CHAPTER 5

APPLICATIONS OF MULTIPLEX ALGORITHMS
TO SOLVE DECOMPOSITION PROBLEMS USING DATA STRUCTURE

5.1 INTRODUCTION

Revised Simplex algorithm is an established and well founded procedure to solve Linear Programming problems. But for solving Large-Scale linear programming problems, simplex algorithm is not suitable since it is an edge oriented univariate search method and therefore it is painfully slow in convergence to seek optimal solution. As an improvement to solve large scale linear programming problems having block angular structure, decomposition principle has been developed which divides a large problem into a number of small subproblems. This is more or less a twin problem like the primal and dual, one known as master program and the other a set of subproblems. Block angular structure problems are solved very efficiently using decomposition principle in conjunction with simplex procedure. Here again it suffers the drawback of being painfully slow in convergence and consumes considerable computer time since the same revised simplex procedure is used for solving subproblems and master program. Hence in order to make the convergence rapid, multiplex and dual multiplex algorithms using decomposition principle have been applied to solve those subproblems and the master program.

Usually large scale real life linear programming problems have a highly sparse constraint matrix. Hence the matrices of the subproblems and the master program for large
scale problems are also highly sparse. In this chapter multiplex algorithms in conjunction with data structure are applied to solve LP problems identifiable as having block angular structure. Data structure concepts of multiplex and dual multiplex algorithms have been employed to store only the non-zero elements and to perform arithmetic operations only on them. This will reduce the wastage of computer memory and also avoid the unnecessary computation to be performed to check the zero entries before every multiply/divide operation.

5.2 DECOMPOSITION PRINCIPLE

The procedure using Decomposition principle is the most efficient when applied to linear programming problems whose constraint coefficient matrices have block angular structure, i.e., one or more independent blocks linked by coupling constraints. It operates by forming an equivalent "master program", with only a few more rows than the coupling constraints in the original problem, but with very many columns. This program is solved without tabulating all the columns, but by generating them only when the algorithm needs, a technique we call "column generation". The resulting algorithm involves iteration between a set of independent subproblems whose objective functions contain variable parameters, and the master program. The subproblems receive a set of parameters (simplex multipliers or pricing vector) from the master program, which combines these with previous solutions in an optimal way and computes new prices. Once all the subproblems are solved, the master program gets augmented with new columns and test for optimality is performed. If the master program fails to be optimal, then new prices are computed and this in turn updates the new prices for the subproblems. The iteration
proceeds until the optimality test is passed. The procedure has an elegant economic interpretation, in which the master program coordinates the actions of the subproblems by setting prices on resources used by these problems.

5.2.1 Column generation

Consider a linear program

\[ \begin{align*}
\text{minimize} & \quad Z = \sum_{j=1}^{n} c_j x_j \\
\text{subject to} & \quad \sum_{j=1}^{n} p_j x_j = b \\
& \quad x_j \geq 0, \ j = 1,2,\ldots,n
\end{align*} \]

where the \( p_j \) and \( b \) are \( m \) component vectors, \( m < n \). Assume that an initial basic feasible solution, \( X_B \), is available, with associated basis matrix \( B \), and cost coefficients \( C_B \). Such a solution, if exists, may be found by using a phase I procedure. The simplex multipliers associated with this basis are

\[ \pi = C_B B^{-1} \]

and are always made available by the simplex method. To improve the basic feasible solution we 'price out' all columns corresponding to nonbasic variables by forming their relative cost coefficients

\[ \bar{c}_j = c_j - p_j \]

If \( \min_j \bar{c}_j = \bar{c}_s < 0 \)
then, barring degeneracy, the current solution may be improved by introducing \( x_s \) into the basis via a pivot transformation.

If there are many columns, i.e., from several thousands to several millions, then finding \( \min c_j \) by computing may be tedious, if not impossible. Fortunately, in such large problems, the set of all columns generally has a well-defined structure, owing to the pattern of the real world situation giving rise to the problem. This is almost always true, and fortunately so, since otherwise the task of gathering unordered numerical data to specify all the columns would be hopeless.

In general, then, let us assume that all columns, \( p_j \), are drawn from a set, \( S \), which typically is the set of all \( m \) vectors satisfying some system of equations or equalities. The column to enter the basis may then be chosen by solving the subproblem

\[
\minimize_{p_j \in S} c(p_j) - \pi p_j
\]

where \( c(p_j) = c_j \) is a given function of \( p_j \). Depending on the structure of \( S \) and the form of \( c(.) \), a variety of techniques may be used to solve this subproblem, e.g., linear programming in the Decomposition principle. This approach is called column generation because, in solving the subproblem, only a small subset of columns in \( S \) are typically examined, and these are generated as and when needed. Thus none of the columns need be kept in computer storage, a definite advantage for large problems.

Linear programs may be large not only in their number of variables but in their number of constraints as well. One
way to deal with this difficulty is to transform the problem into an equivalent linear program with fewer constraints but many more variables. Column generation methods are then applied to the equivalent problem. This is the basis for the Decomposition principle.

5.2.2. Development of Decomposition principle

Consider a linear program whose constraint coefficient matrix has a p-block angular structure, with $p \geq 1$:

$$
A = \begin{bmatrix}
A_1 & A_2 & \ldots & A_p \\
B_1 & B_2 & & \\
& & & B_p
\end{bmatrix}
$$

Any linear program may be considered to have this form with $p = 1$ by partitioning its constraints into two subsets:

minimize $Z = CX$ \hspace{1cm} (5.1)

subject to

$$
\tilde{A}_1 X = \tilde{b}_1 \hspace{1cm} (m_1 \text{ constraints}) \hspace{1cm} (5.2)
$$

$$
A_2 X = b_2 \hspace{1cm} (m_2 \text{ constraints}) \hspace{1cm} (5.3)
$$

$$
X \geq 0 \hspace{1cm} (5.4)
$$
Assume that the convex polyhedron

\[ S_2 = \{ X | A_2 X = b_2, X \geq 0 \} \tag{5.5} \]

is bounded. Then any element of \( S_2 \) may be written as

\[ X = \sum_j \lambda_j X^j \tag{5.6} \]

where

\[ \sum_j \lambda_j = 1, \quad \lambda_j \geq 0 \tag{5.7} \]

and the \( X^j \) are extreme points of the polyhedron \( S_2 \).

The original problem (5.1) to (5.4) may be viewed as follows: Choose, from all solutions of (5.3) and (5.4), those which satisfy (5.2) and minimize \( Z \). To enforce satisfaction of (5.2), substitute (5.6) into (5.2), obtaining

\[ \sum_j (A_1 X^j) \lambda_j = b_1 \tag{5.8} \]

Substituting (5.6) into (5.1) gives an expression for \( Z \) in terms of the variables \( \lambda_j \):

\[ Z = \sum_j (A_1 X^j \lambda_j) \tag{5.9} \]

defining

\[ A_1 X^j = p_j \tag{5.10} \]

\[ C X^j = f_j \tag{5.11} \]

relations (5.7) to (5.9) are seen to compromise a linear program in \( \lambda_j \).
minimize $\sum_j f_j \lambda_j$ \hfill (5.12)

subject to

\[ \sum_j P_j \lambda_j = b_1 \] \hfill (5.13)

\[ \sum \lambda_j = 1 \] \hfill (5.14)

\[ \lambda_j \geq 0 \] \hfill (5.15)

This program called the master program, is completely equivalent to the original. It has only $m_1 + 1$ rows, compared to the $m_1 + m_2$ rows of the original problem, a sizable saving if $m_2$ is large. It also has as many columns as the polyhedron $S_2$ has extreme points, which may be many thousands if $m_2$ is large.

Rather than tabulating all these columns, a column-generation technique is used, creating columns to enter the basis as they are needed. To see how this is done, consider the relative cost coefficient for the variable $\lambda_j$:

\[
- f_j = f_j - \pi \begin{bmatrix} P_j \\ 1 \end{bmatrix}
\] \hfill (5.16)

partitioned as

\[
\pi = (\pi_1', \pi_0')
\]

where $\pi_1$ corresponds to the constraints (5.13) and the scalar $\pi_0$ to the single constraint (5.14). Then, using the definitions of $f_j$ and $p_j$ in (5.10) and (5.11), $f_j$ may be written as

\[
\tilde{f}_j = (C - \pi_1 A_1)X^s - \pi_0
\] \hfill (5.17)
The usual simplex criterion needs to find

$$\min_j f_j = f_s = (C - \pi_1 \lambda_1)X^s - \pi_0 \tag{5.18}$$

in order to choose a variable, $\lambda_s$, to enter the basis. Recalling that $X^j$ is an extreme point of $S_2$, it is noted that $f_j$ is linear in $X^j$. Since an optimal solution of a linear program whose constraint set is bounded always occurs at an extreme point of that set, (5.18) is equivalent to the subproblem

$$\text{minimize } (C - \pi_1 \lambda_1)X \tag{5.19}$$

subject to $A_2X = b_2; X \geq 0; \tag{5.20}$

To find a column to enter the basis of the master program, this subproblem is solved to obtain a solution, $X^s$. Then the column to enter the basis is

$$p_s = \begin{bmatrix} \lambda_1 X^s \\ \vdots \\ 1 \end{bmatrix} \tag{5.21}$$

with cost coefficient

$$f_s = CX^s \tag{5.22}$$

This approach becomes particularly attractive if $p$, the number of independent blocks in the angular structure, is greater than one, i.e., if the problem to be solved is of the form
minimize $Z = \sum_{i=1}^{p} c_i x_i^i + \sum_{i=2}^{p} c_i x_i^i + \ldots + c_x x_{11}^p$ (5.23)

subject to $A_1 x_1 + A_2 x_2 + \ldots + A_p x_p = b_0$

$B_1 x_1 = b_1$

$B_2 x_2 = b_2$ (5.24)

$B_p x_p = b_p$

$x_1 \geq 0, x_2 \geq 0, \ldots, x_p \geq 0, p > 1$ (5.25)

then the subproblem defined by (5.19) and (5.20) becomes

minimize $\sum_{i=1}^{p} (c_i - \pi_i a_i) x_i^i$ (5.26)

subject to $B_i x_i = b_i, x_i \geq 0, i = 1, 2, \ldots, p$ (5.27)

Since the objective, (5.26), is additively separable in the $x_i$ and the constraints on the $x_i$, (5.27), are independent, this problem reduces to the $p$ independent subproblems

minimize $(c_i - \pi_i a_i) x_i$ (5.28)

subject to $B_i x_i = b_i, x_i \geq 0$ (5.29)

5.2.3. Decomposition algorithm

A two-level algorithm for the solution of (5.23) to (5.25) may now be formulated, with the master on the second level and the subproblems defined by (5.28) and (5.29) on the first. Assume that an initial basic feasible solution for the
master program, (5.12) to (5.15), is available, with basis matrix B and simplex multipliers \((\pi_1, \pi_0)\).

**Step 1:** Using the simplex multipliers \(\pi_1\), solve the subproblems defined by (5.28) and (5.29) to obtain solutions \(X_1(\pi_1)\) and optimal objective values \(Z^0_1\).

Let \(X(\pi_1) = (X_1(\pi_1), \ldots, X_p(\pi_1))\).

**Step 2:** Compute \(\min f_j\), which is given by

\[
\min f_j = \frac{p}{i=1} Z^0_i - \pi_0
\]

(5.30)

if \(\min f_j \geq 0\) then stop. (5.31)

The optimal solution to (5.23) to (5.25) is

\[
X^0 = \sum_j \lambda_j X^j
\]

(5.32)

where the \(X^j\) are the extreme points of \(S_2\) corresponding to basic \(\lambda_j\).

**Step 3.** if \(\min f_j < 0\) form the column

\[
p = \left(\sum_i A_i X_i(\pi_1)\right)
\]

(5.33)

transform it by multiplication by \(B^{-1}\), and, using the simplex pivot operation, obtain a new basis inverse and a new vector of simplex multipliers. Go back to step 1 and repeat.

If the master program is nondegenerate, then each iteration decreases \(Z\) by a nonzero amount. Since there are only a finite number of possible bases, and none is repeated,
the decomposition principle will find the optimal solution in a finite number of iterations.

Note that the optimal solution, $X^0$, is not necessarily any one of the subproblem proposals. By $(5.32)$, $X^0$ is some convex combination of a number of such solutions. Thus the master program cannot obtain the overall optimum simply by sending appropriate prices, $\pi_1$, to the subproblems; it must have the freedom to combine subproblem solutions into an overall plan. In this sense, the decomposition obtained here cannot be viewed as complete decentralization of the decision-making process. A better term would be "centralized planning without complete information at the center".

5.2.4. Restricted master program

As previously formulated, the Decomposition principle solves optimization problems on the first level, but not at the second, where only a single pivot operation is performed. By defining a restricted master program a more symmetric formulation is possible in which both levels solve linear programs. This is simply the master program $(5.12)-(5.15)$ with all columns dropped but those in the current basis and that column about to enter. Then the restricted master program is

$$\begin{align*}
\text{minimize} & \quad f_1 \lambda_1 + \ldots + f_m \lambda_m + f \lambda \\
\text{subject to} & \quad p_1 \lambda_1 + \ldots + p_m \lambda_m + p \lambda = b_0 \\
& \quad \lambda_1 + \ldots + \lambda_m + \lambda = 1 \\
& \quad \lambda_i \geq 0, \quad i = 1, \ldots, m, \quad \lambda \geq 0
\end{align*}$$

(5.34, 5.35, 5.36, 5.37)
where \( m = m_1 + 1 \), the \( \lambda_i \) are the current basic variables and \( \lambda \) is the variable entering. Assuming that the variable \( \lambda \) had an \( f < 0 \), it enters the basis. If the current basis is nondegenerate, the variable leaving will have a positive \( f \), so the new solution will be optimal. The only case in which more than one iteration needed to pass the optimality test is, when the current basis is degenerate and the pivot element in the entering column is negative. Thus, little is gained by solving (5.34) to (5.37). However, as we now show, other formulations of the master program can make the restricted master program much more worthwhile.

5.3 MULTIPLEX AND DUAL MULTIPLEX ALGORITHMS WITH DATA STRUCTURE

The algorithm presented in this chapter exploits the concept of Data structure using linked list to represent the matrices which are involved in solving Linear programming problems. This algorithm employs two types of linked list representations namely columnwise circularly linked list and Multiply linked list for different purposes.

In this algorithm, all the independent blocks are treated as subproblems. When all the subproblems are solved, augmenting columns to the master program is done. This process of solving all the subproblems and augmenting the columns to the master program is referred to as one iteration of the master program. In each iteration

i. All the \( p \) subproblems are solved independently and

ii. Using the results of those subproblems, master program is constructed.
Master program contains the number of coupling constraints and one constraint for each block. Moreover, each time \( p \)-number of columns are generated. Each time when a new column is generated per block for the master program a new column header node is created and the values are stored in that column. At the same time the columns corresponding to the nonbasic variables are dropped out from the master program which will be equal to the number of columns augmented to the master program. Number of elements in each generated column will be equal to the number of coupling constraints + 1 only. Hence irrespective of the sparsity of the problem the master program is a sparse one.

Since normally the real life large scale problems are highly sparse, all the subproblems also will be sparse. Without data structure representation of the coefficient matrices, each time when a subproblem is solved all the elements in the constraint matrix corresponding to the subproblems have to be accessed. Whereas using data structure representation, only address of all column header nodes alone need be accessed with the help of pointers. Hence access time is considerably reduced each time when the subproblems are solved. Summarizing the above facts that the Data structure implementation for decomposition principle enables faster convergence and reduction of memory used for the following reasons:

i. Only non-zero elements of all the constraint matrices are stored,

ii. All the arithmetic operations are performed only on non-zero elements and unnecessary checking of zero/non-zero elements is thus avoided and

iii. Access time for coefficient matrix is reduced.
In the decomposition principle, constraint coefficient matrices for all the subproblems will remain constant throughout the process and only the objective function coefficients will change. But the master program consists of the optimal basic columns of the previous master program (all other columns are dropped) plus p number of newly augmented columns. Multiplex and dual multiplex algorithms using data structure concept is invoked to solve all the subproblems and master program for every iteration.

In the Decomposition principle, memory size for processing will be considerably reduced since the size of the constraint coefficient matrix of each subproblem is \((1/p)\)th of main program size. For solving the subproblems and master program, the proposed algorithm requires the construction of

i. Constraint coefficient matrices for all the subproblems
ii. Constraint coefficient matrix for the master program and
iii. \(M^{-1}\) matrices while solving the subproblems and the master program.

Constraint coefficient matrices for all the subproblems and the master program are represented using columnwise circularly linked list since they are always used for post multiplication purpose. \(M^{-1}\) matrices are represented using multiply linked list since they are used for both pre multiplication and post multiplication purposes. This algorithm is illustrated with the following example.
Consider the problem

\[
\begin{align*}
\text{minimize } & \quad Z = -x_1 - x_2 - 2y_1 - y_2 \\
\text{subject to } & \quad x_1 + 2x_2 + 2y_1 + y_2 \leq 40 \\
& \quad x_1 + 3x_2 \leq 30 \\
& \quad 2x_1 + x_2 \leq 20 \\
& \quad y_1 \leq 10 \\
& \quad y_2 \leq 10 \\
& \quad y_1 + y_2 \leq 15
\end{align*}
\]

and all the variables are nonnegative. This problem has one coupling constraint and two independent blocks. Each block is treated as a subproblem. The constraint coefficient matrices for the subproblems corresponding to blocks 1 and 2 respectively are given by

\[
\begin{align*}
B_1 &= \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \\
B_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}
\end{align*}
\]

The feasible region for these blocks are shown shaded in Figure 5.1(a) and (b).
Figure 5.1 Feasible solution regions
Let \( X \) and \( Y \) be feasible solutions for blocks 1 and 2, respectively, and write

\[
X = \sum_{i} a_X^i, \quad Y = \sum_{i} a_Y^i
\]

with the \( X^i, Y^i \) extreme points of these blocks. The coupling constraint and objective function are written in vector notation as

\[
Z = C_1 X + C_2 Y
\]

\[
A_1 X + A_2 Y + s = 40
\]

where \( C_1 = (-1, -1), C_2 = (-2, -1), A_1 = (1,2), A_2 = (2, 1) \)

and \( s \geq 0 \) is a slack variable. The master program then becomes

\[
\text{minimize } Z = \sum_{i} \big( C_1^i a_X^i \big) + \sum_{i} \big( C_2^i a_Y^i \big) \quad \text{subject to } \sum_{i} \big( A_1^i a_X^i \big) + \sum_{i} \big( A_2^i a_Y^i \big) + s = 40
\]

\[
\sum_{i} a_X^i = 1 \quad \sum_{i} a_Y^i = 1
\]

\( a_X^i \geq 0, \quad a_Y^i \geq 0 \)
Iteration 1:

The first set of subproblems is

1. Minimize $Z_1 = c_1 x = -x_1 - x_2$ (5.44)

subject to

\[ x_1 + 3x_2 \leq 30 \] (5.45)

\[ 2x_1 + x_2 \leq 20 \]

and

2. Minimize $Z_2 = c_2 y = -2y_1 + y_2$ (5.46)

subject to

\[ y_1 \leq 10 \]

\[ y_2 \leq 10 \] (5.47)

\[ y_1 + y_2 \leq 15 \]

and

the initial master program is

Minimize $Z = 0$

subject to

\[ s = 40 \]

\[ a_1 = 1 \]

\[ b_1 = 1 \]
The representation of constraint coefficient matrices for the master program and the two subproblems using columnwise circularly linked list are shown in Figures 5.2, 5.3 and 5.4 respectively.

The solution for the master program may be obtained as

\[ X^1 - Y^1 = (0,0) \]

\[ q_1 = \frac{b_1}{1} = 1 \]

The initial \( M^{-1} \) matrix for all the subproblems and master program are represented using multiply linked list and the representation for master program is shown in Figure 5.5. From this representation one can observe that \( M^{-1} \) matrix has \( m+1 \) rows and \( m \) columns. The first row represents \([C_B B^{-1}]\).

These two subproblems are solved using multiplex and dual multiplex algorithms with data structure. The result of these two subproblems are obtained as follows.

\[ X^2 = (6,8), \quad Z_1^0 = -14 \]

\[ Y^2 = (10,5), \quad Z_2^0 = -25 \]

The minimum relative cost factors are

\[ \min f_X = Z_1^0 - \pi_{01} = -14 \]

\[ \min f_Y = Z_2^0 - \pi_{02} = -25 \]
Figure 5.2 Columnwise circularly linked list

HN: Head Node  RN: Row Number
VAL: Value of the element  DLINK: Pointer
HN: Head Node  RN: Row Number
VAL: Value of the element  DLINK: Pointer

Figure 5.3 Columnwise circular linked list
Figure 5.4 Columnwise circulary linked list

HN: Head Node  RN: Row Number
VAL: Value of the element  DLINK: Pointer
Figure 5.5 Multiply linked list
Associated with each subproblem solution is a column of the master program. The columns corresponding to the current solutions are

\[
\begin{align*}
C_1X^2 & \quad -14 & C_2Y^2 & \quad -25 \\
A_1X^2 & \quad 22 & A_2Y^2 & \quad 25 \\
1 & \quad 1 & 1 & \quad 0 \\
0 & \quad 0 & 1 & \quad 1
\end{align*}
\]

Thus the solution of these subproblems result in the generation of the following new columns for the master program:

\[
\begin{bmatrix}
\alpha_2 \\
\beta_2
\end{bmatrix}
= \begin{bmatrix}
22 \\
1 \\
0
\end{bmatrix}
\begin{bmatrix}
25 \\
0 \\
1
\end{bmatrix}
\]

These two columns are represented using columnwise circularly linked list and augmented to the basic variable columns \((s, \alpha_1, \beta_1)\) of the master program. Hence the updated master program is given by

\[
\begin{align*}
\text{minimize} & \quad -14 \alpha_2 - 25 \beta_2 \\
\text{subject to} & \quad s + 22 \alpha_2 + 25 \beta_2 = 40 \\
& \quad \alpha_1 + \alpha_2 = 1 \\
& \quad \beta_1 + \beta_2 = 1
\end{align*}
\]

with all variables nonnegative.
The columnwise circularly linked list representation of the above master program is shown in Figure 5.6. The above mentioned master program is solved by multiplex and dual multiplex algorithm. The $M^{-1}$ matrix for the master program corresponding to the optimal solution is

$$M^{-1} = \begin{bmatrix}
1 & -14/22 & 0 & -200/22 \\
0 & 1/22 & 0 & -25/22 \\
0 & -1/22 & 1 & 25/22 \\
0 & 0 & 0 & 1
\end{bmatrix} \quad (5.48)$$

The solution is given by

$$a_1 = 7/22$$

$$a_2 = 15/22$$

$$b_2 = 1$$

and

$$Z = -34(6/11)$$

Hence the new solution to the primal problem is

$$X = a_1^X + a_2^X = 7/22(0,0) + 15/22 (6,8)$$

$$Y = b_2^Y = 1 \times (10,5) \text{ and } Z = -34(6/11)$$
Figure 5.6 Column-wise circular linked list

HN: Head Node  RN: Row Number  VAL: Value of the element  DLINK: Pointer
Iteration 2:

The new simplex multipliers are found in the top row of the $M^{-1}$ matrix for the master program

$$(\pi, \pi_{01}, \pi_{02}) = (-14/22, 0, -200/22)$$

These are used to form new subproblem objectives

$$Z_1 = (C_1 - \pi A_1)X = [(-1, -1) + 14/22 (1, 2)]X$$
$$= -8/22x_1 + 6/22x_2$$

$$Z_2 = (C_2 - \pi A_2)Y = [(-2, -1) + 14/22 (2, 1)]Y$$
$$= -16/22y_1 - 8/22y_2$$

and the constraint coefficient matrices for the subproblems are the same as equations (5.42) and (5.43).

These two subproblems are solved by multiplex and dual multiplex algorithms with data structure and the optimal solution is

$$X^3 = (10, 0), \quad Z_1^0 = -80/22$$

$$Y^3 = (10, 5), \quad Z_2^0 = -200/22$$

with minimal relative cost factors

$$\min f_X = Z_1^0 - \pi_{01} = -80/22$$

$$\min f_Y = Z_2^0 - \pi_{02} = 0$$
Associated with each subproblem solution is a column of the master program. The columns corresponding to the current solutions are

\[
\begin{align*}
C_1X^3 & \quad -10 & C_2X^3 & \quad -25 \\
A_1X^3 & \quad 10 & A_2X^3 & \quad 25 \\
1 & \quad 0 & 0 & 0 \\
0 & \quad 0 & 1 & 1
\end{align*}
\]

The new master program consists of the optimal basic columns of the previous master program (all other columns are dropped) plus the following new columns:

\[
\begin{bmatrix}
\alpha_3 \\
\beta_3
\end{bmatrix} =
\begin{bmatrix}
10 \\
1 \\
0
\end{bmatrix} \quad \begin{bmatrix}
25 \\
0 \\
1
\end{bmatrix}
\]

These two columns are represented using columnwise circularly linked list and augmented to the basic variable columns \((\alpha_1, \alpha_2, \beta_2)\) of the master program. But the columns corresponding to the nonbasic variables\((\beta_1)\) are dropped out from the master program and the new master program is
minimize \(-14 \alpha_2 - 25 \theta_2 - 10 \alpha_3 - 25 \theta_3\)

subject to \(22 \alpha_2 + 25 \theta_2 + 10 \alpha_3 + 25 \theta_3 = 40\)

\[\begin{align*}
\sum \alpha_i + \alpha_2 + \alpha_3 &= 1 \\
\theta_2 + \theta_3 &= 1
\end{align*}\]

The matrix representation of this master program using columnwise circularly linked list is shown in Figure 5.7. It is seen from the Figures 5.6 and 5.7, the size of the master program is always constant. The above mentioned master program is solved by multiplex and dual multiplex algorithms with data structure. The optimal solution for the previous master program is an initial basic feasible solution for this program.

The \(M^{-1}\) matrix for the master program corresponding to the optimal solution is

\[
M^{-1} = \begin{bmatrix}
1 & -4/22 & -80/12 & -200/12 \\
0 & 1/12 & -10/12 & -25/12 \\
0 & -1/12 & 22/12 & 25/12 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(5.49)

The solution is given by

\[\begin{align*}
\alpha_2 &= 5/12 \\
\alpha_3 &= 7/12 \\
\theta_2 &= 1
\end{align*}\]

and \(Z = -110/3\)
Figure 5.7 Columnwise circular linked list

HN: Head Node  RR: Row Number  VAL: Value of the element  DLINK: Pointer
Hence the new solution to the primal problem is

\[ X = a_2 x^2 + a_3 x^3 = \frac{5}{12}(6,8) + \frac{7}{12}(10,0) \]

\[ Y = 8_2 y^2 = 1 \times (10,5) \text{ and } Z = -\frac{110}{3} \]

**Iteration 3:**

The new simplex multipliers and subproblems objectives are

\[ (\pi, \pi_{01}, \pi_{02}) = -(4/12, 80/12, 200/12) \]

\[ Z_1 = (-(1,-1)+4/12(1,2))X = \frac{-2}{3}x_1 - \frac{1}{3}x_2 \]

\[ Z_2 = (-(2,-1) + 4/12(2,1))Y = \frac{-4}{3}y_1 - \frac{2}{3}y_2 \]

The optimal subproblem solutions are

\[ X^4 = (6,8) \text{ or } X^5 = (10,0), \quad Z^0_1 = -\frac{20}{3} \]

\[ Y^4 = (10,5), \quad Z^0_2 = -\frac{50}{3} \]

with minimal relative cost factors

\[ \min f^*_X = Z^0_1 - \pi_{01} = 0 \]

\[ \min f^*_Y = Z^0_2 - \pi_{02} = 0 \]

\[ \min \bar{c}_s = -\pi_{01} = \frac{80}{12} \]
Since the coefficients are nonnegative the current primal solution is optimal and the solution is

\[ Z = \frac{-110}{3} \]

\[ x_1 = \frac{25}{3} \]

\[ x_2 = \frac{10}{3} \]

\[ y_1 = 10 \]

\[ y_2 = 5 \]

The following observations are made from the above example.

The constraint coefficient matrices for the subproblems are represented using columnwise linked list. To access the elements in the constraint matrix for solving a subproblem every time, only the addresses of all the header nodes of constraint matrix corresponding to that particular subproblem are required. Each of the constraint coefficient matrix corresponding to the other subproblems will be represented in a similar manner. Once it is represented, the structure will always be constant since the elements of all the subproblems will not be varied throughout.

Each time when a subproblem is solved, it will generate a new column for the master program. It is observed that the number of elements "generated" per column is equal to the number of coupling constraints + 1 only. Hence the master program always contains 'm' columns corresponding to the basic variables (the columns corresponding to the nonbasic variables are dropped) and 'p' number of newly
created columns. It is seen that irrespective of the type of the problem, the master program is of sparse one. The p-number of newly generated columns are represented using columnwise circularly linked list and augmented to the 'm' basic variable columns of master program. But at the same time the columns corresponding to the nonbasic variables are dropped from the master program. Thus the nodes which are having nonbasic variable elements are deleted and could be used for generation of new columns when necessary.

5.4 RESULTS AND DISCUSSIONS

The proposed algorithm is compared with the following algorithms:

i. Revised simplex using decomposition principle,

ii. Revised simplex with data structure using decomposition principle and

iii. Multiplex and dual multiplex using decomposition principle.

Few problems are solved using all the four algorithms and the results are tabulated in Table 5.1. Problems amenable for solution using decomposition principle seem to behave differently as may be observed from graphs shown in Figure 5.8. A set of linear programming problems was modeled to fall in the pattern solvable by using the well known decomposition principle.
### Table 5.1
Comparison of Computation Times with Decomposition Principle Introduced for Different Methods

<table>
<thead>
<tr>
<th>S.No</th>
<th>Variables</th>
<th>Constraints</th>
<th>Simplex</th>
<th>Simplex with Data Structure</th>
<th>Multiplex and Dual Multiplex with Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>20</td>
<td>4.07</td>
<td>3.13</td>
<td>2.96</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>25</td>
<td>4.47</td>
<td>3.52</td>
<td>3.45</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
<td>30</td>
<td>5.91</td>
<td>4.52</td>
<td>4.02</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
<td>50</td>
<td>9.91</td>
<td>7.57</td>
<td>6.43</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>60</td>
<td>11.21</td>
<td>8.32</td>
<td>7.21</td>
</tr>
<tr>
<td>6</td>
<td>330</td>
<td>65</td>
<td>11.66</td>
<td>8.46</td>
<td>7.36</td>
</tr>
</tbody>
</table>

**Note:** The table lists the computation times in seconds for different methods and configurations.
Figure 5.8 Comparison of Revised Simplex, Multiplex with & without Data Structures using Decomposition principle.
Curves are fitted connecting the number of constraints and the computational time for various algorithms as follows:

<table>
<thead>
<tr>
<th>Sl No.</th>
<th>Algorithm</th>
<th>Equation of fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Revised simplex</td>
<td>$T = 0.4 + 0.18m$</td>
</tr>
<tr>
<td>2.</td>
<td>Revised simplex with data structure</td>
<td>$T = 0.6 + 0.13m$</td>
</tr>
<tr>
<td>3.</td>
<td>Multiplex and Dual Multiplex</td>
<td>$T = 0.9 + 0.1m$</td>
</tr>
<tr>
<td>4.</td>
<td>Multiplex and Dual Multiplex with data structure</td>
<td>$T = 1.19 + 0.06m$</td>
</tr>
</tbody>
</table>

It may be observed that the curves belong to a family of straight lines and the goodness of fit varies from 98.31 percent to 99.47 percent. It is to be noted that when the same problem is solved using the conventional simplex procedure the curve is exponential [32] but when decomposed it follows a straight line. A comparison of the slopes of the various fits shows the computational efficiency of the method proposed in this dissertation.

A typical problem with 10 subproblems is solved using decomposition principle. The same problem is then solved nine more times by modifying the number of subproblems as 9, 8, 7, 6, 5, 4, 3, 2 and 1. All the above mentioned problems are tested with the proposed method and the other three methods. The results and comparisons are furnished in Table 5.2. The frequency with which variables enter and leave the basis for all the above problems are tabulated in Table 5.3. A graph connecting the number of subproblems and the time taken to
**TABLE 5.2**

**COMPARISON OF COMPUTATION TIMES WITH DECOMPOSITION PRINCIPLE INTRODUCED FOR A 700 VARIABLES 100 CONSTRAINTS PROBLEM WITH DIFFERENT NUMBER OF SUBPROBLEMS.**

<table>
<thead>
<tr>
<th>SL. NO.</th>
<th>NUMBER OF SUB PROBLEMS</th>
<th>SIMPLEX WITH DATA STRUCTURE</th>
<th>SIMPLEX WITH DATA STRUCTURE</th>
<th>MULTIPLEX AND DUAL MULTIPLEX WITH DATA STRUCTURE</th>
<th>MULTIPLEX AND DUAL MULTIPLEX WITH DATA STRUCTURE</th>
<th>SPEED RATIO WITH SIMPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>47.79</td>
<td>46.34</td>
<td>18.50</td>
<td>13.23</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>58.20</td>
<td>51.17</td>
<td>20.24</td>
<td>13.48</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>61.16</td>
<td>55.42</td>
<td>21.56</td>
<td>14.57</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>70.38</td>
<td>56.23</td>
<td>24.30</td>
<td>14.82</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>81.24</td>
<td>62.35</td>
<td>26.44</td>
<td>15.32</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>99.27</td>
<td>70.42</td>
<td>32.12</td>
<td>16.25</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>125.19</td>
<td>93.28</td>
<td>40.30</td>
<td>18.10</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>157.27</td>
<td>112.58</td>
<td>48.67</td>
<td>21.24</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>202.58</td>
<td>146.12</td>
<td>55.20</td>
<td>25.15</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>267.41</td>
<td>182.34</td>
<td>68.23</td>
<td>31.20</td>
<td>9</td>
</tr>
</tbody>
</table>
**TABLE 5.3**

**COMPARISON OF FREQUENCY WITH DECOMPOSITION PRINCIPLE**, introduced for a 700 VARIABLES 100 CONSTRAINTS PROBLEM WITH DIFFERENT NUMBER OF SUBPROBLEMS.

<table>
<thead>
<tr>
<th>SL. NO.</th>
<th>NUMBER OF SUBPROBLEMS</th>
<th>FREQ. OF VARIABLES ENTERING AND LEAVING THE BASIS</th>
<th>NUMBER OF ITERATIONS</th>
<th>SAVINGS IN TERMS OF ITERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SMP MULT 2</td>
<td>SMP MULT 3</td>
<td>SMP MULT 4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>10</td>
<td>56</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>9</td>
<td>60</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>8</td>
<td>65</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>7</td>
<td>77</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6</td>
<td>84</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>5</td>
<td>93</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>4</td>
<td>106</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>3</td>
<td>131</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>2</td>
<td>166</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1</td>
<td>185</td>
<td>56</td>
</tr>
</tbody>
</table>
find solution by different algorithms is shown in Figure 5.9 as shown below:

<table>
<thead>
<tr>
<th>Sl No.</th>
<th>Algorithm</th>
<th>Equation of fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Revised simplex</td>
<td>$T = 278.16e^{-0.18m}$</td>
</tr>
<tr>
<td>2.</td>
<td>Revised simplex with data structure</td>
<td>$T = 180.7e^{-0.15m}$</td>
</tr>
<tr>
<td>3.</td>
<td>Multiplex and Dual Multiplex</td>
<td>$T = 72.27e^{-0.14m}$</td>
</tr>
<tr>
<td>4.</td>
<td>Multiplex and Dual Multiplex with data structure</td>
<td>$T = 28.56e^{-0.08m}$</td>
</tr>
</tbody>
</table>

It may be observed from the exponentially decaying curves that all the algorithms tend to be computationally more efficient when the number of subproblems is large.

It has been observed that the proposed multiplex and dual multiplex algorithms with data structure concept using decomposition principle is faster than the other three algorithms and

i. When the size of the problem is large, the percentage saving in computational effort in the proposed algorithm is also large. Hence it can be concluded that this algorithm is highly suitable for large scale problems.

ii. Multiplex and dual multiplex algorithms minimize the popping tendency of variables and
Figure 5.9 Comparison of computation times with Decomposition Principle introduced for a 100 x 700 problem with different number of subproblems.
iii. The popping tendency of variables is reduced as may be observed from Table 5.3 when a given problem is split into a number of subproblems.

Thus this chapter is concluded with the analysis of the savings obtained in the proposed algorithm when compared individually with the above said three algorithms while solving problems amenable to solution using decomposition principle.