ and $K'_0 = \{(i,j) | y_{ij}$ is fixed at 0$\}$ and K'_2 be the set of ordered pairs (i,j) corresponding to free variables y_{ij}. The optimal solution to programming problem at this node is given by

$$x_{ijk} = \begin{cases} 1 & \text{if } C_{ijk} + \dfrac{g_{ik}}{n_k} = \underset{(h,\ell)\in(K'_1 \cup K'_2)}{\text{Min}} \left[C_{hj\ell} + \dfrac{g_{h\ell}}{n_h}\right] \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$y_{ik} = \begin{cases} 0 & \text{if } (i,k)\in K'_0 \\ 1 & \text{if } (i,k)\in K'_1 \\ \left(\dfrac{1}{n_i}\right) \sum_{j\in P_i} x_{ijk} & \text{if } (i,k)\in K'_2 \end{cases} \tag{9}$$

$$g_{ik} = \begin{cases} f_{ik} & \text{if } (i,k)\in K'_2 \\ 0 & \text{if } (i,k)\in K'_1 \end{cases} \tag{10}$$

Some useful simplifications, which significantly reduce computational effort are mentioned below:

1. Let $L_{i_2}, L_{i_3}, \ldots, L_{i_{k_i}}$ $(i = 1,\ldots,m)$ be the abscissi of the points of discontinuity of gradients for cost function of the plant i (see Fig(1)) and $L_{i_1} = 0$ $(i = 1,\ldots,m)$ then if, for some integers h_i

$$\begin{cases} \sum_{j\epsilon P_i} D_j \geq L_{ih_i} \text{ , and} \\ \sum_{j\epsilon P_i} D_j < L_{ih_i+1} \end{cases} \tag{11}$$

Then,

$$y_{ik} = 0 \quad k = h_i+1,\ldots,k_i \quad (i = 1,2,\ldots,n) ,$$

in all the solutions, i.e. if the total demand of the potential customers, which can be supplied by the plant i is less than L_{ih_i+1} and is equal or greater than L_{ih_i} then the integer variable y_{ik} can be kept fixed at zero for $k = h_i+1,\ldots,k_i$.

N.B. This simplification and the next one are carried out before any calculation.

2. If

$$\text{Min}(D_1,D_2,\ldots,D_n) > \text{Max}(L_{1h},L_{2h},\ldots,L_{nh}) \tag{12}$$

for some h, and there exists some i for which

$$\text{Min}(D_1,D_2,\ldots,D_n) < L_{ih+1} , \tag{13}$$

then

$$y_{ik} = 0 \quad i = 1,\ldots,m , \quad k = 1,\ldots,h-1 ,$$

in all the solutions.

Some simplifications, which have been mentioned in [1,2] can be used here, with some modifications. The modified form of those simplifications are listed below:

3. This simplification determines a minimum bound for opening a plant. If this bound is positive the plant is fixed open. Mathematically this can be stated as:

If $(i,\ell)\epsilon K_2'$, $j\epsilon P_i$,

$$\nabla_{ij\ell} = \underset{(h,k)\in(K_1'\cup K_2')\ \&\ h\in N_j\ \&\ (h,k)\neq(i,\ell)}{\text{Min}} [\text{Max} \ (C_{hjk} - C_{ij\ell}, 0)] \tag{14}$$

$$\Delta_{i\ell} = \sum_{j\in P_i} \nabla_{ij\ell} - f_{i\ell}$$

It is clear that if $\Delta_{i\ell} \geq 0$, then $y_{i\ell} = 1$, and $y_{ik} = 0$ $(\underset{k\neq\ell}{k = 1,\ldots,k_i})$ for all the branches emanating from this node.

4. This simplification provides a means of reducing n_i. If for some plant i and customer j $j\in P_i$

$$\underset{\ell\in M_i}{\text{Max}} \left[\underset{(h,k)\in K_1'\ \&\ h\in N_j}{\text{Min}} (C_{hjk} - C_{ij\ell}) \right] < 0 , \tag{15}$$

then n_i is reduced by one. If the inequality holds for all $j\in P_i$, then $P_i = \phi$, $n_i = 0$ and $y_{i1} = y_{i2} = \ldots = y_{ik_i} = 0$ for all the branches emanating from the node. Clearly if an already open plant can supply a customer j cheaper than any of free plants, then such a customer should not be considered as a potential customer of the free plants at the node.

5. This simplification determines a maximum bound on the cost reduction for opening a plant. If this bound is negative the plant will be fixed closed. For $(i,k)\in K_2'$, $j\in P_i$ define

$$\omega_{ijk} = \underset{(h,\ell)\in K_1'\ \&\ h\in N_j\ \&\ (h,\ell)\neq(i,k)}{\text{Min}} [\text{Max} \ (C_{hj\ell} - C_{ijk}, 0)] \tag{16}$$

$$\Omega_{ik} = \sum_{j\in P_i} \omega_{ijk} - f_{ik} \tag{17}$$

If $\Omega_{ik} < 0$, then

$y_{ik} = 0$ for all the branches emanating from the node.

By considering the last three modified simplifications the author suggests another simplification as:

6. It was mentioned earlier that if for some plant i_0 and $j_0 \epsilon P_{i_0}$, the inequality

$$\operatorname*{Max}_{\ell \epsilon K_{i_0}} \left[\operatorname*{Min}_{(h,k)\epsilon K_1' \ \& \ h \epsilon N_j} (C_{hj_0k} - C_{i_0j_0\ell_0}) \right] < 0 \qquad (18)$$

then n_{i_0} is reduced by one. Therefore the total demand which can be supplied from plant i_0 is reduced by D_{j_0}. Now compute

$$T_{i_0} = \sum_{\substack{j \epsilon P_i \\ j \neq j_0}} D_j \ , \qquad (19)$$

if

$$L_{ih+1} > T_{i_0} \geq L_{ih} \ ,$$

then set $y_{ik} = 0$, $k = h+1,\ldots,k_i$,
for all the branches emanating from the node.

7. An Efficient Method for Solving the LP Problems at Nodes

If for some j_0 and $(i_0,k_0)\epsilon(K_1' \cup K_2')$ with $i_0 \epsilon N_{j_0}$ either $(i_0,k_0)\epsilon K_1'$ and $\nabla_{i_0j_0k_0} > 0$ or $(i_0,k_0)\epsilon K_2'$ and $\nabla_{i_0j_0k_0} > f_{i_0k_0}/n_{i_0}$ then

$$x_{i_0j_0k_0} = 1 \ , \ \text{and } x_{ij_0k} = 0 \ \text{ otherwise} \qquad (20)$$

Proof: First suppose $(i_0,k_0)\epsilon K_1'$ & $i_0 \epsilon N_{j_0}$, and

$$\nabla_{i_0j_0k_0} > 0 \ ,$$

simply this means that

$$\underset{(h,k)\in M}{\text{Min}} \left[\text{Max}(C_{hj_0k} - C_{i_0j_0k_0}, 0)\right] > 0 \tag{21}$$

where

$$M = \{(i_1,k_2) \mid (i_1,k_2)\in(K_1'UK_2') \& i_1\in N_{j_0} \& (i_1,k_2) \neq (i_0,k_0)\} .$$

From (21) it is deduced that:

$$\text{Max}(C_{hj_0k} - C_{i_0j_0k_0}, 0) > 0 , \tag{22}$$

so

$$C_{hj_0k} > C_{i_0j_0k_0} \quad \text{for all } (h,k)\in M .$$

As $(i_0,k_0)\in K_1'$, therefore $g_{i_0k_0} = 0$, so (22) may be expressed as:

$$C_{i_0j_0k_0} + \frac{g_{i_0k_0}}{n_{i_0}} < C_{hj_0k} + \frac{g_{hk}}{n_h} \tag{23}$$

for all $(h,k)\in M$, since $g_{hk} \geq 0$.
(23) is equivalent to

$$C_{i_0j_0k_0} + \frac{g_{i_0k_0}}{n_{i_0}} = \underset{(h,k)\in K_1'UK_2'}{\text{Min}} \left(C_{hj_0k} + \frac{g_{hk}}{n_h}\right) , \tag{24}$$

therefore from (24) and (8) it is deduced that

$$x_{i_0j_0k_0} = 1 \quad \text{and } x_{hj_0k} = 0 \text{ otherwise} .$$

Now let $(i_0,k_0)\in K_2'$, $i_0\in N_{j_0}$, and

$$\nabla_{i_0j_0k_0} > \frac{f_{i_0k_0}}{n_{i_0}} \tag{25}$$

From (14) it is deduced that,

$$\text{Max}(C_{hj_0k} - C_{i_0j_0k_0}, 0) > \frac{g_{i_0k_0}}{n_{i_0}} > 0$$

For all $(h,k) \epsilon M$.

In a similar manner to the above it can be deduced that:

$$C_{i_0j_0k_0} + \frac{g_{i_0k_0}}{n_{i_j}} = \underset{(h,k)\epsilon K_1' \cup K_2'}{\text{Min}} \left(C_{hj_0k} + \frac{g_{hk}}{n_h}\right), \tag{26}$$

therefore $x_{i_0j_0k_0} = 1$, and $x_{hj_0k} = 0$ otherwise.

As the ∇_{ijk} are calculated as part of previous simiplification, little extra computational cost is required in applying the above theorem.

The following point is worthwhile mentioning. Suppose for plant i the original handling cost is piecewise linear but not concave. Then as above this plant can be decomposed into several plants with 'linear' costs and different fixed charges. However, if during computation it is desired to fix $y_{ik} = 1$ then the following constraint must also be imposed

$$L_{ik} \leq \sum_{j\epsilon S} D_j < L_{ik+1} \tag{27}$$

where S = {j| customer j is supplied by plant i}. else the solution generated to the problem will be invalid. In the case of a concave cost function this constraint (27) will hold in an optimal solution anyway as any solution where (27) does not hold there always will be another better solution in which (27) does hold.

Branching Decision Rules

The branch and bound method requires that a plant is selected from the set of free plants at the node from which further branching is to be done. The selected plant is constrained to be closed and open respectively to yield two additional nodes. The selection of such a plant is called a 'branching decision', and the rule used for this selection is called the branching decision. Delta-rules, omega-rules, y-rules, and demand-rules can be applied to the problem in hand,for further details see [4.2].

Example: Consider the following problem with five plants and seven customers. Details plant and delivery costs are given the following table:

i \ J	1	2	3	4	5	6	7
1	-	3	4	-	4	-	3
2	4	-	6	5	-	5	4
3	3	-	-	-	4	6	-
4	4	-	5	4	-	4	-
5	-	4	-	6	-	-	5
D_j	30	25	40	20	50	30	60

t_{ij}

Table 3-1

i \ J	1	2	3	4	5	6
1	3.0	2.5	2.1	2.0	1.8	1.5
2	3.0	2.8	2.3	2.1	2.0	1.8
3	3.0	2.2	2.0	1.8	1.5	1.4
4	3.0	2.3	2.0	1.9	1.5	1.3
5	3.5	3.0	2.5	2.0	1.5	1.4

λ_{ij}

Table 3-2

In this problem $k_1 = k_2 = \ldots = k_5 = 6$,

i \ k	1	2	3	4	5	6
1	60	140	220	300	500	M
2	70	150	210	320	510	M
3	80	130	200	340	450	M
4	70	140	240	300	550	M
5	50	150	200	310	600	M

L_{ik}

Table 3-3

where M is an arbitrary large number.
Given $f_{11} = 20$, $f_{21} = 25$, $f_{31} = 18$, $f_{41} = 20$ and $f_{51} = 28$.
As it is being assumed that

$$\lambda_{ik}L_k + f_{ik} = \lambda_{ik+1}L_k + f_{ik+1} ,$$

all other f's can be calculated from the formula

$$f_{ik+1} = L_k(\lambda_{ik} - \lambda_{ik+1}) + f_{ik} \qquad \begin{array}{l} i = 1,2,\ldots,m \\ k \in M_i \\ k \neq 1 \end{array}$$

This is shown in Table 3-4

i \ k	1	2	3	4	5	6
1	20	50	106	128	188	338
2	25	39	114	156	188	290
3	18	82	108	148	250	295
4	20	69	111	135	255	365
5	28	53	128	228	383	443

f_{ik}

Table 3-4

The first simplification is then applied

Let

$$A_1 = \sum_{j\epsilon P_1} D_j = 175 \qquad A_2 = \sum_{j\epsilon P_2} D_j = 180 \qquad A_3 = \sum_{j\epsilon P_3} D_j = 110$$

$$A_4 = \sum_{j\epsilon P_4} D_j = 120 \qquad A_5 = \sum_{j\epsilon P_5} D_j = 105$$

$L_{12} = 140 < A_1 < 220 = L_{13}$ $\qquad$ $L_{22} = 150 < A_2 < L_{23} = 210$

$L_{31} = 80 < A_3 < 130 = L_{32}$ $\qquad$ $L_{41} = 70 < A_4 < L_{42} = 140$

$L_{51} = 50 < A_5 < 150 = L_{52}$, therefore

$y_{14} = y_{15} = y_{16} = 0 \qquad y_{24} = y_{25} = y_{26} = 0 \qquad y_{33} = y_{34} = y_{35} = y_{36} = 0$

$y_{43} = y_{44} = y_{45} = y_{46} = 0 \qquad y_{53} = y_{54} = y_{55} = y_{56} = 0$

Having applied the first simplification, out of 18 of the 30 variables y_{ij} become zero.

The C_{ijk} are now calculated, and are shown in Table 3-5

	1	2	3	4	5	6	7
1	-	C_{121}=150 C_{122}=137.5 C_{123}=127.5	C_{131}=280 C_{132}=260 C_{133}=244	-	C_{151}=350 C_{152}=325 C_{153}=305	-	C_{171}=360 C_{172}=330 C_{173}=306
2	C_{211}=210 C_{212}=204 C_{213}=189	-	C_{231}=360 C_{232}=352 C_{233}=332	C_{241}=160 C_{242}=156 C_{243}=146	-	C_{261}=240 C_{262}=234 C_{263}=219	C_{271}=420 C_{272}=408 C_{273}=378
3	C_{311}=180 C_{312}=156	-	-	-	C_{351}=350 C_{352}=310	C_{361}=270 C_{362}=246	-
4	C_{411}=210 C_{412}=189	-	C_{431}=320 C_{432}=292	C_{441}=140 C_{442}=126	-	C_{461}=210 C_{462}=189	-
5	-	C_{521}=187.5 C_{522}=175	-	C_{541}=190 C_{542}=180	-	-	C_{571}=510 C_{572}=480

Table 3-5

$$\Delta_{121} = 0 \quad \Delta_{122} = 0 \quad \Delta_{123} = 10 \,,$$
$$\Delta_{131} = 0 \quad \Delta_{132} = 0 \quad \Delta_{133} = 16 \quad \Delta_{151} = 0 \quad \Delta_{152} = 0 \quad \Delta_{153} = 5 \,,$$
$$\Delta_{171} = 0 \quad \Delta_{172} = 0 \quad \Delta_{173} = 24 \,,$$

so

$$\begin{cases} \Delta_{11} = 0 - 20 = -20 \\ \Delta_{12} = 0 - 50 = -50 \\ \Delta_{13} = (10 + 16 + 24 + 5) - 106 = -51 \,. \end{cases}$$

Similarly it can be shown that

$$\begin{cases} \Delta_{21} = -25 \\ \Delta_{22} = -39 \\ \Delta_{23} = -114 \end{cases} \quad \begin{cases} \Delta_{31} = -18 \\ \Delta_{32} = -42 \end{cases} \quad \begin{cases} \Delta_{41} = -20 \\ \Delta_{42} = -34 \end{cases} \quad \begin{cases} \Delta_{51} = -28 \\ \Delta_{52} = -53 \end{cases}$$

The optimal solution to the linear program at this node is

$x_{312} = 1 \;\; x_{122} = 1 \;\; x_{133} = 1 \;\; x_{442} = 1 \;\; x_{153} = 1 \;\; x_{462} = 1 \;\; x_{173} = 1$,
and all $x_{ijk} = 0$ otherwise, and $y_{13} = \frac{3}{4}$, $y_{12} = \frac{1}{4}$, $y_{32} = \frac{1}{3}$, $y_{42} = \frac{2}{4}$
and all the other y's are zero. Therefore

$$K_0 = \{(5,1), (5,2), (4,1), (3,1), (2,1), (2,2), (2,3), (1,1)\}$$
$$K_2 = \{(3,2), (4,2), (1,3), (1,2)\}$$
$$K_1 = \phi \,,$$
$$Z_1 = 1617.80$$

Note in finding optimal solution to LP at this node $\Delta_{462} = 21 > \frac{69}{4}$, is being used to set $x_{462} = 1$.

Now by applying y-rules y_{13} is set to 1 and $y_{11} = y_{12} = 0$. Applying the 4th simplification to plant 3 for which customer 5 can be supplied

more cheaply from plant 1 which is already fixed open $n_3 = 3 - 1 = 2$, and the optimal solution to the linear program at this node is
$x_{312} = x_{442} = x_{123} = x_{123} = x_{133} = x_{153} = x_{173} = 1$, and all $x_{ijk} = 0$ otherwise, and

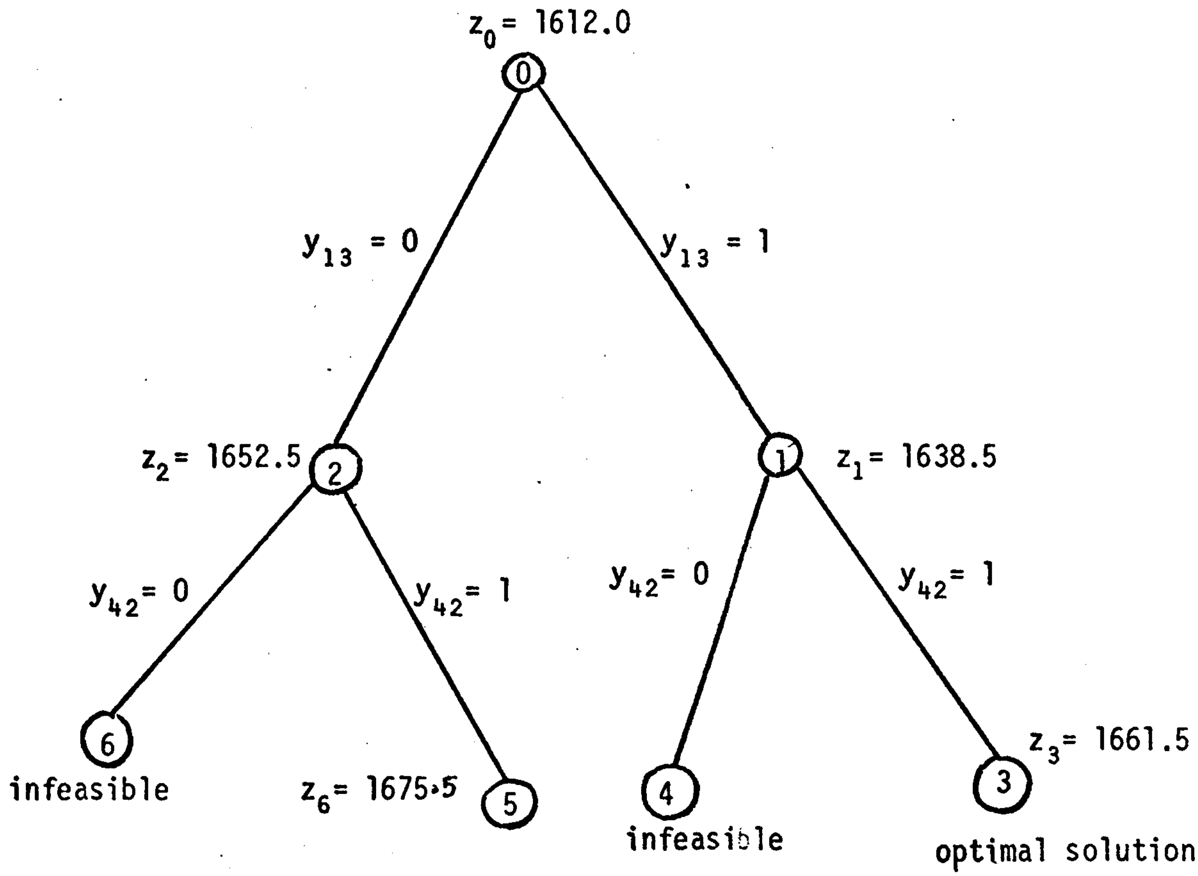
$y_{13} = 1 \quad y_{42} = \frac{1}{2} \quad y_{32} = \frac{1}{2}$, so

$K_1 = \{(1,3)\}$

$K_2 = \{(4,2), (3,2)\}$

$K_0 = \{(5,1), (5,2), (4,1), (3,1), (2,1), (2,2), (2,3), (1,1), (1,2)\}$

Following the procedure, the optimal solution is $z_3 = 1661.5, y_{13} = 1, y_{42} = 1$. The related branch and bound tree is shown in Fig(3).



Fig(3)

4.4 Concluding Remarks and Computational Experience

The algorithm mentioned in section 3 has been programmed by the author in FORTRAN IV. In writing this program the following features have been introduced.

In a real life problem a customer cannot be supplied by all the plants. Therefore in the tableau containing C_{ijk} many of the blocks are kept blank. By using the graph related to this problem these blank blocks are not stored. Therefore problems of considerable size can be handled by this program.

A major limitation of the branch and bound algorithm is the amount of computer storage required to store all the eligible nonterminal nodes and associated information. However, it is found that these storage requirements can be reduced by deleting nodes which are no longer processed by the algorithm. The storage used for these deleted nodes is effectively used over and over again for the new nodes that are generated as the algorithm proceeeds.

This program is flexible in its design and it is possible to use any of the eight branching decision rules mentioned in [4.2]

5 test problems with the following characteristics have been solved by this program.

Problem 1. 5 plants, 7 customers, and the cost function of each plant contains 6 segments (30 integer variables).

Problem 2. 10 plants, 20 customers (36 integer variables). The cost functions of the plants have altogether 36 segments.

Problem 3. 14 plants, 30 customers, cost function for each plant contains 2 segments (28 integer variables) for each plant.

Problem 4. 15 plants, 25 customers, cost function of each plant contains 3 segments (75 integer variables).

Problem 5. 15 plants, 40 customers, with 80 integer variables.

The following results are obtained (*)

The number of iterations required to obtain the optimal solution is less than the number of integer variables in the problems. The number of integer variables which are fixed at zero or one in the first iteration is very high in proportion.

In spite of the limited computational experience the characteristics of algorithm lead us to believe that it can be equally effective for large scale problems.

(*) y-Rules in [4.2] have been used for the solution in all these problems.

References 4

4.1 Effroymson, M.A. and Ray, T.L., A Branch-Bound Algorithm for Plant Location, Operations Research, vol 14 (May-June 1966).

4.2 Khumawala, B.M., An Efficient Branch and Bound Algorithm for the Warehouse Location Problem, Management Science, vol 18, No 12, August 1972.

4.3 Balinski, M.L., On Finding Integer Solutions to Linear Programs, Mathematica, Princeton, New Jersey (May 1964).

4.4 Spielberg, K., An Algorithm for the Simple Plant Location Problem, with Some Side Conditions, Operations Research, vol 17 (January-February 1969).

4.5 ____________ Plant Location with Generalized Search Origin, Management Science, vol 16 (November 1969).

CHAPTER FIVE

Chinese Representation of Integers and its Application in an Algorithm to Find the Smith Normal Form for an Integer Matrix

5.0 Summary

An algorithm which transforms a nonsingular integer matrix to its Smith Normal Form has been proposed. The algorithm is based on the chinese representation for integers, and is considered to be more efficient than any other known algorithms used for this purpose.

5.1 Introduction

To analyse a pure integer programming problem as a group knapsack problem over a cyclic group [5.1, 5.2, 5.3, 5.4] it is necessary to consider an auxiliary problem well known in the literature as ILPC i.e. Integer Linear Programming over Cone. If the solution to the ILPC associated with the given problem is also a solution of this problem then the ILPC is called an asymptotic integer linear program. For a given ILP the corresponding ILPC can be easily analysed by considering its equivalent representations. There exist two classical canonical representations [5.7] called Hermite Normal Form and Smith Normal Form which may be used to obtain the desired equivalent representations of the problem. Obtaining the Smith Normal Form corresponding to the optimal basis matrix of the ILPC is the crucial step of this analysis. In this study an efficient algorithm to find the Smith Normal Form for a nonsingular integer matrix has been proposed.

In section 5.2 the definition and existence of these normal forms are discussed and a general algorithm [5.2] for obtaining the Smith Normal Form is stated. The chinese (modular) representation of an integer is considered in section 5.3. Section 5.4 contains a description of the proposed algorithm and an example. The computational implications of the proposed algorithm set against other known algorithms are discussed in section 5.5. The appendix 5.1 contains a short note on finding the gcf (greatest common factor) of a set of integers represented in the chinese form. Appendix 5.2 contains a proof of a theorem stated in section 5.3.

5.2 Canonical Representations, [5.3],[5.7].

Two canonical representations are known from the middle of last century and these are stated without proof in the two following theorems.

Theorem 1. Hermite Normal Form: Given an mth order nonsingular integer matrix B there exists an mth order, unimodular, integer matrix K such that

$$[f_{ij}] \equiv F = BK ,$$

where

$$\begin{array}{lll} \text{(i)} & f_{ij} = 0 , & \text{for all } j > i , \\ \text{(ii)} & f_{ii} > 0 , & \text{for all } i , \\ \text{(iii)} & f_{ij} < 0 \text{ and } |f_{ij}| < f_{ii} , & \text{for all } i, \text{ and } j < i . \end{array} \qquad (1)$$

The matrix F is known as the Hermite Normal Form of B and is unique for a given B.

Theorem 2. Smith Normal Form: Given an mth order nonsingular integer matrix B, there exist mth order, unimodular, integer matrices R and C such that

$$[\delta_{ij}] \equiv \Delta = RBC ,$$

where

(i) Δ is a diagonal matrix ($\delta_{ij} = 0,\ i \neq j$),

(ii) the diagonal elements denoted for convenience as $\delta_{ii} = \delta_i$, $i = 1,2..m$, are all positive, (2)

(iii) δ_i is a divisor of δ_{i+1} , $i = 1,2...m-1$.

The matrix Δ is called the Smith Normal Form of B; for a given B the corresponding Δ is unique but the unimodular matrices R and C corresponding to the row and column operations are not unique.

Starting from the relationship,

$$\det\Delta = |\det RBC| = |\mathrm{Det}R| \times |\mathrm{Det}B| \times |\mathrm{Det}C| = |\mathrm{Det}B| = D \ ,$$

it can be deduced that (3)

$$\prod_{i=1}^{m} \delta_i = D \ .$$

In the following algorithm which transforms an integer matrix B into its Smith Normal Form, a column and a row of B are referred to as b_j^c and b_j^r respectively and the elements as b_{ij}, $i,j = 1,2...m$.

Step. 0. Set the cycle number $t = 1$.

Step. 1. In the matrix of order $(m-t+1)$ interchange the columns and rows such that the leading diagonal element b_{tt} has the least absolute value of all the nonzero elements of the matrix.

Step. 2. If b_{tt} divides b_{tj} exactly, for all $j = t+1,...m$, goto step. 3. Otherwise for some j, say $j = k$, b_{tt} does not divide b_{tk}. In this case let,

$$b_{tk} = nb_{tt} + q \ , \qquad (4)$$

where n is an integer and $0 < q < b_{tt}$.

Construct a column such that

$$\bar{b}_k^c = b_k^c - nb_t^c , \tag{5}$$

where the element $\bar{b}_{tk} = q$ is strictly less than b_{tt}. Replace the column b_k^c by $\bar{b}_k^c$ and goto step 1.

Step 3. If b_{tt} divides b_{it} exactly for $i = t+1,\ldots m$, goto step 4. Otherwise for some i, say i = k, b_{tt} does not divide b_{kt}. In this case let,

$$b_{kt} = nb_{tt} + q , \tag{6}$$

where n is an integer and $0 < q < b_{tt}$.

Construct a row such that

$$\bar{b}_k^r = b_k^r - nb_t^r , \tag{7}$$

where the element $\bar{b}_{kt} = q$ is strictly less than b_{tt}. Replace the row b_k^r by $\bar{b}_k^r$ and go to step 1.

Step 4. Reduction Operation. At this stage if $b_{tt} \not> 0$ then negate the t^{th} row of the matrix. The element b_{tt} divides all the elements in the t^{th} row and the t^{th} column of the matrix such that

$$\begin{aligned} b_{tj} &= n_j b_{tt} , \quad n_j \text{ integer} , \quad j = t+1,\ldots m, \\ \text{and } b_{it} &= \ell_i b_{tt} , \quad \ell_i \text{ integer} , \quad i = t+1,\ldots m. \end{aligned} \tag{8}$$

Construct the columns,

$$\overline{b}^c_j = b^c_j - n_j b^c_t, \text{ whereby } b_{tj} = 0,\ j = t+1,\ldots m, \tag{9}$$

and replace b^c_j by $\overline{b}^c_j$ for $j = t+1,\ldots m$, further set $b_{it} = 0$, $i = t+1,\ldots m$.

For the cycle $t = 1$ this transformation leads to the matrix shown in Tableau 1.

$$\left(\begin{array}{c|cccc} b_{11} & 0 & 0 & \cdots & 0 \\ \hline 0 & b_{22} & b_{23} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & b_{m2} & b_{m3} & \cdots & b_{mm} \end{array}\right)$$

(Tableau 1)

Step 5. If b_{tt} divides exactly $b_{ij}(i,j = t+1,\ldots m)$, then set $t = t+1$. If $t = m$ then goto Exit, otherwise goto step 1. On the other hand if for some i,j the following relationship holds,

$$b_{ij} = n.b_{tt} + q_{ij} ,$$

$$\text{where } 0 < q_{ij} < b_{tt} , \tag{10}$$

then find $\min\limits_{\substack{i,j \\ \{0 < q_{ij} < b_{tt}\}}} \{q_{ij}\} = q_{\ell k}$ say .

By a combination of row and column operation similar to those set out in step 2 and step 3, it is possible see Hu [5.2] to make $q_{\ell k}$ the leading diagonal element of the remaining matrix of order $m - t + 1$. Thus new value of $b_{tt} = q_{\ell k}$. Now goto step 1.

Exit. If $b_{mm} < 0$ then set $b_{mm} = -b_{mm}$.

The matrix is now transformed to its Smith Normal Form.

T.C. Hu [5.2] provides an upper bound on the number of times the loop, step 1 through step 5 should be obeyed, he has also proposed an improved algorithm for obtaining the Smith Normal Form for a given matrix.

5.3 Some Relevant Theoretical Results and the Chinese Representation of Integer. [5.5],[5.6].

Given a set of integers $m_1, m_2 \ldots m_n$, and their gcf d this may be expressed as

$$(m_1, m_2, \ldots m_n) = d \ . \qquad (11)$$

The following theorems connecting $m_1, m_2, \ldots m_n$ and d are well known [5.5].

Theorem 3. There exists a set of integer multipliers $k_1, k_2 \ldots k_n$ such that

$$d = k_1 m_1 + k_2 m_2 \ldots k_n m_n \ . \qquad (12)$$

Theorem 4. If ℓ divides $m_1, m_2 \ldots m_n$ the ℓ also divides d. The proof of this theorem follows directly from Theorem 3.

Theorem 5. If Δ is the Smith Normal Form of the Matrix B, i.e. RBC = Δ as in (2) then δ_1 is the gcf of the set of integer elements b_{ij}, $i,j = 1,2 \ldots m$, of the B matrix. The author's proof of this theorem as set in [5.4] is presented in Appendix 5.2.

Given a positive integer n, and the set of the first k prime numbers $p_1, p_2 \ldots p_k$, the following k congruences may be stated

$$\begin{aligned} n &\equiv r_1 (\text{mod } p_1) \;, \quad p_1 = 2 \\ n &\equiv r_2 (\text{mod } p_2) \;, \quad p_2 = 3 \\ n &\equiv r_k (\text{mod } p_k) \;, \quad p_k = \text{kth prime} \end{aligned} \tag{13}$$

where $0 \leq r_i < p_i$, $i = 1,2 \ldots k$. From these congruences a representation of the integer n is given as

$$n \sim (r_1, r_2, \ldots r_k) \;, \tag{14}$$

and for n lying in the range $0 \leq n < \prod_{i=1}^{k} p_i$ the representation in (14) is unique. This is well known in the literature [5.6] as the chinese or the modular representation. The attraction of this representation from the computational point of view is that given the chinese representations of two numbers their sum, difference and product may be obtained by sum, difference and product operations carried out modulo the prime numbers used to obtain the components (remainders) in the representation. This is illustrated below.

An Example.

Consider the number 678 which may be expressed as

$$\begin{aligned} 678 &\equiv 0 \pmod{2} \\ &\equiv 0 \pmod{3} \\ &\equiv 3 \pmod{5} \\ &\equiv 6 \pmod{7} \\ &\equiv 7 \pmod{11} \\ &\equiv 2 \pmod{13} \\ &\equiv 15 \pmod{17} \end{aligned} \qquad (15)$$

or $678 \sim (0, 0, 3, 6, 7, 2, 15)$;

similarly 143 may be expressed as

$$143 \sim (1, 2, 3, 3, 0, 0, 7)$$

$$\begin{aligned} \text{Thus } (678 + 143) &\sim (1, 2, 1, 2, 7, 2, 5) \sim 821 , \\ (678 - 143) &\sim (1, 1, 0, 3, 7, 2, 8) \sim 535 , \\ \text{and } (678 \times 143) &\sim (0, 0, 4, 4, 0, 0, 3) \sim 96954 . \end{aligned} \qquad (16)$$

An important implication of the above operations is that these may be carried out in parallel in a computer with parallel processing facility. However, in the present study our immediate concern is to exploit this representation to obtain the gcf of a set of numbers with a minimum number of division operations between integers. Note that the division operation in a computer is an order of magnitude longer in time than the multiplication and the addition operation, also note that the representation cannot be extended to the division operation which yields a dividend and a remainder. However, when the division is exact i.e. the remainder is zero and the divisor is a prime, the chinese representation may be exploited again, see appendix 5.1. Reference [5.5] may be consulted to find an algorithm for converting a chinese representation into its decimal form; note that the proposed algorithm does not require this conversion.

An important corollary arising out of this representation is that the gcf 1 for a set of relatively prime integer numbers may be established at a glance i.e. a direct search in the context of automatic computation. In Appendix 5.1 the theory underlying this approach is more formally set out.

An Example.

Consider the numbers,

$$\begin{aligned} 64 &\sim (0,\ 1,\ 4,\ 1) \\ 25 &\sim (1,\ 1,\ 0,\ 4) \\ 33 &\sim (1,\ 0,\ 3,\ 5) \end{aligned} \tag{17}$$

these are relatively prime as none of the columns formed by the corresponding components is a null vector. Note that in all the other known algorithms it requires a lot more computational effort to establish this unit gcf. In Appendix 5.1 an algorithm for finding the gcf of a set of integers presented in their chinese form is outlined, where these numbers are not relatively prime, i.e. they have one or more primes as common factor.

5.4 An Algorithm for Finding the Smith Normal Form Based on the Chinese Representation of Integers.

Given the integer matrix,

$$[b_{ij}] \equiv B \ , \tag{18}$$

the matrix R is defined as

$$[(r_1,\ r_2,\ \ldots\ r_k)_{ij}] \equiv R \tag{19}$$

where $b_{ij} \sim (r_1,\ r_2,\ \ldots\ r_k)_{ij}$ for all i,j is the chinese representation of b_{ij}

Step 0. Obtain the matrix R from the given integer matrix B. Set cycle number $t = 1$ and $\delta_o = 1$.

Step 1. Obtain the gcf d for all b_{ij}, $i,j = t, t+1, \ldots m$, and set $\delta_t = \delta_{t-1} \times d$, where d is obtained by applying the algorithm given in appendix 1. Note that by the end of this step the gcf d is taken out of the remaining matrix.

Step 2. Either (a) there exists one element of magnitude unity in the remaining matrix. In this case goto step 3. Or (b) there is no element of unit magnitude in this matrix therefore carry out the 'Auxiliary Sequence' which makes one of the elements of the matrix unit in magnitude.

Step 3. Let $|b_{\ell p}| = 1$ be the unit element of the matrix, then by at most two operations (one row, and one column) this element is made the leading element b_{tt} of the matrix B. Corresponding permutation operations are carried out on the matrix R as well.

Step 4. The leading element $b_{tt} = 1$ divides all the elements in the remaining $(m - t + 1) \times (m - t + 1)$ matrix. The Reduction Operation as stated in step 4, section 5.2 is now carried out on the matrix B and also on matrix R. At the end of this step in the t^{th} cycle the matrix B and R are of the form displayed in Tableau 2 and Tableau 3. Set $t = t+1$, if $t < m$ goto step 1.

Exit If $b_{mm} < 0$ then set $b_{mm} = -b_{mm}$. Now set $\delta_m = \delta_{m-1} . b_{mm}$ Smith Normal Form for B is now obtained.

Not updated

$$
B=\left[\begin{array}{cccc|ccc}
\delta_1 & & \underline{0} & & & & \\
& \delta_2 & & & & \underline{0} & \\
\underline{0} & & \ddots & & & & \\
& & & \delta_t & & & \\
\hline
& & & & b_{t+1,t+1} & \cdots & b_{t+1m} \\
& \underline{0} & & & \vdots & & \vdots \\
& & & & b_{m,t+1} & \cdots & b_{mm}
\end{array}\right]
\qquad
R=\left[\begin{array}{c|ccc}
\text{Not updated} & & \text{Not updated} & \\
\hline
\text{Not updated} & (r_1,\ldots r_k)_{t+1,t+1} & \cdots & (r_1,\ldots r_k)_{t+1,m} \\
& \vdots & & \vdots \\
& (r_1,\ldots r_k)_{m,t+1} & \cdots & (r_1,\ldots r_k)_{mm}
\end{array}\right]
$$

(Tableau 2) (Tableau 3)

'Auxiliary Sequence'

In this sequence in a series of steps one of the elements of the remaining $(m - t + 1) \times (m - t + 1)$ matrix is made equal to one.

Step 1. Search for a set S of minimum cardinality such that its elements are relatively prime. If all the elements of S are in the same row or same column then goto step 2. Otherwise goto step 3.

Step 2. By integer linear combinations of the elements in S (all in one row, or in one column) an element of magnitude one is generated. Goto step 4.

Step 3. Construct a square submatrix of minimum order in which the elements of the chosen set appear. Applying to this submatrix the relevant steps of the general algorithm (section 5.2) make the leading element unity. Note that the transformation in this step must be applied to the full rows and columns of the remaining matrix, the elements of the submatrix being used only to generate the transformation matrix.

Step 4. Return to the calling step.

An Example.

Consider the integer matrix

$$B = \begin{bmatrix} 2 & 0 & 2 \\ 2 & 0 & -4 \\ -12 & 12 & 12 \end{bmatrix}, \tag{20}$$

the corresponding R is

$$R = \begin{bmatrix} (0,2,2) & (0,0,0) & (0,2,2) \\ (0,2,2) & (0,0,0) & (0,2,1) \\ (0,0,3) & (0,0,2) & (0,0,2) \end{bmatrix}.$$

Set $\delta_o = 1$, $t = 1$.

All the first components of the chinese representation are zero; therefore 2 is a common factor. Taking this out of B, and R matrix (for the latter operation see appendix 1) it follows,

$$B = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & -2 \\ -6 & 6 & 6 \end{bmatrix}, R = \begin{bmatrix} (1,1,1) & (0,0,0) & (1,1,1) \\ (1,1,1) & (0,0,0) & (0,1,2) \\ (0,0,4) & (0,0,1) & (0,0,1) \end{bmatrix}. \tag{21}$$

From the entries of the matrix R it is obvious that the corresponding elements in B are relatively prime, hence

$$d = 1 \times 2 = 2 \; ,$$

and (22)

$$\delta_1 = \delta_o . d = 1 \times 2 = 2 \; .$$

The matrix B is reduced to

$$B = \begin{bmatrix} \delta_1 & 0 & 0 \\ 0 & 0 & -3 \\ 0 & 6 & 12 \end{bmatrix}$$

and the corresponding R becomes (23)

not updated

$$R = \left[\begin{array}{c|cc} & & \\ \hline & (0,0,0) & (1,0,2) \\ & (0,0,1) & (0,0,2) \end{array}\right] .$$

Set t = 2.
From R in (23) it is again deduced that 3 is a common factor for the entries in the remaining matrix B. Taking out this common factor the remaining matrix in B and R become

$$\begin{bmatrix} 0 & -1 \\ 2 & 4 \end{bmatrix} , \quad \begin{bmatrix} (0,0,0) & (1,2,4) \\ (0,2,2) & (0,1,4) \end{bmatrix} . \qquad (24)$$

The elements of this matrix are relatively prime, therefore

$$d = 3 \times 1 = 3 \; ,$$

and $\delta_2 = \delta_1 . d = 2 \times 3 = 6$. (25)

Reducing B again, R is not considered further, the matrix

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \tag{26}$$

is obtained.

Set t = 3 ; since $b_{33} = 2 > 0$, δ_3 is computed as,

$$\delta_3 = \delta_2 . 2 = 12 ,$$

and the required Smith Normal Form

$$\Delta = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 12 \end{bmatrix}, \tag{27}$$

is obtained.

5.5 Some Comments on the Computational Implications.

The algorithm stated in section 5.4 transforms an integer matrix to its Smith Normal Form, exactly in (m-1) iterations. The diagonal element δ_i is known at the beginning of the ith iteration, therefore the reduction operation is carried out only once in this iteration. This is in contrast with the repeated application of the reduction step in the general algorithm stated in section 5.2. For integers set out in the chinese representation, formal addition (subtraction), multiplication operations are replaced by their corresponding look up tables, see appendix 5.1. The reduction operation of the matrix R is therefore carried out only by look up of these operation tables.

At the reduction step of any algorithm used to obtain the Smith Normal Form it is necessary to compute the gcf of the set of elements of a matrix. The chinese representation used by the authors prove to be of advantage in that:

(i) whenever the gcf is unity this is established immediately from the representation,

(ii) otherwise the common factors which multiply to produce the gcf are obtained immediately from the representation.

The number of operations by which the leading element is generated has an upper-bound of $\phi(D)$ where ϕ is a monotone increasing function of D the determinant of the matrix, see T.C. Hu [5.2], p379. Since the gcf d is taken out at the reduction step the determinant D reduces to $D/d^{(m-t+1)}$ therefore the number of operations are expected to reduce in relation to this upperbound.

The algorithm has been programmed by the author in Fortan IV and has been used to put the optimal bases of all the problems in Haldi [5.8] to their Smith Normal Form.

Appendix 5.1.

An algorithm which exploits the chinese representation of integers to obtain the gcf of a set of positive integers (not relatively prime) is described in this appendix.

Consider a set of positive integers $n_1, n_2 \ldots n_p$ expressed in their chinese form as:

$(r_1,r_2,\ldots r_k)_1$, $(r_1,r_2,\ldots r_4)_2$ ... $(r_1,r_2,\ldots r_k)_p$ respectively.

The steps of the algorithm to obtain the gcf d are set out below.

Step 0. Set d = 1.

Step 1. If the ith components ($1 \le i \le k$) of the representations of integers are zero for all the integers i.e.

$$(r_i)_j = 0 \ , \quad \text{for } j = 1, 2, \ldots p \ , \tag{28}$$

then goto step 5.

Step 2. Let $s = \min\{n_1,n_2\ldots n_p\}$. Find s and decompose s into its prime factors such that $s = p_1^{\alpha_1}.p_2^{\alpha_2}\ldots p_k^{\alpha_k}\ldots p_l^{\alpha_l}$, where some α_i may be zero. If s is discovered to be a prime go to step 4.

Step 3. If the integer part of $\sqrt{s}$ is less than p_k then the set of integers are relatively prime goto exit.

Step 4. For $i = k+1,k+2,\ldots l$ and $\alpha_i \ge 1$ determine if p_i is a common factor of the set of integers $\{n_1,n_2\ldots n_p\}$. If yes goto step 5 else goto exit.

Step 5. The prime p_i is a common factor of the set of integers $n_1,n_2\ldots n_p$. Divide to obtain n'_j ,

$$\left.\begin{array}{l} n'_j = (n_j/p_i) \\ \text{and update the corresponding chinese} \\ \text{representation } (r'_1, r'_2 \ldots r'_k)_j \quad ; \end{array}\right\} j = 1,2,\ldots p \tag{29}$$

The updating of the chinese representation is described in the Note which follows.

Set $d' = d \times p_i$;

Update $n_j = n_j'$ $j = 1, 2 \ldots p$ (30)

$(r_1, r_2, \ldots r_k)_j = (r_1', r_2', \ldots r_k')_j$

and $d = d'$,

and goto step 1.

Exit   d is now the gcf of the original set of p integers.

Note: Given an integer n_j, one of its factors p_i and the chinese representations of n_j and p_i ,

$$n_j \sim (r_1, r_2, \ldots r_k) ,$$
$$p_i \sim (\Pi_1, \Pi_2, \ldots \Pi_k) ; \qquad (31)$$

the chinese representation of n_j'

$$n_j' = (n_j / p_i)$$
$$n_j' \sim (r_1', r_2', \ldots r_k')_j , \qquad (32)$$

can be obtained in a minimum number of divisions by the following method. It is assumed that a set of k multiplication tables $T_1, T_2, \ldots T_k$, of dimensions $p_1 \times p_1, p_2 \times p_2 \ldots p_k \times p_k$ corresponding to the prime numbers $p_1, p_2, \ldots p_k$ are available for this method. Obtain the ith component r_i' the remainder of the division operation of n_j' by p_i (one division). From the multiplicative relations

$$r_\ell' \times \Pi_\ell = r_\ell (\text{mod } p_\ell) , \quad \ell = 1, 2, \ldots p , \ \ell \neq i , \qquad (33)$$

obtain r_ℓ' by looking up table T_ℓ. Thus the conversion involves only one division operation.

The following example illustrates the method.

Let $n_j = 21 \sim (1,0,1)$

$p_2 = 3 \sim (1,0,3)$

Therefore $n_j' = n_j/p_2 = 7 \sim (r_1', r_2', r_3')$ to be obtained.

$r_1' \times 1 = 1 \bmod(2)$; from T_1 , $r_1' = 1$;

$r_3' \times 3 = 1 \bmod(5)$; from T_3 , $r_3' = 2$.

Further r_2' = remainder of $(7/3) = 1$

Therefore $n_j' = 7 \sim (1,1,2)$.

Multiplication Tables

$p_1 = 2$	0	1
0	0	0
1	0	1

Table T1

$p_2 = 3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Table T2

$p_3 = 5$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Table T3

Appendix 5.2.

The Theorem 5 stated in the text of this paper has been proposed by Garfinkel and Nemhauser[4.4]. This Theorem is proved in this appendix and forms the basis of the algorithm proposed by the author .

Proof: Let d be the gcf of b_{ij}, $i,j = 1, 2 \ldots m$, it is required to prove that $\delta_1 = d$. As d is the gcf it must divide any 'integer linear combination' of the elements of B. It follows from the operations by which δ_1 is obtained there exists a set of integer multipliers (hence the term 'integer linear combination') k_{ij} $(i,j = 1, 2 \ldots m)$ such that δ_1 is expressed as

$$\delta_1 = \sum_{i=1}^{m} \sum_{j=1}^{m} k_{ij} b_{ij} . \tag{34}$$

Therefore d divides δ_1 which implies,

$$d \le \delta_1 . \tag{35}$$

By back substitution it can be proved that δ_1 divides all b_{ij} , therefore δ_1 divides their gcf d. This implies that

$$\delta_1 \le d . \tag{36}$$

From (35) and (36) it is deduced that

$$\delta_1 = d .$$

References 5

5.1. Gomory, R.E. On the relation between integer and non-integer solutions to linear programs, Proc. Nat. Acad. Soc., U.S.A., 53(2), pp260-265.

5.2. Hu,T.C. Integer Programming and Network Flows, Addison-Wesley Publishing Company, 1970.

5.3. Jahanshahlou, G.R. and Mitra, G. Group Theory and its Applications in Mathematical Programming, Technical Report, STR/7, September 1975, Department of Statistics & O.R., Brunel University.

5.4. Garfinkel, R.S. and Nemhauser, G., Integer Programming. John Wiley & Sons, 1972.

5.5. Griffin, H. Elementary Theory of Numbers, McGraw-Hill, New York, 1954.

5.6. Lehmer, D.H. The Machine Tools of Combinatorics, in Applied Combinatorial Mathematics, edited by F. Beckenbach, John Wiley & Sons, New York, 1964.

5.7. Bradley, G.H. Equivalent Integer Programs and Canonical Problems, Management Science, 17, pp354-366.

5.8. Haldi, J. 25 Integer Programming Test Problems, Working Paper No.43, Graduate School of Business, Stanford University, December 1964.

CHAPTER SIX

Hybrid Gradient and Simplex Method for the Solution of Linear Program

6.0 Summary

A mixture of gradient and simplex method is used to obtain an optimal solution to a linear programming problem. It seems that for some problems this method in contrast to the simplex method might arrive at the optimal solution with a fewer number of iterative simplex steps.

6.1 Introduction

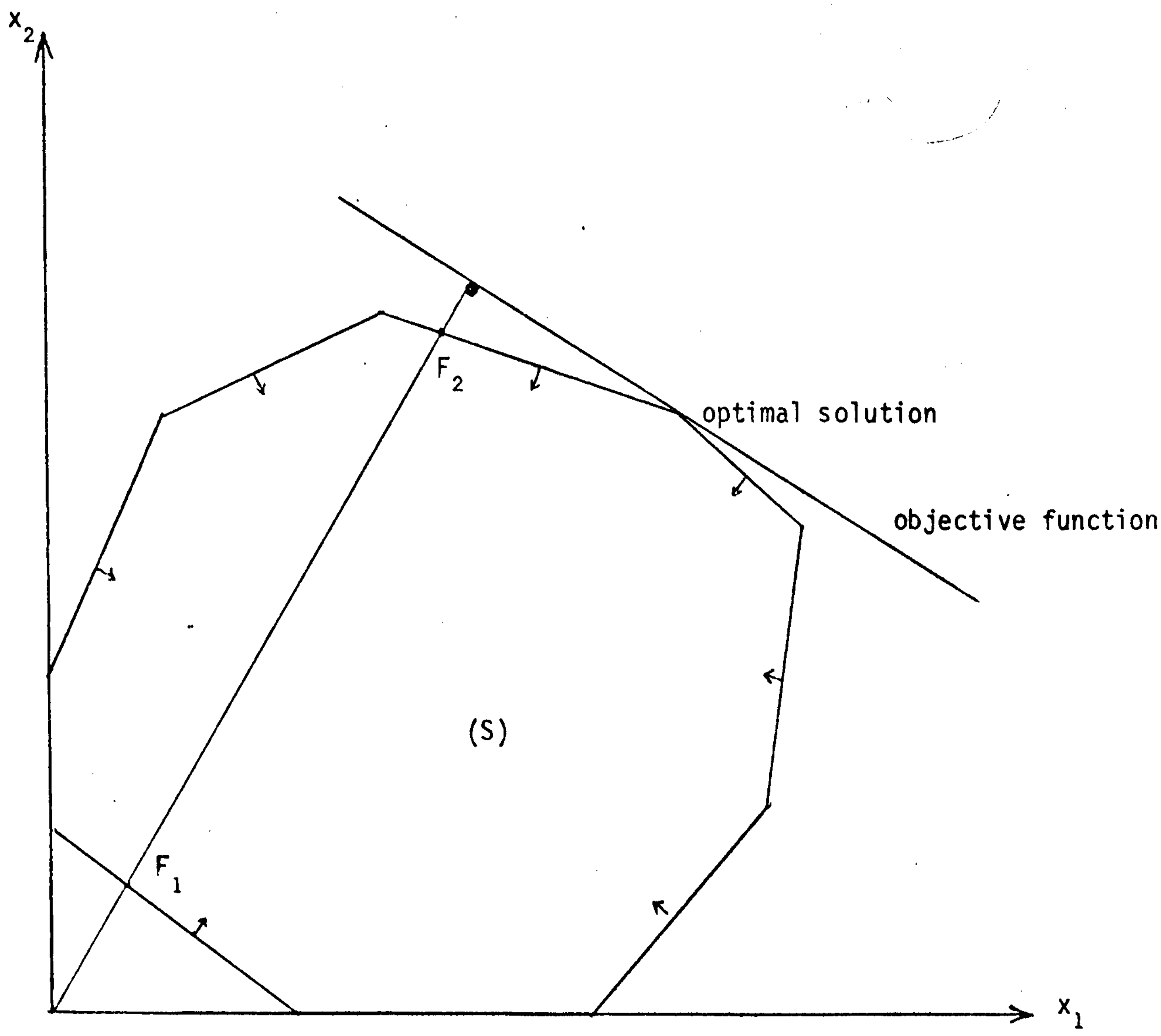
The simplex method is the most attractive and powerful method for solving a linear programming problem, and was developed by G.B. Dantzig. This is an iterative method which converges to an optimal solution in a finite number of iterations. The number of iterations depends on the number of constraints and on the number of variables.

To illustrate the idea underlying the present approach consider the linear programming problem shown graphically in Fig(1) and defined mathematically as:

$$\text{Max } z = c_1x_1 + c_2x_2$$

$$\text{subject to} \quad Ax \le b \qquad (1)$$

$$x \ge 0 \quad , \quad \text{where } x = (x_1, x_2)$$



Fig(1)

For simplicity assume that the vector $\vec{c} = (c_1, c_2) \geq 0$. The vector $\lambda \vec{c}$ (where λ is a scalar) is perpendicular to the hyperplane $c_1x_1 + c_2x_2$. Suppose for some $\lambda = \lambda_0, \lambda_0'$, the vectors $\lambda_0\vec{c}, \lambda_0' c$ cut the region S (region S is defined by the set of inequalities $Ax \leq b,\ x \geq 0$) at the points F_1 and F_2 . If F_2 is chosen as a starting point (note that the solutions corresponding to the point F_1 and F_2 are feasible but not necessarily basic) at most in two iterations the optimal solution is obtained.

In section 2 of this chapter the algorithm based on the above mentioned idea is described, and in section 3 an example is worked out by this algorithm. Section 4 contains a discussion on the possible ways of extending this algorithm.

6.2 Algorithms

Consider the general linear programming problem:

$$\text{Max } z = c_1x_1 + c_2x_2 + \dots + c_nx_n ,$$

subject to

$$\begin{aligned} &a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \leqslant b_i && (i = 1,\dots,p_1) \\ &a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geqslant b_i && (i = p_1 + 1,\dots,p_2) \qquad (2) \\ &a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i && i = p_2 + 1,\dots,m \\ &\quad x_j \geq 0 \quad (j = 1,\dots,n) \text{ , and} \end{aligned}$$

it is assumed that all the b's are non-negative.

After introducing slack, surplus, and artificial variables (2) can be written as:

$$\text{Max } z = \sum_{j=1}^{n} c_jx_j$$

subject to

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n + x_{n+i} = b_i & i = 1,\dots,p_1 \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n - x_{n+i} = b_i & i = p_1 + 1,\dots,p_2 \qquad (3) \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n + v_{i-p_2} = b_i & i = p_2 + 1,\dots,m \\ x_1,x_2,\dots,x_{n+p_2} \geq 0 \text{ , and } v_i-p_2 = 0 \text{ ,} & i = p_2+1,\dots,m . \end{cases}$$

Consider two cases:

CASE 1. $p_2 = m$ i.e., there is no equality constraint.

Define

$$P = \{j \mid c_j > 0\} \quad , \tag{4}$$

and set

$$x_j = \begin{cases} c_j t & \text{if } j \in P \\ 0 & \text{otherwise, } (j = 1,\ldots,n) \end{cases} \tag{5}$$

Substitution of these x's in (3) leads to the relation

$$\begin{aligned} x_{n+i} &= b_i - \left(\sum_{j=1}^{n} c_j a_{ij}\right)t & i &= 1,\ldots,p_1 \\ x_{n+i} &= \left(\sum_{j=1}^{n} c_j a_{ij}\right)t - b_i & i &= p_1+1,\ldots,m \; . \end{aligned} \tag{6}$$

Let

$$\alpha_i = \left(\sum_{j=1}^{n} c_j a_{ij}\right) \qquad i = 1,\ldots,m \quad , \tag{7}$$

whereby (6) can be expressed as

$$\begin{cases} x_{n+i} = b_i - \alpha_i t & i = 1,\ldots,p_1 \\ x_{n+i} = \alpha_i t - b_i & i = p_1+1,\ldots,m \; . \end{cases} \tag{8}$$

Let

$$Q = \{i \mid \alpha_i \neq 0\} \quad , \tag{9}$$

set $x_{n+i} = 0$ for all $i \in Q$, this leads to

$$t_i = \frac{b_i}{\alpha_i} \quad \text{for all } i \in Q \; .$$

Suppose

$$t_1^* = \text{Min}\ \{t_i > 0 \mid i \in (Q \quad \{1,2,\ldots,p_1\})\}$$
$$t_2^* = \text{Max}\ \{t_i > 0 \mid i \in (Q \quad \{p_1+1,\ldots,m\})\} \quad , \tag{10}$$

then obtain

$$\begin{cases} x^*_{n+i} = b_i - \alpha_i t_1^* \geq 0 & i = 1, \ldots, p_1 \\ x^*_{n+i} = \alpha_i t_2^* - b_i \geq 0 & i = p_1+1,\ldots,m \ . \end{cases} \tag{11}$$

If $t_1^* \geq t_2^* \geq 0$ it can be immediately deduced that the constraints are consistent and two feasible solutions may be constructed as

$$x^1 = (x_1^1,\ldots,x_n^1,b_1-\alpha_1 t_1^*,\ldots,b_{p_1}-\alpha_{p_1} t_1^*,\alpha_{p_1+1} t_1^*-b_{p_1+1},\ldots,\alpha_m t_1^*-b_m)$$

$$x^2 = (x_1^2,\ldots,x_n^2,b_1-\alpha_1 t_2^*,\ldots,b_{p_1}-\alpha_{p_1} t_2^*,\alpha_{p_1+1} t_2^*-b_{p_1+1},\ldots,\alpha_m t_2^*-b_m)$$

where (12)

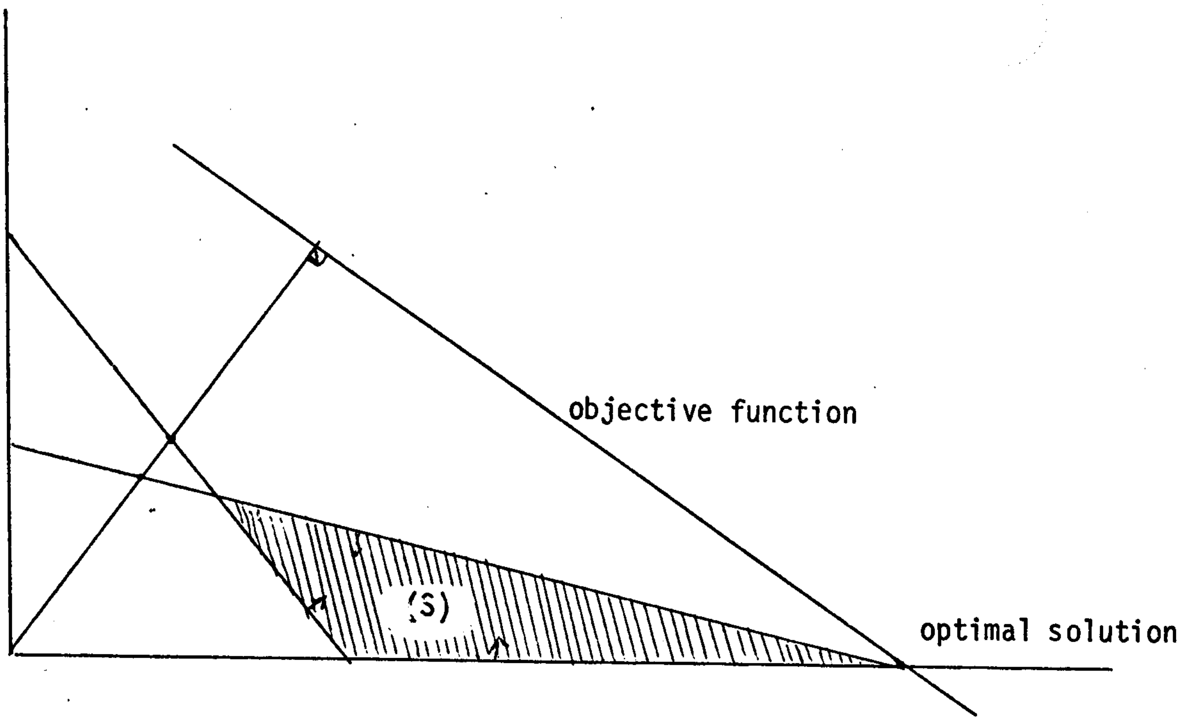
$$x_j^1 = \begin{cases} c_j t_1^* & \text{if } j \in P \\ 0 & \text{otherwise} \end{cases} \qquad x_j^2 = \begin{cases} c_j t_2^* & \text{if } j \in P \\ 0 & \text{otherwise} \end{cases}$$

and

$$\sum_{j=1}^{n} c_j x_j^1 \geq \sum_{j=1}^{n} c_j x_j^2 \tag{13}$$

It follows from these relations that x^1 is a feasible solution. Later on it is shown how a basic feasible solution may be obtained from this solution •

If $t_2^* > t_1^*$, it cannot be deduced that the constraints are not consistent.



Fig(2)

This is shown in Fig(2). Under these circumstances the problem is considered as CASE 2.

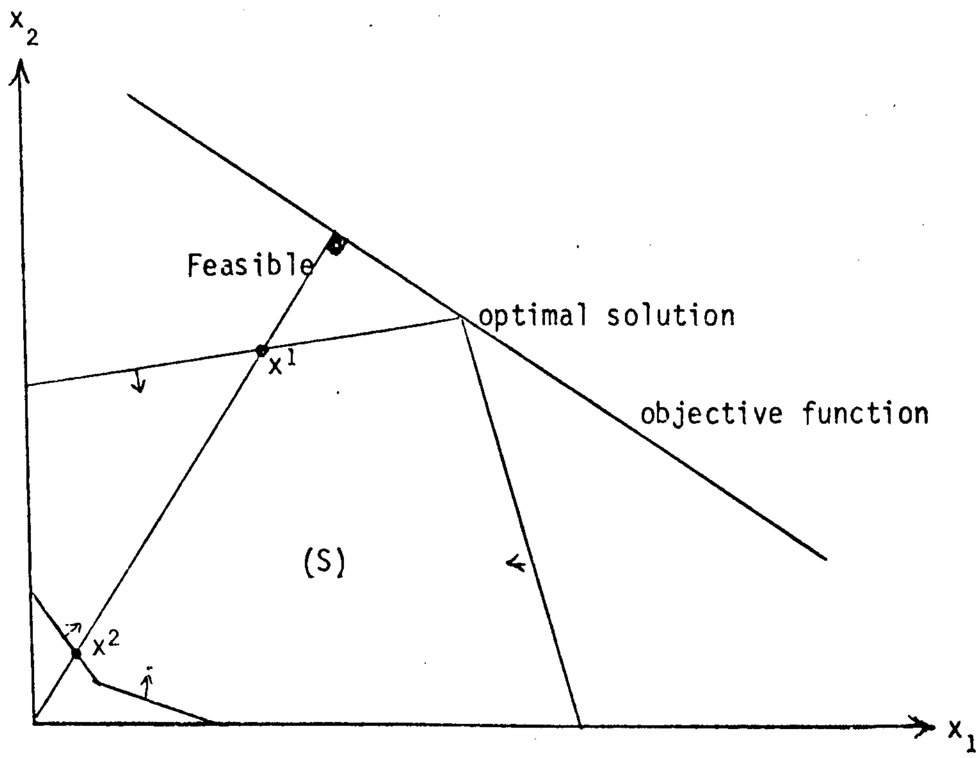
Example 1.

$$\text{Max } z = 2x_1 + 3x_2 \ ,$$

subject to

$$\begin{aligned} -x_1 + 3x_2 &\le 28 \ , \\ 3x_1 + x_2 &\le 54 \ , \\ x_1 + 3x_2 &\ge 6 \ , \\ 2x_1 + x_2 &\ge 4 \ , \\ x_1, x_2 &\ge 0 \ . \end{aligned} \tag{14}$$

Graphically this problem is shown in Fig(3).



Fig(3)

After introducing slack and surplus variables the problem may be written as:

$$\text{Max } z = 2x_1 + 3x_2 \quad ,$$

subject to

$$\begin{cases} -x_1 + 3x_2 + x_3 = 28 \\ 3x_1 + x_2 + x_4 = 54 \\ x_1 + 3x_2 - x_5 = 6 \\ 2x_1 + x_2 - x_6 = 4 \end{cases} \quad \text{or} \quad \begin{cases} x_3 = 28 - (-x_1 + 3x_2) \\ x_4 = 54 - (3x_1 + x_2) \\ x_5 = (x_1 + 3x_2) - 6 \\ x_6 = (2x_1 + x_2) - 4 \end{cases} \qquad (15)$$

$$x_i \geq 0 \quad i = 1,\ldots,6 \qquad\qquad x_i \geq 0 \quad i = 1,\ldots,6$$

By setting $x_1 = 2t \quad x_2 = 3t \quad p = \{1,2\}$, and substituting these in (15) gives

$$\begin{cases} x_3 = 28 - 7t \\ x_4 = 54 - 9t \\ x_5 = 11t - 6 \\ x_6 = 7t - 4 \end{cases} .$$

Putting $x_i = 0$ for $i = 3,4,5,6$ gives

$$t_1 = 28/7 = 4 \quad t_2 = 54/9 = 6 \quad t_3 = 6/11 , \quad t_4 = 4/7$$

$$t_1^* = \text{Min}\ \{4,6\} = 4$$

$$t_2^* = \text{Max}\ \{6/11, 4/7\} = 4/7$$

so $t_1^* > t_2^* \geq 0$, and the feasible solution which is chosen as the starting point is

$$x^1 = (8,12,0,18,38,24) , \qquad (16)$$

which is a feasible, but not basic solution. Later on it is shown how a basic feasible solution can be obtained from this feasible solution.

<u>CASE 2</u>. $p_2 < m$ i.e., there are some equality constraints. In this case an infeasibility form is introduced as:

$$w = v_1 + v_2 + \ldots + v_{m-p_2} - x_{n+p_1+1} - x_{n+p_1+2} - \ldots - x_{n+p_2} =$$

$$= \sum_{i=1}^{m-p_2} v_i - \sum_{i=1}^{p_2-p_1} x_{n+p_1+i} = \sum_{i=p_2+1}^{m} v_i - p_2 - \sum_{i=p_1+1}^{p_2} x_{n+i} =$$

$$= \sum_{i=p_2+1}^{m} [b_i-(a_{i1}x_1 + \ldots + a_{in}x_n)] + \sum_{i=p_1+1}^{p_2} [b_i-(a_{i1}x_1+\ldots+a_{in}x_n)]$$

$$= \sum_{i=p_1+1}^{m} [b_i - (a_{i1}x_1 + \ldots + a_{in}x_n)] =$$

$$= \sum_{i=p_1+1}^{m} b_i - \left(\sum_{i=p_1+1}^{m} a_{i1} \right) x_1 - \ldots - \left(\sum_{i=p_1+1}^{m} a_{in} \right) x_n \;, \quad \text{so}$$

$$w = \beta_0 - \beta_1 x_1 - \beta_2 x_2 - \ldots - \beta_n x_n \;, \tag{17}$$

where $\beta_j = \sum_{i=p_1+1}^{m} a_{ij} \quad (j = 0,1,\ldots,n)$

(17) can be written as:

$$-\beta_0 = -w - \beta_1 x_1 - \beta_2 x_2 - \ldots - \beta_n x_n \;. \tag{18}$$

Let

$$R = \{ j \mid \beta_j > 0 \;, \quad j = 1,\ldots,n \} \;, \tag{19}$$

introduce a parameter t, and set

$$x_j = \begin{cases} \beta_j t & \text{if } j \in R \\ 0 & \text{otherwise} \end{cases} \;, \quad (j = 1,\ldots,n) \;, \tag{20}$$

substituting these in (3) and solving the equations for x_{n+i} $(i = 1,\ldots,p_2)$, and v_i $(i = 1,\ldots,m-p_2)$ the following is obtained

$$\begin{cases} x_{n+i} = b_i - (\sum_{j=1}^{n} a_{ij}\beta_j)t & (i = 1,\ldots,p_1) \\ x_{n+i} = (\sum_{j=1}^{n} a_{ij}\beta_j)t - b_i & (i = p_1+1,\ldots,p_2) \\ v_{i-p_2} = b_i - (\sum_{j=1}^{n} a_{ij}\beta_j)t & (i = p_2+1,\ldots,m) \end{cases} \tag{21}$$

or (21) may be written as:

$$\begin{cases} x_{n+i} = b_i - \delta_i t & (i = 1,\ldots,p_1) \ , \\ x_{n+i} = \delta_i t - b_i & (i = p_1+1,\ldots,p_2) \ , \\ v_{i-p_2} = b_i - \delta_i t & (i = p_2+1,\ldots,m) \ , \end{cases} \tag{22}$$

where $\delta_i = (\sum_{j=1}^{n} a_{ij}\beta_j)$, $(i = 1,\ldots,m)$.

Let

$$T = \{i \mid \delta_i \neq 0 \ , \quad i = 1,\ldots,m\} \tag{23}$$

set x_{n+i}, v_{i-p_2} equal zero for those $i \in T$ and solve the equations for t, which gives

$$t_i = \frac{b_i}{\delta_i} \quad \text{for all } i \in T \tag{24}$$

t* is chosen as:

$$t^* = \min \{t_i \mid t_i > 0 \ \text{ and } i \in T\}$$

substituting t* in (22) gives

$$\begin{cases} x_{n+i} = \gamma_{n+i} = b_i - \beta_i t^* & i = 1,\ldots,p_1 \\ x_{n+i} = \gamma_{n+i} = \beta_i t^* - b_i & i = p_1+1,\ldots,p_2 \\ v_{i-p_2} = \gamma_{n+i} = b_i - \beta_i t^* & i = p_2+1,\ldots,m \\ x_i = \gamma_i = \begin{cases} c_i t^* & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases} \ , \end{cases} \tag{25}$$

and

$$w = M_1 = \beta_0 - \beta_1\gamma_1 - \beta_2\gamma_2 - \beta_3\gamma_3 - \dots - \beta_n\gamma_n$$
$$z = M_2 = c_1\gamma_1 + c_2\gamma_2 + \dots + c_n\gamma_n \quad . \qquad (26)$$

In tableau representation this is shown in (Tableau -0).

Note. The elements of the ℓth tableau are denoted with superscript ℓ .

F-rule

Initial step set $\ell = 1$

Step 0. Choose a column, say j_0 , such that $\gamma_{j_0}^{\ell} > 0$ and $-\beta_{j_0}^{\ell} > 0$ and x_{j_0} is not a basic variable and go to step 4. If no such column exists go to step 1.

Step 1. Choose a column, say j_0 , such that

$$(-\beta_{j_0}^{\ell}) = \min\{(-\beta_{j_0}^{\ell}) \mid (-\beta_{j_0}^{\ell}) < 0\} \qquad (27)$$

go to step 2. If no such column exists go to step 6.

Step 2. For finding pivot row carry out ratio test as

$$\frac{\gamma_{r_{i_0}}^{\ell}}{a_{i_0 j_0}^{\ell}} = \min\left\{\min\left\{\frac{\gamma_{r_i}^{\ell}}{a_{ij_0}^{\ell} > 0}, \gamma_{r_i}^{\ell} \geq 0\right\}, \min\left\{\frac{\gamma_{r_i}^{\ell}}{a_{ij_0}^{\ell}}, \gamma_{r_i}^{\ell} \leq 0, a_{ij_0}^{\ell} < 0\right\}\right\} . \qquad (28)$$

Choose the i_0th row as a pivot row, do pivotal transformation, set $\gamma_{r_{i_0}}^{\ell+1} = \gamma_{j_0}^{\ell} + \dfrac{\gamma_{r_{i_0}}^{\ell}}{a_{i_0 j_0}^{\ell}}$, update all the entries of the $(\ell+1)^{th}$ tableau, set $\ell = \ell+1$, go to step 3.

Step 3. If $w = 0$ go to step 7, otherwise go to step 0.

	1	γ_1 x_1		γ_n x_n	x_{n+1}		x_{n+p_1}	x_{n+p_1+1}		x_{n+p_2}	v_1		v_{m-p_2}	z	w
$-w$	M_1	$-\beta_1$	$\cdots$	$-\beta_n$	0	$\cdots$	0	0	$\cdots$	0	0	$\cdots$	0	0	1
z	M_2	$-c_1$	$\cdots$	$-c_n$			0	0		0	0		0	1	
x_{n+1}	γ_{n+1}	a_{11}	$\cdots$	a_{1n}	1		0	0		0	0		0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	0	$\ddots$									
x_{n+p_1}	γ_{n+p_1}	$a_{n+p_1\ 1}$		$a_{n+p_1\ n}$	0		1	0		0	0		0	0	0
x_{n+p_1+1}	γ_{n+p_1+1}	$a_{n+p_1+1\ 1}$		$a_{n+p_1+1\ n}$				-1					0	0	0
x_{n+p_2}	γ_{n+p_2}	$a_{n+p_2\ 1}$		$a_{n+p_2\ n}$						-1			0	0	
v_1	γ_{n+p_2+1}	$a_{n+p_2+1\ 1}$		$a_{n+p_2+1\ n}$	0		0	0			1		0	0	0
												$\ddots$			
v_{m-p_m}	γ_{m+n}	a_{m1}		a_{mn}	0		0	0		0	0		1	0	0

(Tableau -0)

Step 4. As $(-\beta_{j_0}^{\ell}) > 0$, therefore by decreasing x_{j_0}, $-w$ can be increased. Let

$$r = \min_i \left\{ \min \left\{ \frac{\gamma_{r_i}^{\ell}}{|a_{ij_0}^{\ell}|},\ a_{ij_0}^{\ell} < 0,\ \gamma_{r_i}^{\ell} \geq 0 \right\}, \gamma_{j_0}^{\ell} \right\} . \qquad (29)$$

If $r = \gamma_{j_0}^{\ell}$, set

$$\gamma_{j_0}^{\ell+1} = 0, \text{ and } \gamma_{r_i}^{\ell+1} = \gamma_{r_i}^{\ell} + r a_{ij_0}^{\ell} \quad (i = 1,\ldots,m)$$

$$M_1^{\ell+1} = M_1^{\ell} + (-\beta_{j_0}^{\ell}) r,\ M_2^{\ell+1} = M_2^{\ell} + (-c_{j_0}^{\ell}) r ,$$

all the other entries of the $(\ell+1)^{th}$ tableau are the same as ℓth tableau, set $\ell = \ell+1$, go to step 0.

If $r = \dfrac{\gamma_{r_{i_0}}^{\ell}}{|a_{i_0 j_0}^{\ell}|}$ for some $i = i_0$, then set

$$\gamma_{j_0}^{\ell+1} = \gamma_j^{\ell} - r \text{ and } \gamma_{r_i}^{\ell+1} = \gamma_{r_i}^{\ell} + r a_{ij_0}^{\ell} \quad (i = 1,\ldots,m) ,$$

all the other entries of the $(\ell+1)^{th}$ tableau are the same as ℓth tableau, set $\ell = \ell+1$, go to step 5.

Step 5. It follows from the above operation that for $i = i_0$, $\gamma_{r_{i_0}}^{\ell} = 0$ and $a_{i_0 j_0}^{\ell} < 0$, choose $a_{i_0 j_0}^{\ell}$ as the pivot element, carry out a pivotal transformation, set $\gamma_{r_{i_0}}^{\ell+1} = \gamma_{j_0}^{\ell}$, update all the element of the $(\ell+1)^{th}$ tableau, set $\ell = \ell+1$, go to step 0.

Step 6. If $w \neq 0$, problem has no feasible solution, go to step 8.

Step 7. The present representation contains a feasible solution.

Step 8. Stop.

B-rule which can be applied to get a basic feasible solution from a given feasible solution is discussed now. B-rule in some way is similar to F-rule.

B-rule

Initial step. Set $\ell = 1$ go to step 0.

Step 0. Choose a column, say j_0 , such that, $\gamma_{j_0}^{\ell} > 0$ and $(-c_{j_0}^{\ell}) > 0$ and x_{j_0} is a nonbasic variable, go to step 4. If no such column exists go to step 1.

Step 1. Choose a column, say j_0, such that $\gamma_{j_0}^{\ell} > 0$ and $(-c_{j_0}^{\ell}) = \min_j \left\{(-c_j^{\ell}) \mid (-c_j^{\ell}) < 0 \text{ , } \gamma_j^{\ell} > 0\right\}$, go to step 2. If no such column exists go to step 6.

Step 2. Do ratio test for finding the pivot row as usual, i.e.

$$\frac{\gamma_{r_{i_0}}^{\ell}}{a_{i_0 j_0}^{\ell}} = \min \left\{ \frac{\gamma_{r_i}^{\ell}}{a_{ij_0}^{\ell}} \text{ , } a_{ij_0}^{\ell} > 0 \right\} , \tag{30}$$

if all $a_{ij_0}^{\ell} \leq 0$ then the problem is unbounded, go to step 7, otherwise choose i_0th row as a pivot row, carry out pivotal transformation update all the entries of the $(\ell+1)^{th}$ tableau,

set $\gamma_{r_{i_0}}^{\ell+1} = \gamma_{j_0}^{\ell} + \gamma_{r_{i_0}}^{\ell} / a_{i_0 j_0}^{\ell}$, set $\ell = \ell+1$, go to step 3.

Step 3. If all the nonbasic variables are zero, go to step 6, otherwise go to step 0.

Step 4. As $(-c_j^{\ell}) > 0$, therefore, by decreasing x_{j_0}, the objective function can be increased, let

$$r = \min \left\{ \min_i \left\{ \frac{\gamma_{r_i}^{\ell}}{|a_{ij_0}^{\ell}|} \text{ , } a_{ij_0}^{\ell} < 0 \right\} \text{ , } \gamma_{j_0}^{\ell} \right\} . \tag{31}$$

If $r = \gamma_{j_0}^{\ell}$, then set, $M_2^{\ell+1} = M_2^{\ell} + r(-c_{ij_0}^{\ell})$, $\gamma_{j_0}^{\ell+1} = 0$,

$\gamma_{r_i}^{\ell+1} = \gamma_{r_i} + ra_{ij_0}^{\ell}$, $i = 1,2,\ldots,m$ and all the other entries

of the $(\ell+1)^{th}$ tableau are the same as the ℓth tableau, set
$\ell = \ell+1$, go to step 0.

If $r = \dfrac{\gamma_{r_{i_0}}^{\ell}}{a_{i_0j_0}^{\ell}}$, for some $i = i_0$, set $\gamma_{j_0}^{\ell+1} = \gamma_{j_0}^{\ell} - r$, and

$\gamma_{r_i}^{\ell+1} = \gamma_{r_i}^{\ell} + ra_{i_0j_0}^{\ell}$, $i = 1,\ldots,m$, $M_2^{\ell+1} = M_2^{\ell} + r(-c_{ij_0}^{\ell})$,

and all the other entries of the $(\ell+1)^{th}$ tableau is the same as ℓth tableau, set $\ell = \ell+1$, go to step 5.

Step 5. It follows from the above operation that for $i = i_0, \gamma_{r_{i_0}}^{\ell} = 0$, choose $a_{i_0j_0}^{\ell} < 0$ as a pivotal element, carry out the pivotal transformation, update all the element of the $(\ell+1)^{th}$ tableau, set $\gamma_{r_{i_0}}^{\ell} = \gamma_{j_0}^{\ell}$, and

$\ell = \ell+1$, go to step 0.

Step 6. The solution is a basic feasible solution moving from a feasible vertex in the steepest direction to increase the objective function.

In the tableau containing a basic feasible solution. Let

$$L = \{j \mid c_j^{\ell} > 0 \text{ for some } j \quad 1 \le j \le m+n\}$$

$$K = \{j \mid x_j \text{ is nonbasic variable}\}$$

set

$$x_j = \begin{cases} c_j^{\ell}t & \text{if } j \in L \cap K \\ 0 & \text{otherwise} \end{cases}$$

where t is a parameter as in (20). Substitute these x's into the equations obtained from the corresponding tableau, carry out the operations as defined in (22), (23), (24) and (25). This gives a solution to the equations obtained from tableau.

Now the steps of the algorithm may be stated as follows:

Step 0. If $p_2 = m$, use CASE 1 to obtain a solution, if the solution is basic feasible go to step 4. If it is feasible but not basic go to step 1. If the solution is not feasible go to step 2.

Step 1. Apply B-rule, if a basic feasible solution can be obtained go to step 4; otherwise corresponding to the unbounded exist of the B-rule go to step 6.

Step 2. Use CASE 2 if a basic feasible solution is obtained go to step 3. If the solution is feasible, but not basic, go to step 1. If solution is not feasible go to step 3.

Step 3. Apply F-rule, if a feasible solution is obtained go to step 4; otherwise go to step 8.

Step 4. If $-c_j^{\ell} \geq 0$ for $j = 1,\ldots,m+n$, go to step 7, otherwise go to step 5.

Step 5. Move from the given feasible vertex in the steepest direction to increase the objective function, whereby you get an improved solution and go to step 1.

Step 6. Problem is unbounded go to step 9.

Step 7. The corresponding tableau contains an optimal solution go to step 9.

Step 8. The problem has no feasible solution go to step 9.

Step 9. Stop.

6.3 Example

$$\text{Max } z = 5x_1 + 16x_2$$

subject to

$$2x_1 + x_2 \le 10$$

$$x_1 + 2x_2 \le 10$$

$$4x_1 - 2x_2 \ge 1$$

$$-2x_1 + 4x_2 \ge 1$$

$$x_1, x_2 \ge 0$$

By introducing negative slack, and slack variables, the problem may be rewritten as:

$$\text{Max } z = 5x_1 + 16x_2 \text{ ,}$$

subject to

$$\begin{cases} 2x_1 + x_2 + x_3 = 10 \\ x_1 + 2x_2 + x_4 = 10 \\ 4x_1 - 2x_2 - x_5 = 1 \\ -2x_1 + 4x_2 - x_6 = 1 \\ x_1, x_2, \dots, x_6 \ge 0 \end{cases} \qquad \begin{cases} x_3 = 10 - (2x_1 + x_2) \\ x_4 = 10 - (x_1 + 2x_2) \\ x_5 = -1 + (4x_1 - 2x_2) \\ x_6 = -1 + (-2x_1 + 4x_2) \\ x_1, \dots, x_6 \ge 0 \end{cases} \qquad \text{(a)}$$

It can be easily seen that CASE 1 cannot be applied, therefore the infeasibility form is introduced as:

$$w = -x_5 - x_6 = 1 - (4x_1 - 2x_2) + 1 - (-2x_1 + 4x_3) = 2 - 2x_1 - 2x_2$$

or $-2 = -w - 2x_1 - 2x_2$.

By substituting $x_1 = 2t$, $x_2 = 2t$ in (a) the following equations are obtained.

$$\begin{cases} x_3 = 10 - 6t \\ x_4 = 10 - 6t \\ x_5 = -1 + 4t \\ x_6 = -1 + 4t \end{cases} \qquad \text{(b)}$$

setting $x_i = 0$, $i = 3,\ldots,6$, gives

$$t_1 = t_2 = 5/3 , \quad t_3 = t_4 = 1/4$$

$$t^* = \text{mm}\,\{1/4, 5/3\} = 1/4 .$$

Substituting $t^* = 1/4$ gives the following values for x's

$$x_1 = x_2 = 1/2 , \quad x_3 = x_4 = 17/2 , \quad x_5 = x_6 = 0 , \qquad \text{(c)}$$

in the tableau form this may be written as

	1	x_1	x_2	x_3	x_4	x_5	x_6	z	w
-w	0	0	-2	-2	0	0	0	0	1
z	21/2	-5	-16	0	0	0	0	1	0
x_3	17/2	2	1	1	0	0	0	0	0
x_4	17/2	1	2	0	1	0	0	0	0
x_5	0	-4	2	0	0	1	0	0	0
x_6	0	2	-4	0	0	0	1	0	0

$x_1 = x_2 = \frac{1}{2}$

Tableau (6-0)

The tableau (6-0) contains a feasible solution, which is not basic. Now B-rule is applied to get a basic feasible solution. The related steps of this rule are carried out in tableau (6-1), and tableau (6-2).

	1	x_1	x_2	x_3	x_4	x_5	x_6	z
z	21/2	-37	0	0	0	8	0	1
x_3	17/2	4	0	1	0	-1/2	0	0
x_4	1/2	5	1	0	0	-1	0	0
x_2	1/2	-2	1	0	0	1/2	0	0
x_6	0	-6	0	0	0	2	1	0

$x_1 = \frac{1}{2}$

Tableau (6-1)

	1	x_1	x_2	x_3	x_4	x_5	x_6	z
z	21/2	0	0	0	0	-13/3	-37/6	1
x_3	17/2	0	0	1	0	5/6	+5/6	0
x_4	1/2	0	1	0	0	2/3	5/6	0
x_2	1/2	0	1	0	0	-1/6	-2/6	0
x_1	1/2	1	0	0	0	-1/3	-1/6	0

Tableau (6-2)

Tableau (6-2) contains a basic feasible solution, which is not optimum.

Now put $x_5 = \frac{13}{3} t \quad x_6 = \frac{37}{6} t$

By substituting these into the equations

$$\begin{cases} x_3 = 17/2 - (5/6x_5 + 4/6x_6) \\ x_4 = 17/2 - (2/3x_5 + 5/6x_6) \\ x_2 = 1/2 + (1/6x_5 + 2/6x_6) \\ x_1 = 1/2 + (1/3x_5 + 1/6x_6) \end{cases} \quad (c)$$

which are obtained from tableau (6-2),the following are deduced.

$$\begin{cases} x_3 = 17/2 - (5/6.13/3 + 4/6.37/6)t \\ x_4 = 17/2 - (2/3.13/3 + 5/6.37/6)t \\ x_2 = \ 1/2 + (-/6.13/3 + 2/6.37/6)t \\ x_1 = \ 1/2 + (1/3.13/3 + 1/6.37/6)t \ . \end{cases} \qquad (d)$$

By putting $x_i = 0$ for i = 1,2,3,4 in (d) one gets

$$t_1 = 1.1007 \qquad t_2 = 1.0588$$

substituting t* in (d) gives the following solution

$$x_1 = 3.1177,\ x_2 = 3.4411,\ x_3 = 0.3235,\ x_4 = 0,\ x_5 = 4.5882,\ x_6 = 6.5294,$$

which is a feasible solution to the problem. This solution in tableau form is represented as:

	1	x_1	x_2	x_3	x_4	x_5	x_6	z
z	70.646	0	0	0	0	-13/3	-37/6	1
x_3	0.3235	0	0	1	0	5/6	4/6	0
x_4	0.0	0	0	0	1	2/3	5/6	0
x_2	3.4411	0	1	0	0	-1/6	-2/6	0
x_1	3.1177	1	0	0	0	-1/3	-1/6	0

$x_5 = 4.5882$

$x_6 = 6.5294$

Tableau (6-3)

All the entries of the tableau (6-3) are the same as the tableau (6-2) except the values for x's. Now a pivotal transformation is carried out on the tableau (6-3) to make x_6 basic variable. This is shown in tableau (6-4).

	1	x_1	x_2	x_3	x_4	x_5	x_6	z
z	70.6461	0	0	0	37/5	3/5	0	1
x_3	0.3235	0	0	1	-4/5	3/10	0	0
x_6	6.5294	0	0	0	6/5	4/5	1	0
x_2	3.4411	0	1	0	+2/5	1/10	0	0
x_1	3.1177	1	0	0	2/5	-1/5	0	0

$x_5 = 4.5882$

Tableau (6-4)

By decreasing x_5 the objective function is increased. Consider

$$r = \min\{15.5885, 4.5882\} = 4.5882 \quad ,$$

therefore set $x_5 = 0$ and the optimal solution is

$$x_3 = 0.3235 + (4.5882)(0.3) = 1.7$$

$$x_6 = 6.5294 + (4.5882)(0.8) = 10.2$$

$$x_2 = 3.4411 + (4.5882)(0.1) = 3.9$$

$$x_1 = 3.1177 - (4.5882)(0.2) = 2.200 = 2.2$$

$$z = 73.4$$

6.4 Discussion

The ideas put forward by Hadley [1.3] and Zoutendijk [6.4] in applying gradient method to solve the mathematical programming problem, takes a simple form by mixing that idea with simplex method, and taking advantage of the structure of Linear programming problem. The preliminary investigation reported in this chapter leads to the following question

Is it possible to carry out the algorithm mentioned in this chapter in the context of product form, rather than tableau, which is used throughout?

A similar method may be developed to solve the quadratic programming problem or in general a convex programming problem.

References 6

6.1 Claude McMillan, Jr. Mathematical Programming. An Introduction to the Design and Application of Optimal Decision Machines, John Wiley & Sons Inc., 1970.

6.2 Lemke, C.E., The Constrained Gradient Method of Linear Programming, Journal of the Society for Industrial and Applied Mathematics, 9, 1961.

6.3 Zoutendijk, G., Maximizing a Function in a Convex Region, Journal of the Royal Statistics Society (B), 21, 1959.

6.4 ———————— Method of Feasible Directions, Amsterdam, Elsevier Publishing Company, 1960.

Appendix R1

This Appendix contains a FORTRAN program for finding all the vertices of a convex polyhedron S, using algorithm 1 in chapter 2. The set S is defined by the set of inequalities,

$$Ax \le b \ ,$$
$$x \ge 0 \ ,$$

and it is assumed that all the components of b are non-negative.

The Data Deck

To use the program, a data deck should be prepared as follows:

First Card. This card contains 3 values. These may be punched in whatever fashion the user desires, but FORMAT statement number 100 must be changed accordingly. The variable names into which these 3 data are read, and their purposes are as follows:

IW The number of rows of constraint equations

IZ The number of columns in (e) including the column of constants in the constraint

IY The number of real variable + 1; a real variable meaning variables in the set of inequalities $Ax \le b$, i.e., other than slack variables which are introduced to convert $Ax \le b$ into the set of equation

$$Ax + IU = b \qquad (e)$$

Second and Subsequent Cards: Onto the next set of cards the coefficient of the matrices including the constants defining the set of equations $Ax + IU = b$ are punched, and if necessary the FORMAT statement number 102 is changed in such a fashion that these data are read into the array.

D(M,N) M = 1 to IW and N = 1 to IZ as follows:

D(1,N) Holds the coefficient (elements) in the first row(thus the first constraint equation)

. ⋮ . . . .

D(IW,N) Holds the coefficient (elements) in the M^{th} row and final row (thus the final constraint equation).

Example problem.

It is required to find all the vertices defined by the set of inequalities

$$\begin{cases} 5x_1 + 3x_2 + x_3 \le 1050 \\ 4x_1 + 3x_2 + 2x_3 \le 1000 \\ x_1 + 2x_2 + 2x_3 \le 400 \\ x_1, x_2, x_3 \ge 0 \ . \end{cases} \qquad \text{(a)}$$

Adding slack variables x_4, x_5 and x_6 (a) may be written as:

$$\begin{cases} 5x_1 + 3x_2 + x_3 + x_4 = 1050 \\ 4x_1 + 3x_2 + 2x_3 + x_5 = 1000 \\ x_1 + 2x_2 + 2x_3 + x_6 = 400 \ . \end{cases} \qquad \text{(b)}$$

The data deck is prepared as follows:

```
 1.00     2.00     2.00     0.00     0.00     1.00  400.00
 4.00     3.00     2.00     0.00     1.00     0.00 1000.00
 5.00     3.00     1.00     1.00     0.00     0.00 1050.00
 3        7        4
```

FORTRAN STATEMENT

A scratch file is used in the program to write the generated tableaux for further reference.

```
      DEFINE      FILE10(220,500,U,KSVE)
      INTEGER       SMALL(200,30)  ,AR(30)
      DIMENSIOND(30,50),IBV(40),X(50)
      COMMON//KSVE,NSVE,LSVE,D,SMALL,N,IW,IZ,IY,K1,IBV,AR,X,IX
 100   FORMAT(3I6)
 102  FORMAT(10F8.2)
 103  FORMAT(1H1,40X,15HINITIAL TABLEAU)
 104  FORMAT(7X,12F8.1/(7X,12F8.1)/(7X,12F8.1))
 105  FORMAT(1H1,20X,22HCOORDINATE OF VERTICES)
 106  FORMAT(1X,2HX(,I3,2H)=,F10.4)
      READ(2,100)IW,IZ,IY
      IX=IZ-1
      DO      1        M=1,IW
 1    READ(2,102)(D(M,N),N=1,IZ)
      WRITE(3,103)
      DO      2        M=1,IW
 2    WRITE(3,104)(D(M,N),N=1,IZ)
      DO    3      N=IY,IX
      DO    4       L=1,IW
       IF(D(L,N).EQ.1)     GO    TO    6
 4    CONTINUE
 6     IBV(L)=N
 3     CONTINUE
      WRITE(3,105)
      DO        7        I=1,IW
 7    X(IBV(I))=D(I,IZ)
      DO       8       J=1,IY-1
 8    WRITE(3,106)J,X(J)
      DO          707            I=IY,IX
 707  SMALL(1,I-IY+1)=I
      K1=1
      LSVE,KSVE=1
      CALL   IOTAB
      NSVE=1
      N=0
 10    N=N+1
 300  LSVE=3
      K2=KSVE
      CALL   IOTAB
      KSVE=K2
 1000  IF(N-IX)114,114,15
 15   NSVE=NSVE+1
      N=0
      IF(NSVE-KSVE)80,80,81
 80   GO     TO     10
 81   GO    TO    800
 114   DO        9          I=1,     IW
 14   IF(N-IBV(I))9,10,9
 9     CONTINUE
 500  CALL PIVOTE(&10)
      GO     TO    10
 800  STOP
      END
```

```
      SUBROUTINE PIVOTE(*)
      INTEGER      SMALL(200,30)   ,AR(30)
      DIMENSIOND(30,50),IBV(40),X(50)
      COMMON//KSVE,NSVE,LSVE,D,SMALL,N,IW,IZ,IY,K1,IBV,AR,X,IX
      SNALL=999999.0
      DO     30      I=    1,    IW
       IF(D(I,N))30,30,400
400     QUALL=D(I,IZ)/D(I,N)
      IF(QUALL-SNALL)60,30,30
60    SNALL=QUALL
      KR=I
30    CONTINUE
      IF(KR)99,99,999
999   IBV(KR)=N
      DO         31       J=1,IW
31     AR(J)=IBV(J)
      CALL         ORIBV
      DO         34     K=1,K1-1
      DO       32    J=1,IW
      IF(SMALL(K,J)-SMALL(K1,J))34,32,34
32    CONTINUE
      GO     TO    33
34    CONTINUE
      GO     TO    40
33     K1=K1-1
99    RETURN1
40    BM=D(KR,N)
      DO     37    M=1,IW
      CRANK=D(M,N)
       DO         36     J=  1,    IZ
      IF(M-KR)135,37,135
135   RS=CRANK
36    D(M,J)=D(M,J)-(D(KR,J)/BM)*RS
37    CONTINUE
      DO       35    I=1,IZ
35     D(KR,I)=D(KR,I)/BM
       KSVE=KSVE+1
        LSVE=2
       DO           702      I=1,IX
702    X(I)=0.0
       DO      700     I=1,IW
700    X(IBV(I))=D(I,IZ)
      WRITE(3,107)K1
107   FORMAT(1H0,22HTHIS IS   THE VERTEX NO,I3)
      DO     803        I=1,IY-1
803   WRITE(3,110)       I,X(I)
110   FORMAT(2X,2HX(,I3,2H)=,F10.4)
      CALL   IOTAB
      CONTINUE
      RETURN 1
      END
```

```
      SUBROUTINE    IUTAB
      INTEGER       SMALL(200,30)   ,AR(30)
      DIMENSIOND(30,50),IBV(40),X(50)
      COMMON//KSVE,NSVE,LSVE,D,SMALL,N,IW,IZ,IY,K1,IBV,AR,X,IX
       IF(LSVE-2)60,60,62
60    WRITE(10'KSVE)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
       KSVE=KSVE-1
      RETURN
62    KSVE=NSVE
63    READ (10'KSVE)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      KSVE=KSVE-1
      RETURN
      END
      SUBROUTINE    ORIBV
      INTEGER       SMALL(200,30)   ,AR(30)
      DIMENSIOND(30,50),IBV(40),X(50)
      COMMON//KSVE,NSVE,LSVE,D,SMALL,N,IW,IZ,IY,K1,IBV,AR,X,IX
      K1=K1+1
      DO    204       I=1,IW
204   SMALL(K1,I)=100.
      DO      200    I=1,     IW
      DO    202    M=1,IW
      IF(AR(M)-SMALL(K1,I))  203,202,202
203   SMALL(K1,I)=AR(M)
      K3=M
202    CONTINUE
      AR(K3)=999
200    CONTINUE
      RETURN
       END
      FINISH
```

Appendix R2

In this Appendix two FORTRAN programs are presented. Program 1 solves the problem of finding all the vertices of a convex polyhedron S as defined in Appendix R1, via algorithm II in chapter 2. The data deck for this program is prepared exactly in the same way as that described in Appendix R1.

Program II is used to find all the vertices of a convex polyhedron S defined by

$$\begin{cases} Bv = b \\ v \geq 0 \end{cases}, \qquad \text{(a)}$$

via algorithm II in chapter 2.

The data deck for this program contains the following cards

First Card. This card contains 4 data values. These may be punched in whatever fashion one desires, but FORMAT statement number 104 in the SUBROUTINE SIMPLEX must be changed accordingly. The variable names into which these 4 values are read, and their purposes are as

IW The number of rows in the set of equations (a)

IZ The number of columns, including the columns corresponding to the artificial variables, which one introduced to get a feasible solution and the column associated with the constant in the right-hand side of (a)

IY The number of components of v_1 plus one. After introducing artificial variable (a) may be written in the form

$$Dv_1 + Iv_2 = b \qquad \text{(d)}$$

I30 The number of artificial variables express

Second card. This card contains only one datum : a1 or 0 in the first column. If one wishes to have the successive tableaux printed out as the iterative process progresses to get a basic feasible solution, 1 should be punched in the first column.

Third Card. In the third card the user punches the coefficients of the infeasibility form used to get a basic feasible solution to a FORMAT statement number 102 in the SUBROUTINE SIMPLEX may be modified accordingly for reading this into the array P(J), J = 1 to IZ-1.

Fourth and subsequent Cards. Onto these cards the coefficient of the equations (d) are punched as explained in Appendix R1.

Example Problem

Find all the vertices of a convex polyhedron defined by

$$\begin{cases} 2x_1 + 3x_2 + x_3 + x_4 = 2 \\ 3x_1 - 2x_2 + x_3 = 3 \\ 3x_1 + 4x_2 + 5x_3 + x_5 = 4 \\ x_1, x_2, \ldots, x_5 \geq 0 \end{cases}$$

By introducing x_6 as an artificial variable, the starting basic feasible solution is the optimum solution of the linear program

$$\text{Max } z = -999X6$$

subject to

$$2x_1 + 3x_2 + x_3 + x_4 = 2$$

$$3x_1 - 2x_2 + x_3 + x_6 = 3$$

$$3x_1 + 4x_2 + 5x_3 + x_5 = 4$$

$$x_i \geq 0 \quad i = 1, \ldots, 6 \quad .$$

The data deck are punched as

```
3.00   4.00   5.00   0.00   1.00   0.00    4.00
3.00  -2.00   1.00   0.00   0.00   1.00    3.00
2.00   3.00   1.00   1.00   0.00   0.00    2.00
0.00   0.00   0.00   0.00   0.00   0.00 -999.00

3      7      4      1
```

C FOR COMMENT | STATEMENT NUMBER | FORTRAN STATEMENT

PROGRAM I

```
      DEFINE   FILE10(200,500,U,KSVE)
      DIMENSION D(20,27),IBV(27),IBN(220,27),IBR(220,27),X(27)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
100    FORMAT(3I6)
102   FORMAT(10F8.2)
105   FORMAT(1H1,20X,22HCOORDINATE OF VERTICES)
      READ(2,100)IW,IZ,IY
      IX=IZ-1
      DO      1      M=1,IW
1     READ(2,102)(D(M,N),N=1,IZ)
      DO    3     N=IY,IX
      DO     4      L=1,IW
       IF(D(L,N).EQ.1)      GO     TO    6
4     CONTINUE
6      IBV(L)=N
3      CONTINGE
      WRITE(3,105)
      DO       9       I=1,100
      DO   10      J=1,IZ
10    IBN(I,J)=J
      DO       11       K=1,IW
11    IBR(I,K)=K
9     CONTINUE
      K5=0
      NSVE,KSVE,LSVE=1
      CALL   IOTAB
      N=0
12    LSVE=3
      N=K5
19    N=N+1
      K8=NSVE
      K2=KSVE
      CALL   IOTAB
      KSVE=K2
      DO    55    J=KSVE+1,200
      DO             25          I=1,IX
25    IBN( J     ,I)=IBN(NSVE,I)
55    CONTINUE
      IF(N-IX)14,14,15
15    NSVE=NSVE+1
      K8=NSVE
```

```
      N=0
      IF(NSVE-KSVE)17,17,18
17    GO    TO      19
14    IF(IBN(K8,N))19,20,20
20    DO     21        I=1,IW
      IF(N-IBV(I))21,19,21
21    CONTINUE
      KS=N
      KR=0
      SNALL=99999.0
      DO          50       I=1,IW
      IF(IBR(K8,I))50,50,31
31    IF(D(I,N))50,50,33
33    QUALL=D(I,IZ)/D(I,N)
      IF(QUALL-SNALL)60,50,50
60    SNALL=QUALL
      KR=I
50    CONTINUE
      IF(KR)19,19,70
70    CALL   PIVOTE(&12)
18    K10=KSVE
      LSVE=3
      NSVE=0
      DO          30                 I=1,K10
      NSVE=NSVE+1
      CALL          IOTAB
      WRITE(3,39)I
39    FORMAT(30X,22HTHIS IS THE TABLEAU NO,I3)
      WRITE(3,49)(IBN(I,J),J=1,IX)
49    FORMAT(8X,13I8)
      DO        40        M=1,IW
40    WRITE(3,41)IBR(I   ,M),IBV(M),(D(M,K),K=1,IZ)
41    FORMAT(1H0,1X,I3,2HX(,I2,1H),13F8.2/(9X,13F8.2)/(9X,13F8.2))
      DO         48        L=1,IX
48    X(L)=0.0
      DO        38        L=1,IW
38    X(IBV(L))=D(L,IZ)
      IY2=IY-1
      DO       42       L=1,IY2
42    WRITE(3,43)L,X(L)
43    FORMAT(2X,2HX(,I2,2H)=,F8.4)
30    CONTINUE
      STOP
      END
```

```
      SUBROUTINE PIVOTE(*)
      DIMENSION D(20,27),IBV(27),IBN(220,27),IBR(220,27),X(27)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
      N1=0
      KR=0
      K9=NSVE
44    SNALL=999999.0
      DO     30      I=1,IW
      IF(IBR(K9,I))30,30,31
31    IF(D(I,N))30,30,33
33    QUALL=D(I,IZ)/D(I,N)
      IF(QUALL-SNALL)60,30,30
60    SNALL=QUALL
      KR=I
30    CONTINUE
      IF(KR)99,99,51
99    IF(N1-IX)89,50,50
50    RETURN1
51    BM=D(KR,N)
      IBV(KR)=N
      N1=0
      KSVE=KSVE+1
      K10=KSVE
      DO       70       I=1,IW
70    IBR(KSVE,I)=IBR(K9,I)
      IBR(KSVE ,KR)=-KR
      IBN(K9,N)=-N
72    DO           37       I=1,IW
      CRANK=D(I,N)
      DO       36         J=1,IZ
      IF(I-KR)11,37,11
11    RS=CRANK
36    D(I,J)=D(I,J)-(D(KR,J)/BM)*RS
37    CONTINUE
      DO       32        I=1,IZ
32    D(KR,I)=D(KR,I)/BM
      WRITE(3,120)KSVE,NSVE,K10,K9
120   FORMAT(20X,4I8)
      KR=0
      LSVE=1
      CALL   IOTAB
      DO       71        I=1,IW
      IF(IBR(KSVE,I))71,80,80
71    CONTINUE
      RETURN
```

```
80    LSVE=3
89    N1=N1+1
      K3=NSVE
      NSVE=KSVE
      K4=KSVE
      CALL    IOTAB
      NSVE=K3
      KSVE=K4
      IF(N1-IX)84,84,85
85    RETURN
84    IF(IBN(K10,N1))89,90,90
90    DO    91    I=1,IW
      IF(N1-IBV(I))91,89,91
91    CONTINUE
      N=N1
      K9=KSVE
      GO   TO        44
      RETURN
      END
      SUBROUTINE    IOTAB
      DIMENSION D(20,27),IBV(27),IBN(220,27),IBR(220,27),X(27)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
       IF(LSVE-2)60,60,62
60    WRITE(10'KSVE)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
       KSVE=KSVE-1
      RETURN
62    KSVE=NSVE
63    READ (10'KSVE)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      KSVE=KSVE-1
      RETURN
      END
      FINISH
```

```
      MASTER         CONVEX      SET
      DEFINE  FILE10(220,500,U,K200),12(220,500,U,K100)
      DIMENSION    D(10,20),IBV(20),IBN(400,20),IBR(400,10),X(20)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
     1/AREA111/K100,K200
      CALL   SIMPLEX
      WRITE(3,105)
105   FORMAT(1H1,20X,22HCOORDINATE OF VERTICES)
      DO      9       I=1,100
      DO   10      J=1,IZ
10    IBN(I,J)=J
      DO        11       K=1,IW
11    IBR(I,K)=K
9     CONTINUE
      K5=0
      NSVE,KSVE,LSVE=1
      CALL  IOTAB
      N=0
12    LSVE=3
      N=K5
19    N=N+1
      K8=NSVE
      K2=KSVE
      CALL  IOTAB
      KSVE=K2
      DO    55    J=KSVE+1,400
      DO            25          I=1,IX
25    IBN( J     ,I)=IBN(NSVE,I)
55    CONTINUE
      IF(N-IX)14,14,15
15    NSVE=NSVE+1
      K8=NSVE
      N=0
      IF(NSVE-KSVE)17,17,18
17    GO    TO       19
14    IF(IBN(K8,N))19,20,20
20    DO      21        I=1,IW
      IF(N-IBV(I))21,19,21
21    CONTINUE
      K5=N
      KR=0
      SNALL=99999.0
      DO        50      I=1,IW
      IF(IBR(K8,I))50,50,31
31    IF(D(I,N))50,50,33
33    QUALL=D(I,IZ)/D(I,N)
      IF(QUALL-SNALL)60,50,50
60    SNALL=QUALL
      KR=I
50    CONTINUE
      IF(KR)19,19,70
70    CALL   PIVOTE(&12)
18    K10=KSVE
      LSVE=3
      NSVE=0
```

```
      DO        30              I=1,K10
      NSVE=NSVE+1
      CALL       IOTAB
      WRITE(3,39)I
 39   FORMAT(30X,22HTHIS IS THE TABLEAU NO,I3)
      WRITE(3,49)(IBN(I,J),J=1,IX)
 49   FORMAT(8X,13I8)
      DO      40      M=1,IW
 40   WRITE(3,41)IBR(I   ,M),IBV(M),(D(M,K),K=1,IZ)
 41   FORMAT(1H0,1X,I3,2HX(,I2,1H),13F8.2/(9X,13F8.2)/(9X,13F8.2))
      DO       48       L=1,IX
 48   X(L)=0.0
      DO      38      L=1,IW
 38   X(IBV(L))=D(L,IZ)
      IY2=IY-1
      DO     42     L=1,IY2
 42   WRITE(3,43)L,X(L)
 43   FORMAT(2X,2HX(,I2,2H)=,F8.4)
 30   CONTINUE
      STOP
      END
      SUBROUTINE PIVOTE(*)
      DIMENSION    D(10,20),IBV(20),IBN(400,20),IBR(400,10),X(20)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
     1/AREA111/K100,K200
      N1=0
      KR=0
      K9=NSVE
 44   SNALL=999999.0
      DO     30     I=1,IW
      IF(IBR(K9,I))30,30,31
 31   IF(D(I,N))30,30,33
 33   QUALL=D(I,IZ)/D(I,N)
      IF(QUALL-SNALL)60,30,30
 60   SNALL=QUALL
      KR=I
 30   CONTINUE
      IF(KR)99,99,51
 99   IF(N1-IX)89,50,50
 50   RETURN1
 51   BM=D(KR,N)
      IBV(KR)=N
      N1=0
      KSVE=KSVE+1
      K10=KSVE
      DO        70          I=1,IW
 70   IBR(KSVE,I)=IBR(K9,I)
      IBR(KSVE ,KR)=-KR
      IBN(K9,N)=-N
 72   DO          37      I=1,IW
      CRANK=D(I,N)
      DO      36          J=1,IZ
      IF(I-KR)11,37,11
 11   RS=CRANK
 36   D(I,J)=D(I,J)-(D(KR,J)/BM)*RS
```

```
37    CONTINUE
      DO      32       I=1,IZ
32    D(KR,I)=D(KR,I)/BM
      WRITE(3,120)KSVE,NSVE,K10,K9
120   FORMAT(20X,4I8)
      KR=0
      LSVE=1
      CALL   IOTAB
      DO      71       I=1,IW
      IF(IBR(KSVE,I))71,80,80
71    CONTINUE
      RETURN1
80    LSVE=3
89    N1=N1+1
      K3=NSVE
      NSVE=KSVE
      K4=KSVE
      CALL     IOTAB
      NSVE=K3
      KSVE=K4
      IF(N1-IX)84,84,85
85    RETURN1
84    IF(IBN(K10,N1))89,90,90
90    DO      91       I=1,IW
      IF(N1-IBV(I))91,89,91
91    CONTINUE
      N=N1
      K9=KSVE
      GO   TO          44
      RETURN
      END
      SUBROUTINE      IOTAB
      DIMENSION    D(10,20),IBV(20),IBN(400,20),IBR(400,10),X(20)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
     1/AREA111/K100,K200
       IF(LSVE-2)60,60,62
   60 IF(KSVE-220)100,100,101
  100 K200=KSVE
      WRITE(10'K200)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      RETURN
  101 K100=KSVE-220
      WRITE(12'K100)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      RETURN
62    KSVE=NSVE
      IF(KSVE-220)200,200,202
  200 K200=KSVE
      READ (10'K200)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      RETURN
  202 K100=KSVE-220
      READ (12'K100)(((D(M,I),I=1,IZ),IBV(M)),M=1,IW)
      RETURN
      END
```

```
      SUBROUTINE SIMPLEX
      DIMENSION    D(10,20),IBV(20),IBN(400,20),IBR(400,10),X(20)
     1,SC(40),P(40)
      COMMON//D,IBV,IBR,IBN,X,IX,IZ,IY,N,NSVE,LSVE,KSVE,K9,K8,N1,IW,K10
     1/AREA111/K100,K200
101   FORMAT (I1)
 104 FORMAT(4I4)
 102 FORMAT(20F4.0)
 103 FORMAT(20F4.0)
106   FORMAT (1H0,11HTABLEAU NO.,I6)
108   FORMAT (1H1,9H SOLUTION)
 109 FORMAT(1H0,8HVARIABLE,4X,5HVALUE)
110   FORMAT (1X,2HX(,I3,4H) = ,F12.2)
111   FORMAT (1H0,28H ALL OTHER VARIABLES = ZERO.)
112   FORMAT (1H1,21H THE INITIAL TABLEAU.)
113   FORMAT (11X,10F10.4/ (11X, 10F10.4))
 300 FORMAT(11X,10I10/(11X,10I10))
301   FORMAT (1H0,2X,2HX(,I2,1H),3X,10F10.3/ (11X, 10F10.3))
302   FORMAT(1H0,12H SIMPLEX  CR,10F10.3/ (11X,10F10.3))
305   FORMAT (1H0,9HOBJ FNCTN, 1X, 10F10.3/ (11X,10F10.3))
 789 FORMAT(1H0,10X,28HOBJECTIVE FUNCTION VALUE IS ,F15.5)
      READ(2,104)IW,IZ,IY,I30
      IX=IZ-1
      READ(2,101)ITAB
      READ(2,102)(P(J),J=1,IX)
      DO 15M=1,IW
 15   READ(2,103)  (D(M,N),N=1,IZ)
      WRITE(3,112)
      WRITE(3,305)(P(M),M=1,IX)
      DO 16 M=1,IW
 16   WRITE(3,113) (D(M,N),N=1,IZ)
      DO 20 N=IY,IX
      DO 30 L=1,IW
      IF(D(L,N).EQ.1.) GO TO   40
 30   CONTINUE
      GO  TO  20
 40   IBV(L)=N
 20   CONTINUE
      Z=0.
      DO 210 M=1,IW
      IBVM=IBV(M)
210   Z=Z+D(M,IZ)* P(IBVM)
      NOPIVS=0
      IF(ITAB.NE.1)GO TO 13
 13   SCMAX=0.
      DO 31 N=1,IX
      DO 32 I=1,IW
      IF(N.EQ.IBV(I)) GO TO 31
 32   CONTINUE
      SUM=0.
      DO 33 I=1,IW
      J=IBV(I)
 33   SUM=SUM+P(J)* D(I,N)
      SC(N)=P(N)-SUM
      IF(SC(N).LE.SCMAX)GO TO 31
      SCMAX=SC(N)
      IPIVCO=N
```

```
 31  CONTINUE
     DO 200 M=1,IW
     IBVM=IBV(M)
200  SC(IBVM)=0.
     IF(SCMAX.LE.0) GO TO 14
     NOPIVS=NOPIVS+1
     SMLVAL=999999.
     DO 4 M=1,IW
     IF(D(M,IPIVCO)) 4, 4, 5
  5  QUONT=D(M,IZ)/D(M,IPIVCO)
     IF(QUONT-SMLVAL) 6,4,4
  6  IPIVRO=M
     SMLVAL=QUONT
  4  CONTINUE
     IBV(IPIVRO)=IPIVCO
     DIV=D(IPIVRO,IPIVCO)
     DO 7 N=1,IZ
     CRANK=D(IPIVRO,N)
  7  D(IPIVRO,N)=CRANK/DIV
     IF(ITAB.NE.1) GO TO 12
     WRITE(6,302)  (SC(J),J=1,IX)
     N100=NOPIVS  +1
     WRITE(3,789) Z
     WRITE(3,106)N100
     WRITE(3,300)(N,N=1,IX)
 12  DO 10 M=1,IW
     IF(M-IPIVRO)9,8,9
  9  RM=-D(M,IPIVCO)
     DO 11 N=1,IZ
     BM=D(IPIVRO,N)*RM
     SINK=D(M,N)+BM
     D(M,N)=SINK
 11  CONTINUE
  8  IF(ITAB.NE.1) GO TO 10
     WRITE(3,301)IBV(M),(D(M,N),N=1,IZ)
 10  CONTINUE
     Z=Z+SMLVAL*SCMAX
     GO  TO  13
 14  WRITE(3,108)
     WRITE(3,109)
     DO 21 M=1,IW
 21  WRITE(3,110)IBV(M),D(M,IZ)
     WRITE(3,111)
     WRITE(3,789)  Z
     I31=IZ-I30
     IX=IX-I30
     DO    2777    I=1,IW
2777 D(I,I31)=D(I,IZ)
     IZ=IZ-I30
     RETURN
     END
     FINISH
```

Appendix R3

Two FORTRAN programs are described in this Appendix. In the first program Lemke's method is applied to the Fundamental Problem

$$\begin{cases} -Mz + IW = q \\ w,z \geq 0 \\ w^T z = 0 \end{cases} \qquad \text{(f)}$$

A data deck for this program is prepared as follows:

First Card. This contains two values. These may be punched in whatever fashion the user desires, but FORMAT statement number 100 must be changed accordingly. The variable names corresponding to these values are as follows:

IM The number of rows in the set of equations $-Mz + IW = q$.

IN The number of column of the matrix [M,I] plus two.

Second and subsequent cards. Onto the next set of cards the user punches the coefficient of the equation

$$MZ + IW + e'^T z_0 = q \quad , \qquad \text{(g)}$$

where e' = (-1,-1,...,-1). These should be in conformance with the FORMAT statement number 101, in such fashion these data are read into the array D(I,J), I = 1 to IM, and J = 1 to IN, as follows:

D(1,IN) Holdsthe coefficient (elements) in the first row (thus the first equation in (g)).

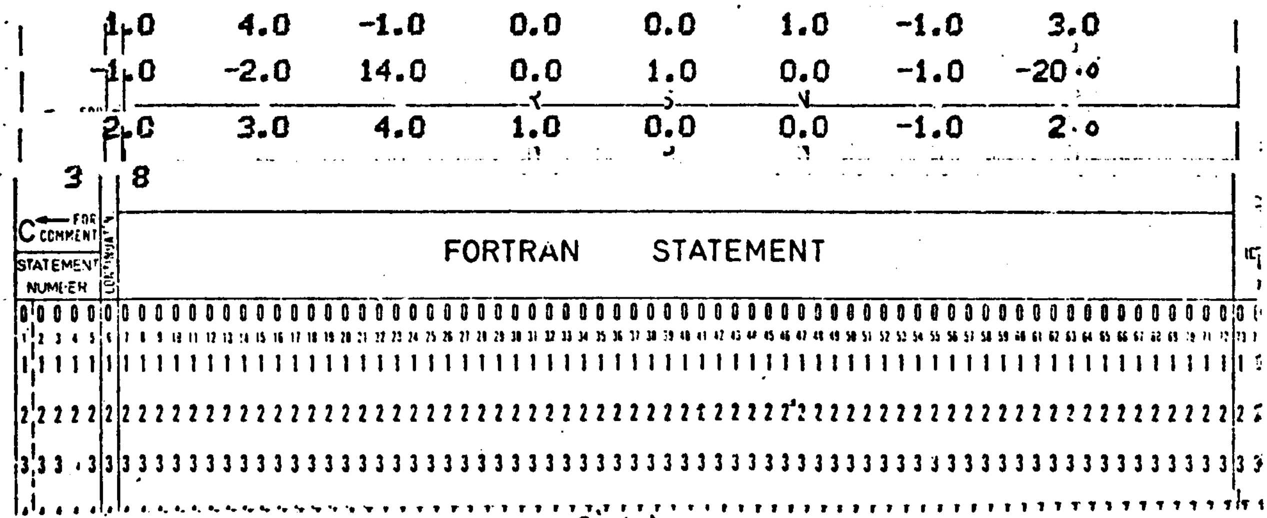
... ...

D(IM,J) Holds the coefficient (elements) in the IMth and final row (thus final equation in (g)).

Example

$$w_1 = 2 + 2z_1 + 3z_2 + 4z_3$$
$$w_2 = -20 - z_1 - 2z_2 + 14z_3$$
$$w_3 = +3 + z_1 + 4z_2 - z_3$$
$$w_1, w_2, w_3, z_1, z_2, z_3 \geq 0$$
$$w_i z_i = 0 \quad i = 1,2,3$$

The data deck is shown in Fig(1).



Fig(1)

Program II solves the Fundamental Problem via the algorithm proposed by the author in chapter 3.

The Fundamental Problem may be written in full as

$$-m_{i1}z_1 - \quad \ldots - m_{in}z_n + w_i = q_i \ , \quad i = 1,\ldots,n \ . \qquad \text{(h)}$$
$$w_i, z_i \geq 0 \ , \quad z_i w_i = 0 \qquad i = 1,\ldots,n$$

Define

$$Q_1 = \{i \mid q_i \geq 0\} \ , \ \text{and} \ \ Q_2 = \{i \mid q_i < 0\} \ .$$

Then (h) is written in the form

$$\begin{cases} -m_{i1}z_1 - \quad \ldots - m_{in}z_n + w_i = q_i & \text{if } i \in Q_1 \\ m_{i1}z_1 + \quad \ldots + m_{in}z_n - w_i + v_i = -q_i & \text{if } i \in Q_2 \end{cases} \qquad \text{(f)}$$
$$w_i, z_i \geq 0 \quad w_i z_i = 0 \qquad i = 1,\ldots,m$$
$$v_i = 0 \quad i \in Q_2 \ .$$

To use the program one must prepare a data deck as described below

1. Read in conformance with FORMAT statement 104 in the SUBROUTINE SIMPLEX, value for the following variables

IW = number of rows in the set of equations (h).

IZ = number of the columns of matrix [M,I] + the cardinality of the set Q_2 + 1.

I30 = the cardinality of the set Q_2.

If the set of equation is expressed in the form

$$M_2 v = M_1 v_1 + I v_2 = q' \qquad (q' \geq 0) \qquad (k)$$

then

IY = number of components of the vector v_1 plus one.

2. Read the coefficient of the artificial variable in the infeasibility form introduced to get a basic feasible solution to (k), into the array P(J),J = 1 to IZ-1.
Where,

$$P(J) = \begin{cases} 0 & \text{if } J^{th} \text{ component of v is not artificial} \\ -M & \text{if } J^{th} \text{ component of v is artificial} \end{cases}$$

where M is very large positive number.

3. Read the coefficients of the equation in (f) into the array D(M,N), M = 1 to IW and N = 1 to IZ, in conformance with the FORMAT statement number 103.

Example

$$w_1 = 10 + 2z_1 + 2z_2 - z_3 + z_4$$

$$w_2 = -2 + 3z_1 - 3z_2 + 4z_3 - z_4$$

$$w_3 = 3 - z_1 + 4z_2 + 10z_3 + 2z_4$$

$$w_4 = -4 + z_1 - 5z_2 - z_3 + 3z_4$$

$$w_i z_i \geq 0 \quad w_i z_i = 0$$

may be expressed as

$$-2z_1 - 2z_2 + z_3 - z_4 + w_1 = 10$$

$$3z_1 - 3z_2 + 4z_3 - z_4 - w_2 + v_1 = 2$$

$$z_1 - 4z_2 - 10z_3 - 2z_4 + w_3 = 3$$

$$z_1 - 5z_2 - z_3 + 3z_4 - w_4 + v_2 = 4$$

The data cards are as:

```
   1. -5. -1.   3.   0.   0.   0. -1.   0.   1.   4.
   1. -4.-10.  -2.   0.   0.   1.   0.   0.   0.   3.
   3. -3.   4. -1.   0.  -1.   0.   0.   1.   0.   2.
  -2. -2.   1. -1.   0.   0.   0.   0.   0.   0.  10.
   0.  0.   0.   0.   0.   0.   0.   0.-99.-99.

    4  11   6   2
```

C FOR COMMENT | STATEMENT NUMBER | CONTINUATION | FORTRAN STATEMENT

PROGRAM I

```
    MASTER LEMKE
    DIMENSION D(40,100), IBV(40)
    COMMON D,IBV,KR,LR,IM,IN,K2  ,ICOLUM   ,M1
100 FORMAT(2I4)
101 FORMAT(10F8.1)
    READ(2,100)  IM,IN
    LX=IN-1
    DO  102  I=1,IM
102 READ(2,101)(D(I,J),J=1,IN)
    M1=0
    RC=0.0
    DO  150  I=1,IM
    IF(D(I,IN))151,150,150
151 IF(D(I,IN)-RC)152,152,150
152 RC=D(I,IN)
150 CONTINUE
    DO  104  I=1,IM
    D(I,IN)=D(I,IN)+ABS(RC)
    IBV(I)=IM+I
104 CONTINUE
    WRITE(3,111)
111 FORMAT(30X,23HTHIS IS INITIAL TABLEAU)
    WRITE(3,201)(I,I=1,IM),(J,J=1,IM+1)
201 FORMAT(11X,6(2HX(,I2,1H),3X ),7(2HW(,I2,1H),3X))
    DO  108  I=1,IM
    IF(D(I,IN))108,109,108
109 KR=I
    M10=IBV(KR)
    GO  TO  160
108 CONTINUE
160        DO  106  I=1,IM
    I20=IBV(I)-IM
    IF(I20)400,400,401
400 WRITE(3,107)(IBV(I),D(I,J),J=1,IN)'
    GO   TO    106
401 WRITE(3,402)I20,(D(I,J),J=1,IN)
106 CONTINUE
107  FORMAT(1X,2HX(,I3,2H)=,15F8.1)
402 FORMAT(1X,2HW(,I3,2H)=,15F8.1)
    ICOLUM=M10-IM
    D(KR,IN)=ABS(RC)
    LR=IN-1
    IBV(KR)=LR
200 CALL PIVOT
141 CALL CHECK
    IF(K2)114,140,114
114 LR=ICOLUM
    SMALL=999999.
    DO  115  I=1,IM
    IF(D(I,ICOLUM))115,115,199
199        SIM=D(I,IN)/D(I,ICOLUM)
```

```
      SIM1=SIM-SMALL
      IF(SIM1)196,115,115
  196 SMALL=SIM
      KR=I
  115 CONTINUE
       I3=IBV(KR)-LX
      M12=IBV(KR)
      IBV(KR)=ICOLUM
      CALL  PIVOT
      IF(I3)116,118,116
  118         WRITE(3,172)
  172 FORMAT(10X,8HSOLUTION)
      DO   171   I=1,IM
       I30=IBV(I)-IM
      IF(I30)600,600,601
  600         WRITE(3,173)IBV(I),D(I,IN)
  173 FORMAT(5X,2HX(,I3,2H)=,F8.1)
      GO    TO    171
  601 WRITE(3,602)I30,D(I,IN)
  602 FORMAT(5X,2HW(,I3,2H)=,F8.1)
  171 CONTINUE
      WRITE(3,174)
  174 FORMAT(10X,19HALL OTHER VARIABL=0)
      GO    TO     300
  116 I4=M12-IM
      IBV(KR)=ICOLUM
      IF(I4)121,120,120
  120 ICOLUM=I4
      GO    TO     141
  121 ICOLUM=M12+IM
      GO    TO     141
  140 WRITE(3,202)
  202 FORMAT(1X,23HPROBLEM HAS NO SOLUTION)
  300 STOP
      END
      SUBROUTINE PIVOT
      DIMENSION    D(40,100),IBV(40)
      COMMON D,IBV,KR,LR,IM,IN,K2,ICOLUM,M1
      K6=IN
      IF(M1)17,16,17
   16 IN=IN-1
   17 DIV=D(KR,LR)
      DIV IS  PIVOT  ELEMENT
      DO 7      N=1,IN
      CRANK=D(KR,N)/DIV
    7 D(KR,N)=CRANK
      DO   10    M=1,IM
      IF(M-KR)9,10,9
    9 RM=-D(M,LR)
      DO   11     N=1,IN
      BM=D(KR,N)*RM
      SINK=D(M,N)+BM
```

```
      D(M,N)=SINK
   11 CONTINUE
   10 CONTINUE
      IN=K6
      M1=M1+1
      WRITE(3,110)M1
  110 FORMAT(1X,18HTHIS IS TABLEAU NO,I3)
      DO 112  I=1,IM
      I20=IBV(I)-IM
      IF(I20)200,200,201
  200 WRITE(3,113)IBV(I),(D(I,J),J=1,IN)
  113 FORMAT(1X,2HX(,I3,2H)=,15F8.1)
      GO     TO     112
  201 WRITE(3,202)I20,(D(I,J),J=1,IN)
  202 FORMAT(1X,2HW(,I3,2H)=,15F8.1)
  112 CONTINUE
      RETURN
      END
      SUBROUTINE  CHECK
      DIMENSION    D(40,100),IBV(40)
      COMMON D,IBV,KR,LR,IM,IN,K2  ,ICOLUM   ,M1
      K2=0
      DO 300 I=1,IM
      IF(D(I,ICOLUM))300,300,302
  302 K2=1
      GO TO    310
  300 CONTINUE
  310 RETURN
      END
      FINISH
```

PROGRAM II

```
      MASTER LINEAR COMPLEMENTARY PIVOT
      DEFINE FILE 10(200,500,U,K1),11(200,120,U,K2)
      DIMENSION D(20,40),IBV(20),N(40),M(20),M1(20),N1(40)   ,K(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
     1LR,ICOLUM,ICY,K,I30 ,M10
      K1=1
      CALL SIMPLEX
  101 DO 102 I=1,IW
  102 N(IBV(I))=1
      DO 126 I=1,IW
  126 M(I)=I
      K(K1)=0
      I31=IZ-I30
      DO   177  I=1,IW
  177 D(I,I31)=D(I,IZ)
      IZ=IZ-I30
      L1=1
      IX=IX-I30
      DO 103 J=1,IX
      IF(N(J)-1)110,103,110
  110 IF(N(J)-2)104,103,104
  104 I10=J-IW
      IF(I10)105,105,106
  105 I11=J+IW
      DO 107 I=1,IW
      IF(I11-IBV(I))107,109,107
  107 CONTINUE
      N(J),N(I11)=2
      K(K1)=K(K1)+1
  109 GO TO 103
  106 DO 108 I=1,IW
      IF(I10-IBV(I))108,103,108
  108 CONTINUE
      N(J),N(I10)=2
      K(K1)=K(K1)+1
  103 CONTINUE
C     UP TO HERE WE HAVE CALCULATED KILTER NUMBER
      WRITE(3,111)K1,K(K1)
  111 FORMAT(3X,24HKILTER NUMBER IN TABLEAU,I3,2HIS,I3)
C     UP TO HERE WE HAVE CHECKED FISIBILITY&CONSISTENTLY OF TABLEAU
      LK,LW=0
      ICOLUM=1
      CALL  IOTAB
  120 DO 112 J=1,IX
  112 N1(J)=N(J)
      DO 113 I=1,IW
  113 M1(I)=M(I)
      ICOLUM=1
      IF(K(K1))114,114,266
  266 M10=0
      GO    TO    333
  114 M10=0
      CALL  BRULE
      IF (ICY )176,176,117
```

```
116 LK=1
    CALL   IOTAB
    DO 122 I=1,IW
122 M1(I)=M(I)
    DO 130 J=1,IX
130 N1(J)=N(J)
    IF(K(K1))310,310,311
310 M10=0
    CALL  BRULE
    IF(ICY)312,312,117
312 CALL   OUTPUT
    K1=K1+1
320 IF(K1-L1)116,116,128
311 M10=0
    CALL   BRULE
    IF(ICY)314,314,117
314 CALL   OUTPUT
    K1=K1+1
    GO   TO   320
117 L1=L1+1
    KFIX=K1
    K1=L1
    LK=1
    CALL IOTAB
    DO 138 I=1,IW
138 M1(I)=M(I)
    DO 139 J=1,IX
139 N1(J)=N(J)
    K1=KFIX
300 IF(K(L1))131,131,132
131 M10=1
    CALL  BRULE
    IF(ICY)133,133,134
133 GO   TO   116
134 GO TO 117
132 M10=1
333 CALL   BRULE
    IF(ICY)135,135,136
135 GO   TO   116
136 GO TO 117
100 WRITE(3,140)
140 FORMAT(10X,23HPROBLEM HAS NO SOLUTION)
176 CALL   OUTPUT
128 STOP
    END
    SUBROUTINE IOTAB
    DIMENSION D(20,40),IBV(20),M(20),N(40),N1(40),M1(20) ,K(200)
    COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
   1LR,ICOLUM,ICY,K
    IF(LK)203,203,205
203 IF(LW)200,200,201
200 WRITE(10'K1)((IBV(I),(D(I,J),J=1,IZ)),I=1,IW)
    K1=K1-1
201 K2=K1
```

```
      WRITE(11'K2)(M(I),I=1,IW),(N(J),J=1,IZ),ICULUM
      K2=K2-1
      GO TO 204
205    READ(10'K1)((IBV(I),(D(I,J),J=1,IZ)),I=1,IW)
      K1=K1-1
      K2=K1
      READ(11'K2)(M(I),I=1,IW),(N(J),J=1,IZ),ICOLUM
      K2=K2-1
204   RETURN
      END
      SUBROUTINE PIVOT
      DIMENSION D(20,40),IBV(20)
      COMMON//D,IBV,IW,IZ/AREA2/KR,LR
      DIV=D(KR,LR)
      DO 7 N=1,IZ
      CRANK=D(KR,N)/DIV
    7 D(KR,N)=CRANK
      DO 10 M=1,IW
      IF(M-KR)9,10,9
    9 RM=-D(M,LR)
      DO 11 N=1,IZ
      BM=D(KR,N)*RM
      SINK=D(M,N)+BM
      D(M,N)=SINK
   11 CONTINUE
   10 CONTINUE
      RETURN
      END
      SUBROUTINE OUTPUT
      DIMENSION D(20,40),IBV(20),M(20),N(40),N1(40),M1(20) ,K(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
     1LR,ICOLUM,ICY,K,I30 ,M10
      WRITE(3,500)K1,K(K1)
500   FORMAT(1X,18HTHIS IS TABLEAU NO,I3,20HAND KILTER NUMBER IS,I3)
      WRITE(3,351)(N(J),J=1,IX)
351   FORMAT(10X,15I8)
      DO 504 I=1,IW
      I20=IBV(I)-IW
      IF(I20) 200,200,201
200   WRITE(3,113)M(I),IBV(I),(D(I,J),J=1,IZ)
113   FORMAT(1X,I3,2HX(,I3,2H)=,15F8.1)
      GO TO 504
201   WRITE(3,202)M(I),I20,(D(I,J),J=1,IZ)
202   FORMAT(1X,I3,2HW(,I3,2H)=,15F8.1)
504   CONTINUE
      RETURN
      END
      SUBROUTINE BRULE
      DIMENSION D(20,40),IBV(20),M(20),N(40),N1(40),M1(20) ,K(200)
     1,ICH(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
     1LR,ICOLUM,ICY,K,I30 ,M10    /AREANEW/ICH,JIM
      SMALL=999999.0
      IROW=0
```

```
      ICY=0
      IF(M10)9000,9000,9001
 9000 IF(K(K1))9003,9003,9004
 9004 CALL CHECK
      IF(I30)9003,9003,888
 9001 IF(K(L1))9003,9003,9004
 9003 CALL   CHOOSE
      IF(I30)9950,9950,9951
 9951 J=JIM
      GO    TO    803
 9950 J=0
  802 J=J+1
      IF(J.GT.IX)GO TO 888
      IF(N(J)-1)2000,802,2000
 2000 IF(N(J))802,1802,803
 1802 DO 604 I=1,IW
      IF(M(I))604,604,605
  605 IF(D(I,J))604,604,606
  606 RM=D(I,IZ)/D(I,J)
      IF(RM-SMALL)607,604,604
  607 SMALL=RM
      IROW=I
  604 CONTINUE
      IF(IROW)802,802,1609
 1609 I22=J-IW
      IF(I22)610,610,1699
  610 I22=J+IW
 1699 IF(IBV(IROW)-I22)681,680,681
C     THIS IS THE CASE IN WHICH COMPLEMENTARY PIVOTE IS POSSIBLE
  680 N1(J)=-3
      N(I22)=-3
      M1(IROW)=-M(IROW)
      M(IROW)=-M(IROW)
      N(J)=1
      IBV(IROW)=J
      GO TO 699
C     IN THIS CASE PRINCIPAL PIVOTING NOT POSSIBLE
  681 I221=IBV(IROW)
      I231=I221-IW
      IF(I231)650,650,651
  650 I231=I221+IW
  651 DO 698 I=1,IW
      IF(IBV(I)-I231)698,660,698
  660 IF(M(I))661,697,697
  698 CONTINUE
      GO TO 503
C     THE VARIABLE IS IN BASIC AND FLAGGED
  661 N1(J)=-3
      N(J)=1
      N(IBV(IROW))=-3
      M(IROW)=-M(IROW)
  555 DO 500 I2=1,IW
      IF(IBV(I2)-I231)500,510,500
  500 CONTINUE
  510 IF(M(I2))1509,1510,1510
 1510 M1(I2)=-M(I2)
 1509 IBV(IROW)=J
```

```
      GO TO 699
  697 N1(J)=-3
      N(J)=1
      N(IBV(IROW))=0
      M(IROW)=-M(IROW)
      GO TO 555
  503 DO 504 I2=1,IW
      IF(IBV(I2)-I22)504,1505,504
  504 CONTINUE
      GO   TO      1506
 1505 IF(M(I2))1506,1507,1507
 1507 M1(I2)=-M(I2)
 1506 N1(J)=-3
      N(J)=1
      IF(N(I231)+3)506,505,506
  506 N(I231),N(IBV(IROW))=2
      IBV(IROW)=J
      GO TO 507
  505 N(IBV(IROW))=2
  507 M(IROW)=-M(IROW)
      IBV(IROW)=J
      IF(M10)1681,1681,1680
 1681 K(L1+1)=K(K1)+1
      GO TO 4000
 1680 K(L1+1)=K(L1)+1
      GO TO 4000
  699 IF(M10)691,691,692
  691 K(L1+1)=K(K1)
      GO TO 4000
  692 K(L1+1)=K(L1)
      GO TO 4000
  803 IROW=0
      DO 804 I=1,IW
      IF(M(I))804,804,805
  805 IF(D(I,J))804,804,806
  806 RM=D(I,IZ)/D(I,J)
      IF(RM-SMALL)807,804,804
  807 SMALL=RM
      IROW=I
  804 CONTINUE
      IF(IROW)802,802,1809
 1809 I22=J-IW
      IF(I22)810,810,899
  810 I22=IW+J
  899 I221=IBV(IROW)
      I231=I221-IW
      IF(I231)850,850,851
  850 I231=I221+IW
  851 DO    852    I=1,IW
      IF(IBV(I)-I231)852,853,852
  852 CONTINUE
      IF(N(I231)+3)351,350,351
  350 N(I221)=2
      GO    TO    855
```

```
 351 N(I221),N(I231)=2
     GO   TO   855
 853 IF(M(I))700,700,701
 700 N(I221)=-3
     GO  TO  702
 701 N(I221)=0
 702 IF(M10)910,910,911
 910 K(L1+1)=K(K1)-1
     GO   TO   815
 911 K(L1+1)=K(L1)-1
     GO  TO  815
 855 IF(M10)222,222,223
 222 K(L1+1)=K(K1)
     GO   TO   815
 223 K(L1+1)=K(L1)
 815 ICOLUM=J
     IBV(IROW)=ICOLUM
     M(IROW)=-M(IROW)
     N1(ICOLUM)=-3
     N(ICOLUM)=1
     N(I22)=-3
4000 KR=IROW
     LR=J
     CALL PIVOT
     L1=L1+1
     LK,LW=0


     IFIX=K1
     K1=L1
     CALL IOTAB
     K1=IFIX
     L1=L1-1
     ICY=1
     DO 820 I=1,IW
 820 M(I)=M1(I)
     DO 822 J=1,IX
 822 N(J)=N1(J)
     IF(M10)320,320,321
 320 LK=0
     LW=1
     CALL   IOTAB
     GO   TO   888
 321 IFIX=K1
     K1=L1
     LK=0
     LW=1
     CALL IOTAB
     K1=IFIX
 888 RETURN
     END
```

```
      SUBROUTINE CHOUSE
      DIMENSION D(20,40),IBV(20),M(20),N(40),N1(40),M1(20) ,K(200)
     1,ICH(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
     1LR,ICOLUM,ICY,K,I30 ,M10    /AREANEW/ICH,JIM
      I3,I30,J1,JIM=0
      K3=IX/2
      IF(M10)100,100,101
  100 IF(ICH(K1))102,102,103
  101 IF(ICH(L1))102,102,104
  103 J1=ICH(K1)+IW
      GO   TO   105
  104 J1=ICH(L1)+IW
  105 DO   106   I=1,IW
      IF(M(I))106,106,107
  107 IF(D(I,J1))106,106,108
  106 CONTINUE
      IF(I3)114,114,102
  114 I30=0
      GO  TO    120
  108 I30=100
      JIM=J1
      IF(I3)130,130,121
  121 IF(M10)122,122,123
  122 ICH(K1)=J1
      GO  TO    130
  123 ICH(L1)=J1
  130 GO   TO   120
  102 J1=J1+1
      I3=0
      J2=J1+IW
      IF(J1.GT.K3)  GO  TO   120
      IF((N(J1).EQ.2).AND.(N(J2).EQ.2))GO  TO  116
      GO  TO   102
  116 I3=1
      GO   TO   105
  120 RETURN
      END
      SUBROUTINE SIMPLEX
      DIMENSION D(20,40),P(39),IBV(20),SC(39)     ,K(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA2/KR, LR,ICOLUM,ICY,K  ,I30
101   FORMAT (I1)
 104  FORMAT(4I4)
 102  FORMAT(20F4.0)
 103  FORMAT(20F4.0)
106   FORMAT (1H0,11HTABLEAU NO.,I6)
108   FORMAT (1H1,9H SOLUTION)
 109  FORMAT(1H0,8HVARIABLE,4X,5HVALUE)
110.  FORMAT (1X,2HX(,I3,4H) = ,F12.2)
111   FORMAT (1H0,28H ALL OTHER VARIABLES = ZERO.)
112   FORMAT (1H1,21H THE INITIAL TABLEAU.)
113   FORMAT (11X,10F10.4/ (11X, 10F10.4))
 300  FORMAT(11X,10I10/(11X,10I10))
301   FORMAT (1H0,2X,2HX(,I2,1H),3X,10F10.3/ (11X, 10F10.3))
302   FORMAT(1H0,12H SIMPLEX  CR,10F10.3/ (11X,10F10.3))
305   FORMAT (1H0,9HOBJ FNCTN, 1X, 10F10.3/ (11X,10F10.3))
 789  FORMAT(1H0,10X,28HOBJECTIVE FUNCTION VALUE IS ,F15.5)
```

```
      READ(2,104)IW,IZ,IY,I30
      IX=IZ-1
      READ(2,101)ITAB
      READ(2,102)(P(J),J=1,IX)
      DO 15M=1,IW
  15  READ(2,103)  (D(M,N),N=1,IZ)
      WRITE(3,112)
      WRITE(3,305)(P(M),M=1,IX)
      DO 16 M=1,IW
  16  WRITE(3,113) (D(M,N),N=1,IZ)
      DO 20 N=IY,IX
      DO 30 L=1,IW
      IF(D(L,N).EQ.1.) GO TO   40
  30  CONTINUE
      GO  TO  20
  40  IBV(L)=N
  20  CONTINUE
      Z=0.
      DO 210 M=1,IW
      IBVM=IBV(M)
 210  Z=Z+D(M,IZ)* P(IBVM)
      NOPIVS=0
      IF(ITAB.NE.1)GO TO 13
  13  SCMAX=0.
      DO 31 N=1,IX
      DO 32 I=1,IW
      IF(N.EQ.IBV(I)) GO TO 31
  32  CONTINUE
      SUM=0.
      DO 33 I=1,IW
      J=IBV(I)
  33  SUM=SUM+P(J)* D(I,N)
      SC(N)=P(N)-SUM
      IF(SC(N).LE.SCMAX)GO TO 31
      SCMAX=SC(N)
      IPIVCO=N
  31  CONTINUE
      DO 200 M=1,IW
      IBVM=IBV(M)
 200  SC(IBVM)=0.
      IF(SCMAX.LE.0) GO TO 14
      NOPIVS=NOPIVS+1
      SMLVAL=999999.
      DO 4 M=1,IW
      IF(D(M,IPIVCO)) 4, 4, 5
   5  QUONT=D(M,IZ)/D(M,IPIVCO)
      IF(QUONT-SMLVAL) 6,4,4
   6  IPIVRO=M
      SMLVAL=QUONT
   4  CONTINUE
      IBV(IPIVRO)=IPIVCO
      DIV=D(IPIVRO,IPIVCO)
      DO 7 N=1,IZ
      CRANK=D(IPIVRO,N)
   7  D(IPIVRO,N)=CRANK/DIV
```

```
      IF(ITAB.NE.1) GO TO 12
      WRITE(6,302)  (SC(J),J=1,IX)
      N100=NOPIVS  +1
      WRITE(3,789) Z
      WRITE(3,106)N100
      WRITE(3,300)(N,N=1,IX)
12    DO 10 M=1,IW
      IF(M-IPIVRO)9,8,9
9     RM=-D(M,IPIVCO)
      DO 11 N=1,IZ
      BM=D(IPIVRO,N)*RM
      SINK=D(M,N)+BM
      D(M,N)=SINK
11    CONTINUE
 8    IF(ITAB.NE.1) GO TO 10
      WRITE(3,301)IBV(M),(D(M,N),N=1,IZ)
10    CONTINUE
      Z=Z+SMLVAL*SCMAX
      GO  TO  13
14    WRITE(3,108)
      WRITE(3,109)
      DO 21 M=1,IW
21    WRITE(3,110)IBV(M),D(M,IZ)
      WRITE(3,111)
      WRITE(3,789)  Z
      RETURN
      END
      SUBROUTINE   CHECK
      DIMENSION D(20,40),IBV(20),M(20),N(40),N1(40),M1(20) ,K(200)
      COMMON//D,IBV,IW,IZ,IX,Z/AREA1/M,N,M1,N1,K1,K2,L1,LK,LW/AREA2/KR,
     1LR,ICOLUM,ICY,K,I30 ,M10
      J1,I30=0
      K3=IX/2
      I10=-3
400   J1=J1+1
      IF(J1.GT.K3)  GO   TO   200
      IF(N(J1)-I10)400,401,400
401   IF((N(J1).EQ.I10).AND.(N(J1+IW).EQ.I10)) GO TO  202
      GO   TO   400
202   I30=100
200   RETURN
      END
       FINISH
```

Appendix R4

The FORTRAN program in this Appendix solves the plant location problem with unlimited capacity, and concave cost function using the algorithm discussed in chapter 4.

To use the program one must prepare a data deck as described below.

1. Read in, in conformance with FORMAT statement number 100, values for the following variables:

LAST = number of arcs of the graph related to the given problem. (This is a directed graph.)

IM = number of plants.

IN = number of customers.

2. Read the nodes of the graph from which arcs start, into the array M1(I),I = 1 to LAST, in conformance with FORMAT statement number 101.

3. Read the nodes of the graph to which arcs end,into the array N1(I),I = 1 to LAST, in conformance with FORMAT statement number 101.

4. Read the number of customers that can be supplied from each plant, into the array N2(I),I = 1 to IM, in conformance with the FORMAT statement number 109.

5. Read the number of segment of each cost function of the plants, into the array, IK(I),I = 1 to IM, in conformance with the FORMAT statement number 109.

6. Read the points of discontinuity of gradient of the cost function of the plants, into the array L(I,J),I = 1 to IM and J = 1,IK(I), in conformance with the FORMAT statement number 105.

7. Read the slope of the lines, into the array ALAM(I,J),I = 1 to IM, J = 1 to IK(I), in conformance with the FORMAT statement number 107.

8. Read, the demand of each customer, into the array D(J),J = 1 to IN, in conformance with the FORMAT statement number 108.

9. Read the transportation cost of a unit from plants to the customers,into the array T (I,J),I = 1 to IM and J = 1 to N2(I), in conformance with the FORMAT statement number 104.

10. Read the least fixed charge at each plant, into the array F(I,1),I = 1 to IM, in conformance with the FORMAT statement number 112.

```
      MASTER PLANT LOCATION
C     THIS PROGRAM SOLVES PLANT LOCATION PROBLEM WHEN
C     PLANTCOST ARE CONCAVE FUNCTION*
      DEFINE    FILE10(100,60,U,KSVE)
      DIMENSION C(15,20,5),X(15,20,5),
     1          Z(800),Y(15,5),F(15,5),ALAM(15,6)
      INTEGER M1(200),N1(200),N2(15),N4(99),L(15,6),D(20),   N3(15),
     1IK(15),SUM(15)    ,T(15,20)
     1,NODE(200)
      COMMON//IM,IN,LINK,IB,I1,J1,LI,LK,ANS/AREA1/Y,IK/AREA3/
     1M1,N1/AREA4/KSVE,NSVE,LSVE/AREA2/C,F,N2,X/AREA9/
     2L,SUM               /AREA11/D/AREA12/KR,KR1,KR2    /AREA10/LAST,N3
     3/AREA20/J5,Z,NODE,M11,M10,N4
C     READING NUMBER OF ARCS NUMBER OF PLANT, NUMBER OF
C     CUSTOMERS
  100 FORMAT(3I6)
C     READING NODES FOR NETWORK
  101 FORMAT(40I2)
  102 FORMAT(10X, 29HTHESE ARE THE ARCS OF NETWORK)
  103 FORMAT(15X,1H(,I6,1H-,I6,1H))
C     READING THE TRANSPORTATION COST
  104 FORMAT(20I4)
C     READING THE POINT OF DISCONTINUITY
  105 FORMAT(10I8)
C     READING NUMBER OF SEGMENTS
  106 FORMAT(I8)
C     READING THE SLOPS OF THE LINES
  107 FORMAT(10F8.1)
C     READING DEMAND
  108 FORMAT(20I4)
C     READING NUMBER OF CUSTOMER THAT CAN BE SUPPLIED
  109 FORMAT(20I4)
  110 FORMAT(20X,32HPROBLEM HAS NO FEASIBLE SOLUTION)
C     READING INITIAL FIXED COST
  112 FORMAT(10F8.1)
      READ(2,100)LAST,IM,IN
      READ(2,101)(M1(I),I=1,LAST         )
      READ(2,101)(N1(I),I=1,LAST)
      WRITE(3,102)
      DO       113    I=1,LAST
  113 WRITE(3,103)M1(I),N1(I)
      READ(2,109)(N2(J),J=1,IM)
      READ(2,109)(IK(I),I=1,IM)
C     READING THE POINTS OF DISCONTINUITY
      DO 5 I=1,IM
    5 READ(2,105)(L(I,J),J=1,IK(I))
C     READING SLOPES OF THE LINES
      DO 8 I=1,IM
    8 READ(2,107)(ALAM(I,J),J=1,IK(I))
C     READING DEMAND
      READ(2,108)(D(J),J=1,IN)
      DO     3   I=1,IM
      READ(2,104)(T(I,J),J=1,N2(I))
      N3(I)=N2(I)
```

```
    3 CONTINUE
C       READING INITIAL FIXED COST
        READ(2,112)(F(I,1),I=1,IM)
C       ***SIMPLICATIFICATIUN ONE***
C       THIS CALCULATION MUST BE DONE BEFORE ANY OTHER
C       IT SHOWS HOW MAY SEGMENTS CAN BE USED FOR EACH PLANT
        DO      200     I=1,IM
  200 SUM(I)=0
        DO 9 I1=1,IM
        DO 10 J1=1,IN
        IB=I1
        CALL NETFLW
        IF(LINK)10,10,11
   11 SUM(I1 )=SUM(I1)+D(J1)
   10 CONTINUE
    9 CONTINUE
        DO 13 I=1,IM
        IF(SUM(I).LT.L(I,1)) GO TO 15
        DO 14 J=1,IK(I)
        IF((SUM(I).GE.L(I,J)).AND.(SUM(I).LT.L(I,J+1)))GO TO 77
   14 CONTINUE
   15 IK(I)=1
        GOTO 13
   77 IK(I)=J +1
   13 CONTINUE
C                 END OF THE SIMPLIFICATION ONE
C                 ***
C                 *
        WRITE(3,100)IM,IN,LAST
        DO    224    I=1,IM
  224 WRITE(3,105)(L(I,J),J=1,IK(I))
        DO      223    I=1,IM
  223 WRITE(3,109)(T(I,J),J=1,N2(I))
        DO   222       I=1,IM
  222 WRITE(3,106)SUM(I)
        WRITE(3,105)(IK(I),I=1,IM)
C       ***THIS PART OF THE PROGRAM CALCULATES PLANCOST***
        K6=0
        DO 16 I1=1,IM
        DO 17 J1=1,IN
        IB=I1
        CALL NETFLW
        IF(LINK)17,17,117
  117 K6=K6+1
        DO 18 K=1,IK(I1)
   18 C(I1,K6,K)=(T(I1,K6)+ALAM(I1,K))*D(J1)
   17 CONTINUE
        K6=0
   16 CONTINUE
C       *** THIS PART CALCULATES FIXED CHARGE COST***
        DO 19 I=1,IM
        DO 20 K=2,IK(I)
   20 F(I,K)=ALAM(I,K-1)*L(I,K-1)+F(I,K-1)-ALAM(I,K)*L(I,K-1)
   19 CONTINUE
```

```
      DO    230     I=1,IM
      DO       231      K=1,IK(I)
      WRITE(3,112)(C(I,J,K),J=1,N2(I))
  231 Y(I,K)=2.0
  230 CONTINUE
C     ***SIMPLIFICATION TWO **
C     THIS SIMPLIFICATION REDUCES THE NUMBER OF Y'S
      SNALL=99999999.0
      DO 30 J=1,IN
      IF(D(J)-SNALL)31,31,30
   31 SNALL=D(J)
   30 CONTINUE
C     SMALL IS MINIMUM OF THE DEMAND
      IH=0
   34 IH=IH+1
      DO 32 I=1,IM
      IF(L(I,IH)-SNALL)32,32,33
   32 CONTINUE
      GO TO 34
   33 IF(IH-1)35,35,36
   36 DO 37 I=1,IM
      DO 38 K=1,IH-1
   38 Y(I,K)=0.0
   37 CONTINUE
   35 DO    225    I=1,IM
      WRITE(3,112)(F(I,K),K=1,IK(I))
      DO    226     K=1,IK(I)
  226 CONTINUE
  225 CONTINUE
      DO      228     I=1,IM
  228 WRITE(3,112)(Y(I,K),K=1,IK(I))
      CALL   SIMFIC
      M10=99
      DO      280     I=1,M10
  280 N4(I)=0
      J5=1
      KSVE=1
      LSVE=1
      NODE(J5)=1
      CALL   SIMPLX
      CALL    IOTAB
      IF(KR)21,21,23
   21 WRITE(3,110)
      GO   TO   80
   23 CALL  CHECK
      IF(ANS-1.0)2780,2799,2780
 2799 WRITE(3,2999)
 2999 FORMAT(38HOPTIMAL SOLUTION OCCURED AT FIRST NODE)
      GO   TO  80
 2780 UPB=9999999.0
C     ***  BRANCHING  STRATS  FROM  HERE***
C                   *******
C                     *****
C                       ***
```

```
C                  PART  ONE
C      *** CHOOSING    THE  NODE  FOR  FURTHER  BRANCHING ***
 8000 WRITE(3,8899)UPB
      SMALL=9999999.0
 8899 FORMAT(3X,13HT8IS IS UPER=,F12.2)
      DO    8891    I=1,J5
 8891 WRITE(3,8892)I,Z(I)
 8892 FORMAT(11X,4H****,I5,F12.2)
      DO     1023    I=1,J5
      IF(Z(I)-SMALL)1024,1024,1023
 1024 SMALL=Z(I)
      M11=I
 1023 CONTINUE
C      *****    END    ******
C      READING      THE    RECORD
      LSVE=2
      M4=KSVE
      NSVE=NODE(M11)
      CALL     IOTAB
      KSVE=M4
      CALL    YRULE
      CALL    SIMFIC
      DO      1290    I=1,M10
      IF(N4(I)) 1290,1290,1292
 1292 KSVE=N4(I)
      N4(I)=0
      GO    TO    1350
 1290 CONTINUE
 1400 KSVE=KSVE+1
 1350 LSVE=1
      CALL    IOTAB
      J5=J5+1
      NODE(J5)=KSVE
      CALL    SIMPLX
C             ***    END    OF   THE   FIRST   BRANCH    ***
      KR1=KR
      IF(KR1)24,24,25
   24 DO      2500      I=1,M10
      IF(N4(I))2501,2501,2500
 2501 N4(I)=NODE(J5)
      Z(J5)=9999999.0
      GO      TO     2524
 2500 CONTINUE
 2524 LSVE=2
C          *****READING   THE   AGAIN   ***
      NSVE=NODE(M11)
      M4=KSVE
      CALL       IOTAB
      KSVE=M4
      Y(LI,LK)=0.0
      DO       1390       I=1,M10
      IF(N4(I))1390,1390,1392
 1392 KSVE=N4(I)
      N4(I)=0
```

```
      GO    TO       2509
 1390 CONTINUE
 2400 KSVE=KSVE+1
 2509 LSVE=1
      CALL     IOTAB
      CALL     SIMFIC
      J5=J5+1
      NODE(J5)=KSVE
      CALL     SIMPLX
      KR2=KR
      IF(KR2)1064,1064,1065
 1064 DO       2600    I=1,M10
      IF(N4(I)) 2061,2061,2600
 2061 N4(I)=NODE(J5)
      Z(J5)=9999999.0
      GO  TO       2664
 2600 CONTINUE
 2664 IF((KR1.EQ.0.0).AND.(KR2.EQ.0.0))  GO    TO    21
      GO   TO     1065
   25 CALL     CHECK
      IF(ANS-1.0)133,1066,133
 1066 IF(Z(J5).GE.UPB)     GO    TO    2524
      UPB=Z(J5)
      M12=J5
  133 GO      TO    2524
 1065 DO     2066    I=1,M10
      IF(N4(I))2067,2067,2066
 2067 N4(I)=NODE(M11)
      Z(M11)=9999999.0
      GO     TO   3065
 2066 CONTINUE
 3065 IF(KR2)3069,3069,3066
 3066 CALL CHECK
      IF(ANS.EQ.1.0)     GO     TO     1067
 3069 K8=J5-1
      IF(Z(K8)-UPB)1068,1068,1230
 1067 IF(Z(J5).LT.UPB)    UPB=Z(J5)
      M12=J5
 1068 M8=0
      DO      1069      I=1,J5
      IF(Z(I).EQ.9999999.0)    GO   TO    1069
      IF(Z(I)-UPB)1442,1444,1444
 1442 M8=M8+1
      GO     TO       1069
 1444 Z(I)=9999999.0
      DO      1070      J=1,M10
      IF(N4(J))1071,1071,1070
 1071 N4(J)=NODE(I)
      GO     TO       1069
 1070 CONTINUE
 1069 CONTINUE
      IF(M8.EQ.0)     GO     TO     8888
      GO  TO  8000
 8888 WRITE(3,2800) M12
```

```
 2800 FORMAT(3X,40HTHIS IS OPTIMAL SOLUTION OCCURED AT NODE,I6)
      WRITE(3,2801)UPB
 2801 FORMAT(8HOPTIMAL=,F12.3)
      GO   TO     80
 1230 DO        1231    I=1,M10
      IF(N4(I))1232,1232,1231
 1232 N4(I)=NODE(K8)
      Z(K8)=9999999.0
      GO        TO        1068
 1231 CONTINUE
      GO        TO        1068
   80 STOP
      END
C     *** THIS SUBROUTINE CHOOSES FREE PLANT***
      SUBROUTINE YRULE
      DIMENSION Y(15,5),IK(15)
      COMMON//IM,IN,LINK,IB,I1,J1,LI,LK/AREA1/Y,IK
      ALARGE=0.0
      DO 1000 I=1,IM
      DO 2000 K=1,IK(I)
      IF((Y(I,K).EQ.1.0).OR.(Y(I,K).EQ.0.))GO TO 2000
      IF(Y(I,K)-ALARGE) 2000,2000,3000
 3000 ALARGE=Y(I,K)
      LI=I
      LK=K
 2000 CONTINUE
 1000 CONTINUE
      Y(LI,LK)=1.0
      DO    4000      K=1,IK(LI)
      IF(K.EQ.LK)    GO    TO    4000
      Y(LI,K)=0.0
 4000 CONTINUE
      RETURN
      END
C     **** END OF THE SUBROUTINE YRULES ****
C     * THIS SUBROUTINE SOLES LINEAR PROGRAM AT NODE *
      SUBROUTINE SIMPLX
      DIMENSION IK(15),Y(15,5),F(15,5),C(15,20,5),G(15,5),L1(15),
     1X(15,20,5),N2(15),Z(800)
     1,NODE(200)   ,N4(99)
      COMMON//IM,IN,LINK,IB,I1,J1 /AREA1/Y,IK/AREA2/C,F,N2,X
     1/AREA4/KSVE,NSVE,LSVE/AREA12/KR,KR1,KR2
     3/AREA20/J5,Z,NODE,M11,M10   ,N4
      WRITE(3,1240)M11
 1240 FORMAT(2X,17HTHE LAST NODE WAS,I5)
      WRITE(3,1220)J5
 1220 FORMAT(2X,16HTHIS IS THE NODE,I5)
      DO     7777    I=1,IM
      DO     7778    K=1,IK(I)
 7778 WRITE(3,7779)Y(I,K)
 7779 FORMAT(F12.6)
 7777 CONTINUE
      K10=J5
      Z(K10)=0.0
```

```
      DO 40 I=1,IM
   40 L1(I)=0
      SMALL=999999.0
      DO 4 J1=1,IN
      SMALL=999999.0
      KR=0
      DO 5 I1=1,IM
      IB=I1
      CALL NETFLW
      IF(LINK)5,5,7
    7 L1(I1)=L1(I1)+1
      DO 6 K=1,IK(I1)
      IF(Y(I1,K)) 48,216,48
  216 X(I1,L1(I1),K)=0.0
      GO   TO   6
   48 IF(Y(I1,K)-1.0)51,46,51
   46 IF(X(I1,L1(I1),K))45,45,49
   49 X(I1,L1(I1),K)=1.0
      IF(K.GE.IK(I1))   GO  TO   77
      DO      73     K2=K+1,IK(I1)
   73 X(I1,L1(I1),K2)=0.0
   77 IF(I1.GE.IM)   GO    TO   55
      K20=I1
      DO     70      I1=K20+1,IM
      IB=I1
      CALL    NETFLW
      IF(LINK)70,70,71
   71 L1(I1)=L1(I1)+1
      DO    72     K2=1,IK(I1)
   72 X(I1,L1(I1),K2)=0.0
   70 CONTINUE
      I1=K20
      GO TO 55
   45 GO TO 59
   51 IF( X(I1 ,L1(I1),K)-F(I1,K)/N2(I1))59,59,52
   52 X(I1,L1(I1),K)=1.0
      IF(K.GE.IK(I1))      GO    TO     4008
      DO    4009    K2=K+1,IK(I1)
 4009 X(I1,L1(I1),K2)=0.0
 4008 K20=I1
      DO     4000      I1=K20+1,IM
      IB=I1
      CALL       NETFLW
      IF(LINK)4000,4000,4001
 4001 L1(I1)=L1(I1)+1
      DO        5000       K2=1,IK(I1)
 5000 X(I1,L1(I1),K2)=0.0
 4000 CONTINUE
      I1=K20
      GO TO 55
   59 X(I1,L1(I1),K)=0.0
      IF(Y(I1,K)-1)9,8,9
    8 G(I1,K)=0.0
```

```
    9 G(I1,K)=F(I1,K)
   11 IF(C(I1,L1(I1),K)+G(I1,K)/N2(I1)-SMALL)12,6,6
   12 SMALL = C(I1,L1(I1),K)+G(I1,K)/N2(I1)
      KH=I1
      KM=L1(I1)
      KR=K
    6 CONTINUE
    5 CONTINUE
      IF(KR)60,60,61
   60 Z(K10)=99999999.0
      WRITE(3,1222)
 1222 FORMAT(36HPROBLEN HAS NO SOLUTION AT THIS NODE        )
      GO TO 20
   61 X(KH,KM,KR)=1.0
      GO TO 63
   55 KH=I1
      KM=L1(I1)
      KR=K
   63 WRITE(3,41)KH,KM,KR,X(KH,KM,KR)
   41 FORMAT(2X,2HX(,I2,1H,,I2,1H,,I2,2H)=,F5.2)
      Z(K10)=Z(K10)+C(KH,KM,KR)
    4 CONTINUE
      DO     1100     I=1,IM
      WRITE(3,1005)L1(I)
 1005 FORMAT(I20)
 1100 CONTINUE
      DO 21 I1=1,IM
      DO 22 K=1,IK(I1)
      IF(Y(I1,K).EQ.1.0)GO     TO    220
      SUM=0.0
      DO 23 J=1, N2(I1)
   23 SUM=SUM+X(I1,J,K)
      IF(N2(I1))8133,8133,8134
 8133 Y(I1,K)=0.0
      GO      TO     22
 8134 Y(I1,K)=SUM/N2(I1)
      Z(K10)=Z(K10)+Y(I1,K)*F(I1,K)
      GO      TO    22
   22 CONTINUE
  220 Z(K10)=Z(K10)+F(I1,K)
   21 CONTINUE
      DO 24 I=1,IM
   24 WRITE(3,145)(Y(I,K),K=1,IK(I )       )
  145 FORMAT(1X,10F8.2)
      WRITE(3,144)  Z(K10)
  144 FORMAT(1X,21HOBJECTIVE FUNCTION IS ,F14.2)
   20 RETURN
      END
C     *** END OF THE SUBROUTINE SIMPLX ***
C     *** SUBROUTINE FOR CHECKING FEASIBILITY ***
      SUBROUTINE CHECK
      DIMENSION Y(15,5),IK(15)
      COMMON//IM,IN,LINK,IB,I1,J1,LI,LK,ANS/AREA1/Y,IK
      ANS=1.0
```

```
      DO 1 I=1,IM
      DO 2 K=1, IK(I)
      IF((Y(I,K).EQ.1.).OR.(Y(I,K).EQ.0.)) GO TO 2
      ANS=0
      GO TO 3
    2 CONTINUE
    1 CONTINUE
    3 RETURN
      END
C     *** END OF THE SUBROUTINE CHECK ***
C     * THIS SUBROUTINE CHECKS THE NETFLOW *
      SUBROUTINE NETFLW
      DIMENSION M1(200),N1(200)   , N3(15)
      COMMON//IM,IN,LINK,IB,I1,J1/AREA3/M1,N1,I9/AREA10/LAST   ,N3
      I9=0
      LINK=0
      M3=3
      K11=IB
      DO     5    J=K11,LAST,M3
      IF(M1(J)-IB)5,6,6
    5 CONTINUE
    6 IF(M1(J)-IB)11,10,11
   10 J=J-1
      IF(J) 6,11,6
      GO    TO    6
   11 IB=J+1
      DO     1    I=IB,IB+N3(I1)-1
      IF(N1(I)-J1)1,4,1
    1 CONTINUE
      GO TO 7
    4 LINK=1
      I9=I
    7 RETURN
      END
C     * THIS SUBROUTINE IS ONLY FOR READ AND WRITING
C     IN SCRATCH FILE
      SUBROUTINE IOTAB
      DIMENSION Y(15,5),IK(15)
      COMMON//IM,IN/AREA1/Y,IK/AREA4/KSVE,NSVE,LSVE
      IF(LSVE-2)60,62,62
   60 WRITE(10'KSVE)((Y(I,K),K=1,IK(I)),I=1,IM)
      KSVE=KSVE-1
      RETURN
   62 KSVE=NSVE
      READ(10'KSVE)((Y(I,K),K=1,IK(I)),I=1,IM)
      KSVE=KSVE-1
      RETURN
      END
C     *** END OF THE SORT FOR DISC FILE ***
C     ***** MAIN SUBROUTINE *****
C     * SUBROUTINE FOR SIMPLIFICATION
      SUBROUTINE SIMFIC
      DIMENSION X(15,20,5),Y(15,5),L1(15),N2(15),
     1C(15,20,5)      ,L(15,6),IK(15),
```

```
     1DOM(15,20,5) ,T(15,20),F(15,5) ,N1(200) ,M1(200)
      INTEGER D(20),SUM(15)     ,I13(15)
      COMMON//IM,IN,LINK,IB,I1,J1        /AREA1/Y,IK/
     1AREA2/C,F,N2,X/AREA9/L,SUM/AREA11/D   /AREA3/M1,N1 ,I9
      SMALL=9999999.0
      I1=0
      DO 230 I=1,IM
      DO 240 K=1,IK(I)
      T(I ,K)=0.0
      DO           1240    J=1,N2(I )
 1240 X(I ,J,K)=0.0
  240 CONTINUE
  230 CONTINUE
  201 I1=I1+1
      IF(I1.GT.IM)GO TO 250
      DO 203 K=1,IK(I1)
      IF(Y(I1,K).EQ.1.0)     I1=I1+1
      IF(I1.GT.IM)  GO    TO   250
  203 CONTINUE
      K3=1
 1023 DO  800    K=K3,IK(I1)
      IF(Y(I1,K))800,800,290
  800 CONTINUE
      GOTO 201
  290 LK=K
      LR=I1
      DO 202 I=1,IM
  202 L1(I)=0
      DO 204 J1=1,IN
      I1=LR
      SMALL=99999999.0
      IB=LR
      CALL NETFLW
      WRITE(3,1234)LR,J1,LINK
 1234 FORMAT(30X,5HGRAPH,3I5)
      IF(LINK)2040,2040,206
 2040 K20=I1
      DO         2071       I1=     1,IM
      IB=I1
      CALL     NETFLW
      IF(LINK)2071,2071,2072
 2072 L1(I1)=L1(I1)+1
 2071 CONTINUE
      I1=K20
      GO    TO    204
  206 RH=C(LR,L1(LR)+1,LK)
      DO 207 I1=1,IM
      IB=I1
      CALL NETFLW
      IF(LINK)207,207,209
  209 L1(I1)=L1(I1)+1
      DO 210 K1=1,IK(I1)
      IF(I1.EQ.LR.AND.K1.EQ.LK)GO TO 210
      IF(Y(I1,K1))299,210,299
```

```
  299 DELTA=C(I1,L1(I1),K1)-RH
      WRITE(3,1222) DELTA
 1222 FORMAT(11X,5HDELTA,F12.3)
      IF(DELTA)219,219,212
  212 X3=DELTA
      GO TO 300
  219 X3=0.0
  300 R1=X3
      IF(R1-SMALL)220,220,210
  220 SMALL=R1
  210 CONTINUE
  207 CONTINUE
      IF(SMALL.EQ.99999999.0)    SMALL=0.
      X(LR,L1(LR),LK)=SMALL
      WRITE(3,55554)X(LR,L1(LR),LK)
55554 FORMAT(11X,5HSMALL,F12.3)
      WRITE(3,2345)LR,J1,LK,SMALL
 2345 FORMAT(3X,3HROW,I2,3HCOL,I3,3HSEG,I3,5HSMALL,F8.2)
  204 CONTINUE
      DO  231   J=1,N2(LR)
  231 T(LR,LK)=T(LR,LK)+X(LR,J,LK)
      T(LR,LK)=T(LR,LK)-F(LR,LK)
      WRITE(3,9999)LR,LK,T(LR,LK)
 9999 FORMAT(2X,10HIMPORTANT=,2I5,F13.3)
      IF(T(LR,LK))260,260,262
  262 Y(LR,LK)=1.0
  260 K=LK+1
      K3=K
      I1=LR
      WRITE(3,6677)LR,L1(LR)
 6677 FORMAT(10HVALUEOFL1R,2I20)
      IF(K3-IK(I1))1023,1023,201
  250 WRITE(3,5555)
 5555 FORMAT(12X,7HSIMFICX)
C     *END OF THE DELTA SIMPLIFICATION
C     * THIS PART OF SUBROUTINE REDUCES N2'S; THAT IS
C     THE SUM OF CUSTOMERS THAT CAN BE SUPPLIED
C     FROM EACH PLANT. IF N2'S BECOMES ZERO
C     THAT PLANT WILL BE FIXED CLOSED.
      I1=0
      WRITE(3,1777)
 1777 FORMAT(19H*THIS IS PART 2 ***)
  400 I1=I1+1
      IF(I1.GT.IM)GO TO 450
      DO 403 K=1,IK(I1)
      IF(Y(I1,K)-1.0)403,400,403
  403 CONTINUE
      DO 514 K=1,IK(I1)
      IF(Y(I1,K))514,514,4016
  514 CONTINUE
      GO TO 400
 4016 LR=I1
      IB=I1
      DO 420 I=1,IM
```

```
  420 L1(I)=0
      DO 404 J1=1,IN
      IB=LR
      SMALL=9999999.0
      CALL NETFLW
      IF(LINK)1400,1400,406
 1400 K20=I1
      DO     1401    I1=1,IM
      IB=I1
      CALL    NETFLW
      IF(LINK)1401,1401,1402
 1402 L1(I1)=L1(I1)+1
 1401 CONTINUE
      I1=K20
      GO  TO     404
  406 K=0
      DO    11113    I=1,IM
11113 I13(I)=0
      RH=0.0
  480 K=K+1
      SMALL=9999999.0
      IF(K.GT.IK(LR))GO TO 490
      IF(Y(LR,K))481,482,481
  482 T(LR,K)=-100.0
      GO TO 480
  481 IF(RH)9000,9000,9001
 9000 RH=C(LR,L1(LR)+1,K)
      GO        TO    11112
 9001 RH=C(LR,L1(LR),K)
11112 WRITE(3,4091)RH
 4091 FORMAT(25X,3HRH=,F15.4)
      DO 407 I1=1,IM
      IB=I1
      LK=K
      CALL NETFLW
      IF(LINK)407,407,409
  409 I13(I1)=I13(I1)+1
      IF(I13(I1)-1)4086,4086,4099
 4086 L1(I1)=L1(I1)+1
      WRITE(3,4089)    I1,L1(I1)
 4089 FORMAT(11X,3HI1=,I5,7HL1(I1)=,I5)
 4099 DO 470 K2=1,IK(I1)
      IF(Y(I1,K2).NE.1. ) GO TO 470
      DEF=C(I1,L1(I1),K2)-RH
      WRITE(3,4000)I1,L1(I1),K2,C(I1,L1(I1),K2)
 4000 FORMAT(1X,11HTHE COEFFE=,3I3,F12.4)
      IF(DEF-SMALL)411,411,470
  411 SMALL=DEF
  470 CONTINUE
  407 CONTINUE
      IF(SMALL.EQ.9999999.0)    SMALL=0.0
      T(LR,LK)=SMALL
      GO TO 480
  490 ALARGE=-1000000.
```

```
      DO      491    K2=1,IK(LR)
      IF(T(LR,K2)-     ALARGE)491,491,492
  492 ALARGE=T(LR,K2)
  491 CONTINUE
      DO      1779     I=1,IK(LR)
 1779 WRITE(3,1778)T(LR,I)
 1778 FORMAT(8HNEGATIVE,F12.4)
      IF(ALARGE)1493,4004,4004
 4004 I1=LR
      GO    TO    404
 1493 IF(N2(LR))497,497,493
  497 DO 498 J3=1,IK(LR)
  498 Y(LR,J3)=0.0
      GO TO 400
  493 N2(LR)=N2(LR)-1
      IF(N2(LR))5000,5000,5001
 5000 DO   3000I=1,IK(LR)
 3000 Y(LR,I)=0.0
 5001 I1=LR
      WRITE(3,4278)I1,J1
 4278 FORMAT(20X,11HARC OF THE=,2I6)
      IB=LR
      CALL   NETFLW
      WRITE(3,9977)I9
 9977 FORMAT(11X,14HTHIS HOUSE IS=,I9)
      N1(I9)=0
      DO      8000    K=1,IK(LR)
      DO      8001    J=L1(LR),N2(LR)
 8001 C(LR,J,K)=C(LR,J+1,K)
      C(LR,N2(LR)+1,K)=0.0
 8000 CONTINUE
      L1(LR)=L1(LR)-1
      WRITE(3,1555)   N2(LR)
 1555 FORMAT(17HTHE REDUCEDN2.IS=,I5)
      SUM(LR)=SUM(LR)-D(J1)
      LH=0
      DO 494 K=1,IK(LR)
      IF((SUM(LR).GE.L(LR,K)).AND.(SUM(LR).LT.L(LR,K+1)))   GO    TO500
  494 CONTINUE
  500 LH=K+1
      LH=LH+1
      IF(LH.GT.IK(LR))    GO    TO    404
      DO 501 K=LH,IK(LR)
  501 Y(LR,K)=0.0
  404 CONTINUE
      I1=LR
      GOTO 400
  450 DO     7001     J1=1,IN
      WANS=0.0
      DO     7002    I1=1,IM
      DO     7003    K=1,IK(I1)
      IF(Y(I1,K)-1.0)7003,7100,7003
 7003 CONTINUE
      GOTO 7002
```

```
 7100 IB=I1
      CALL   NETFLW
      IF(LINK)7600,7600,7700
 7600 GO  TO    7002
 7700 WANS=1.0
      GO     TO    7001
 7002 CONTINUE
      IF(WANS)7300,7300,7001
 7300 GO    TO   650
 7001 CONTINUE
      IF(WANS)650,650,9666
C     *** END OF THIS PART ***
C     * THIS SIMPLIFICATION DETERMINES A MAXIMUM
C     BOUND ON THE COST REDUCTION FOR OPENING
C     A PLANT. IF THIS BOUND IS NEGATIVE THE PLANT
C     WILL BE FIXED CLOSED***.
 9666 DO     3200     I=1,IM
      DO        3400     K=1,IK(I)
      IF(Y(I,K).EQ.1.0)   GO    TO       600
 3400 CONTINUE
 3200 CONTINUE
      GO    TO          650
  600 I1=0
      DO 920 I=1,IM
      DO 621 K=1,IK(I)
  621 T(I,K)=0.0
  920 CONTINUE
  601 I1=I1+1
      IF(I1.GT.IM) GO TO 650
      DO 603 K=1,IK(I1)
      IF(Y(I1,K).EQ.1.0) GO TO 601
  603 CONTINUE
      K3=1
 3023 DO      808  K=K3,IK(I1)
      IF(Y(I1,K))808,808,690
  808 CONTINUE
      GO    TO   601
  690 LK=K
      LR=I1
      DO 602 I=1,IM
  602 L1(I)=0
      DO 604 J1=1,IN
      SMALL=99999999.
      I1=LR
      IB=I1
      CALL NETFLW
      IF(LINK)1604,1604,606
 1604 K20=I1
      DO     3071    I1=1,IM
      IB=I1
      CALL   NETFLW
      IF(LINK)3071,3071,3072
 3072 L1(I1)=L1(I1)+1
 3071 CONTINUE
```

```
      I1=K20
      GO   TO   604
  606 RH=C(LR,L1(LR)+1,LK)
      LH=L1(LR)+1
      DO 607 I1=1,IM
      IB=I1
      CALL NETFLW
      IF(LINK)607,607,609
  609 L1(I1)=L1(I1)+1
      DO 610 K1=1,IK(I1)
      IF(I1.EQ.LR.AND.K1.EQ.LK)GO   TO   610
      IF(Y(I1,K1)-1.0)610,699,610
  699 AMEGA=C(I1,L1(I1),K1)-RH
      IF(AMEGA)619,619,612
  612 X3=AMEGA
      GO TO 700
  619 X3=0.0
  700 R2=X3
      IF(R2-SMALL)620,620,610
  620 SMALL=R2
  610 CONTINUE
  607 CONTINUE
      IF(SMALL.EQ.99999999.0)   SMALL=0.
      DOM(LR,L1(LR),LK)=SMALL
  604 CONTINUE
      DO     622     J=1,N2(LR)
  622 T(LR,LK)=T(LR,LK)+DOM(LR,J,LK)
      T(LR,LK)=T(LR,LK)-F(LR,LK)
      IF(T(LR,LK))652,651,651
  652 Y(LR,LK)=0.0
  651 K=LK+1
      K3=K
      I1=LR
      IF(K3-IK(I1))3023,3023,601
      DO      3340       I=1,IM
 3340 WRITE(3,3339)(T(I,K),K=1,IK(I))
 3339 FORMAT(8HNEGATIVE,13F10.2)
  650 RETURN
      END
      FINISH
```

Appendix R5

The program in this Appendix transforms an integer matrix B into its Smith Normal Form via the algorithm developed by the author in the chapter 5.

To use the program one must prepare a data deck as described below:

1. Read in value for the following in conformance with the FORMAT statement number 600

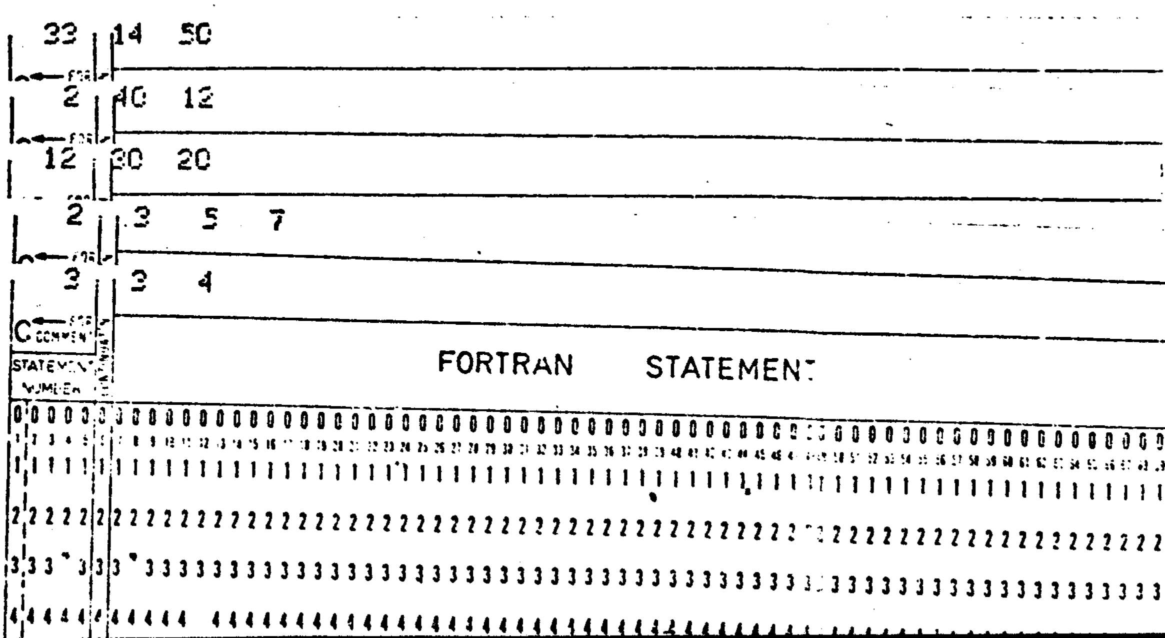
   IM=IN= number of rows or column of B.
   IK = number of the prime numbers for converting the given integer to its chinese representation.

2. Read the prime numbers which are used in converting the given integers to their chinese representation, into subscripted variable , B(I),I = 1 to IK, in conformance with FORMAT statement number 601.

3. Read the element of the matrix B, into the subscripted variable D(I,J),I = 1 to IM, J = 1 to IN, in conformance with the FORMAT statement number 603.

An example will illustrate the use of program.

$$B = \begin{bmatrix} 12 & 30 & 20 \\ 2 & 40 & 12 \\ 33 & 14 & 50 \end{bmatrix}$$



```
      MASTER SMITH
      INTEGER D(40,40),C(40,40,8),B(8),K2(40),K1(40) ,DETER
     1,S(8,19,19),P(8,19,19),D2(40)
     2,DELTA
      COMMON//D,C,IM,IN,KM,DELTA,DETER             /AREA1/IR,IC,K20/AREA2
     1N,I20/AREA3/JI4,MF1,KIM/AREA21/B,IK/AREA20/S,P
     1/AREA10/D2
      KM=1
C     READING NO ROW COLUMN PRIME NUMBER
      DETER=164
      READ(2,600)IM,IN,IK
  600 FORMAT(3I4)
      READ(2,601)(B(I),I=1,IK)
  601 FORMAT(20I4)
      DO  602  I=1,IM
  602 READ(2,603)(D(I,J),J=1,IN)
  603 FORMAT(20I4)
      CALL CHINESE
      KM=KM-1
      CALL TABLEAU
      WRITE(3,604)
  604 FORMAT(1X,23HTHIS IS INITIAL TABLEAU)
      DO 605  I=1,IM
  605 WRITE(3,606)(D(I,J),J=1,IN)
  606 FORMAT(1X,28I4)
      DO  607  I=1,IM
  607 WRITE(3,608)((C(I,J,K),K=1,IK),J=1,IN)
  608 FORMAT(10(8I2,1X))
      KM=0
      IY=IM-1
  699 KM=KM+1
      IF(KM-IY)610,610,611
  610 CALL GRETCD
      CALL CHECK
      IF(K20)612,612,777
  777 IF(IR-KM)613,680,613
  680 IF(IC-KM)613,682,613
  613 CALL POSIT
  682 CALL SUBTRACT
      GO TO 699
  612 CALL PRINROW
      IF(KIM)614,614,615
  615 CALL OBTAINROW
      IF(IR-KM)800,801,800
  801 IF(IC-KM)800,802,800
  800 CALL  POSIT
  802 CALL SUBTRACT
      GO TO 699
  614 CALL  PRINCOL
      IF(KIM)616,616,617
  617 CALL OBTAINCOL
      IF(IR-KM)400,401,400
  401 IF(IC-KM)400,402,400
  400 CALL POSIT
```

```
402 CALL SUBTRACT
    GO TO  699
616 WRITE(3,900)
900 FORMAT(1X,25HMORE SUBROUTINE IS NEEDED)
611 IF(D(KM,KM))693,692,692
693 D(KM,KM)=-D(KM,KM)
692 D2(KM)=D2(KM-1)*D(KM,KM)
    D(KM,KM)=D2(KM)
    WRITE(3,640)
640 FORMAT(20X,17HSMITH NORMAL FORM)
    DO  641  I=1,IM
641 WRITE(3,606)(D(I,J),J=1,IN)
    STOP
    END
    SUBROUTINE CHECK
    INTEGER  D(40,40),C(40,40,8),DELTA
    COMMON//D,C,IM,IN,KM,DELTA/AREA1/IR,IC,K20
    K20=0
    DO  400  I=KM,IM
    DO  401  J=KM,IN
    IF(IABS(D(I,J))-DELTA)401,402,401
402 IR=I
    IC=J
    K20=1
    GO TO  403
401 CONTINUE
400 CONTINUE
403 RETURN
    END
    SUBROUTINE GRETCD
    INTEGER D(40,40),C(40,40,8),B(8),N(8),DELTA  ,D2 (40 )
    COMMON//D,C,IM,IN,KM,DELTA/AREA2/N,I20/AREA21/B,IK
   1/AREA10/D2
    DELTA=1
222 DO  200  K=1,IK
    DO  201  I=KM,IM
    DO  202  J=KM,IN
    IF(C(I,J,K))200,202,200
202 CONTINUE
201 CONTINUE
    DELTA=DELTA*B(K)
    DO     211  I=KM,IM
    DO  210   J=KM,IN
210 D(I,J)=D(I,J)/B(K)
211 CONTINUE
    CALL CHINESE
    GO     TO    222
200 CONTINUE
    M1=(IM-KM)+1
    WRITE(3,204)M1,DELTA
204 FORMAT(2X,18HMATRIX IS OF ORDER,I3,3X,10HAND G.C.F=,I3)
    IF(KM-1)240,241,240
241 D2(1)=DELTA
    DELTA=1
    GO   TO   260
```

```
240 DP(KM)=DP(KM-1)*DELTA
    DELTA=1
260 RETURN
    END
    SUBRUUTINE CHINESE
    INTEGER  D(40,40),C(40,40,8),B(8)
    COMMON//D,C,IM,IN,KM/AREA21/B,IK
    DO     100   I=KM,IM
    DO    101   J=KM,IN
    DO   102   K=1,IK
    IB=IABS(D(I,J))/B(K)
    C(I,J,K)=IABS(D(I,J))-IB*B(K)
    IF(D(I,J))110,102,102
110 IF(C(I,J,K))102,102,112
112 C(I,J,K)=B(K)-C(I,J,K)
102 CONTINUE
101 CONTINUE
100 CONTINUE
    RETURN
    END
    SUBRUUTINE  TABLEAU
    INTEGER S(8,19,19),P(8,19,19),B(8)
    COMMUN/AREA20/S,P/AREA21/B,IK
    DO  660   K=1,IK
    S(K,1,1)=0
    P(K,1,1)=0
    IN1=1
    DO 661 I=2,B(K)
    I1=I-1
    IN1=IN1+1
    S(K,I,1)=S(K,I1,1)+1
    S(K,1,I)=S(K,I,1)
    P(K,I,1),P(K,1,I)=0
    DO  662  J=2,IN1
    J1=J-1
    P(K,I,J)=I1*J1
    IF(P(K,I,J)-B(K))670,670,671
671 IX=P(K,I,J)/B(K)
    P(K,I,J)=P(K,I,J)-IX*B(K)
670 P(K,J,I)=P(K,I,J)
    S(K,I,J)=S(K,I,J1)+1
    SAM=S(K,I,J)-B(K)
    IF(SAM)664,665,666
665 S(K,I,J),S(K,J,I)=0
    GO  TO  662
664 S(K,J,I)=S(K,I,J)
    GO TO  662
666 S(K,I,J)=SAM
    S(K,J,I)=S(K,I,J)
662 CONTINUE
661 CONTINUE
660 CONTINUE
    DO  680  K=1,IK
    DO  681  I=1,B(K)
681 WRITE(3,683)(S(K,I,J),J=1,B(K)),(P(K,I,J1),J1=1,B(K))
680 CONTINUE
```

```
683 FORMAT(5X,35I3)
    RETURN
    END
    SUBROUTINE  SUBTRACT
    INTEGER  D(40,40),C(40,40,8),MULT(8),SINK,SINK1,S(8,19,19),P(8,19,
   119),B(8),DELTA,DETER,D2(40)
    COMMON//D,C,IM,IN,KM,DELTA,DETER/AREA21/B,IK,MULT,MULT1,J60,ICAL,
   1CAL/AREA20/S,P/AREA10/D2
    IZ=KM+1
    IF(D(KM,KM)) 510,510,511
510 DO  512  I=KM,IM
    DO   800  K=1,IK
    IF(C(I,KM,K))800,800,801
801 C(I,KM,K)=B(K)-C(I,KM,K)
800 CONTINUE
512 D(I,KM)=-D(I,KM)
511 DO  500  J=IZ,IN
    IF(D(KM,J)) 514,500,515
514 DO  516  I=KM,IM
    DO   700  K=1,IK
    IF(C(I,J,K))700,700,702
702 C(I,J,K)=B(K)-C(I,J,K)
700 CONTINUE
516 D(I,J)=-D(I,J)
515 MULT1=D(KM,J)/D(KM,KM)
    J60=1
    CALL CONVERT
    DO  501  I=KM,IM
    D(I,J)=D(I,J)-MULT1*D(I,KM)
520 DO  502  K=1,IK
    SINK=P(K,MULT(K)+1,C(I,KM,K)+1)
    IF(SINK)503,503,504
503 C(I,J,K)=S(K,C(I,J,K)+1,SINK+1)
    GO  TO  502
504 SINK1=B(K)-SINK
    C(I,J,K)=S(K,C(I,J,K)+1,SINK1+1)
502 CONTINUE
501 CONTINUE
500 CONTINUE
    DO     550    I=KM+1,IM
550 D(I,KM)=0
    D(KM,KM)=D2(KM)
    DO         530          I=1,IM
530 WRITE(3,532)(D(I,J),J=1,IN)
532 FORMAT(1X,20I3)
    DO    663    K=1,IK
    DO    660    I=IZ,IM
660 WRITE(3,661)((C(I,J,K),K=1,IK),J=IZ,IN)
661 FORMAT(30I3)
663 CONTINUE
    RETURN
    END
    SUBROUTINE  POSIT
    INTEGER D(40,40),C(40,40,8),B(8),K1(40),K2(40)
    COMMON//D,C,IM,IN,KM,DELTA,K1,K2/AREA1/IR,IC,K20/AREA21/B,IK
```

```
      DO    700  I=KM,IM
      K1(I)=D(I,IC)
      D(I,IC)=D(I,KM)
      D(I,KM)=K1(I)
      DO  701   K=1,IK
      K2(I)=C(I,IC,K)
      C(I,IC,K)=C(I,KM,K)
  701 C(I,KM,K)=K2(I)
  700 CONTINUE
      DO    702    J=KM,IN
      K1(J)=D(IR,J)
      D(IR,J)=D(KM,J)
      D(KM,J)=K1(J)
      DO  703   K=1,IK
      K2(J)=C(IR,J,K)
      C(IR,J,K)=C(KM,J,K)
  703 C(KM,J,K)=K2(J)
  702 CONTINUE
      RETURN
      END
      SUBROUTINE  PRINROW
      INTEGER  D(40,40),C(40,40,8),B(8),N(8)
      COMMON//D,C,IM,IN,KM/AREA2/N ,I20/AREA3/JIM,MF1,KIM/AREA21/B,IK
      IY=IM
      KIM=0
      MF=KM-1
  920 MF=MF+1
      IF(MF-IY)991,991,910
  991 IF(D(MF,KM))911,920,911
  911 IN1=IN
      KM1=KM+1
      DO  930   J=KM1,IN1
      I5=0
      DO  940   K=1,IK
  904 IF(C(MF,KM,K))922,923,922
  923 J1=J
      IF(C(MF,J1,K))922,930,922
  922 J1=J
      IF(C(MF,KM,K)-C(MF,J1,K))940,925,940
  925 I5=I5+1
  940 CONTINUE
      I6=I5
      IF(I6-IK)941,930,941
  941 MF1=MF
      JIM=J1
      KIM=1
      WRITE(3,966)JIM,MF1,KIM
  966 FORMAT(3X,4HJIM=,I5,4HMF1=,I5,4HKIM=,I6)
      GO TO  910
  930 CONTINUE
      GO    TO    920
  910 RETURN
      END
      SUBROUTINE PRINCOL
      INTEGER D(40,40),C(40,40,8),B(8),N(8)
      COMMON//D,C,IM,IN,KM/AREA2/N ,I20/AREA3/JIM,MF1,KIM/AREA21/B,IK
```

```
      IY=IN
      KIM=0
      MF=KM-1
  920 MF=MF+1
      IF(MF-IY)991,991,910
  991 IF(D(KM,MF))911,920,911
  911 IM1=IM
      KM1=KM+1
      DO  930  I=KM1,IM1
      I5=0
      DO  940  K=1,IK
  904 IF(C(KM,MF,K))922,923,922
  923 I1=I
      IF(C(I1,MF,K))922,930,922
  922 I1=I
      IF(C(KM,MF,K)-C(I1,MF,K))940,925,940
  925 I5=I5+1
  940 CONTINUE
      I6=I5
      IF(I6-IK)941,930,941
  941 MF1=MF
      JIM=I1
      KIM=1
      GO  TO  910
  930 CONTINUE
      GO  TO  920
  910 RETURN
      END
      SUBROUTINE CONVERT
      INTEGER D(40,40),C(40,40,8),B(8),MULT(8)
      COMMON//D,C/AREA21/B,IK,MULT,MULT1,J60,ICAL,JCAL
      IF(J60)101,101,102
  101 IF(D(ICAL,JCAL))105,106,106
  105 D(ICAL,JCAL)=-D(ICAL,JCAL)
  106 DO  100 K=1,IK
  100 C(ICAL,JCAL,K)=D(ICAL,JCAL)-(D(ICAL,JCAL)/B(K))*B(K)
      GO  TO  104
  102 DO  103  K=1,IK
  103 MULT(K)=MULT1-(MULT1/B(K))*B(K)
  104 RETURN
      END
      SUBROUTINE OBTAINROW
      INTEGER D(40,40),C(40,40,8),B(8)
     1,MULT(8),SINK,SINK1   ,S(8,19,19),P(8,19,19)
     2,DETER,DELTA
      COMMON//D,C,IM,IN,KM,DELTA,DETER/AREA3/JIM,MF1/
     2AREA1/IR,IC,K20/AREA21/B,IK
     1,MULT,MULT1,J60,ICAL,JCAL/AREA20/S,P
      IF(D(MF1,KM))111,112,112
  111 DO  100  I=KM,IM
      DO   300  K=1,IK
      IF(C(I,KM,K))300,300,301
  301 C(I,KM,K)=B(K)-C(I,KM,K)
  300 CONTINUE
  100 D(I,KM)=-D(I,KM)
  112 IF(D(MF1,JIM))113,114,114
```

```
113 DO    101  I=KM,IM
    DO    400  K=1,IK
    IF(C(I,JIM,K))400,400,401
401 C(I,JIM,K)=B(K)-C(I,JIM,K)
400 CONTINUE
101 D(I,JIM)=-D(I,JIM)
114 IF(D(MF1,KM)-D(MF1,JIM))116,119,117
116 IA=D(MF1,JIM)/D(MF1,KM)
    IDIV=D(MF1,JIM)-IA*D(MF1,KM)
    IF(IDIV)118,119,118
119 IR=MF1
    IC=KM
    GO  TO  200
118 DO  120  I=KM,IM
    J60=1
    MULT1=IA
    CALL  CONVERT
    DO    220    K=1,IK
    SINK=P(K,MULT(K)+1,C(I,KM,K)+1)
    IF(SINK)221,221,222
221 C(I,JIM,K)=S(K,C(I,JIM,K)+1,SINK+1)
    GO    TO    220
222 SINK1=B(K)-SINK
    C(I,JIM,K)=S(K,C(I,JIM,K)+1,SINK1+1)
220 CONTINUE
    D(I,JIM)=D(I,JIM)-IA*D(I,KM)
120 CONTINUE
    GO  TO  114
117 IA=D(MF1,KM)/D(MF1,JIM)
    IDIV=D(MF1,KM)-IA*D(MF1,JIM)
    IF(IDIV)130,131,130
131 IR=MF1
    IC=JIM
    GO TO 200
130 DO  140  I=KM,IM
    J60=1
    MULT1=IA
    CALL  CONVERT
    DO    202   K=1,IK
    SINK=P(K,MULT(K)+1,C(I,JIM,K)+1)
    IF(SINK)203,203,204
203 C(I,KM,K)=S(K,C(I,KM,K)+1,SINK+1)
    GO    TO    202
204 SINK1=B(K)-SINK
    C(I,KM,K)=S(K,C(I,KM,K)+1,SINK1+1)
202 CONTINUE
    D(I,KM)=D(I,KM)-IA*D(I,JIM)
140 CONTINUE
    GO TO 114
200 RETURN
    END
    SUBROUTINE OBTAINCOL
    INTEGER D(40,40),C(40,40,8),B(8)
   1,MULT(8),SINK,SINK1   ,S(8,19,19),P(8,19,19)
   2,DELTA,DETER
    COMMON//D,C,IM,IN,KM,DELTA,DETER/AREA3/JIM,MF1/
```

```
     2AREA1/IR,IC,K20/AREA21/B,IK
     1,MULT,MULT1,J60,ICAL,JCAL/AREA20/S,P
      IF(D(KM,MF1))111,112,112
 111 DO    100  J=KM,IN
     DO    300  K=1,IK
     IF(C(KM,J,K))300,300,301
 301 C(KM,J,K)=B(K)-C(KM,J,K)
 300 CONTINUE
 100 D(KM,J)=-D(KM,J)
 112 IF(D(JIM,MF1))113,114,114
 113 DO  101   J=KM,IN
     DO  400   K=1,IK
     IF(C(JIM,J,K))400,400,401
 401 C(JIM,J,K)=B(K)-C(JIM,J,K)
 400 CONTINUE
 101 D(JIM,J)=-D(JIM,J)
 114 IF(D(KM,MF1)-D(JIM,MF1))116,119,117
 116 IA=D(JIM,MF1)/D(KM,MF1)
     IDIV=D(JIM,MF1)-IA*D(KM,MF1)
     IF(IDIV)118,119,118
 119 IR=KM
     IC=MF1
     GO  TO  200
 118 DO  120  J=KM,IN
     J60=1
     MULT1=IA
     CALL   CONVERT
     DO   202   K=1,IK
     SINK=P(K,MULT(K)+1,C(KM ,J,K)+1)
     IF(SINK)203,203,204
 203 C(JIM,J,K)=S(K,C(JIM,J,K)+1,SINK+1)
     GO   TO   202
 204 SINK1=B(K)-SINK
     C(JIM,J,K)=S(K,C(JIM,J,K)+1,SINK1+1)
 202 CONTINUE
     D(JIM,J)=D(JIM,J)-IA*D(KM,J)
 120 CONTINUE
     GO  TO  114
 117 IA=D(KM,MF1)/D(JIM,MF1)
     IDIV=D(KM,MF1)-IA*D(JIM,MF1)
     IF(IDIV)130,131,130
 131 IR=JIM
     IC=MF1
     GO  TO  200
 130 DO  140  J=KM,IN
     J60=1
     MULT1=IA
     CALL  CONVERT
     DO  220  K=1,IK
     SINK=P(K,MULT(K)+1,C(JIM,J,K)+1)
     IF(SINK)221,221,222
 221 C(KM,J,K)=S(K,C(KM,J,K)+1,SINK+1)
     GO   TO   220
 222 SINK1=B(K)-SINK
     C(KM,J,K)=S(K,C(KM,J,K)+1,SINK1+1)
 220 CONTINUE
     D(KM,J)=D(KM,J)-IA*D(JIM,J)
 140 CONTINUE
     GO  TO  114
 200 RETURN
     END
     FINISH
```

SUPPLEMENTARY MATERIAL

Group Theory and its Application in Mathematical Programming

INTRODUCTION

Recently considerable work has been done towards applying group theory to integer programming problems. While studying the literature [2,3,5] it became evident that theoretical background of the relevant aspects of group theory and that of integer programming are not available in one source document. In the present study we have, therefore, set out to provide some of the pertinent theoretical results which may form the basis of further study of this topic.

In the first part the concept of binary operation in a set, group, subgroup, normal subgroup of a group, quotient group, homomorphism, kernel of homomorphism, isomorphism, isomorphic, and direct sum group are briefly studied. In the second part the group minimization problem, and solving integer programming problems by means of the knapsack problem are discussed.

PART ONE

Definition. A "mapping" f, from S to T is a subset of ordered pairs of S x T (by S x T, we mean the Cartesian product of S and T) such that for $s \in S$, there is a unique $t \in T$, such that the ordered pair $(s,t) \in f$; this is shown as $f : S \to T$ or $S \xrightarrow{f} T$. If t is the image of s under f we shall represent this fact by $t = f(s)$. Indeed this notation is used instead of writing $(s,t) \in f$.

Definition. A binary operation in S is a mapping of S x S to T, denoted in this note by $\oplus$, therefore,

$$\oplus \ : \ S \times S \to T \text{ or } S \times S \xrightarrow{\oplus} T$$

If $t \in T$ is the image of ordered pair (s_1,s_2) under binary operation, we denote this by $t = s_1 \oplus s_2$ instead of $t = \oplus(s_1,s_2)$

Example: Addition is a binary operation in the set of real numbers, R, i.e.,

$$+ \ : \ R \times R \to R, \text{ or } R \times R \xrightarrow{+} R$$

which is defined $+(a,b) = c$, and is expressed in the form $(a+b) = c$.

1.1 Group

A nonempty set of elements G is said to form a group, if in G there is defined a binary operation such that the following holds:

(1) $a,b\in G$, implies that $a \oplus b\in G$, i.e., the set G is closed under this operation.

(2) $a,b,c\in G$ implies that,

$$(a \oplus b) \oplus c = a \oplus (b \oplus c).$$

(3) There exists an element $e\in G$ whereby

$$a \oplus e = e \oplus a = a \text{ for all } a\in G$$

(e is called the identity element in G).

(4) For every $a\in G$ there exists an element $(-a)\in G$, such that $(-a) \oplus a = a \oplus (-a) = e$, (the existence of inverse in G).

Definition. A group G is said to be 'abelian' if for every $a,b\in G$, $a \oplus b = b \oplus a$.

Example 1. The set of all square nonsingular matrices of order two under the multiplication defined for matrices, forms a group. This group of course is not abelian, since (1)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (1)$$

does not hold for all a,b,c,d,a',b',c',d'.

Example 2. The set of all integers under the addition forms an abelian group.

Example 3. Let p be a real number, δ, a positive integer, and x the remainder when p is divided by δ; that is $p = m\delta+x$, where m is an integer, and $0 \leq x < \delta$. We say that x is congruent p modulo δ and write this relation

$$x \equiv p(\text{mod}\,\delta)$$

For example $7\equiv 43 \pmod{12}$. We will prove that the set $S=\{0,1,2,3,\ldots,\delta-1\}$ under the addition with modulo δ forms an abelian group. For this purpose we should verify that all the conditions (1) to (4) hold, and if $p,q\in S$, then we have $(p+q)(\mathrm{mod}\,\delta) \equiv (q+p)(\mathrm{mod}\,\delta)$.

(1) Let $p,q\in S$, therefore $0\leq p<\delta$, and $0\leq q<\delta$
if $p+q<\delta$, then $p+q\in S$, but
if $p+q>\delta$, we can write $p+q=\delta+r_3$ or $r_3\equiv(p+q)(\mathrm{mod}\,\delta)(0\leq r_3<\delta)$
i.e., $r_3\in S$ or $(p+q)(\mathrm{mod}\,\delta)\in S$.

By similar arguments the other statements can be verified, therefore, the set S under the binary operation defined in it forms an abelian group.

Example 4. This example proves very useful in applying group theory to integer programming. Let $S=\{g_0,g_1,\ldots g_{\delta-1}\}$ and binary operation in S be defined as :

$$g_i \oplus g_j = g_{(i+j)(\mathrm{mod}\,\delta)} .$$

From Example 3 it follows that the set S under the binary operation defined as above forms an abelian group and let this be denoted by $G(\delta)$.

Definition. A subset H of a group G is said to be a subgroup of G if under the binary operation defined in G, H itself forms a group.

Theorem 1. A nonempty subset H of the group G is a subgroup of G if and only if,

(1) $a,b\in H$ implies that $a \oplus b\in H$.

(2) $a\in H$ implies that $(-a)\in H$.

Theorem 2. If H is nonempty finite subset of a group G and H is closed under the binary operation defined in G, then H is a subgroup of G.

A natural characteristic of a group is the number of elements it contains. known as the order of G, and denoted by $|G|$. This number is of course most interesting when it is finite, in that case G is a finite group.

Definition. If G is a group, and $a \varepsilon G$ define

$$\underbrace{a \oplus a \oplus , \dots \oplus a}_{m \text{ times}} = ma ,$$

and also define $e = oa$

The order of $a \varepsilon G$ is the least positive integer m such that $ma = e$, and will be denoted by $|a|$. It can be easily shown that if G is a finite group, and $a \varepsilon G$, then $|a| \, \Big| \, |G|$ i.e., $|a|$ divides $|G|$.

Definition. A group G is said to be 'cyclic' if there exists an element in G, say a, such that

$$|a| = |G| .$$

For the group $G(\delta)$, Example 4, $\delta g_1 = g_0$, therefore $|g_1| = \delta$, i.e., $G(\delta)$ is a cyclic group.

Definition. If H is a subgroup of group G, and $a \varepsilon G$, then $H \oplus a = \{h \oplus a | h \varepsilon H\}$ is called a 'right coset' of H in G. Similarly the left coset of H in G can be defined.

Normal subgroup of a group. A subgroup N of G is said to be a normal subgroup of G if, every left coset of N in G is also a right coset of N in G; i.e., for every $a \varepsilon G$, $N \oplus a = a \oplus N$. Of course when G is an abelian group each subgroup of G is a normal subgroup of it; but the converse is not always true. Note it can be shown that for $a,b \varepsilon G$, and $a \neq b$ either $N \oplus a = N \oplus b$ or $(N \oplus a) \cap (N \oplus b) = \phi$, and furthermore $\bigcup_{a \varepsilon G} (N \oplus a) = G$.

Let G/N (N is a normal subgroup of G) denote the collection of right cosets of N in G (that is the element of G/N are certain subsets of G) and we use the binary operation of set G to yield for us a binary operation in G/N. For this binary operation we claim that

$X,Y \varepsilon G/N$ implies that $X \oplus Y \varepsilon G/N$; for $X = N \oplus a, Y = N \oplus b$ for some $a,b \varepsilon G$, and $X \oplus Y = (N \oplus a) \oplus (N \oplus b) = N \oplus (a \oplus b) = N \oplus c \varepsilon G/N$, where $c = a \oplus b$. (a)

The other three conditions can be verified as above; therefore the set G/N under binary operation $\oplus$, forms a group, which is called "quotient group" or <u>factor group of G by N.</u>

N.B. If G is abelian, then G/N is abelian as well.

<u>Example.</u> Let G be the group of integers under addition, and N be the set of all multiples of 3. We shall write the coset of N in G as N + a rather than as $N \oplus a$, since the binary operation in G is addition. Consider three cosets N, N+1, N+2. We claim that these are all the cosets of N in G. For $a \varepsilon G$, $a=3b+C$ where $b \varepsilon G$ and C=0,1, or 2 (C is remainder of a on division by 3). Thus $N+a = N+3b+C = (N+3b)+C = N+C$, since $3b \varepsilon N$. Thus every coset is, as we stated one of N, N+1 or N+2, and

$$G/N = \{N, N+1, N+2\}$$

How do we add elements in G/N? Our formula $(N \oplus a) \oplus (N \oplus b) = N \oplus (a \oplus b)$ translates into:

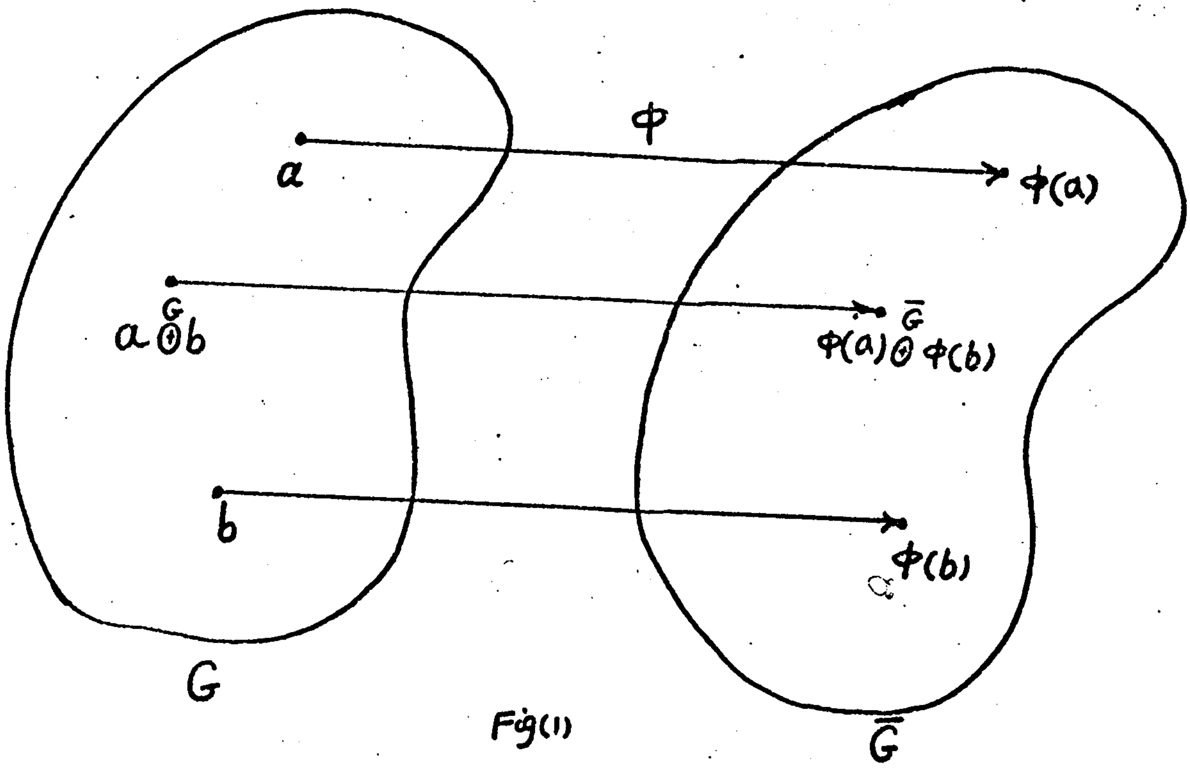
$(N+1) + (N+2) = N+(1+2) = N+3 = N$ since $3 \varepsilon N$;
$(N+2) + (N+2) = N+(2+2) = N+4 = (N+3)+1 = N+1$, and so on.
Clearly what we did for 3 we could emulate for any integer n.

1.2 Homomorphism

A mapping ϕ from a group G into a group $\bar{G}$ is said to be a "homorphism" if for all $a,b \varepsilon G$

$$\phi(a \overset{G}{\oplus} b) = \phi(a) \overset{\bar{G}}{\oplus} \phi(b), \qquad \text{(b)}$$

by $\underset{G}{\oplus}$, and $\underset{\bar{G}}{\oplus}$ we mean the binary operation defined in G, and $\bar{G}$ respectively. Fig. 1 is an illustration of the relationship in (b).



Fig(1)

Example: Let G be the group of all real numbers under addition, and let $\bar{G}$ be the group of nonzero real numbers with the binary operation multiplication of real numbers. Define the mapping,

$$\phi : G \to \bar{G} \text{ by } \phi(a) = 2^a.$$

In order to verify that this mapping is a homomorphism we must check if

$$\phi(a + b) = \phi(a).\phi(b)$$

i.e., we must check if $2^{a+b} = 2^a.2^b$, which is indeed true. Since 2^a is always positive the image of ϕ is not all of $\bar{G}$, so ϕ is a homomorphism of G into $\bar{G}$, but not onto $\bar{G}$; cf p. 12 [1].

Example: If G is a group, N a normal subgroup of G; define the mapping ϕ from G into G/N by

$$\phi(x) = N \oplus x$$

for all $x \in G$. Then ϕ is a homorphism of G onto G/N.

Kernel of a homomorphism.

If ϕ is a homomorphism of G into $\bar{G}$, the "kernel" of ϕ, K_ϕ, is defined by

$$K_\phi = \{x | x \in G \text{ and, } \phi(x) = e, \text{ where } e \text{ is the identity element of } \bar{G}\}$$

It can be easily shown that if ϕ is a homomorphism of G into $\bar{G}$ with kernel K_ϕ, K_ϕ is a normal subgroup of G.

Definition: A homomorphism ϕ of G into $\bar{G}$ is said to be an "isomorphism" if ϕ is one-to-one.

Definition: Two groups G, G^* are said to be isomorphic, if there is an isomorphism of G onto G^*. In this case we write $G \simeq G^*$.

Two isomorphic groups are the same mathematical object, only their representations are different.

Lemma. Let ϕ be a homomorphism of G onto $\bar{G}$ with kernel K, then

$$G/K \simeq \bar{G};$$ see 1 ; p. 50.

This result is frequently used in the present study.

1.3 Direct Sum Groups

Let $S = \{(a_i, b_k) | i=0,1,2,\ldots,\delta_1-1, k=0,1,2,\ldots,\delta_2-1\}$ and $(a_i, b_k) \oplus (a_j, b_\ell) = (a_{(i+j)(\text{mod}\,\delta_1)}, b_{(k+\ell)(\text{mod}\,\delta_2)})$ then it can be shown that S under this operation forms an abelian group of order $\delta_1 \cdot \delta_2$, this is defined as $G(\delta_1, \delta_2)$, where $G(\delta_1, \delta_2)$ is said to be the direct sum group of $G(\delta_1)$ and $G(\delta_2)$ and expressed as $G(\delta_1, \delta_2) = G(\delta_1) \oplus G(\delta_2)$. In exactly the same way a direct sum group $G(\delta_1, \delta_2, \ldots, \delta_m)$ of order $\prod_{i=1}^{m} \delta_i$ can be defined as

$$G(\delta_1, \delta_2, \ldots, \delta_m) = G(\delta_1) \oplus G(\delta_2) \ldots \oplus G(\delta_m).$$

For example consider G(2,3). The elements of G(2,3) are the ordered pairs $g_{0,0} = (a_0, b_0)$, $g_{1,0} = (a_1, b_0)$, $g_{0,1} = (a_0, b_1)$, $g_{1,1} = (a_1, b_1)$, $g_{0,2} = (a_0, b_2)$, and $g_{1,2} = (a_1, b_2)$. Note in G(2,3),

$$g_{i,k} \oplus g_{j,\ell} = g_{(i+j)(\text{mod}2),(k+\ell)(\text{mod}3)}.$$

Example: The groups G(6) and G(2,3) are isomorphic. The correspondence between the elements, are set out graphically in Fig. (2).

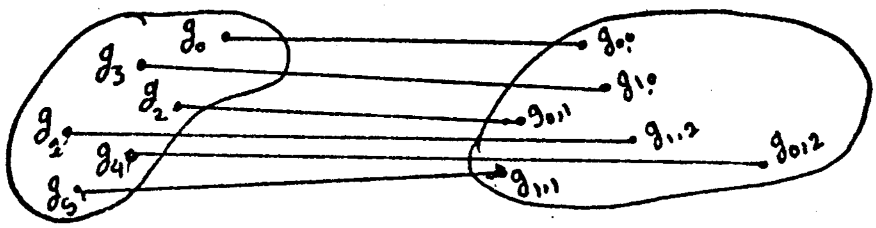


Fig (2)

For example choose two elements in G(6), say g_1, g_2, which correspond to $g_{1,2}$, $g_{0,1}$ respectively.

$$g_1 \oplus g_2 = g_3,$$

$$g_{1,2} \oplus g_{0,1} = g_{1,0},$$

as it is shown g_3 corresponds to $g_{1,0}$. Similarly it can be checked that, the mapping Φ is a homomorphism of G onto G(2,3), and is one-to-one, therefore G(6) and G(2,3) are isomorphic.

Note. The group G(6) is cyclic, therefore G(2,3) is also cyclic.

Let $\mathbb{R}^m$, $\mathbb{Z}^m$ be the sets of column vectors with m components of real and integer entries respectively. These sets under the usual operation of addition form abelian groups. Clearly the group $\mathbb{Z}^m$ is a subgroup of the group $\mathbb{R}^m$.

Let $A = [a_{ij}]$ be an mxn matrix expressed as a set of column vectors $A = [a_1, a_2, \ldots, a_n]$ where any vector a_j is made of integer components. Define the set

$$\{A\} = \{x | x = \sum_{j=1}^{n} p_j a_j, p_j \text{ integer}, j = 1,2,\ldots,n\} ,$$

then this set {A} under addition forms an abelian group. If the matrix A contains an mxm identity matrix, then $\mathbb{Z}^m = \{A\}$. In general the group {A} is a subgroup of the group $\mathbb{Z}^m$. Assume that A is of rank m, and $B = [b_1, b_2, \ldots, b_m]$ is an mxm submatrix of A, also of rank m. We can consider the abelian group formed by the set {B} defined as follows:

$$\{B\} = \{y | y = \sum_{j=1}^{m} p_j b_j, p_j \text{ integer}, j=1,2,\ldots,m\} ,$$

then {B} forms a group under addition. In general group {B} is a subgroup of group {A}. Let

$$\alpha = [\alpha_{ij}] = B^{-1}A, \text{ and}$$

$$\{\alpha\} = \{z | z = \sum_{j=1}^{n} p_j \alpha_j, p_j \text{ integer } j=1,2,\ldots,n\}$$

The set {α} under the usual binary operation of addition forms an abelian group.

Let $\bar{\alpha}$ be the set made up of the fractional part of α such that

$$\alpha = \bar{\alpha} + L.$$

Then the set $\{\bar{\alpha}\} := \{\bar{w} | w = \sum_{j=1}^{n} p_j\bar{\alpha}_j, p_j \text{ integer}, j=1,2,\ldots,n, \text{ and } \bar{w} = w \bmod(1)\}$,

forms a group which is generated by the fractional parts of column vectors of α under addition (mod 1). Thus given a group $\{\Lambda\}$, B^{-1} is used to map $\{\Lambda\}$ into the group $\{\alpha\}$, and let ϕ be the mapping from group $\{\alpha\}$ to the group $\{\bar{\alpha}\}$. This can be indicated as follows:

$$\{A\} \xrightarrow{B^{-1}} \{\alpha\} \xrightarrow{\phi} \{\bar{\alpha}\}$$

The composite mapping f defined as $f = \phi B^{-1}$ may be proved to be a homomorphism from $\{A\}$ into $\{\bar{\alpha}\}$. Let $a_1, a_2 \varepsilon \{A\}$, and $\bar{\alpha}_1, \bar{\alpha}_2 \varepsilon \{\bar{\alpha}\}$, such that,

$$\phi B^{-1}(a_1) = \bar{\alpha}_1 \text{ and } \phi B^{-1}(a_2) = \bar{\alpha}_2.$$

From earlier definition it follows

$$B^{-1}(a_1) = L + \bar{\alpha}_1,\ B^{-1}(a_2) = L + \bar{\alpha}_2,$$

or $$a_1 = BL + B\bar{\alpha}_1, a_2 = BL + B\bar{\alpha}_2,$$

$$a_1 + a_2 = BL + B(\bar{\alpha}_1 + \bar{\alpha}_2),$$

so $$B^{-1}(a_1+a_2) = L + (\bar{\alpha}_1+\bar{\alpha}_2).$$

Now applying the mapping ϕ,

$$\phi B^{-1}(a_1+a_2) = \bar{\alpha}_1 + \bar{\alpha}_2 = \phi B^{-1}(a_1) + \phi B^{-1}(a_2).$$

so the composite mapping $f = \phi B^{-1}$ is a homomorphism of group $\{A\}$ onto group $\{\bar{\alpha}\}$, see below.

<u>Theorem.</u> The quotient group $\{A\}/K_f$ is isomorphic with the group $\{\bar{\alpha}\}$, i.e., $\{A\}/K_{\phi B^{-1}} \cong \{\bar{\alpha}\}$ The proof is straightforward, because ϕB^{-1} is a homomorphism $\{A\}$ onto $\{\bar{\alpha}\}$.

Theorem. The kernel of $\phi B^{-1} = \{B\}$. To see this suppose $a \varepsilon K_{\phi B^{-1}} \subseteq \{A\}$; therefore $\phi B^{-1}(a) = 0$*, we know that

$a = \sum_{j=1}^{n} p_j a_j$ for some p_j, $j=1,2,\ldots,n$,

so $$\phi B^{-1}(a) = \phi B^{-1}\left(\sum_{j=1}^{n} p_j a_j\right) = 0,$$

so $$B^{-1}\left(\sum_{j=1}^{n} p_j a_j\right) = \Gamma, \quad (\Gamma \text{ is an integer vector})$$

or

$$\sum_{j=1}^{n} p_j a_j = B\Gamma = \sum_{j=1}^{m} b_j \Gamma_j.$$

So

$$a \varepsilon K_{\phi B^{-1}} \text{ implies } a \varepsilon \{B\} \text{ therefore } K_{\phi B^{-1}} \subseteq \{B\}$$

Similarly we can prove that $\{B\} \subseteq K_{\phi B^{-1}}$, thus,

$$K_{\phi B^{-1}} = \{B\}.$$

Let us study the structure of the group $\mathbb{Z}^m/\{B\}$.
Since $\mathbb{Z}^m$ is m-dimensional, the unit vectors $e_i (i=1,2,\ldots,m)$ serve as a basis for $\mathbb{Z}^m$, and certainly for the group $\{B\}$; where

$b_j = \sum_{i=1}^{m} b_{ij} e_i$, $(j=1,2,\ldots,m)$. Therefore the matrix B expresses every

$$\begin{array}{c} \\ e_1 \\ e_2 \\ \\ \\ \\ e_m \end{array}
\begin{array}{c} \begin{array}{cccccc} b_1 & b_2 & \cdot & \cdot & \cdot & b_m \end{array} \\ \left[\begin{array}{cccccc} b_{11} & b_{12} & \cdot & \cdot & \cdot & b_{1m} \\ b_{21} & b_{22} & \cdot & \cdot & \cdot & b_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ b_{m1} & b_{m2} & \cdot & \cdot & \cdot & b_{mm} \end{array}\right] \end{array} = B$$

b_i in term of e_j. By changing the basis vector b_i, and the unit vector e_j, we can diagonalize the matrix by a series of elementary transformations such that it is of the form

* (0 is the identity element of the group $\{\bar{\alpha}\}$)

$$\begin{array}{c} \\ e'_1 \\ e'_2 \\ \\ \\ e'_m \end{array} \begin{array}{c} \begin{array}{cccc} b'_1 & b'_2 & & b'_m \end{array} \\ \left[\begin{array}{cccc} \delta_1 & & & \\ & \delta_2 & 0 & \\ & \cdot & & \\ & 0 & & \\ & & & \delta_m \end{array}\right] \end{array} = \hat{B} ,$$

where δ_i is a divisor of $\delta_{i+1}(i=1,2,\ldots,m-1)$. The matrix $\hat{B}$ is called the "Smith Normal Form" of the matrix B (the process of transforming a matrix into Smith Normal Form can be found in [3] or in our report [4]. Since the process does not change the determinant $D=|\det B|=\det B=\delta_1\delta_2\ldots\delta_m$, and $b'_i = \delta_i \cdot e'_i$ $(i=1,2,\ldots,m)$ where $e'_i(i=1,2,\ldots,m)$ are basis for $\mathbb{Z}^m$.

It is well known that $\mathbb{Z}^m$ can be expressed as a direct sum group.

$$\mathbb{Z}^m = Ze'_1 \oplus Ze'_2 \oplus \ldots \oplus Ze'_m$$

and $b'_i(i=1,2,\ldots,m)$ are basis for the group {B}, therefore it can also be expressed as a direct sum group

$$\{B\} = Zb'_1 \oplus Zb'_2 \oplus \ldots \oplus Zb'_m \text{ (Z any integer)}$$

$$= Z\delta_1e'_1 \oplus Z\delta_2e'_2 \oplus \ldots \oplus Z\delta_me'_m$$

Hence the quotient group $\mathbb{Z}^m/\{B\}$ may be expressed as,

$$\mathbb{Z}^m/\{B\} = \frac{Ze'_1 \oplus Ze'_2 \oplus \ldots \oplus Ze'_m}{Ze'_1\delta_1 \oplus Ze'_2\delta_2 \oplus \ldots \oplus Ze'_m\delta_m}$$

This group and the group

$$\frac{Z}{Z\delta_1} \oplus \frac{Z}{Z\delta_2} \oplus \ldots \oplus \frac{Z}{Z\delta_m}$$

are isomorphic, therefore $\mathbb{Z}^m/\{B\}$ and the direct sum of m cyclic groups are isomorphic, and the ith cyclic group is of order $\delta_i(i=1,2,\ldots,m)$. Further the order of this group is

$$D = \delta_1\delta_2\ldots\delta_m$$

Now, $$D = \left| \mathbb{Z}^m / \{B\} \right| = \left| \mathbb{Z}^m / \{A\} \right| \left| \{A\} / \{B\} \right|$$

as $\{B\} \subseteq \{A\}$, so $\left| \{A\} / \{B\} \right|$ divides D. If A should contain identity matrix $\left| \mathbb{Z}^m / \{A\} \right| = 1$.

In the theorem of page 10 it is proved that thekernel K_f of the homomorphism is the group {B} , and Z^m is {A} if A contains an identity matrix. Therefore from the theorem in page 9 it follows

$Z^m/\ \{B\}$ is isomorphic with the group $\{\bar{\alpha}\}$, and they should be of the same order, i.e., $|\ \{\bar{\alpha}\}| = D$.

The important result concerning the group $\{\bar{\alpha}\}$constructed out of the fractional elements of the matrix $\alpha = B^{-1}A$, obtained under the operation of addition modulo 1, and the direct sum group constructed out of the diagonal elements of the Smith Normal form may be summarized as:

Two groups $\{\bar{\alpha}\}$ and $\frac{Z}{Z\delta_1} + \ldots + \frac{Z}{Z\delta_m}$ are isomorphic.

PART TWO

2.1. Knapsack problem

This is the classical problem that a hiker faces in deciding how to pack his knapsack.

Let a_j be the weight of the jth item, c_j be the value of the jth item, x_j be the number of items of type j that the hiker carries with him, and let b denote the total weight limitation. Then the hikers problem may be expressed as

$$\max \sum_{j=1}^{n} c_j x_j, (c_j \text{ integer}, j = 1,2,\ldots,n),$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_j x_j \leq b (a_j, b \text{ positive integer}), \tag{1}$$

$$x_j \geq 0, \text{ and integer.}$$

The knapsack problem can be solved by any of the general Integer Linear Programmi (ILP) algorithms, however it has only one constraint and more direct algorithms may be used for its solution. A general ILP in bounded variables can be transformed into a knapsack problem as well [3].

2.2. Group knapsack problem

Consider the finite abelian group G, and $H = \{g_{i_1},\ldots,g_{i_n}\}$ a subset of G, and let the set Q be made up of the subscripts such that $Q = \{i_1,i_2,\ldots,i_n\}$. Consider the problem of finding non-negative integers t_{i_j} $j=1,2,\ldots,n$ such that

$$\bigoplus_{j\varepsilon Q} t_j g_j = t_{i_1} g_{i_1} \oplus \cdots \oplus t_{i_1} g_{i_n} = g^{*} \varepsilon G. \tag{2}$$

Now for integer p and m such that

$$p + m|g_k \geq 0 \tag{3}$$

$$(p + m|g_k|)g_k = pg_k + m|g_k|g_k = pg_k + mg_o = pg_k$$

Thus if one component t_k of the solution of (2) is given by $t_k = p$, $k \notin Q$ there are solutions with $t_k = p + m|g_k|$ for all m such that $p + m|g_k| \geq 0$. Related to the pure integer programming problem, there exists the group knapsack problem

$$f_n(g^*) = \min \sum_{j \in Q} t_j d_j \tag{4}$$

subject to

$$\bigoplus_{j \in Q} t_j g_j = g^*$$

$$t_j \geq 0 \text{ and integer, } j \in Q.$$

where d_j are given for all $j \in Q$. Note that $d_j \geq 0$ implies that if (4) has a solution, it has an optimal solution with $t_j \leq |g_j|$ for all $j \in Q$. However, if there exists j^* such that $d_{j^*} < 0$, and (4) has a solution, then it is unbounded. It can be shown that this problem can be solved as knapsack problem, and the relation between (4) and (1) is as follows :

By introducing a slack variable x_{n+1} to the constraint in (1), (1) can be written in the form

$$\max \ x_o = \sum_{j=1}^{n+1} c_j x_j \tag{5}$$

subject to

$$\sum_{j=1}^{n+1} a_j x_j = b, \tag{5a}$$

$$x_j \geq 0, \text{ and integer, } j = 1,2,\ldots,n+1$$

where $a_{n+1} = 1$ and $c_{n+1} = 0$.

Assume that the variables in (5) are ordered so that

$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n} \geq \frac{c_{n+1}}{a_{n+1}}$. An optimal solution to the LP corresponding to (5) is given by $x_1 = \frac{b}{a_1}$, $x_o = \frac{c_1}{a_1} b$, $x_j = 0$ for $j \geq 2$.

From (5a) x_1 may be expressed as

$$x_1 = \frac{b}{a_1} - \frac{1}{a_1} \sum_{j=2}^{n+1} a_j x_j. \tag{5b}$$

Substituting (5b) in (5) gives

$$\max x_0 = \frac{c_1 b}{a_1} - \frac{1}{a_1} \sum_{j=2}^{n+1} (c_1 a_j - a_1 c_j). \tag{5c}$$

Now maximizing (5c) subject to (5a) is equivalent to

$$\min \sum_{j=2}^{n+1} (c_1 a_j - c_j a_1) x_j = \sum_{j=2}^{n+1} d_j x_j \text{ say,} \tag{6}$$

$$\text{subject to } \frac{b}{a_1} - \sum_{j=2}^{n+1} \frac{a_j x_j}{a_1} \geq 0, \text{ and integer} \tag{7}$$

$$x_j \geq 0, \text{ and integer } j = 2,3,\ldots,n+1$$

(We have assumed that $\frac{b}{a_1}$ is not integer).

If b is large enough, so that x_1 is certain to be positive in an optimal solution, the non-negativity on x_1 can be dropped. Equation (7) can be written in the form

$$\sum_{j=2}^{n+1} \frac{a_j x_j}{a_1} = \frac{b}{a_1} \pmod 1,$$

or

$$\sum_{j=2}^{n+1} a_j x_j = b \pmod{a_1}, \tag{8}$$

or

$$\sum_{j=2}^{n+1} p_j x_j = p_0 \pmod{a_1},$$

where $p_j = a_j \pmod{a_1}$ $j = 2,3,\ldots, n+1$, and

$$p_0 = b \pmod{a_1}.$$

Note that (8) can be written as a group equation over the group $G(a_1)$.

$$\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 6 & 2 & 0 \\ \hline & & & 1 & 0 & 0 \\ & & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array} \sim \begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 2 & 1 \\ 0 & 0 & 1 & 6 & -4 & 0 \\ \hline & & & 1 & -1 & 0 \\ & & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array}$$

$$\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 2 & 1 \\ -6 & 0 & 1 & 0 & -4 & 0 \\ \hline & & & 1 & -1 & 0 \\ & & & 0 & 1 & 0 \\ & & & 0 & 0 & 1 \end{array} \sim \begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & -2 \\ -6 & 0 & 1 & 0 & 0 & 4 \\ \hline & & & 1 & 0 & 1 \\ & & & 0 & 0 & -1 \\ & & & 0 & 1 & 0 \end{array}$$

$$\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ -6 & 0 & 1 & 0 & 0 & 4 \\ \hline & & & 1 & 0 & 1 \\ & & & 0 & 0 & -1 \\ & & & 0 & 1 & 2 \end{array}$$

, Therefore

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -6 & 0 & 1 \end{pmatrix} \quad \hat{B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 1 & 2 \end{pmatrix}$$

Let $Q = \{2,3,4,\ldots,n+1\}$, and $g^* = g_{p_0}$, (it has been assumed that $d_j \neq d_i$ $i = j,i,j=2,\ldots,n+1$, otherwise see [3]), then (8) can be expressed as:

$$\bigoplus_{i\varepsilon Q} t_i g_i = g^* \tag{9}$$

where $t_i = x_j$ $i\varepsilon G$ and $i = p_j$.

Now for large b (7) may be written as the group knapsack problem

$$\text{minimize} \quad \sum_{i\varepsilon Q} d_i t_i,$$

subject to

$$\bigoplus_{i\varepsilon Q} t_i g_i = g^*, \tag{10}$$

$t_i \geq 0$ and integer, $i\varepsilon Q$.

Example: Consider the problem

$$\max x_0 = 10x_1 + 6x_2 + 3x_3 + 2x_4 + x_5,$$

subject to

$$6x_1 + 4x_2 + 3x_3 + 2x_4 + 5x_5 \leq 40 \tag{7a}$$

$$x_1,x_2,x_3,x_3,x_4,x_5 \geq 0 \text{ and integer.}$$

Introduce x_6 as slack variable, the problem (7a) may be expressed as

$$\max x_0 = 10x_1 + 6x_2 + 3x_3 + 2x_4 + x_5 + 0x_6, \tag{7b}$$

subject to $6x_1 + 4x_2 + 3x_3 + 2x_4 + 5x_5 + x_6 = 40,$

$$x_1,x_2,x_3,x_4,x_5,x_6 \geq 0, \text{ and integer.}$$

The optimal solution to the problem (7b) ignoring the integrality condition on the variables is $x_1 = \frac{40}{6}$, $x_0 = \frac{400}{6}$, and $x_i=0$ $i \geq 2$. Writing x_0 and x_1 in terms of the remaining variables one obtains

$$\begin{aligned} x_0 &= \frac{400}{6} - \frac{4}{6}x_2 - \frac{12}{6}x_3 - \frac{8}{6}x_4 - \frac{44}{6}x_5 - \frac{10}{6}x_6, \\ x_1 &= \frac{40}{6} - \frac{4}{6}x_2 - \frac{3}{6}x_3 - \frac{2}{6}x_4 - \frac{5}{6}x_5 - \frac{1}{6}x_6. \end{aligned} \tag{7c}$$

Ignoring the non-negativity condition on x_1 (7c) can be written as:

minimize $\quad 4x_2 + 12x_3 + 8x_4 + 44x_5 + 10x_6,$

subject to

$$4x_2 + 3x_3 + 2x_4 + 5x_5 + x_6 \equiv 4 \pmod 6, \qquad (7\text{c})$$

$$x_2, x_3, x_4, x_5, x_6 \geq 0,$$

and the group minimization problem corresponding to (7d) becomes

minimize $\quad 4t_4 + 12t_3 + 8t_2 + 44t_5 + 10t_1,$

subject to

$$g_4t_4 + g_3t_3 + g_2t_2 + g_5t_5 + g_1t_1 = g_4,$$

$$t_4, t_3, t_2, t_5, t_1 \geq 0 \text{ integer.}$$

The optimal solution is $t_4 = 1$, $t_i = 0$, $i \neq 4$. Therefore the optimal solution corresponding to (7a) is

$$x_2 = 1, x_1 = 6, x_3 = x_4 = x_5 = x_6 = 0, \text{ and } x_0 = 66.$$

The advantage of representing (5), (5a) as a group knapsack problem is that the order of the group is only a_1, where $a_1 \leq b$. In (5), (5a) the number of calculation is in a lose sense proportional to the magnitude of b and in (10) it is proportional to the magnitude of a_1.

2.3. Relation Between Integer Programming and the Group Knapsack Problem

Consider the pure integer program

$$\max \bar{c}\,\bar{x}$$
$$\text{subject to } \bar{A}\,\bar{x} \leq b, \qquad (11)$$
$$\bar{x} \geq 0, \text{ and integer}$$

where $\bar{A}$ is m x n integer matrix, b an integer m-vector, and $\bar{c}$ an integer n-vector. Alternatively the integer program (11) can be written as:

$$\max \ cx$$
$$\text{subject to } \ Ax = b, \qquad (12)$$

where A is an m x (m+n) integer matrix, c an (m+n) vector, and x is an (m+n) vector which includes the slack variables introduced to convert the inequalities (11) to equations of (12). Partitioning A as $[B,N]$, (12) may be rewritten as

$$\max C_B x_B + C_N x_N$$
$$\text{subject to } \ Bx_B + Nx_N = b, \qquad (13)$$
$$x_B, x_N \quad 0 \text{ and integer,}$$

where B is an $m \times m$ non-singular matrix. Expressing x_B in term of x_N, i.e., $x_B = B^{-1}b - B^{-1}Nx_N$, we can write (13) as:

$$\max C_B B^{-1}b - (C_B B^{-1}N - CN)\, x_N$$

subject to

$$x_B + B^{-1}Nx_N = B^{-1}b \tag{14}$$

$$x_B,\ x_N \geq 0 \text{ integers.}$$

If we consider (14) as a linear program, i.e., drop the integer restriction on x_B, and x_N and if B is the optimal basis of the linear program, then the optimum solution to the linear program is

$$x_B = B^{-1}b,\ x_N = 0,$$

where $C_B\, B^{-1}N - C_N \geq 0$. If $B^{-1}b$ happens to be an integer vector, then,

$$x_B = B^{-1}b,\ x_N = 0,$$

is obviously the optimum solution to integer program (14). When $B^{-1}b$ is not an integer vector, x_N must be increased from zero to some non-negative integer vector such that

$$x_B = B^{-1}b - B^{-1}N\, x_N \geq 0, \text{ and integer.}$$

This leads to two questions:

(1) Under what conditions $B^{-1}b - B^{-1}Nx_N \geq 0$ holds ?

(2) When is $B^{-1}b - B^{-1}Nx_N$ an integer vector ?

To start with, consider the relaxation of (14) in which the nonnegativity condition $x_B \geq 0$ and the integer restriction are omitted; the problem becomes

$$\max C_B B^{-1}b - (C_B B^{-1}N - C_N)\, x_N$$

subject to

$$x_B = B^{-1}(b - Nx_N), \tag{15}$$

$$x_N \geq 0.$$

In the n-dimensional space over which the components of x_N are defined the feasible solutions to (15) correspond to the cone defined by the non-negative orthant. For this reason, LP's of form (15) are called LP's over cone: and

$$\max C_B B^{-1}b - (C_B B^{-1}N - C_N)\, x_N$$

$$x_B = B^{-1}(b - Nx_N)\ x_B \text{ integer} \tag{16}$$

$$x_N \geq 0, \text{ integer}$$

are ILP's in which the corresponding LP's are over cones. Thus problems in the form of (16) are called ILP's over cone or ILPC's. An ILPC is a relaxation of the corresponding ILP in which, for a given B the non-negativity restriction on x_B are omitted.

It will be seen that an ILPC is considerably easier to solve than the corresponding ILP. In fact, an ILPC can be solved as a group knapsack problem over a direct sum group which is of order $D = |\det B|$.

To answer the second question stated above, note that:

The condition x_B be an integer vector is equivalent to

$$B^{-1}(b-Nx_N) \quad 0(\mathrm{mod}\ 1).$$

Eliminating the constant term from the objective function of (16) and changing from max into min we obtain the ILPC statement

$$\min\ (C_B B^{-1}N-C_N)\ x_N$$

subject to

$$B^{-1}Nx_N = B^{-1}b(\mathrm{mod}\ 1), \tag{17}$$

$$x_N \geq 0,\ \text{integer},$$

where $(C_B B^{-1}N-C_N) \geq 0$. Assume that x_N^* is an optimal solution to (17) so that the corresponding value of x_B is

$$x_B^* = B^{-1}(b-Nx_N^*).$$

We can think of $B^{-1}Nx_N^*$ as a minimum cost correction to $B^{-1}b$ that yields x_B^* integer. If the correction is such that $x_B^* \geq 0$, then (x_B^*, x_N^*) is the optimal solution to ILP (12).

For this reason it is intuitively appealing to choose B such that $B^{-1}b \quad 0$. Thus one generally works with an ILPC and an associated optimal basis B. However, the theory and the algorithm apply to any ILPC generated by a dual feasible basis.

2.4. Equivalent ILPC Representation

Our objective is to transform the ILPC constraints of (16), by changing variables, into a form more suitable for analysis. Some classical result on the solution of simultaneous linear equations in integers provides the background. Let

$$S = \{x | Bx = b,\ x\ \text{integer}\}$$

$$T = \{y | \bar{B}y = \bar{b},\ y\ \text{integer}\}$$

where B and $\overline{B}$ are m^{th}-order matrices, and b, $\overline{b}$, m-dimensional integer vectors. If there is a one-to-one correspondence between the elements of S and T given by y = px, where p is an m^{th}-order integer matrix, then Bx = b, x integer and $\overline{B}y = \overline{b}$,y integer are said to be equivalent representation. To obtain a representation equivalent to Bx = b,x integer, the following theorem proves to be useful.

Theorem 1. Let E be an m^{th}-order unimodular integer matrix, then for every integer vector y there exists a unique x such that y = Ex.

Theorem 2. Let R and C be m^{th} order unimodular integer matrices, and let $RBC = \hat{B}$, then Bx = b,x integer, and $\hat{B}y = Rb$,y integer are equivalent representation.

Proof: Multiplying Bx = b on the left by R yields RBx = Rb. (19)

Since C^{-1} exists it is also true that

$$RB = \hat{B}\,C^{-1} \tag{20}$$

From (19), (20) it follows that

$$\hat{B}\,C^{-1}x = RBx = Rb. \tag{21}$$

Let

$$C^{-1}x = y \tag{22}$$

Note that C unimodular and integer implies that C^{-1} is unimodular, and integer. Using Theorem 1, it follows that there is a one-to-one correspondence between the integer value x and y in (22).

Substituting (22) into (21) yields

$$\hat{B}y = Rb. \tag{23}$$

Consider a particular integer solution $x_N = x_N'$, in (16), then

$$Bx_B = b - Nx_N' \text{ is integer} \tag{24}$$

An equivalent representation of (24) is therefore,

$$\hat{B}y = R(b - Nx_N'),\ y \text{ integer}. \tag{25}$$

Thus problem (16) can be made easier to analyse by obtaining a particular form for $\hat{B}$, say diagonal form. This form is simpler to handle than that of the original B matrix.

Example: Consider the ILP (taken from [3])

$$\max\ 2x_1 + x_2$$

subject to

$$x_1 + x_2 + x_3 = 5 \tag{26}$$

$$-x_1 + x_2 + x_4 = 0$$

$$6x_1 + 2x_2 + x_5 = 21,\ x_1,x_2,x_3,x_4,x_5 \geq 0, \text{ and integer}$$

The optimal solution to the corresponding LP is

$x_B = (x_1,x_2,x_4) = (\frac{11}{4}, \frac{9}{4}, \frac{1}{2})$ and $x_N = (x_3,x_5) = (0,0)$.

The ILPC corresponding to the optimal basis LP is

$$\max 2x_1 + x_2 \tag{26}$$

subject to

$$x_1 + x_2 + x_3 = 5$$
$$-x_1 + x_2 + x_4 = 0 \tag{26}$$
$$6x_1 + 2x_2 + x_5 = 21$$
$$x_3, x_5 \geq 0 \text{ and integer}$$
$$x_1, x_2, x_4 \text{ integer} \tag{26}$$

Thus (26)" can be written as follows:

$$\begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \\ 6 & 2 & 0 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_4 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \\ 21 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x_3 \\ x_5 \end{pmatrix} \tag{27}$$

Let R and C by any unimodular matrices such that RBC = B, and these unimodular matrices are chosen such that B is the Smith Normal Form of B. This is computed as illustrated below.

$$Rb = \begin{pmatrix} 5 \\ 5 \\ -9 \end{pmatrix} \qquad RNx_N = \begin{pmatrix} x_3 \\ x_3 \\ -6x_3 + x_5 \end{pmatrix}$$

Let $y = (y_1, y_2, y_3)$

$$\hat{B}y = \begin{pmatrix} y_1 \\ y_2 \\ 4y_3 \end{pmatrix} = \begin{pmatrix} 5 - x_3 \\ 5 - x_3 \\ -9 + 6x_3 - x_5 \end{pmatrix} \quad \text{or}$$

$$\begin{aligned} y_1 &= 5 - x_3 \\ y_2 &= 5 - x_3 \\ 4y_3 &= -9 + 6x_3 - x_5 \end{aligned} \tag{28}$$

Now (28) is a simpler representation than (27), in the sense that it immediately provides necessary and sufficient condition on (x_3, x_5) for y to be integer, and equivalently for x_B to be integer. In particular any (x_3, x_5) integer yields (y_1, y_2) integer, and y_3 is integer if and only if

$$-9 + 6x_3 - x_5 = 0 \pmod 4$$

or

$$3 + 2x_3 + 3x_5 = 0 \pmod 4 \tag{29}$$

or

$$2x_3 + 3x_5 = 1 \pmod 4$$

Thus $(x_3, x_5) \geq 0$ and integer yields a feasible solution to the ILPC if and only if (29) holds. Therefore the problem reduces to

$$\text{Min } Z = \frac{x_3}{2} = \frac{x_5}{4}$$

subject to $2x_3 + 3x_5 = 1 \pmod 4$, (30)

$x_3, x_5 \geq 0$, and integers,

and this is a group Knapsack problem over the group G(4).

2.5 Group Knapsack Representation of an ILPC

Suppose $\hat{B}$ is the Smith Normal Form of B, then

$$\hat{B}y = R(b-Nx_N),\ y \text{ integer}, \tag{31}$$

is equivalent to

$$Bx_B + Nx_N = b,\ x_B \text{ integer},$$

where $\hat{B} = RBC$, and $y = C^{-1}x$. Therefore (17) can be stated as:

$$\min z_o = (C_B \bar{B}^{-1} N - C_N)\, x_N$$

$$\hat{B}y = R(b-Nx_N) \tag{32}$$

$$y \text{ integer},$$

$$x_N \geq 0, \text{ and integer}$$

Denote the i^{th} row of R by $R_i (i = 1,2,\ldots,m)$ then the i^{th} row of $\hat{B}y = R(b-Nx_N)$ is

$$\delta_i y_i = R_i(b - Nx_N) \tag{33}$$

Since for x_N integer, the right-hand side of (33) is an integer, there exists an integer y_i satisfying (33) if and only if

$$R_i(b - Nx_N) = 0 \pmod{\delta_i} \text{ or} \tag{34}$$

equivalently

$$R_i Nx_N = R_i\, b \pmod{\delta_i}\ i = 1,2,\ldots,m.$$

Note that if $\delta_i = 1$, (34) is superfluos, in the sense that it is satisfied by any integer vector x_N.

If $\delta_1 = \delta_2 = \ldots = \delta_{k-1} = 1, \delta_k > 1$, then from (33), (31) can be stated as

$$\min Z_o = (C_B B^{-1} N - C_N) x_N$$

$$R_i N x_N = R_i b (\mathrm{mod}\ \delta_i)\ i = k, \ldots, m \tag{35}$$

$$x_N \geq 0, \text{ integer}$$

Note that $D > 1$ implies that $\delta_m > 1$.

Suppose k=m, so that there is exactly one constraint in (35). Let $x_N = (x_1, x_2, \ldots, x_r)$, $C_B\ B^{-1}\ a_j - C_j = d_j$, and $p_{mj} = R_m a_j (\mathrm{mod}\ \delta_m)$ $p_{mo} = R_m b (\mathrm{mod}\ {}_m)$; then (35) reduces to

$$\min Z_o = \sum_{j=1}^{r} d_j x_j$$

subject to

$$\sum_{j=1}^{r} p_{mj} x_j = p_{mo} (\mathrm{mod}\ \delta_m), \tag{36}$$

$$x_j \geq 0, \text{ integer } j = 1, 2, \ldots, r.$$

As stated earlier $\sum_{j=1}^{r} p_{mj} x_j = p_{mo} (\mathrm{mod}\ \delta_m)$ is a group equation over $G(\delta_m)$ and (36) is the corresponding group knapsack problem. The objective coefficient d_j can be transformed into integer by multiplying the objective function by D.

In the general case where $1 \leq k < m$ $p_{ij} = R_i a_i (\mathrm{mod}\ \delta_i)$, and $p_{io} = R_i b (\mathrm{mod}\ \delta_i)$

then (35) can be stated as

$$\text{minimize } z_o = \sum_{j=1}^{r} d_j x_j \tag{37}$$

subject to

$$\sum_{j=1}^{r} p_{ij} x_j = p_{io} (\mathrm{mod}\ \delta_i),\ i = k, \ldots, m,$$

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

The congruences of (37) taken together are equivalent to a group equation over the direct sum group $G(\delta_k, \delta_{k+1}, \ldots, \delta_m)$; this sum group is of order $|D| = \delta_k \delta_{k+1}, \ldots, \delta_m$ and (37) is a group knapsack problem. Represent p_{ij} by the group element $g_{p_{ij}}$ in $G(\delta_i)$, denote the element of the group $G(\delta_k, \delta_{k+1}, \ldots, \delta_m)$ by $g_{i_k, \ldots, i_m}$, where $0 \leq i_\ell \leq \delta_\ell$ $(\ell = k, \ldots, m)$. Therefore $g_{p_{kj}, \ldots, p_{mj}}$ is an element of the group $G(\delta_k, \ldots, \delta_m)$.

Example

$$\text{Max } z_o = -x_3 - 2x_4$$

Subject to

$$2x_1 + 4x_2 + x_3 = 12,$$

$$12x_1 + 8x_2 + x_4 = 60, \tag{38}$$

$$x_1,\ x_2,\ x_3,\ x_4 \geq 0, \text{ and integer.}$$

The optimal LP solution to this problem is given by $(x_1, x_2, x_3, x_4) = (\frac{9}{2}, \frac{3}{4}, 0, 0)$; and the optimal basis is

$$B = \begin{pmatrix} 2 & 4 \\ 12 & 8 \end{pmatrix}$$

$$\begin{array}{cc|cc} 1 & 0 & 2 & 4 \\ 0 & 1 & 12 & 8 \\ \hline & & 1 & 0 \\ & & 0 & 1 \end{array} \sim \begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 0 & 1 & 12 & -16 \\ \hline & & 1 & -2 \\ & & 0 & 1 \end{array} \sim \begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 6 & -1 & 0 & 16 \\ \hline & & 1 & -2 \\ & & 0 & 1 \end{array}$$

So $R = \begin{bmatrix} 1 & 0 \\ 6 & -1 \end{bmatrix}$ $C = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$, therefore $\delta_1 = 2, \delta_2 = 16$

$B = \begin{bmatrix} 2 & 0 \\ 0 & 16 \end{bmatrix}$, $D = 32$ and $N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Thus the ILPC associated with the optimal LP basis is given by $\max(C_B B^{-1} N - C_N) x_N$ or

$$\max x_3 + 2x_4$$

Subject to

$$R_i(b - Nx_N) \equiv 0 \pmod{\delta_i},\ i = 1, 2 \tag{39}$$

or $$12 - x_3 \equiv 0 \pmod{2}$$

$$12 - 6x_3 + x_9 \equiv 0 \pmod{16}$$

i.e.,

$$x_3 \equiv 0 \pmod{2} \qquad (39)''$$

$$6x_3 - x_4 \equiv 12 \pmod{16}$$

where $x_3, x_4 \geq 0$, and integer.

This can be represented as a group knapsack problem over the group G(2,16). In particular the coefficient of x_3 and x_4 corresponds to $g_{1,6}$ and $g_{0,15}$ respectively. Introduce the two integer variables t_1, t_2 corresponding to x_3 and x_4 respectively, and the group knapsack problem becomes

$$\text{minimize } t_1 + 2t_2$$

$$\text{subject to } t_1 g_{1,6} \oplus t_2 g_{0,15} = g_{0,12}, \qquad (40)$$

$$t_1, t_2 \geq 0, \text{ and integer.}$$

An optimal solution to (40) is $t_1 = 2$, $t_2 = 0$, this yields

$x_1 = 5$, $x_2 = 0$, $x_3 = 2$, $x_4 = 0$ which is a feasible solution to the ILP and therefore optimal.

To obtain the group knapsack problem the basis matrix B has been diagonalized into Smith Normal Form. However, it is clear that any unimodular R and C such that RBC = $\hat{B}$, where $\hat{B}$ is a diagonal matrix with positive integer diagonal elements $(\delta_1, \delta_2, \ldots, \delta_,)$ will yield a group knapsack problem. Smith Normal Form is preferred for computation because it yields the simplest representation of the group.

Consider now the problem of finding sufficient conditions for an ILPC to solve an ILP. The objective is to get an upper bound of Nx^*_N, where x^*_N is an optimal solution to ILPC(17). Then given an optimal basis B one looks for a sufficient condition such that

$$x^*_B = B^{-1} (b - Nx^*_N) \geq 0. \qquad (41)$$

If (41) holds, (x^*_B, x^*_N) solves the ILP (12).

An upper bound on $||Nx^*_N||$ (by $||Nx^*_N||$ we mean the Euclidean length of the vector Nx^*_N), may be obtained which depends on the coefficients of N and the magnitude of D. The bound is mainly of theoretical interest, since it is frequently very loose. The upper bound is derived from a bound on the variable in the corresponding group knapsack problem.

Consider the problem

$$\text{Min} \sum_{j\varepsilon Q} d_j t_j \tag{42}$$

subject to

$$\bigoplus_{j\varepsilon Q} t_j g_j = g^*, \quad t_j \geq 0, \text{ and integer,}$$

over the group G. It follows from earlier discussions (see if (42) has a feasible solution it has an optimal solution t^*, with $t^*_j \leq |G|-1$, for all $j\varepsilon Q$.

A stronger bound on t^*_j is given by,

$$\sum_{j\varepsilon Q} t^*_j \leq |G|-1.$$

References

1. Herstein, I.N. Topics in Algebra, Blaisdell Publishing Company, 19

2. Hu, T.C. Integer Programming and Network Flow, Addison-Wesley Publishing Company, 1970.

3. Garfinkel R.S. and Nemhauser G., Integer Programming, John Wiley & Sons, 1972.

4. Jahanshahlou G.R., and Mitra G., Chinese Representation of Integers and its Application in an Algorithm to Find the Smith Normal Form for an Integer Matrax. Technical Report TR/7 Dept. of Statistics & O.R. Brunel University.

5. Gomory R.E., on the Relation Between Integer and Non-Integer Solutions to Linear Programs, Proc.Nat.Acad.Sci., U.S.A., 53 (2), pp 260-265.