CHAPTER 3

ALGORITHMS FOR MODEL REDUCTION

3.1 INTRODUCTION

Algorithms for identifying and disposing redundancies (rows and columns) *apriori* in a linear programming problem have been developed before seeking final optimal solution. While structuring a problem, model builders sometimes because of inadvertency and anxiety not to miss any likely active constraint sprinkle redundancies, which influence the computational efficiency. An attempt is made in this dissertation to reduce a given model as far as possible to an effective and equivalent size so that:

i) the time taken to solve a problem is minimized,
ii) when the size of the problem is made small, the memory requirements are minimized,
iii) formulation errors are revealed while reducing a given model and
iv) a reduced model can be solved with simple algorithms.

3.2 MOTIVATION

The motive for detecting redundancies is to reduce the original model size and thus the cost of solution. It should be clear that for this to happen, the number of redundancies identified and removed must
comparison with the solution cost. Perhaps more importantly the presence of a large number of redundancies is a sign of inefficiency in the problem formulation or model building, which may lead, to wastage of computational effort while finding solution. Column redundancy is also considered for identification, which has the virtue of keeping those variables off the bay, and saves computation and memory.

After a survey of the work done, a model reduction algorithm has been developed by incorporating the following modifications for further investigation.

(i) The redundant constraint detection algorithm has been improved.
(ii) A new selection criterion is suggested for minimization problem.
(iii) The algorithm is modified so as to handle a generalized linear programming model.
(iv) A new approach has been developed to identify the primal redundant variables by identifying its corresponding dual redundant constraints and
(v) A tie-breaking rule is introduced.

3.3 APPLICATION OF SIMPLEX AND MULTIPLEX ALGORITHMS

The original and reduced linear programming models are solved using Simplex and Multiplex algorithms. Simplex algorithm is a proven, well-founded and established procedure to solve linear programming problems. It is essentially a univariate search technique and exhibits slow convergence properties. Multiplex algorithm uses a multivariate search technique and possesses the property of rapid
convergence. It brings a cluster of variables into the basis and reduces the number of iterations considerably. This algorithm also minimizes the tendency of variables popping in and out of the basis. The dual Simplex or the dual Multiplex algorithm can remove infeasibility if any, in the solution.

A brief outline of the algorithms to solve linear programming problems is explained in this section.

### 3.3.1 Simplex algorithm

Linear programming is usually identified with the development of the Simplex algorithm developed in 1947 by George B Dantzig. The Simplex algorithm starts with a non-optimal basis and then updates the basis via the pricing rule until it reaches the optimal basis. Each iteration of the method brings one variable into the basis and sends one out of it. In other words, variables are switched in and out of the basis. The Simplex algorithm may be viewed as a procedure for systematically estimating the set of constraints that are binding at a solution. More specifically, if the current vertex is feasible but not optimal, the next adjacent feasible vertex is chosen at which the objective function has an improved value. In 1953, Dantzig modified the Simplex algorithm so that the implementation of it on a computer could be done more efficiently. This is called the revised Simplex algorithm.

#### 3.3.1.1 The procedure of the algorithm

Consider the standard LP model

$$\text{Extremize } Z = C X$$

subject to $$A X (\leq, =, \geq) b; \quad X \geq 0$$

(3.1)
where

$$A = \begin{pmatrix}
    a_{11} & a_{12} & \ldots & \ldots & a_{1n} \\
    a_{21} & a_{22} & \ldots & \ldots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    a_{m1} & a_{m2} & \ldots & \ldots & a_{mn}
\end{pmatrix}_{(m \times n)}$$

and $a_{ij}$ is the demand resource coefficient of the $i^{th}$ constraint and the $j^{th}$ activity, $c_j$ is the contribution/cost coefficient associated with the output of the activity $x_j$, and $b_i$'s are the limited available resources in the form of men, money, machine and materials.

Let the columns corresponding to the matrix $A$ be denoted by $p_1, p_2, p_3, \ldots, p_n$ where

$$p_j = \begin{pmatrix}
    a_{1j} \\
    a_{2j} \\
    \vdots \\
    \vdots \\
    a_{mj}
\end{pmatrix}$$

Let the vector $X$ be partitioned as

$$X = \begin{pmatrix}
    X_B \\
    X_N
\end{pmatrix}_{(n \times 1)}$$
where the sub-vectors $X_b$ and $X_n$ correspond to the basic and non-basic variables respectively.

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

$$B X_b = b \quad (3.2)$$

and finds the value of the objective function

$$Z = C_B X_b \quad (3.3)$$

The set of equations (3.2) and (3.3) may be put in matrix notation as

$$\begin{pmatrix} 1 & -C_b \\ 0 & B \end{pmatrix} \begin{pmatrix} Z \\ X_b \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad (3.4)$$

Let

$$M = \begin{pmatrix} 1 & -C_b \\ 0 & B \end{pmatrix}$$

The solution vector in terms of the matrix $M$ is

$$\begin{pmatrix} Z \\ X_b \end{pmatrix} = M^{-1} \begin{pmatrix} 0 \\ b \end{pmatrix} \quad (3.5)$$

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

\[
M^{-1} = \begin{pmatrix}
1 & CbB^{-1} \\
0 & B^{-1}
\end{pmatrix}
\]  

(3.6)

The solution vector is given by

\[
\begin{pmatrix}
Z \\
X_B
\end{pmatrix} = \begin{pmatrix}
1 & CbB^{-1} \\
0 & B^{-1}
\end{pmatrix} \begin{pmatrix}
0 \\
b
\end{pmatrix} = \begin{pmatrix}
CbB^{-1}b \\
B^{-1}b
\end{pmatrix}
\]  

(3.7)

The solution is based mainly on identifying \(B^{-1}\) for the current iteration.

The basis matrix \(B\) is different from the preceding or succeeding basis matrix by only one column and so is the matrix \(M\). Let the matrices \(M_c\) and \(M_n\) correspond to the current and next iterations of the revised Simplex procedure. The \(M_n^{-1}\) may be obtained from \(M_c^{-1}\) using linear algebra and this eliminates the computation for direct inversion.

The procedure for determining \(M_n^{-1}\) from \(M_c^{-1}\) is summarized below.

Let the identity matrix \(I\) be represented as

\[ I = (e_1, e_2, \ldots, e_{m+1}) \]

where \(e_i\) is a unit vector of dimension \((m+1) \times 1\) with a unit element at the \(i^{th}\) place. Let \(x_j\) and \(x_r\) be the entering and leaving variables at the start of any iteration. The \(M_n^{-1}\) can then be computed using the relationship

\[
M_n^{-1} = E_0 M_c^{-1}
\]  

(3.8)
where $E_0$ is a transformation matrix of order $(m + 1) \times (m + 1)$ with $m$ unit vectors and one non-unit vector corresponding to the entering variable.

The non-unit column ($\eta$) of the $E_0$ matrix is given by

$$
\eta = \begin{pmatrix}
- \alpha_{0j} / \alpha_{cj} \\
- \alpha_{1j} / \alpha_{cj} \\
. \\
. \\
+ 1 / \alpha_{cj} \\
. \\
. \\
- \alpha_{nj} / \alpha_{cj}
\end{pmatrix}
$$

where $\alpha_{0j} = z_j - c_j$

and $\alpha_{ij} = (B^{-1}p_j)_i ; \quad i = 1, 2, \ldots, m$

The $E_0$ matrix is now formed as

$$
E_0 = (e_1, e_2, \ldots, e_r, \eta, \ldots, e_{m+1})
$$

where $e_1, e_2, \ldots, e_r \ldots e_{m+1}$ are all unit vectors and $\eta$ alone is the non-unit vector corresponding to the entering variable. Thus the $M_{n^{-1}}$ is constructed using the $M_{e^{-1}}$ and the transformation matrix $E_0$.

The various steps involved in the Simplex procedure are as follows:

**step 1: Identify the entering vector $p_j$**

$$
(z_j - c_j) = C_b B^{-1} p_j - c_j = (1, C_b B^{-1}) \begin{pmatrix}
-c_j \\
p_j
\end{pmatrix}
$$

(3.9)
The most promising vector enters the basis. If the objective is one of maximization and all \((z_j - c_j) \geq 0\), then the optimal solution is obtained.

**step 2:** Identify the leaving vector \(p_r\) when the entering vector is \(p_j\) and the current basis matrix is \(B_c\).

\[
\theta = \min_k \left\{ \frac{\left[ B_c^{-1} b \right]_k}{\alpha_{kj}} : \alpha_{kj} > 0 \right\}
\]

where \(\alpha_{kj} = (B_c^{-1} p_j)_k\) and \(k = 1, 2, \ldots, n\)

If all \(\alpha_{kj} \leq 0\) then the problem has no bounded solution.

**step 3:** Compute the next basic solution. The \(E_0\) matrix is constructed as explained above and the \(M_n^{-1}\) is computed by using (3.8) and the solution vector is obtained using (3.5).

Thus \(B_n^{-1}\) is expressed as a function of \(B_c^{-1}\) and the procedure is returned to step 1.

This procedure is repeated until the optimal solution is reached.

**3.3.1.2 Alternate optimality criterion**

It is a known fact that the product of a variable and its corresponding contribution or cost coefficient contributes to the objective function. Since the decision variable is unknown there is no way to get an estimate of a term like \(c_j x_j\) of the objective function. The matrix of intercepts enables to develop an alternate optimality criterion other than one which the Simplex algorithm makes use of, to identify the entering variable and sometimes to break the tie among incoming
variables. Sometimes both the criteria lead to the same conclusion. But where they differ the product criterion works better.

### 3.3.2 Dual Simplex algorithm

The dual Simplex is applied to problems, which satisfy optimality but not feasibility (Hamdy A Taha, 1997). If one or more of the variable(s) in the solution vector become infeasible in the course of computation and if the optimality condition is satisfied, then the infeasibility can be removed using the dual Simplex method and when the basic solution becomes feasible, it is also optimal. This procedure is similar to the Simplex algorithm in that it uses the feasibility and optimality criteria alternately.

To remove infeasibility, if any, from the basis the variable with the most negative value is sent out of the basis. When all the basic variables are non-negative, the process ends and the optimal solution is obtained.

The optimality criterion is employed to select the entering variable from among the non basic variables. If the infeasible variable corresponding to the $i^{th}$ row is leaving, compute

$$d_j = (B^{-1} p_j)_i$$

$$= \sum_{k=1}^{m} b_{ik} a_{kj}$$

for all the non-basic variables. Compute the ratio \( \frac{z_j - c_j}{d_j} \) for all the variables with \( d_j < 0 \). Select the entering variable as the one associated with the smallest ratio if the objective of the problem is minimization or the smallest absolute value of the ratios if the objective is maximization.
After selecting the leaving and the entering variables, $M^{-1}$ is updated in the same manner as the revised Simplex procedure.

### 3.3.3 Multiplex algorithm

Consider the objective function

$$Z = c_1 x_1 + c_2 x_2 + \ldots + c_j x_j + \ldots + c_n x_n$$

The product of the contribution coefficient $c_j$ and the variable $x_j$, is a better estimate of a change in the objective function than the coefficient $c_j$ alone. But the value of the variable $x_j$ is not known. This is where the matrix of intercepts offers a solution. The minimum intercept for $x_j$ is chosen and $c_jx_j$ is calculated to identify the promising variable.

#### 3.3.3.1 Multiple column selection

A matrix of intercepts $\theta$ of the decision variables along the respective axes with respect to the chosen basis is constructed. A typical intercept at the starting pass for the $j^{th}$ variable, $x_j$ due to the $i^{th}$ resource, $b_i$ is

$$\frac{b_i}{a_{ij}} \quad ; \quad a_{ij} > 0$$

It may be observed that the rows of the $\theta$ matrix represent the $k$ number of ($i \leq k \leq m$) intercepts of the variables along their respective axes and the columns represent the $l$ number of intercepts ($j \leq l \leq n$) formed by each of the $m$ constraints.
The construction of the $\theta$ matrix in the subsequent passes is governed by the following rules:

a. If the promising variable is a decision variable then

$$\theta_{qi} = \frac{(B^{-1} b)_i}{\alpha_{ij}} ; \quad \alpha_{ij} > 0$$

where $i = 1, 2, \ldots m$ ; $q = 1, 2, \ldots l$

$j$ is the subscript of the promising variable and

$$\alpha_{ij} = (B^{-1} p)_i$$

b. If the entering variable is a slack variable, then the relationship

$$\theta_{qi} = \frac{(B^{-1} b)_i}{d_{ij}} ; \quad d_{ij} > 0$$

is used,

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

and

$$d_{ik} = i^{th} \text{ row and } k^{th} \text{ column element of } B^{-1}.$$ 

The following step by step method is used to select a set of linearly independent columns to form a basis matrix.

step 1: The $\theta$ matrix is scanned row after row and the position of the smallest intercept in each row is identified and boxed.

step 2: The first entering vector will be the most promising vector. The entering variable corresponding to this vector is found by taking the product of the contribution/ cost-coefficient and
the minimum intercept in the corresponding row. Among these values, the most promising variables will be found out, say the j<sup>th</sup> vector and the corresponding row in the θ matrix be, k. This is the entering vector in place of a vector corresponding to the column in which the minimum intercept lies in the θ matrix, say i<sup>th</sup> column.

step 3: Examine whether any other row has a minimum in the i<sup>th</sup> column. All such rows as well as the k<sup>th</sup> row and the i<sup>th</sup> column are deleted.

step 4: Steps 1 to 3 are repeated until all the rows are covered.

All promising vectors form the basis for the first pass (a pass is none other than a set of iterations).

3.3.4 Dual Multiplex algorithm

As was stated earlier, Multiplex algorithm preserves the property of linear independence but not the feasibility. Infeasibility arising while employing the Multiplex algorithm can be removed either by the dual Simplex algorithm or by the dual Multiplex algorithm. Unlike the dual Simplex algorithm, the dual Multiplex algorithm attempts to remove from the basis as many infeasible variables as possible in each pass. The dual Multiplex algorithm employs the transpose of the θ matrix to choose a set of leaving and entering variables at start of each pass.

The θ matrix is constructed as follows. If the i<sup>th</sup> row variables in the basic solution is infeasible, the d<sub>j</sub>’s for all the non-basic variables are computed using the relationship (3.10).
If $d_j \geq 0$, then $[\theta]_{ij}$ is not computed.

If $d_j < 0$, then $[\theta]_{ij} = \left( \frac{z_j - c_j}{d_j} \right)$ for minimization

and $[\theta]_{ij} = \left( \frac{z_j - c_j}{d_j} \right)$ for maximization objective.

The number of rows in the $\theta$ matrix depends upon the number of infeasible variables in the basis and the column depends upon the number of non-basic variables.

The following steps are employed to implement the dual Multiplex algorithm:

Step 1: Use the above $\theta$ matrix and select a set of leaving and entering variables depending upon the number of infeasible variables in the basis.

step 2: While updating $M^{-1}$ matrix, check for feasibility of the variable corresponding to the entering column. If it is feasible then go to step 4, otherwise.

step 3: Remove the infeasible variable by introducing an appropriate variable and update the $M^{-1}$ matrix.

step 4: Repeat steps 2 and 3 till all the variables in the selected set are exhausted.

These algorithms are applied to the original and reduced model which is explained below. The model reduction algorithm
combined with these algorithms detects the redundancies, if any, and solves the reduced model.

3.4 MODEL REDUCTION ALGORITHM

It is a well established fact that every additional constraint and variable in a linear programming problem increases the computational effort. If a constraint or a variable is redundant, it consumes extra computations which is a sheer waste. Any redundant constraint or variable should be detected and removed before the commencement of the computation.

The following method uses the matrix of intercepts of the original problem as a means to identify the redundancies.

A resource constraint is said to be redundant if it does not form part of the edge of a convex set formed by the constraints. If such constraints are not identified and removed they waste a lot of computational effort contributing to inefficiency. It was observed that while finding the solution for some of the linear programming problems, a set of surplus variables always stays as basic variables irrespective of changes in the other variables from being non-basic to basic and vice versa (Philip et.al 1962). Surplus variables behaving in this manner clearly indicate that no product mix could consume these resources completely and irrespective of the mix, they are always surplus. Such redundant constraints at no stage become active and they do not affect the optimal solution. The θ matrix explained in section 3.3.3.1 may be utilized to detect such redundancies.

The rows of the θ matrix represent the m number of intercepts of the decision variables along their respective axes in the m dimensional requirement space and the columns represent the
intercepts formed by the decision variables of each constraint in the n dimensional decision space.

This model reduction algorithm is based on the theorem (Mattheiss T.H,1973) which is stated below:

**Theorem:** A constraint \( A_i X \leq f_i \) is redundant for the system of linear inequalities \( A X \leq b \) if and only if its associated slack variables \( s_i \) are in the basis of every primary subsystem of the linear programming problem. This, in other words, means a surplus variable \( s_i \) should stay in all the basis until the optimal solution is found.

A simple but generalized heuristic algorithm has been developed to predict *apriori* whether a slack variable \( s_i \) is present in the basis of every primary subsystem of the LP problem or not.

### 3.4.1 Identification of redundant constraints

Consider the linear programming model which has \( m \) constraints and \( n \) variables.

Extremize \( Z = C X \)

subject to \( A X (\leq, =, \geq) b \); \( X \geq 0 \)

**step 1:** A matrix of intercepts is constructed with decision and surplus variables as rows and columns respectively. This matrix is of dimension \( n \times m \).

\( \theta_{ij} = b_i / a_{ij} \); \( a_{ij} > 0 \) for all \( i \) and \( j \)

**step 2:** Identify the pivot element in each row.

\( \beta_j = \min \{ \theta_{ij} \} \) for all \( j \) while the objective is maximization

\( \beta_j = \max \{ \theta_{ij} \} \) for all \( j \) while the objective is minimization.
step 3: Score out the rows and columns corresponding to the entering and leaving variables. If a column has more than one minimum / maximum score out those rows also.

step 4: The constraints corresponding to the surplus variables in the unscored columns, if any, ab initio are assumed and predicted as redundant.

step 5: Remove these redundant constraints tentatively from the original model.

The application of the above steps results in a reduced primal model. The present work has also suggested a new approach for identifying the redundant variables of the reduced primal model. According to the primal-dual properties, for every primal redundant constraint, there exists a dual redundant variable and vice versa. This property when applied alternately to a given problem reduces it to a smaller size.

3.4.2 Identification of redundant variables

This procedure is to identify whether a slack variable associated with a dual constraint is in the basis of every dual primary subsystem of the reduced LP model or not, and hence the identification of the redundant primal variable. In other words the algorithm attempts to identify the irrelevant decision variables.

Consider the dual model corresponding to the reduced primal model which has n constraints and (m-k) variables, where k is the number of identified redundant primal constraints.

step 1: Perform steps 1 to 4 of 3.4.1 on the constructed dual model.
step 2: Identify the redundant constraint(s) from the dual model and thereby the corresponding primal redundant variable(s). If the \( k^{th} \) dual constraint is redundant, then the corresponding \( k^{th} \) primal variable is redundant or in other words an irrelevant variable.

step 3: Remove these redundant variables tentatively from the reduced primal model.

The application of the row-column reduction algorithms results in an overall reduced model.

3.4.3 Validation of the model

The primal and its associated dual basic (feasible) solutions are obtained by solving the reduced model. These solutions are substituted in the identified redundant constraints.

step 1: Substitute the primal solution in the identified redundant primal constraints. Flag those redundant constraints, which do not satisfy the original conditions.

step 2: Substitute the dual solution in the identified dual constraints. Identify those constraints which do not satisfy the solution as violated constraints (if any), and then the corresponding primal variables. Flag those violated primal variables.

step 3: Apply post optimality analysis to those flagged constraint(s) and/or variable(s) of steps 1 and 2.

step 4: Repeat the above steps until there exists no violating constraint(s) or variable(s).
The original and the reduced abstract models of the linear programming problems are pictorially represented in figure 3.1. The original model is the real one and the apparent model is the one identified by the model reduction algorithm. However the reduced model is only tentative and finality is decided only after the apparent model is validated. Computational experiences have revealed that the final model in some cases is closer to the apparent and the other cases nearer to the original.

**Figure 3.1 A pictorial representation of the original and the reduced models**

**3.4.4 Extension of the complimentary slackness theorem**

For every primal redundant constraint there is a dual redundant variable and for every dual redundant constraint there is a primal redundant variable. This extended theorem helps to reduce the original model to a smaller size to improve the computational efficiency.
\[ \sum_{j=1}^{n} (a_{ij} x_j - b_i) y_i = \left( \sum_{i=1}^{m} a_{ij} y_i - c_j \right) x_j = 0 \]
for all \( i \) for all \( j \)

### 3.4.5 A tie-breaking rule

In the model reduction algorithm, while identifying pivot element, we consider for each row \( j \in \{1, 2, \ldots, n\} \) and column \( i \in \{1, 2, \ldots, m\} \).

\[ \beta_j = \min_{i} \{ \theta_{ji} \} \text{ while the objective is maximization} \]
\[ \beta_j = \max_{i} \{ \theta_{ji} \} \text{ while the objective is minimization.} \]

If a tie occurs during this selection, there is a possibility of detecting non-redundant constraint as redundant. For example consider the LP problem,

Maximize \( Z = x_1 + 2x_2 \)
subject to
\[ x_1 + x_2 \leq 3 \] \hspace{1cm} (1)
\[ 3x_1 + x_2 \leq 6 \] \hspace{1cm} (2)
\[ 2x_1 + x_2 \leq 4 \] \hspace{1cm} (3)
\[ x_1, x_2 \geq 0 \] \hspace{1cm} (4)
and the matrix of intercepts of the LP model is given by

<table>
<thead>
<tr>
<th>Surplus/decision variables</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>x₂</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

It is noticed that the values of the boxes corresponding to \((x₁, s₂)\) and \((x₁, s₃)\) are the same as well as minimum. If the boxed value corresponding to \((x₁, s₂)\) is considered as minimum, then the third constraint becomes redundant, which is not true. This causes some additional computation (3.4.3). To avoid such problems, a tie-breaking rule is incorporated in the model reduction algorithm.

If a tie occurs among the exit variables the remaining intercepts in the tied columns are sorted. The column corresponding to the minimum of the minima or the maximum of the maxima is scored out depending on maximization or minimization of the objective respectively.

The lexicographer rule when applied to break the tie, identified the third constraint as redundant, which is not true.

The tie-breaking rule thus minimizes the computational steps and increases the overall efficiency of the algorithm.

### 3.4.6 Alternate criterion

The construction of the matrix of intercepts has been suitably modified in this algorithm to improve the computational efficiency. The size of the \(θ\) matrix being \(n \times m\), as many divisions as the order of the
matrix have to be performed at start. Instead of constructing a matrix of intercepts of the decision variables along the respective axes with respect to the chosen basis, the transpose of the constraint matrix, with respect to the chosen basis, $A^T$ is chosen in place of $\theta$ matrix. The algorithms developed in section 3.4.1 to 3.4.3 are applied to the original model by considering the gradient matrix of the constraints in place of $\theta$ matrix and the pivot element choice is made using the maximum value, i.e., for each row, the column containing the largest component of the gradient vector of the constraints.

3.5 ILLUSTRATIVE EXAMPLE

The following fully solved problem is chosen from (Paulraj S et al., 1998) as an example for illustration. It is a four constraint six variable problem and goes through six iterations with Simplex procedure.

\[
\text{Max } Z = 0.4x_1 + 0.28x_2 + 0.32x_3 + 0.72x_4 + 0.64x_5 + 0.06x_6
\]

subject to

\[
\begin{align*}
0.01x_1 + 0.01x_2 + 0.01x_3 + 0.03x_4 + 0.03x_5 + 0.03x_6 & \leq 850 \\
0.02x_1 + 0.05x_4 & \leq 750 \\
0.02x_2 + 0.05x_5 & \leq 100 \\
0.03x_3 + 0.08x_6 & \leq 900 \\
x_j \geq 0 & \text{ for } j = 1 \text{ to } 6.
\end{align*}
\]

<table>
<thead>
<tr>
<th>Surplus/decision variables</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>85000</td>
<td>37500</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2$</td>
<td>85000</td>
<td>-</td>
<td>5000</td>
<td>-</td>
</tr>
<tr>
<td>$x_3$</td>
<td>85000</td>
<td>-</td>
<td>-</td>
<td>30000</td>
</tr>
<tr>
<td>$x_4$</td>
<td>85000/3</td>
<td>15000</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_5$</td>
<td>85000/3</td>
<td>-</td>
<td>2000</td>
<td>-</td>
</tr>
<tr>
<td>$x_6$</td>
<td>85000/3</td>
<td>-</td>
<td>-</td>
<td>11250</td>
</tr>
</tbody>
</table>
The application of the algorithm developed in section 3.4.1 identifies the first constraint as redundant. The corresponding dual model for the given primal is

Minimize $Y=850y_1 + 750y_2 + 100y_3 + 900y_4$.

subject to

$0.01y_1 + 0.02y_2 \geq 0.40 \quad (1)$
$0.01y_1 + 0.02y_3 \geq 0.28 \quad (2)$
$0.01y_1 + 0.03y_4 \geq 0.32 \quad (3)$
$0.03y_1 + 0.05y_2 \geq 0.72 \quad (4)$
$0.03y_1 + 0.05y_3 \geq 0.64 \quad (5)$
$0.03y_1 + 0.08y_4 \geq 0.06 \quad (6)$

Since the first constraint in the primal model is redundant the corresponding dual variable $y_1$ is redundant.

$y_2 \geq 0.4/0.02 \quad y_2 \geq 20 \quad y_2 - s_1 = 20$
$y_3 \geq 0.28/0.02 \quad y_3 \geq 14 \quad y_3 - s_2 = 14$
$y_4 \geq 0.32/0.03 \quad y_4 \geq 10.67 \quad y_4 - s_3 = 10.67$
$y_2 \geq 0.72/0.05 \quad y_2 \geq 14.4 \quad y_2 - s_4 = 14.4$
$y_3 \geq 0.64/0.05 \quad y_3 \geq 12.8 \quad y_3 - s_5 = 12.8$
$y_4 \geq 0.06/0.08 \quad y_4 \geq 0.75 \quad y_4 - s_6 = 0.75$

The corresponding matrix of intercepts is:

<table>
<thead>
<tr>
<th>Surplus/decision variables</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_2$</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>14.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$y_3$</td>
<td>-</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>12.8</td>
<td>-</td>
</tr>
<tr>
<td>$y_4$</td>
<td>-</td>
<td>-</td>
<td>10.7</td>
<td>-</td>
<td>-</td>
<td>0.75</td>
</tr>
</tbody>
</table>
The last three constraints are redundant as revealed by the above table.

\[ y_2 \geq 20 ; y_3 \geq 14 ; y_4 \geq 10.7 \] are the active constraints.

Therefore the corresponding primal variables \( x_4, x_5 \) and \( x_6 \) are redundant. The primal model ultimately reduces to

\[
\begin{align*}
\text{Max} \quad Z &= 0.4x_1 + 0.28x_2 + 0.32x_3 \\
\text{subject to} \\
0.02x_1 &\leq 750 \quad (1) \\
0.02x_2 &\leq 100 \quad (2) \\
0.03x_3 &\leq 900 \quad (3)
\end{align*}
\]

\[ x_1, x_2, x_3 \geq 0 \]

Therefore, \( x_1 = 37,500 \)
\( x_2 = 5,000 \)
\( x_3 = 30,000 \) and
the optimal solution \( Z = 26,000 \).

The proposed algorithm identifies the first constraint and variables \( x_4, x_5, \) and \( x_6 \) as redundant.

Illustrative example 3.5 reveals the following:

(i) The application of the alternate optimality criterion 3.3.1.2 to select an incoming variable eradicates the popping tendency of variables in and out of the basis.

(ii) The matrix of intercepts identifies the redundant constraints in the primal and dual.

(iii) The primal-dual properties tie a primal redundant constraint with a dual redundant variable and \textit{vice versa}.

(iv) The above complementary property enables to reduce the model to the barest minimum.
(v) This problem does not use the Simplex procedure at all to find optimal solution.
(vi) The problem is self solving.

The following problem is chosen from (Paulraj S et.al, 1998) as an example for illustration. It is a seven constraint ten variable problem. The model reduction algorithm at the first instance identifies five constraints and eight variables as redundant.

Max $Z = 61x_1 + 209x_3 + 325x_5 + 33x_4 + 276x_6 + 285x_7 + 100x_8 + 12x_9 + 282x_{10}$

subject to

1. $16x_1 + 25x_2 + 22x_3 + 4x_4 + 9x_5 + 8x_6 + 11x_7 + 29x_8 + 20x_9 + 22x_{10} \leq 11$  
2. $5x_1 + 22x_2 + 15x_3 + 30x_4 + 24x_5 + 15x_6 + 14x_7 + 28x_8 + 31x_9 + 25x_{10} \leq 53$  
3. $22x_1 + 17x_2 + 32x_4 + 26x_5 + 20x_6 + 16x_7 + 16x_8 + 26x_9 + 24x_{10} \leq 50$  
4. $14x_1 + 9x_2 + 32x_3 + 22x_4 + 30x_5 + 18x_6 + 18x_7 + 32x_8 + 15x_9 + x_{10} \leq 40$  
5. $32x_1 + 30x_2 + 10x_3 + 30x_4 + 7x_5 + 29x_6 + 15x_7 + x_8 + 19x_9 + 26x_{10} \leq 04$  
6. $12x_1 + 4x_2 + 30x_3 + 11x_4 + 23x_5 + 29x_6 + 8x_7 + 2x_8 + 0x_9 + 23x_{10} \leq 31$  
7. $22x_1 + 23x_2 + 26x_3 + 13x_4 + 6x_5 + 13x_6 + 32x_7 + 11x_8 + 8x_9 + 5x_{10} \leq 39$

$x_j \geq 0, j=1,2, \ldots, 10.$

<table>
<thead>
<tr>
<th>Decision/surplus variables</th>
<th>Basic variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1.83</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.44</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.5</td>
</tr>
<tr>
<td>$x_4$</td>
<td>2.75</td>
</tr>
<tr>
<td>$x_5$</td>
<td>1.22</td>
</tr>
<tr>
<td>$x_6$</td>
<td>1.38</td>
</tr>
<tr>
<td>$x_7$</td>
<td>1</td>
</tr>
<tr>
<td>$x_8$</td>
<td>0.38</td>
</tr>
<tr>
<td>$x_9$</td>
<td>0.55</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>$s_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>10.6</td>
<td>2.27</td>
<td>2.86</td>
<td>0.125</td>
<td>2.58</td>
<td>1.77</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>2.41</td>
<td>2.94</td>
<td>4.44</td>
<td>0.13</td>
<td>7.75</td>
<td>1.70</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>3.53</td>
<td>5.56</td>
<td>1.25</td>
<td>0.4</td>
<td>1.03</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>1.77</td>
<td>1.56</td>
<td>1.82</td>
<td>0.13</td>
<td>2.82</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td>2.21</td>
<td>1.92</td>
<td>1.33</td>
<td>0.57</td>
<td>1.35</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>$x_6$</td>
<td>3.53</td>
<td>2.5</td>
<td>2.22</td>
<td>0.138</td>
<td>1.03</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$x_7$</td>
<td>3.79</td>
<td>3.13</td>
<td>2.22</td>
<td>0.27</td>
<td>3.88</td>
<td>1.22</td>
<td></td>
</tr>
<tr>
<td>$x_8$</td>
<td>1.89</td>
<td>3.125</td>
<td>1.25</td>
<td>4</td>
<td>15.5</td>
<td>3.55</td>
<td></td>
</tr>
<tr>
<td>$x_9$</td>
<td>1.71</td>
<td>1.92</td>
<td>2.67</td>
<td>0.21</td>
<td></td>
<td>4.88</td>
<td></td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>2.12</td>
<td>2.08</td>
<td>40</td>
<td>0.15</td>
<td>1.35</td>
<td>7.8</td>
<td></td>
</tr>
</tbody>
</table>
The algorithm identifies the constraints 2, 3, 4, 6 and 7 as redundant. The corresponding dual model for the given primal problem is

\[
\begin{align*}
\text{Min } Y &= 11y_1 + 4y_5 \\
\text{subject to} & \\
16y_1 + 32y_5 & \geq 61 \\
25y_1 + 30y_5 & \geq 209 \\
22y_1 + 10y_5 & \geq 325 \\
4y_1 + 30y_5 & \geq 33 \\
9y_1 + 7y_5 & \geq 276 \\
8y_1 + 29y_5 & \geq 285 \\
11y_1 + 15y_5 & \geq 250 \\
29y_1 + y_5 & \geq 100 \\
20y_1 + 19y_5 & \geq 12 \\
22y_1 + 26y_5 & \geq 282 \\
y_1, y_5 & \geq 0
\end{align*}
\]

The matrix of intercepts of the dual model is:

<table>
<thead>
<tr>
<th>Decision /surplus variables</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
</tr>
</thead>
<tbody>
<tr>
<td>y_1</td>
<td>3.8</td>
<td>8.3</td>
<td>14.7</td>
<td>8.2</td>
<td>30.6</td>
<td>35.6</td>
<td>22.7</td>
<td>3.4</td>
<td>0.6</td>
<td>12.8</td>
</tr>
<tr>
<td>y_5</td>
<td>1.9</td>
<td>6.9</td>
<td>32.5</td>
<td>1.1</td>
<td>39.4</td>
<td>9.8</td>
<td>16.6</td>
<td>100</td>
<td>0.6</td>
<td>10.8</td>
</tr>
</tbody>
</table>

The above table reveals that all the dual constraints except 6 and 8 are redundant. Therefore, the corresponding variables x_1, x_2, x_3, x_4, x_5, x_7, x_9 and x_{10} are redundant, which reduces the primal model to

\[
\begin{align*}
\text{Max } Z &= 285x_6 + 100x_8 \\
\text{subject to} & \\
8x_8 + 29x_9 & \leq 11 \\
29x_8 + x_9 & \leq 4 \\
x_8, x_9 & \geq 0
\end{align*}
\]
The primal-dual solution of the reduced model obtained using the Simplex procedure is given below:

<table>
<thead>
<tr>
<th>Primal</th>
<th>$x_6=0.35$</th>
<th>$x_8=0.13$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual</td>
<td>$y_1=3.14$</td>
<td>$y_5=8.97$</td>
</tr>
</tbody>
</table>

Substituting the primal-dual solution to the reduced model, dual constraints 3, 5 and 7 violate and the corresponding primal variables $x_3, x_5$ and $x_7$ become non-redundant. Applying post-optimality analysis to the variables $x_3, x_5$ and $x_7$, the optional solution is $Z=170.516$, $x_5=0.541$, $x_9=0.211$.

A seven constraint ten variable problem is ultimately reduced to a two constraint five variable problem. The storage requirement is only 14.3 percent of the total required for the original problem.

### 3.6 AVOIDANCE OF CYCLING

It is generally assumed that the objective function value increases/decreases at each iteration and the final solution is optimal when termination takes place after a finite number of iterations. Unfortunately, the convergence property of the Simplex procedure has not been established for problems with cycling property and the objective function actually stalls for several iterations in a few linear optimization problems (Beale 1954, Charnes et al. 1952).

This anomaly occurs when the solution is degenerate, i.e., when the values of one or more variable become(s) zero. At this point there is no assurance that the value of the objective function will improve. It is then possible that the Simplex iterations will enter a loop, which will repeat the same sequence of iterations without ever
reaching the optimal solution. This problem is said to have the property of cycling (Beale 1955).

Degeneracy and the consequent cycling has been plaguing the computational effort in linear programming problems for some time, although this phenomenon has never made it impossible to reach an optimal solution since no practical problem has ever cycled. However Charne's perturbation method (1952) rescues the variable from entering into the vicious circle of cycling. Consider the linear programming problem suggested by Beale (1962).

Maximize \( Z = 0.75x_1 - 20x_2 + 0.5x_3 - 6x_4 \)
subject to \( 0.25x_1 - 8x_2 - x_3 + 9x_4 \leq 0 \) \hspace{1cm} (1)
\( 0.5x_1 - 12x_2 - 0.5x_3 + 3x_4 \leq 0 \) \hspace{1cm} (2)
\( x_3 \leq 1 \) \hspace{1cm} (3)
\( x_1, x_2, x_3, x_4 \geq 0 \)

The optimal solution of the above problem using the proposed model reduction algorithm is:

\( Z = 1.25 \)
\( x_1 = 1.0 \)
\( x_3 = 1.0 \)

The various steps of the Simplex algorithm are given in table 3.1 which illustrate the cycling process. It may be observed that the basis for iteration zero and six are identically the same.
Table 3.1 Example of cycling

<table>
<thead>
<tr>
<th>Iter.</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>Basic Variables</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.75</td>
<td>20</td>
<td>-0.5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$s_1$, $s_2$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-4</td>
<td>-3.5</td>
<td>33</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>$x_1$, $s_2$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$x_1$, $x_2$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>3</td>
<td>0</td>
<td>$x_3$, $x_2$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-0.5</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>$x_3$, $x_4$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>-1.75</td>
<td>44</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>$s_1$, $x_4$, $s_3$</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-0.75</td>
<td>20</td>
<td>-0.5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$s_1$, $s_2$, $s_3$</td>
<td>0</td>
</tr>
</tbody>
</table>

By virtue of the model reduction algorithm the cyclic property of this problem is eliminated. The various steps of model reduction algorithm are given below:

The matrix of intercepts of the problem is

<table>
<thead>
<tr>
<th>Surplus/decision variable</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3$</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The algorithm developed identifies the constraints 1 and 2 as redundant. Since primal model has only one constraint, the dual model is not constructed.

Hence the reduced model becomes

Maximize $Z = 0.75x_1 - 20x_2 + 0.5x_3 - 6x_4$

subject to

$x_3 \leq 1$  \hspace{1cm} (1)

$x_1, x_2, x_3, x_4 \geq 0$  \hspace{1cm} (2)
The first iteration of the reduced model using Simplex procedure is as follows:

**Iteration : 0**

<table>
<thead>
<tr>
<th>Basis</th>
<th>Z</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1</td>
<td>-0.75</td>
<td>20</td>
<td>-0.5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Since $x_1$ has no positive pivot element $x_3$ is considered as entering variable.

**Iteration : 1**

<table>
<thead>
<tr>
<th>Basis</th>
<th>Z</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1</td>
<td>-0.75</td>
<td>20</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Here $x_1$ is promising and the corresponding pivot element is zero. Hence a constraint from the identified redundant constraints is introduced. Using post-optimality the 2nd constraint is introduced and the modified table is given by

<table>
<thead>
<tr>
<th>Basis</th>
<th>Z</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1</td>
<td>-0.75</td>
<td>20</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>s2</td>
<td>0</td>
<td>0.5</td>
<td>-12</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Now $x_1$ enters and $s_2$ leaves.

**Iteration 2:**

<table>
<thead>
<tr>
<th>Basis</th>
<th>Z</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>10.5</td>
<td>0</td>
<td>1.5</td>
<td>0.75</td>
<td>1.25</td>
</tr>
<tr>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x1</td>
<td>0</td>
<td>1</td>
<td>-24</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Thus the model reduction algorithm gives the optimal solution in two iterations. The phenomenon of cycling seems to depend on the initial basis with which one commences to search for optimal solution. If an advantageous basis is chosen, then cycling does no longer plague computations. This is another advantage of the proposed model reduction algorithm.

3.7 VALIDATION OF MODEL REDUCTION ALGORITHM ON PERT NETWORK

The PERT, Project Evaluation and Review Technique is widely used in project management to plan and control the activities. The critical path method (Wiest et al. 1960) is used to solve these kinds of problem. Since the model is linear, it can also be formulated as a LP model. The constraints corresponding to the critical activities are non-redundant and the remaining are redundant.

To validate the model reduction algorithm as an application to PERT network, an example is taken from (Wiest et al. 1960). It consists of five nodes and six activities. Among six activities three activities are non-critical.

From the figure one can infer that the activities 1-2, 2-3 and 3-5 are critical. Now the model reduction algorithm is applied to the
PERT network. In order to apply this algorithm on the PERT network, as a first step, it is modelled as a LP problem.

Minimize

\[ Z = x_5 - x_1 \]

subject to

\[ x_2 - x_1 \geq 2 \quad (y_1) \]  
\[ x_3 - x_1 \geq 1 \quad (y_2) \]  
\[ x_3 - x_2 \geq 3 \quad (y_3) \]  
\[ x_4 - x_2 \geq 2 \quad (y_4) \]  
\[ x_5 - x_3 \geq 4 \quad (y_5) \]  
\[ x_5 - x_4 \geq 3 \quad (y_6) \]

\[ x_1, x_2, x_3, x_4, x_5 \geq 0. \]

The corresponding matrix of intercepts is

<table>
<thead>
<tr>
<th>Surplus/decision variables</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
<th>s6</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x3</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

The columns corresponding to \( s_2 \) and \( s_6 \) are not scored out and hence they are redundant. i.e.,

\[ x_3 - x_1 \geq 1 \quad \text{and} \quad (1) \]
\[ x_5 - x_4 \geq 1 \quad \text{are redundant.} \quad (2) \]
Construct the dual corresponding to the reduced primal model

Maximize \( Y = 2y_1 + y_2 + 3y_3 + 2y_4 + 4y_5 + 3y_6 \)
subject to
\[-y_1 \leq -1 \quad (1)\]
\[y_1 - y_3 - y_4 \leq 0 \quad (2)\]
\[y_3 - y_5 \leq 0 \quad (3)\]
\[y_4 \leq 0 \quad (4)\]
\[y_5 \leq 1 \quad (5)\]

The corresponding matrix of intercept is

<table>
<thead>
<tr>
<th>Surplus/decision variables</th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
<th>s4</th>
<th>s5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_1 )</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( y_3 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( y_4 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( y_5 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

The above table reveals that all the dual constraints except 1 and 5 are redundant. Therefore, the corresponding primal variables \( x_2, x_3, x_4 \) are redundant. However, the model validation routine detects the variables \( x_2 \) and \( x_3 \) as non-redundant. Hence, the algorithm detects only the variable \( x_4 \) as redundant.

The model reduction algorithm identifies the activities 1-3 and 4-5 as non-critical. But in the original problem, there are three non-critical activities whereas the algorithm detects only two. Therefore, the primal model reduction algorithm detects only 66.67 percent of redundancies of this problem.
The model reduction algorithm identifies three events as redundant and the validation routine rejects two of them. Hence, the net effect has been that the model reduction algorithm identifies all the redundancies of the above problem.

A variety of PERT network problems were solved and the model reduction algorithm could detect 50 to 90 percent of the redundancies. Thus the model reduction algorithm is validated on PERT network.

3.8 IMPLEMENTATION OF THE DEVELOPED SOFTWARE

To study the effect of model reduction algorithm on finding solution for LP problems, software has been developed. In order to compare the proposed algorithms, the existing Simplex and Multiplex algorithms were coded first. Then, new routines have been included to study the effect of model reduction on these existing algorithms. The object oriented programming language C++ is used to code these algorithms. Using these algorithms, problems have been analyzed on four different combinations. The detailed design view of the model reduction algorithm is described below:

step 1: Read data.

1.1 Read and store the problem data.

step 2: Construction of the reduced model.

2.1 Identification of redundant constraints

2.1.1 Construction of the matrix of intercepts
2.1.2 If the problem is of maximization, scan the matrix of intercepts and score out the column containing the minimum value and repeat for all the rows. Else, if the problem is of minimization, scan the matrix of intercepts and score out the column containing the maximum value and repeat for all the rows. If there is a tie in the selection process the tie-breaking rule is applied.

2.1.3 Collect all the scored out columns of the above step as "non-redundant constraint" set. The constraints corresponding to the unscored columns form the "redundant constraint" set.

2.2 Identification of the redundant variables.

2.2.1 Construct the dual model using the original primal objective function and the non-redundant constraint set identified in steps 2.1.2 and 2.1.3.

2.2.2 Apply step 2.1.2 and 2.1.3 on the dual model to categorize the dual constraints as redundant and non-redundant.

2.2.3 Using primal-dual property, obtain the set of redundant and non-redundant primal variables.

2.3 By considering the primal non-redundant constraints and variable sets, construct the reduced LP model.
Also, store the redundant constraint and variable sets to be used for the validation of the reduced model.

step 3: Solve the reduced LP model.

3.1 Use Simplex or Multiplex algorithm to solve the reduced LP model.

3.2 If the solution is optimal go to step 4, or else if the solution is unbounded go to step 3.3.

3.3 If there exists no redundant constraint and no redundant variable on the identified sets, then declare the model as unbounded and go to step 5.2, or else go to step 3.4.

3.4 If the redundant constraint set is not empty, identify the constraint which is promising in the redundant constraint set and include it on the non-redundant constraint set, and go to step 3.6, or else go to step 3.5.

3.5 Identify the variable which is promising in the redundant variable set and include it in the non-redundant variable set and go to step 3.6.

3.6 Apply the post-optimality analysis on the identified constraint(s) / variable(s) and go to step 3.

step 4: Validate the reduced model.

4.1 Using the solution obtained from step 3.1 test the validity of the redundant constraint(s) and variable(s).
4.2 If valid go to the next step, or else include the violated constraint(s) / variable(s) on the respective redundant sets and go to step 3.6.

step 5: Declare the results.

5.1 Prepare the solution.

5.2 Terminate the job.

The conceptual design of the developed model reduction algorithm is given in the form of a flow chart in figure 3.2.

3.9 MERITS AND DEMERITS OF THE DEVELOPED ALGORITHM

The developed algorithm is evaluated for its computational efficiency by applying it to a number of LP problems downloaded from the net-library. The algorithm is able to detect redundancies anywhere from 20 to 90 percent of the actual. Some of the results are shown in figures 3.5 and 3.6.

The model reduction algorithm in the worst case may result in a maximum of $2 \cdot (m \times n)$ computation (divisions) to construct the $\theta$ matrix and a few comparisons to detect the redundant constraints and variables \textit{apriori} before seeking solution to the problem. A few more computations are carried out in validating the redundant constraints and variables and post-optimality analysis for the violated constraint(s) / variable(s). However, the proposed algorithm considerably minimizes the computations by removing the redundant rows and columns. For purposes of comparison, the given problem and its reduced model are solved using the developed
Figure 3.2 Schematic representation of the algorithms
algorithms. The proposed method has sped up the computations by a factor up to ten even for medium size problems. The results of a few solved sample problems are furnished in table 3.2.

When the model reduction algorithm is applied on Multiplex algorithm, further reduction in computations is achieved. The model reduction algorithm uses the matrix of intercepts of the given problem constructed for Multiplex algorithm to detect the redundant constraint(s). Therefore a computation of \((m \times n)\) is minimized. Also the Multiplex algorithm minimizes the popping tendency of the variables in and out of the basis. This also gives some more computational savings. Apart from the savings achieved from the model reduction algorithm on Simplex procedure, this algorithm makes less computation to solve a given LP model. Here again, for purposes of comparison, the given model and its reduced model are solved using the developed algorithms. The proposed method has sped up the computations by a factor up to eleven even for medium size problems. As the problem size grows larger and larger, the model reduction algorithm with Multiplex procedure produces more computational savings than the Simplex procedure. The results of a few solved sample problems are furnished in table 3.3.

The developed algorithm is compared with the application to Simplex and Multiplex algorithms. The speed ratios of a few solved sample problems are furnished in figures 3.3 and 3.4. The application of the Multiplex algorithm on the reduced model gives a better performance than the Simplex algorithm.

During the analysis, apart from the computational efficiency, the following observations were also made.
(i) The number of iterations to solve the LP model using model reduction algorithm is slightly more in a few cases than the regular algorithm for the set of problems analyzed. It is noticed that the additional number of iterations is due to the post-optimality analysis applied on the violated redundant constraint(s)/ variable(s). However the total number of computations to solve the problems have been minimized when the model reduction algorithms are used.

(ii) The application of the model reduction algorithm to LP problems reduces the number of iterations for most of the problems among the problems analyzed. This is due to minimizing popping variables.

(iii) The wastage of computations due to degeneracy is eliminated.

(iv) The tie-breaking rules minimize the number of violating constraints in the post-optimality phase.

(v) The algorithm works satisfactorily for all problems (both sparse and dense). However, the model reduction algorithm with alternate criterion works well for the problem having high sparsity.

3.10 STATISTICAL ANALYSIS

About 60 original and reduced models of linear programming problems with varying number of constraints and variables were solved. Since no common inference could be drawn due to the variation in the dimensions of the problems, statistical
Figure 3.3 Comparison of computational time of the original and the reduced models using the Simplex algorithm

Figure 3.4 Comparison of computational time of the original and the reduced models using the Multiplex algorithm
Figure 3.5 Comparison of the actual and detected redundant constraints

Figure 3.6 Comparison of the actual and detected redundant variables
Table 3.2 Comparison of computational operations and the time between the original LP model and the reduced one on application of Simplex algorithm

<table>
<thead>
<tr>
<th>Sl.No.</th>
<th>Dim. (m x n)</th>
<th>Redundancies</th>
<th>Simplex</th>
<th>Reduced model</th>
<th>Speed ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Const. (detected/actual)</td>
<td>Var. (detected/actual)</td>
<td>Original model</td>
<td>Iter.</td>
</tr>
<tr>
<td>1</td>
<td>25x40</td>
<td>4 / 7</td>
<td>21 / 25</td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>2</td>
<td>25x50</td>
<td>8 / 9</td>
<td>19 / 21</td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>20x70</td>
<td>5 / 6</td>
<td>27 / 33</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>25x75</td>
<td>14 / 18</td>
<td>19 / 39</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>15x36</td>
<td>6 / 8</td>
<td>1 / 12</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>28x28</td>
<td>7 / 9</td>
<td>9 / 17</td>
<td></td>
<td>28</td>
</tr>
</tbody>
</table>
analysis of the results was made. Some inference on the basis of mean, standard deviation and coefficient of variance was drawn to merit the efficiency of the algorithms. Simplex and Multiplex algorithms were used to solve different models and the speed ratios for the problems were computed. The results of the statistical analysis are furnished in table 3.4.

The Simplex algorithm when applied to the reduced and the original models increases the speed ratio on an average to 3.15 although the largest and the smallest speed ratios are 10.86 and 1.11 respectively. Similarly the Multiplex algorithm, when applied to the respective models results on an average a speed ratio of 3.37 with the most and the least ratios of 11.69 and 1.20. The difference in the average speed ratios shows the supremacy of the Multiplex algorithm over the Simplex.

The coefficient of variance on the speed ratio is 75.75 for the Simplex and 67.85 for the Multiplex algorithm. Since the coefficient of variance for the Multiplex algorithm on the reduced model is less than that of Simplex on the same model, the Multiplex-reduced model combination is considered to be more consistent than the other combination. The model reduction algorithm detects on an average 65.84% of the redundant constraints with a standard deviation of 25.83 and 58.02% of the redundant variables with a standard deviation of 25.11.

The inference and conclusion of this analysis is that the application of the Multiplex algorithm on the reduced model is more consistent and efficient than the Simplex on the same model.
Table 3.4 Results of statistical analysis

<table>
<thead>
<tr>
<th>Description</th>
<th>Statistical parameters connected with the speed ratios</th>
<th>Percentage of redundancies detected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplex</td>
<td>Multiplex</td>
</tr>
<tr>
<td>Average</td>
<td>3.15</td>
<td>3.37</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>2.386</td>
<td>2.286</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.11</td>
<td>1.20</td>
</tr>
<tr>
<td>Maximum</td>
<td>10.86</td>
<td>11.69</td>
</tr>
<tr>
<td>Coefficient of variance</td>
<td>75.75</td>
<td>67.85</td>
</tr>
</tbody>
</table>

3.11 CONCLUSION

The model reduction algorithm developed was tested on a variety of problems partly downloaded from net-library and partly from other sources. It has worked satisfactorily on all of them. This algorithm uses a simple logic, which can be adopted very easily by the existing packages. It can also be extended to problems, which are directly, or indirectly seeking LP solution procedure. The ease in implementation is an additional advantage of this algorithm.

Consider an example to illustrate the limitation of the proposed algorithm:
Maximize $Z = x_1 + x_2$

subject to $x_1 + x_2 \leq 3$ (1)

$-x_1 + x_2 \leq 2$ (2)

$x_1 - x_2 \leq 2$ (3)

$x_1, x_2 \geq 0$

The model reduction algorithm identifies constraint 1 as redundant and the reduced model is obtained. But it leads to an unbounded solution space. The model validation routine includes the redundant constraint again into the reduced model and by applying post-optimality analysis the optimal solution is obtained. Few problems of similar type are analyzed and conclusions drawn. i.e., even though some additional computations are consumed due to model validation and post-optimality analysis, the overall computational time is minimum because it adopts the divide and govern technique.

The detection and removal of redundancies not only brings down the dimension of the problem, but also minimizes the computational time by the LP solvers. While identifying the redundancies savings are obtained in the form of memory requirement and computational time. It also minimizes the popping tendency of the variables for the simple reason that, the majority of the popping
variables stay as non-basic redundant variables. But the computational savings obtained from the identification of the redundant variables are comparatively less compared with the constraints.

The model reduction algorithm takes in a few cases more number of iterations (about 5%) than the regular LP solvers. This is due to the fact that the model validation is done using the post-optimality analysis, i.e., the constraints or variables or both are heuristically identified and included in the redundant set. Due to the limitations of the model reduction algorithm the model validation routine checks the validity of the identified redundancies. Violated redundancies are added to the reduced model using the post-optimality analysis. During this process it uses a few more iterations. But the computational time consumed and the number of floating point operations performed on the reduced and validated model are less compared to the regular methods. This is due to the fact that the size of the reduced model obtained by the application of the heuristic algorithm is considerably less than the original.

The alternate criterion for the model reduction is also a simple heuristic algorithm to detect redundancies of a given LP model using the gradients of the constraint matrix. But it is observed that the alternate criterion is inferior to the matrix of intercepts criterion for problems having dense constraint matrices. However Karmarker's standard form of LP model can use only the alternate criterion, and reduce the given model. Here only the alternate criterion is applicable for the simple reason that the right hand side values of the constraints of the Karmarker's transformed model of the given LP model are zero. The only exception is the constraint \( \sum \lambda_j = 1 \) (\( j = 1 \) to \( n \)). In such special cases the alternate criterion is applied.
The model reduction algorithm also eliminates the wastage of computation due to degeneracy and cyclic problems. Cyclic avoidance rule is an inherent property of the model reduction algorithm.

After a careful and detailed study, a method is proposed to break the tie during the selection of minimum/maximum of the column entries of the matrix of intercepts. It saves a good deal of computational time by avoiding the wrong detection of redundant constraints/variables by the model reduction algorithm.

The model reduction algorithm is validated on PERT network problems for the identification of non-critical activities. It works strongly on these models in detecting the non-critical activities.

Following the attainment of the optimal solution for the initial reduced model and before validation is checked, the matrix of intercepts is constructed for the reduced model to check whether there is any further redundancy defying detection at the first instance. If there is, then a revised $B^{-1}$ matrix should be constructed before the reduced model is validated. If no further redundancy is identifiable then it is concluded that the matrix of intercepts in general has the limitation of identifying redundancies only in the original model and not in the reduced. There is scope for further probe in this area to detect additional set of redundancies if any to improve the computational efficiency.

In this dissertation the extension of model reduction algorithm to bounded variable problems and decomposition problems are also discussed in the next few chapters.