2 LINEAR PROGRAMMING AND ALGORITHMS

2.1 INTRODUCTION

While solving a linear programming problem, a systematic search is made to find a nonnegative vector X, which extremises a linear objective function[24,28, 29,30]

$$Z = \sum_{j=1}^{n} c_j x_j \qquad (2.1.1)$$

such that it satisfies a set of linear constraints of the form

$$\sum_{j=1}^{n} a_{ij} x_j \lesseqgtr b_i, \quad i = 1,2, \ldots m$$
$$x_j \geqslant 0 ; \quad j = 1,2, \ldots n \qquad (2.1.2)$$

The parameter c_j is the contribution/cost coefficient associated with unit output of the activity x_j. The b_i's are the limited available

resources. The problem is to find that vector X which satisfies the relationships (2.1.2) and extremises the linear objective function (2.1.1).

To find a solution to the above problem, revised simplex method can be used. The revised simplex method uses the same basic principles of the regular simplex method. But at each iteration, the entries in the entire tableau are never calculated. The relevant information for moving from one basic feasible solution to another is directly generated from the original set of equations. Later on, techniques such as multiple column selection, suboptimization procedure, crashing, advantageous starting basis, multiplex method etc., were developed to reduce computations in finding the optimal solution. Algorithms to solve the linear programming problem in a finite number of steps have also been developed in recent years[1,27]. Some heuristic approach[35,46] were tried by some authors to solve linear programming problems.

2.2 REVISED SIMPLEX PROCEDURE

2.2.1 Algorithm

The matrix model of the standard linear program[20] is

$$\begin{aligned} &\text{Maximize (Minimize)} && Z = CX \\ &\text{subject to} && AX \lesseqgtr P_o \\ & && X \geqslant 0 \end{aligned} \qquad (2.2.1)$$

where

$$\underset{(m \times n)}{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\underset{(m \times 1)}{P_o} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \; ; \; \underset{(n \times 1)}{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \; ; \; \underset{(1 \times n)}{C} = (c_1, c_2 \cdots c_n)$$

Let the columns corresponding to the matrix A be denoted by $P_1, P_2, \ldots P_n$ where

$$P_{1\ (m\ x\ 1)} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} \quad \text{and} \quad P_{n\ (m\ x\ 1)} = \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}$$

Let the vector X be partitioned as

$$X_{(n\ x\ 1)} = \begin{bmatrix} X_B \\ \hline X_N \end{bmatrix}$$

where X_B corresponds to the basic variables, and X_N to the nonbasic variables.

The revised simplex procedure solves repeatedly a set of linear algebraic equations of the form

$$B\ X_B = P_o \qquad (2.2.2)$$

and finds the value of the objective function as

$$Z = C_B\ X_B \qquad (2.2.3)$$

The set of equations (2.2.2) and (2.2.3) may be written in matrix notation as

$$\begin{bmatrix} 1 & -C_B \\ 0 & B \end{bmatrix} \begin{bmatrix} Z \\ X_B \end{bmatrix} = \begin{bmatrix} 0 \\ P_o \end{bmatrix} \qquad (2.2.4)$$

Let $M = \begin{bmatrix} 1 & -C_B \\ 0 & B \end{bmatrix}$

Rewriting the equation (2.2.4) as

$$M \begin{bmatrix} Z \\ X_B \end{bmatrix} = \begin{bmatrix} 0 \\ P_o \end{bmatrix}$$

The solution vector in terms of the matrix M is

$$\begin{bmatrix} Z \\ X_B \end{bmatrix} = M^{-1} \begin{bmatrix} 0 \\ P_o \end{bmatrix} \qquad (2.2.5)$$

M^{-1} exists if and only if the basis matrix B is nonsingular. Hence,

$$M^{-1} = \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \qquad (2.2.6)$$

The solution vector is given by[29]

$$\begin{bmatrix} Z \\ X_B \end{bmatrix} = \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ P_o \end{bmatrix}$$

$$= \begin{bmatrix} C_B B^{-1} P_o \\ B^{-1} P_o \end{bmatrix} \qquad (2.2.7)$$

The solution is based mainly in identifying B^{-1} for the current iteration. Though this procedure is not as compact as the Gauss elimination method, it offers the advantage of ultimately dealing only with the basis matrix B rather than with the entire tableau. This should have direct effect on the storage space which the problem occupies in the computer memory. Also it reduces the rounding off errors[13,14,15].

Let B_c and B_n be the basis matrices of the current and the next iteration respectively. B_n is obtained from B_c by replacing the leaving vector by the entering vector. If B_c^{-1} is known, then B_n^{-1} can be directly obtained without inverting B_n using the formula

$$B_n^{-1} = E B_c^{-1}$$

where E is a transformation matrix. M_n^{-1} can be obtained

from B_n^{-1}. Instead of computing B_n^{-1} first and then M_n^{-1}, M_n^{-1} can be obtained directly from M_c^{-1} by using a small modification in the matrix E. Let E_o be the modified version of E, then

$$M_n^{-1} = E_o M_c^{-1} \qquad (2.2.8)$$

where E_o has only one non-unit vector and m unit vectors.

Let the identity matrix I_m be represented as

$$I_m = (e_1, e_2, \ldots, e_i, \ldots, e_m)$$

where e_i is a unit column-vector with a unit element at the ith place and zero elements elsewhere. Let e_r be the unit vector representing the leaving variables x_r. Given x_j is the entering variable, then M_n^{-1} can be computed using the relation given in equation (2.2.8).

The E_o matrix may be written as

$$E_o = \begin{pmatrix} 1 & 0 & \ldots & 0 & -(z_j - c_j)/\alpha_{rj} & 0 & \ldots & 0 \\ 0 & e_1 & \ldots & e_{r-1} & \eta & e_{r+1} & \ldots & e_m \end{pmatrix}$$

$$\text{where} \quad \eta = \begin{bmatrix} -\alpha_{1j}/\alpha_{rj} \\ -\alpha_{2j}/\alpha_{rj} \\ \vdots \\ +\ 1/\alpha_{rj} \\ -\alpha_{r+1,j}/\alpha_{rj} \\ \vdots \\ -\alpha_{mj}/\alpha_{rj} \end{bmatrix}$$

and $\alpha_{ij} = B^{-1}P_j$; $i = 1,2, \ldots, m.$

The determination of η can be achieved systematically as shown below.

	Leaving Vector e_r		Entering Vector e_j		Multiplier		New Vector
rth place	$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$	$\longrightarrow$	$\begin{bmatrix} \alpha_{1j} \\ \vdots \\ \alpha_{rj} \\ \vdots \\ \alpha_{mj} \end{bmatrix}$	$\longrightarrow$	$\begin{bmatrix} -1/\alpha_{rj} \\ \vdots \\ 1/\alpha_{rj} \\ \vdots \\ -1/\alpha_{rj} \end{bmatrix}$	$\longrightarrow$	$\begin{bmatrix} -\alpha_{1j}/\alpha_{rj} \\ \vdots \\ 1/\alpha_{rj} \\ \vdots \\ -\alpha_{mj}/\alpha_{rj} \end{bmatrix}$

Thus M_n^{-1} is computed using M_c^{-1} and E_o.

The revised simplex procedure can be summarized in three basic steps.

Step 1 : Determination of the entering vector P_j. $(z_j - c_j)$ for all the variables are computed using the relation

$$(z_j - c_j) = (1, C_B B^{-1}) \begin{pmatrix} -c_j \\ P_j \end{pmatrix} \qquad (2.2.9)$$

The vector which is having the most negative $(z_j - c_j)$ should enter the basis if the objective is one of maximization. When all $(z_j - c_j) \geq 0$, the optimal solution is obtained.

Step 2 : Determination of the leaving vector P_r. Given the entering vector P_j and the current basis matrix B_c, the leaving vector must correspond to

$$\theta = \min_k \left[\frac{B_c^{-1} P_o}{\alpha_{kj}} , \quad \alpha_{kj} > 0 \right]$$

where $\alpha_{kj} = B_c^{-1} P_j \quad k = 1, 2, \ldots, m.$

If all $\alpha_{kj} \leq 0$, the solution is unbounded.

Step 3 : Determination of the next basic solution. E_o matrix is determined and M_n^{-1} is computed from M_c^{-1} as per the equation (2.2.8) and using the equation (2.2.5), the solution vector is obtained. Replace B_c by B_n and go to step 1.

2.2.2 Transformation Matrix

The transformation matrix, E_o used in the revised simplex procedure has m number of unit vectors and one non-unit vector and the total number of non-zero elements may at the most be $(2m + 1)$. It is this matrix which is responsible for the improvement in the solution from iteration to iteration which can easily be shown by splitting it into a unit matrix and a matrix which improves the solution. A typical transformation matrix is given below.

$$E = \begin{bmatrix} 1 & 0 & 0 & \cdots & -(z_j-c_j)/\alpha_{rj} & \cdots & 0 \\ 0 & 1 & 0 & \cdots & -\frac{\alpha_{1j}}{\alpha_{rj}} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \cdot & \cdot & \cdot & \cdots & \frac{1}{\alpha_{rj}} & \cdots & \cdot \\ \vdots & \vdots & \vdots & & \cdot & & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{\alpha_{mj}}{\alpha_{rj}} & \cdots & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdot & 1 & \cdots & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 0 & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 0 & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & -(z_j - c_j)/\alpha_{rj} & \cdots & 0 \\ 0 & 0 & 0 & \cdots & -\frac{\alpha_{1j}}{\alpha_{rj}} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ & & & & \frac{1}{\alpha_{rj}} - 1 & \cdots & 0 \\ 0 & 0 & 0 & & -\frac{\alpha_{r+1,j}}{\alpha_{rj}} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & 0 & & -\frac{\alpha_{mj}}{\alpha_{rj}} & \cdots & 0 \end{bmatrix}$$

$$E = [I] + [TR] \qquad (2.2.10)$$

where the matrix TR has only one column which has nonzero entries. Let $[Z, X_B]^T_{k-1}$ be the solution at the

end of (k-1)th iteration and $(Z, X_B)^T_k$ be the solution at the end of kth iteration.

$$\begin{bmatrix} Z \\ X_B \end{bmatrix}_k = [E_k]\,[M^{-1}_{k-1}] \begin{bmatrix} 0 \\ P_o \end{bmatrix}$$

$$= [E_k] \begin{bmatrix} Z \\ X_B \end{bmatrix}_{k-1}$$

$$= [I] \begin{bmatrix} Z \\ X_B \end{bmatrix}_{k-1} + [TR] \begin{bmatrix} Z \\ X_B \end{bmatrix}_{k-1}$$

$$= \begin{bmatrix} Z \\ X_B \end{bmatrix}_{k-1} + [TR] \begin{bmatrix} Z \\ X_B \end{bmatrix}_{k-1}$$

The TR matrix does two operations on $[Z, X_B]^T_{k-1}$

i) it performs a translation on m variables (including Z) and

ii) it does a rotation on the remaining variable.

In other words, variables common to the (k-1)th and kth iteration undergo only a translation and the entering variable makes a rotation and changes the basis. On the whole, the solution vector from one iteration to another

undergoes a translation and a rotation and therefore the TR matrix may be said to perform the operation of a simultaneous translation and rotation.

2.2.3 Preservation of Linear Independence Property

A basic feasible solution for a linear programming problem can be obtained only if the basis matrix B is non-singular in the m dimensional requirement space[29]. A square matrix of dimension (m x m) is non-singular if and only if it possesses a set of m linearly independent vectors. A set of vectors $P_1, P_2 \ldots P_m$ is said to be linearly independent, if and only if, for all real λ_j

$$\sum_{j=1}^{m} \lambda_j P_j = 0 \qquad (2.2.11)$$

This is possible only if all $\lambda_j = 0$; where λ_j are all scalars. If

$$\sum_{j=1}^{m} \lambda_j P_j = 0$$

for some $\lambda_j \neq 0$, then the vectors are said to be linearly dependent[11].

The conventional simplex procedure always assures the linear independence of the entering vector. This should be true because, the starting basis matrix is the identity matrix and the new basis matrix differs from the initial basis matrix only in one column. As long as the pivot element of the entering vector is nonzero, the new basis matrix is bound to be nonsingular. The reason for this is that E_o, the transformation matrix is none other than the inverse of the matrix with m unit vectors and only one non unit vector, whose pivot element is non zero as otherwise it cannot enter the basis. This property can be formalised using linear algebra as follows:

$$\text{Let, } H = \begin{bmatrix} 1 & 0 & \cdots & z_j - c_j & \cdots & 0 \\ 0 & 1 & \cdots & \alpha_{1j} & \cdots & 0 \\ 0 & 0 & & \cdot & & \cdot \\ 0 & 0 & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ 0 & \cdot & \cdots & \alpha_{rj} & \cdots & 0 \\ 0 & \cdot & & \cdot & & \cdot \\ 0 & \cdot & & \cdot & & \cdot \\ 0 & 0 & \cdots & \alpha_{mj} & \cdots & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ r+1\text{th row} \\ \\ \\ \end{matrix}$$

and E_o is the inverse of the matrix H. The determinant of H is the pivot element, α_{rj} itself. Hence the inverse of H is

$$E_o = H^{-1} = \begin{bmatrix} 1 & 0 & \cdots & -(z_j-c_j)/\alpha_{rj} & \cdots & 0 \\ 0 & 1 & \cdots & -\alpha_{1j}/\alpha_{rj} & \cdots & 0 \\ 0 & 0 & & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & \frac{1}{\alpha_{rj}} & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & . & \cdots & -\frac{\alpha_{mj}}{\alpha_{rj}} & \cdots & 1 \end{bmatrix} \quad r+1\text{th row}$$

Consider the following matrix,

$$W = \begin{bmatrix} 1 & 0 & \cdots & a_{oj} & \cdots & 0 \\ 0 & 1 & \cdots & a_{1j} & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{rj} & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{mj} & \cdots & 1 \end{bmatrix}$$

If the columns of W are to be linearly independent then no set of m+1 scalars whose one or more values are all other than zero could be found.

$$
\begin{aligned}
\lambda_1 + \quad & + \quad \lambda_{r+1}\, a_{oj} && = 0 \\
\lambda_2 \quad & + \quad \lambda_{r+1}\, a_{1j} && = 0 \\
& \vdots && \vdots \\
& + \quad \lambda_{r+1}\, a_{rj} && = 0 \\
& \vdots && \vdots \\
& \quad \lambda_{r+1}\, a_{mj} \quad + \quad \lambda_{m+1} && = 0
\end{aligned}
$$

From the above, $\lambda_{r+1} = 0$ if $a_{rj} \neq 0$. This, when substituted in the other equations, makes all the other $\lambda_i = 0$, $i = 1,2, \ldots r-1, r, r+2, \ldots, m+1$. Hence W is a nonsingular matrix. The transformation matrix E_o, used to get the next inverse of M matrix from the current inverse, is none other than the inverse of the matrix W, whose nonsingularity has just been established.

2.3 REVISED SIMPLEX PROCEDURE USING THE VARIANT

In the revised simplex procedure a variable which produces the maximum rate of change in the objective function is selected to enter the basis. Alternatively a variable which produces greatest change in the objective function may also be selected to enter the basis, i.e., the product of $\bar{c}_k$ and x_k. For this purpose, step 1 of the revised simplex procedure is modified as

$$\bar{c}_k x_k = \max_j / \min_j [\beta_j (z_j - c_j)] \qquad (2.3.1)$$

where

$$\beta_j = \min_i \left[\frac{(B^{-1}P_{oi})}{\alpha_{ij}} ; \ \alpha_{ij} > 0 \right] \qquad (2.3.2)$$

$$\alpha_{ij} = [B^{-1}P_j]_i$$

The $\bar{c}_k x_k$ obtained using equation (2.3.1) gives the greatest change[4]. All the other steps remain the same.

2.4 CRASH ALGORITHMS

When a linear programming problem is having both lower and upper bound inequalities, introduction of slack variables in all the inequality constraints make the starting solution infeasible. To find solution to the above type of problems, either artificial variable technique or the phase I and phase II techniques are employed. To achieve the feasible solution, from the starting infeasible solution, is known as crashing[7]. This involves making a sequence of iterations, omitting the step of finding the entering variable using the $(z_j - c_j)$ criterion. Instead simply each variable is considered in turn. One variant is as follows. Start with a basis consisting of slacks on all equality constraints and artificials on all others. Make one pass through the matrix, performing a forward transformation on each structural variable in turn and making it basic without creating any new infeasibilities if this

a) removes an artificial variable from the basis
b) reduces the number of infeasibilities
c) has a negative reduced cost.

If any of these conditions holds, then B^{-1} matrix is updated and repeat the step, for the next variable; otherwise abandon this variable and try the next one[7].

2.5 MULTIPLE COLUMN SELECTION ALGORITHM

A related improvement over the crash algorithm is known as multiple column selection[7,36]. The step by step procedure of this method is as follows.

Step 1 : The $(z_j - c_j)$ values of all the variables are computed. The solution is checked for optimality. If it is nonoptimal go to step 2. If the solution is optimal verify feasibility. In case of infeasibility, invoke the dual simplex procedure to remove infeasibility. The processing is terminated when the solution is feasible.

Step 2 : Two or more promising vectors are selected.

Step 3 : All the selected columns and P_o are updated by premultiplying them by B^{-1}.

Step 4 : The ratio test[36] is performed on all of them to determine the entering vector which makes the greatest change in the objective function. The ratio test is defined as follows.

$$\theta_j = \min_i \left[\frac{(B^{-1}P_o)_i}{(B^{-1}P_j)_i} \right] ; (B^{-1}P_j)_i > 0$$

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

$$\beta = \min/\max_j (z_j - c_j)\, \theta_j$$

Step 5 : Update the B^{-1} matrix and the other selected columns. Return to step 4 to see whether another variable can be gainfully introduced. This process is continued until no further progress is possible with the selected variables and the control is transferred to step 1.

In the above algorithm if the variables removed from the basis in that particular pass are also included in step 4, this process is referred to as suboptimisation[36].

2.6 MULTIPLEX METHOD

The multiplex method[42,43] is suggestive to select at a time more than one variable to construct a basis. This procedure cuts across the feasible region in search of a vertex unlike the simplex procedure which religiously sticks to the periphery of the constrained set. The step by step procedure of this method is described below.

Step 1 : Select a point in the interior of the feasible region, that is, such a combination of the values of the basis variables as will make all the variables effectively positive. If necessary S(λ) method can be used to find that point[42,43].

Step 2 : From this initial point move by some method in a direction which is influenced by the direction of the preference vector. The logarithmic potential method can be used to decide about the direction of movement.

Step 3 : Determine the value of λ for which it is largest and the solution is feasible. This will give the optimum solution. Otherwise make the new point as the starting point and repeat step 2.

At each point, selection of the number of variables which are independent is arbitrary and hence the number of the variables plays an important role in finding the optimal solution.

2.7 A POLYNOMIAL ALGORITHM IN LINEAR PROGRAMMING

Hacijan's polynomial algorithm is a procedure for deciding whether the system given by

$$AX \leqslant b \qquad (2.7.1)$$

is consistent. In addition, if the system is consistent it finds the co-ordinates of point satisfying all of the inequalities, or it at least determines them within a small margin of error[10].

For a system given by the equation (2.7.1), let

$$L = \left[\left[\sum_{i=1}^{m} \sum_{j=1}^{n} \log_2 (|a_{ij}| + 1) + \sum_{i=1}^{m} \log_2 (|b_i| + 1) + \log_2 mn \right]\right] + 1$$

where [[x]] denotes the greatest integer less than x. The execution of the algorithm involves $N = 16Ln^2$ iterations.

The principle of the algorithm is like the traditional method of catching fish in a net; casting the net over such a large region that some of what is wanted must be inside; then gradually decreasing the volume of the net. When the volume is sufficiently reduced it becomes obvious whether or not anything has been caught[10]. Similar to this, a large ellipsoid is formed in the beginning. Depending upon the discrepancies in the solution vector the size of the ellipsoid is gradually decreased. When there is no discrepancy in the solution vector, the optimal solution is reached.

2.8 CONCLUSION

The various algorithms available for solving linear programming problems were discussed. The multiplex method developed by Ragner Frisch[42,43] is also given. The lack of fixity in these algorithms has led to wide variety of procedures like double pricing, etc., but such efforts have been constantly

discouraged saying that these methods which will bring more than one variable into the basis in each iteration may not lead to any computational saving. In the next chapter an algorithm which will bring more than one variable in each pass is discussed.