Computer Solution of Linear Programming Problems Using the Decomposition Algorithm

1973

Walter Paul Kraslawsky
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/61

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
COMPUTER SOLUTION OF LINEAR PROGRAMMING PROBLEMS USING THE DECOMPOSITION ALGORITHM

BY

WALTER PAUL KRASLAWSKY
B.S., Florida Technological University, 1972

RESEARCH REPORT
Submitted in partial fulfillment of the requirements for the degree of Master of Science in the Graduate Studies Program of Florida Technological University, 1973.

Orlando, Florida
TABLE OF CONTENTS

INTRODUCTION .................................................. 1

Chapter
I. DECOMPOSITION ALGORITHM .............................. 4
   Notational Conventions
   Two-phase Method
   Principles of Decomposition
   The Final Solution
II. COMPUTER SOLUTION .................................... 15
   Technical Difficulties
   Implementation Procedures
III. CONCLUSIONS ........................................... 18

Appendix
A. USER'S GUIDE ........................................... 20
B. COMPUTER LISTING .................................... 28

BIBLIOGRAPHY ................................................ 42
INTRODUCTION

Theoretically, it is possible to solve any linear programming model by the application of the simplex method. The determination of feasible, bounded, and optimal solutions require a finite number of iterations even when the numbers of variables and constraints exceed one million each. From a practical viewpoint, however, even the most modern computers would be hard pressed to find solutions for such a large problem before the data (and the computer) became obsolete. Given a large model of a system such as the economy of the United States, it is fortunate that the household budget of an individual is subject to constraints and variables that do not apply to his next door neighbor. There are other constraints and variables which will affect the national economy. Proper use of the decomposition algorithm will enable each individual to balance his budget while allowing a significant reduction in complexity of the national budget.

A major and popular use of decomposition lies in the area of decentralized planning. The central headquarters of a national firm has the responsibility for coordinating the efforts of each of hundreds of branch offices. Some decisions which are made will result in modifying the optimal output of a branch such that the overall company goals are more efficiently met. For instance, each branch office can
solve its own problems in work loads while letting the home office determine product prices.

Both of these applications are chiefly concerned with a reduction in the complexity of the problems which must be solved at any one time. The orientation is such that someone doesn't know about or care about the constraints on the values of the variables in someone else's department. All that matters is that some of these variables affect the total problem; therefore, modify the optimal solutions of the other departments so that all constraints are satisfied.

Another application of the decomposition algorithm is of no less importance: reduction of the space requirements of a linear programming problem to permit the use of available computing facilities. A problem of as few as 150 constraints and 100 variables will normally contain a large number of zero coefficients. For instance, a group of three variables might have their values completely constrained by five equations. The number of zero coefficients involved is 920 compared to only 15 non-zero ones. The remaining variables and constraints may show a similar waste of valuable computer memory space.

The purpose of this research report is to describe the decomposition algorithm from the viewpoint of a computerized solution, where the method was selected to reduce the memory requirements of a problem. Due to the increased complexity and size of the object code over simpler programs written to solve linear programming problems, the user must determine
the break even point. An algorithm has been provided to assist in this, and a guide to the modification of this program to permit overlaying subproblems is also suggested. Finally, a user's guide has been provided to permit any individual with a linear programming problem to enter the input and interpret the output without detailed knowledge of decomposition. However, a pre-requisite for understanding the content of the text is working knowledge of decomposition by the "Big M" method.
CHAPTER I

DECOMPOSITION ALGORITHM

Notational Conventions

In view of the computer orientation of this report, the notation chosen is very similar to that of the major computer languages FORTRAN and PL/1, rather than the traditional mathematical symbology. The following conventions will be used consistently and should present no difficulties to the reader.

1. Variable names begin with a letter and may consist of several letters and numbers.
2. Multiplication is denoted by the asterisk (*) symbol.
3. Vector and matrix subscripts follow the name and are enclosed in parentheses.
4. Qualifiers precede the variable name and are separated from the name by a period.

Two-phase Method

Prior to describing the solution of a linear program by decomposition, it would be best to briefly examine the solution of a small problem by the two-phase method. Problem 1 is:
maximize \(-Z = X5 + X6\)
subject to \(X5 + X6 \geq 5\)
\(X5 + 5X6 \leq 50\)

where \(X5, X6 \geq 0\).

The simplex tableau for this problem is shown in Figure 1.

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>X5</th>
<th>X6</th>
<th>S5</th>
<th>S6</th>
<th>-W</th>
<th>-Z</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-Z</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R6</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

FIGURE 1.--Simplex tableau for problem 1

Artificial variables must be removed from the solution of the problem in phase I. The significance of the artificial objective function has been stressed by showing the tableau prior to forming the starting basis. This is done by subtracting each of the constraint rows from the \((-W)\) row. The result of this step is given in Figure 2. This tableau is

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>X5</th>
<th>X6</th>
<th>S5</th>
<th>S6</th>
<th>-W</th>
<th>-Z</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W</td>
<td>-55</td>
<td>-2</td>
<td>-6</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-Z</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R6</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

FIGURE 2.--Starting basis for problem 1

now ready for the simplex procedure. Slack and surplus variables are entered, and objective functions have had the sign
reversed for minimization. Two iterations of the simplex procedure result in the first feasible solution shown in Figure 3. If the solution value of $W$ had been positive

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$-W$</th>
<th>$-Z$</th>
<th>$R_5$</th>
<th>$R_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-W$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$-Z$</td>
<td>10</td>
<td>-0.8</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_6$</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>$S_5$</td>
<td>5</td>
<td>-0.8</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

FIGURE 3.—First feasible solution to problem 1

there would have been no feasible solution. Phase II now allows us to find the optimal solution by bringing $X_5$ into the basis as in Figure 4. It can be seen that the columns

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$-W$</th>
<th>$-Z$</th>
<th>$R_5$</th>
<th>$R_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-W$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$-Z$</td>
<td>50</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$X_5$</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$S_5$</td>
<td>45</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

FIGURE 4.—Optimal solution to problem 1

under the artificial variables (including $W$ and $Z$) form an inverse matrix. By pre-multiplying any column of the starting basis tableau in Figure 2 by this inverse, the column in the optimal solution tableau is generated. This is the key to solving problems by decomposition.
Let us now consider how to handle a change in the objective function of problem 1. The new function will be:

\[
\text{maximize } -Z = X_5 + 2X_6 - S_5
\]

with all other conditions of problem 1 unchanged. The new optimal solution is found in three steps. First, change the optimal solution tableau to show the new function. All coefficients not in the new objective function are set to zero as in Figure 5. Second, perform the row operations necessary to remove the current basis variables from the o.f. row. Since the coefficients of all basis variables are zero in the row of all other basis variables, just subtract each basis row times the o.f. coefficient from the o.f. row. This step is shown in Figure 6. Third, perform the simplex procedure to re-optimize the solution. In this case, the variable \( X_6 \) has a negative objective function coefficient and should be brought into the basis. The results of step three are shown in Figure 7. We have now seen how a change in the objective function does not require performing the entire phase I, phase II process to obtain an optimal solution.

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>( X_5 )</th>
<th>( X_6 )</th>
<th>( S_5 )</th>
<th>( S_6 )</th>
<th>(-W)</th>
<th>(-Z)</th>
<th>( R_5 )</th>
<th>( R_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-W)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(-Z)</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(X_5)</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(S_5)</td>
<td>45</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

FIGURE 5.--Change step 1
solution once one is already known. This is the second key to the decomposition algorithm.

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>X5</th>
<th>X6</th>
<th>S5</th>
<th>S6</th>
<th>-W</th>
<th>-Z</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-Z</td>
<td>5</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X5</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S5</td>
<td>45</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

FIGURE 6.--Change step 2

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>X5</th>
<th>X6</th>
<th>S5</th>
<th>S6</th>
<th>-W</th>
<th>-Z</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-Z</td>
<td>15</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>X6</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>S5</td>
<td>5</td>
<td>-0.8</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

FIGURE 7.--Change step 3

Principles of Decomposition

For many problems the constraints can be divided into subsets of independent equations which might refer to the same time period or production facility. These subsets are related to each other by another set of equations called tie-in constraints. The tie-in constraints might refer to some resources which must be shared by all subsets. Figure 8 represents such a problem consisting of three subsets which has been decomposed and blocked.
Each subproblem is associated with several items which are to be qualified by the appropriate prefix when needed:

1. \( C \) = Vector of objective function coefficients.
2. \( D \) = Vector of artificial function coefficients.
3. \( T \) = Matrix of tie-in constraint coefficients.
4. \( S \) = Matrix of subproblem constraint coefficients.
5. \( B \) = Vector of right-hand sides.
6. \( X \) = Vector of subproblem variables.
7. \( X_n \) = Vector of solution values for \( n \)th optimization.
8. \( V_n \) = Vector of shadow prices for \( n \)th optimization.
9. \( Y_n \) = Generated variable associated with \( n \)th optimization.

A zero prefix indicates that the item is not a part of the set of subproblems and must be handled separately as will be seen in the following discussion.

The problem can now be stated as finding the vectors \( 0.X, 1.X, 2.X, \) and \( 3.X \geq 0 \), which solve the linear programming problem:
subject to \[ 0.T*0.X + 1.T*1.X + 2.T*2.X + 3.T*3.X = 0.B \]
\[ 1.S*1.X = 1.B \]
\[ 2.S*2.X = 2.B \]

where \( L.C \) is an \( L.N \) element row vector,
\( L.T \) is a \( O.M \) by \( L.N \) matrix,
\( L.S \) is an \( L.M \) by \( L.N \) matrix,
\( L.B \) is an \( L.M \) element column vector,
and \( L.X \) is an \( L.N \) element column vector of variables.

The above problem has \( O.M+1.M+2.M+3.M \) constraints, and
\( O.N+1.N+2.N+3.N \) variables, where \( O.N \) is permitted to be zero.

It is at this point that we attempt to reduce the size
of the linear program. Each subproblem can be thought of as
a separate program. If any subproblem has no feasible solution
then the entire problem has no solution. If any has an
unbounded solution, then a general constraint can be added
such that the sum of the variables equals some arbitrarily
large value. The tie-in constraints will very likely reduce
the feasible values of the variables to a subset of the
solution spaces of the subproblems. Since any point in the
feasible region of a subproblem can be expressed as a linear
combination of the extreme points, and the extreme points are
optimal solutions to the subproblems found by using different
objective functions, the principle of decomposition is to
let subproblem optimizations keep track of solutions while
a master program keeps track of how much each optimal subproblem solution is allowed to be used in the overall solution to the complete problem.

The starting basis for the simplex tableau of the restricted master program will now be shown in Figure 9, and the parts will be defined. To start, let \( n = 1 \).

### Admissible Variable Matrix

<table>
<thead>
<tr>
<th>Basis</th>
<th>Sol.</th>
<th>0.X</th>
<th>1.Yn</th>
<th>2.Yn</th>
<th>3.Yn</th>
</tr>
</thead>
<tbody>
<tr>
<td>-W</td>
<td>(0.\text{W}_0)</td>
<td>(0.\text{D})</td>
<td>(1.\text{Dn-1})</td>
<td>(2.\text{Dn-1})</td>
<td>(3.\text{Dn-1})</td>
</tr>
<tr>
<td>-Z</td>
<td>0</td>
<td>0.C</td>
<td>1.Cn</td>
<td>2.Cn</td>
<td>3.Cn</td>
</tr>
<tr>
<td>0.R</td>
<td>0.B</td>
<td>0.T</td>
<td>1.Tn</td>
<td>2.Tn</td>
<td>3.Tn</td>
</tr>
<tr>
<td>1.R</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.R</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3.R</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Restricted Master Inverse

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**FIGURE 9.--** Restricted master simplex tableau

The vector of variables 0.R, and the single variables 1.R,
2.R, and 3.R are all artificial variables which must leave the basis or be driven to zero in order to have a feasible solution to the problem. The phase I, phase II process will enter the variables 0.X or L.Yn (L=1, 2, 3) which have negative coefficients in the artificial function row until the solution is feasible, and then will enter more of these to obtain an optimal solution. Every iteration of the simplex procedure will alter the values in the restricted master tableau, not only for the columns currently present, but also for all future columns which will be generated later. The total effect of all iterations is saved in the restricted master inverse, i.e. pre-multiply each new column by this inverse to catch up. The problem now is to discover how to generate these columns.

The symbols in the restricted master simplex tableau in Figure 9 are defined as follows:

1. \(0.WO = -\sum_{I} 0.B(I) -3\)
2. \(0.D(J) = -\sum_{I} 0.T(I,J)\)
3. \(L.Dn = L.D*L.Xn\)
4. \(L.D(J) = -\sum_{I} L.T(I,J)\)
5. \(L.Cn = L.C*L.Xn\)
6. \(L.Tn = L.T*L.Xn\) \((L = 1, 2, 3)\).

Figure 10 shows the generation tableaus used to find the best objective function coefficients for the subproblems. The subproblems will return the values L.Xn for n = 1, 2, ...
to supply artificial and objective function coefficients as small as possible. Let $K$ be the phase number which we are attempting to complete for the restricted master problem. Then multiply the first $2 + 0.1N$ elements of row $K$ of the restricted master inverse times each of the subproblem matrices in Figure 10 to obtain the desired objective function coefficients for the subproblems. When the optimal solution has been found for subproblem $L$, the restricted master inverse times the subproblem generation matrix times the solution gives the new column in the admissible variable matrix of Figure 9. These will replace the last three columns which are no longer needed.

<table>
<thead>
<tr>
<th>Subproblem 1</th>
<th>Subproblem 2</th>
<th>Subproblem 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.D</td>
<td>2.D</td>
<td>3.D</td>
</tr>
<tr>
<td>1.T</td>
<td>2.T</td>
<td>3.T</td>
</tr>
</tbody>
</table>

**FIGURE 10.**—Generation matrices

The simplex procedure does not require that the most negative coefficient column be entered into the basis first. This is suggested as the means to most rapidly approach the optimal solution. In the case of decomposition, however, it takes many more steps to generate a column of the restricted master tableau than to enter a variable into the basis. Any variable with a negative coefficient will
improve the solution; therefore, enter as many as possible before generating new columns.

The Final Solution

When no more variables can enter the basis for phase I (K=1) the solution is examined. If W is greater than zero, the problem had no feasible solution. If W is not positive, phase II (K=2) can begin, continuing with the same procedure as before. The final basis of the restricted master tableau will contain 0.N+3 variables. For each variable 1.Yn, find \[ \sum_{1}^{n} 1.Yn \cdot 1.Xn \] for the final optimal values of 1.X, and find \[ \sum_{1}^{n} 1.Yn \cdot 1.Vn \] for the final shadow prices. It is in this manner that the linear combination of the optimal solutions of subproblem 1 is calculated. Repeat these steps for subproblems 2 and 3. If any of the variables 0.X have entered the basis of the restricted master program, their values are read directly from the tableau. The shadow prices are also read directly from the tableau for the tie-in constraints.
CHAPTER II

COMPUTER SOLUTION

Technical Difficulties

Ordinarily, a program of this nature would have been written in a structural language such as PL/1. Some features which would have been useful are:

1. dynamic allocation of memory for subproblems.
2. data structures for partitioned matrices.
3. based variables for subproblem qualification.

In fact, clever use of based variables and subscript bounds could have eliminated the need for decomposition entirely as a means for saving memory. Unfortunately, most computer installations do not have full PL/1 capabilities, including the one at which this program was written. Based variables with variable dimensioned arrays has not yet been implemented.

This left the problem of allocating a limited amount of memory to large arrays of integer, real, and double precision information. The problem was solved by equivalencing a maximum size vector of each type, and then chopping pieces out as required. A short subroutine named ALLOC8 would get information on the number of locations required and the size of each data type, and then would return a subscript for the beginning location in the correct vector while updating the
information on remaining space available.

Since the piece requested was usually a matrix, every reference to this matrix had to be handled through a subroutine with a vector argument and a matrix parameter. The variable dimensions of the matrices were passed in COMMON. The subroutine named LOC8 handles the search for information about subproblems by retrieving from tables such items as size, vector address, relative location of names in the whole problem, etc.

Other technical difficulties concerned the inability to program decomposition using the "Big M" method. The reason for this is that $M+5-M$ is not equal to 5 after truncation of significant digits. This same problem made it necessary to use double precision variables in the restricted master inverse matrix. Rather than write two separate routines to perform the simplex iterations, the inverses of the sub-problems were also stored in double precision.

Implementation Procedures

The implementation of this program at other computer installations will involve several modifications to the source deck. The IBM 360 uses 4, 4, and 8 bytes of memory for storage of integer, real, and double precision variables, respectively. If a different computer has other values, change the assignment statements for IINT, ISGL, and IDBL in subroutine INITI8 accordingly. Change the value ITOTAL to the total number of bytes (or words, characters, etc.) available for the allocation procedure. Change the size of
vectors IQ, SQ, and DQ accordingly in subroutine LEVEL3. Finally, a work tape or disk is required for the storage of the intermediate results. Change the value of NWORK in subroutine INITI8 to the appropriate unit assignment number.
CHAPTER III

CONCLUSIONS

When a linear program is too large to fit into the memory available, even the fastest computer can not begin to find a solution. Decomposition can then be a powerful tool, provided that a set of subproblems and tie-in constraints can be found which define the original problem. On the other hand, there is no need to waste valuable time on the slower decomposition technique if a straight linear program solver is available which can handle the problem. It would be useful to know when to use one, and when to use the other.

The first consideration is whether or not the program developed by this research can handle larger problems than available straight codes. The increased complexity of the logic and data management routines result in a considerable loss of usable memory. WATFIV compiles need an extra 30K (K=1024) bytes and FORTRAN IV H compiles need an extra 75K bytes over the simplest available code at the University of South Florida data center. Therefore, a savings of at least that much is required before decomposition can begin to use less memory. Since many installations charge more for programs requiring larger blocks of memory, decomposition can
cost more before execution starts. Other factors to be considered when deciding between the two techniques (again, the assumption made here is that both can be used) include the relative difficulties in entering input, the execution times, and the completeness of sensitivity analysis data.

The one really saving grace of decomposition lies in the reduction of core requirements for very large problems. The program shown in Appendix B has been deliberately designed to permit the use of decomposition's greatest space reduction method: subproblems are only considered one at a time and may therefore be overlayed into the same locations of memory. Due to the drastic increase in run times involved in this method, it was not implemented and should be used only as a last resort. Subroutine LOC8 can be re-written to flip-flop the subproblem matrices between alternate external storage devices instead of retrieving the address of the matrices. A simple switch in subroutine ALLOC8 can then determine whether or not the space requested may be reused. This is suggested more as an exercise in programming than in operations research.

A full-scale project, perhaps on the order of a dissertation, would be a formal statement of the complete sensitivity analysis of a decomposed problem. This would include the effect of changes in objective function coefficients, right-hand sides, constraint coefficients, addition of constraints, and addition of variables. As of the date of this report, there is no known source of this data.
APPENDIX A

USER'S GUIDE

The following data sheets are provided for the individual who will prepare input for the program and must interpret the output. No knowledge of decomposition is assumed other than an understanding of how a linear programming problem can be divided into subproblems and tie-in constraints.

Since every installation has its own requirements for job control language (JCL) or its equivalent, this part has been omitted from this appendix and must be obtained from the computing center staff members. Presumably, the JCL will be provided to interested individuals at the same time as the information on the program in appendix B.
Program solves linear programming problems in form:

\[
\begin{align*}
\text{maximize} \quad Z &= C_0 X_0 + C_1 X_1 + C_2 X_2 + \ldots + C_L X_L \\
\text{subject to} \quad T_0 X_0 + T_1 X_1 + T_2 X_2 + \ldots + T_L X_L &= B_0 \\
S_1 X_1 &= B_1 \\
S_2 X_2 &= B_2 \\
\ldots &= \ldots \\
S_L X_L &= B_L
\end{align*}
\]

and \( X_0, X_1, X_2, \ldots, X_L \geq 0 \)

For minimization problems, change the sign of the \( C \)'s and maximize.

**Input Cards:**

Card 1 .......... Title Card (60 alphanumeric characters)
Card 2 .......... L=Number of subproblems (I2)
Card 3 .......... \( M_0, N_0 = \) Number of tie-in constraints, Number of tie-in variables excluding subproblem variables (2I4)
Next L Cards...\( M_1, N_1 = \) Number of subproblem i constraints, Number of subproblem i variables (2I4)
Next Card......MC,NV= Total number of constraints, Total number of variables (2I4)
Next MC Cards...CNAME,RHS= Name of constraint, Right-hand side of constraint (A8,12X, E12.0). Same order as above.
Next NV Cards...VNAME,OFC= Name of variable, Objective function coefficient (10X,A8, 2X,E12.0). Same order as above.
Next Cards......CNAME,VNAME,CC= Constraint name, Variable name, Non-zero constraint coefficient (A8,2X,A8,2X,E12.0)
Last Data Card..SOLVE= The word "SOLVE" (A8)
Repeat above cards for additional problems
Two (2) Blank Cards for end of input signal
Example:

maximize \[ x_0 = 6x_1 + 7x_2 + 3x_3 + 5x_4 + x_5 + x_6 \]

\[ \begin{align*}
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 50 \\
& x_1 + x_2 \leq 10 \\
& x_2 \leq 8 \\
& 5x_3 + x_4 \leq 12 \\
& x_5 + x_6 \geq 5 \\
& x_5 + 5x_6 \leq 50
\end{align*} \]

with all \( x_j \geq 0 \).

After adding slack and surplus variables we get:

Input Cards:

DECOMPOSITION ALGORITHM EXAMPLE - TAHA PAGE 282, PROBLEM 8-21

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S1 50.
S2 10.
S3 8.
S4 12.
S5 5.
S6 50.

S1 0.
X1 6.
X2 7.
S2 0.
S3 0.
X3 3.
X4 5.
S4 0.
X5 1.
X6 1.
S5 0.
S6 0.

S1 X1 1.
S1 X2 1.

S6 S6 1.
SOLVE blank
blank
Note that constraint names are treated separately from the variable names. For convenience, the constraints were given the same names as the slack and surplus variables.

**Interpretation of Output:**

**Page on core requirements—**

Each subproblem requires a fixed amount of core. This is taken from a large pool of available space. For the example shown, the figures given are in bytes.

Additional core is required for the tie-in constraints as well as for storage of names, intermediate solutions, and final solutions. The figures in the column labeled "LEFT" are how much space still remains in the pool after the required space has been determined. Linear programs which are too large will result in negative values under "LEFT". The program will terminate before beginning the optimization if there is insufficient memory.

**Page echoing input—**

Constraint names with right-hand sides, variable names with objective function coefficients, and constraint name-variable name with non-zero constraint coefficients are listed. Column "CABS" gives absolute position of constraint name in list; "CREL" gives relative position. Column "VABS" gives absolute location of variable name in list; "VREL" gives relative location. Column "SUB" gives number of the subproblem to which the quantity under "VALUE" belongs. If this number is zero, the data refers to the tie-in constraint only. Missing data means input error.

**Page with partial solution—**

Included for interest value only. Final solution is derived by performing indicated multiplications and adding all solutions for each subproblem. Subproblems are optimized using different objective functions. If solution is desirable, it will be selected and shown here.

**Page with final solution—**

The optimal value of the objective function is shown. Quantities under "VALUE" next to constraint names are the shadow prices, i.e. a unit increase in the right-hand side of that constraint results in a change in objective function value by amount under "VALUE" as long as solution remains feasible. Quantities under "VALUE" next to variable names are optimal solution values for those variables.

**Pages with dump information—**

Non-zero value under VREL indicates basic variable at time of dump. See input echo for variable name. Column 2 gives current value of variable. Columns 3, 4, 5, ... are simplex tableau (minimization type) columns for first, second, third, ... variable. First row is artificial function and second row is objective function.
<table>
<thead>
<tr>
<th>SUBPROB</th>
<th>CONSTRAINTS</th>
<th>VARIABLES</th>
<th>CORE: REQD</th>
<th>LEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>284</td>
<td>47716</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>180</td>
<td>47536</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>284</td>
<td>47252</td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
<td></td>
<td>936</td>
<td>46316</td>
</tr>
</tbody>
</table>
## DECOMPOSITION ALGORITHM TEST - TAHAN PAGE 282, PROBLEM 8-21

<table>
<thead>
<tr>
<th>CONSTRT VARIABLE</th>
<th>VALUE</th>
<th>CABS</th>
<th>CREL</th>
<th>VABS</th>
<th>VREL</th>
<th>LSUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>0.500000E+02</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>0.100000E+02</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>0.800000E+01</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>0.120000E+02</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>0.500000E+01</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>0.500000E+02</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| X1               | 0.100000E+01 | 1    | 1    | 0    |
| X2               | 0.100000E+01 | 2    | 1    | 1    |
| X3               | 0.100000E+01 | 3    | 2    | 1    |
| X4               | 0.100000E+01 | 4    | 3    |

| X5               | 0.100000E+01 | 5    | 4    |
| X6               | 0.100000E+01 | 6    | 1    |
| S5               | 0.100000E+01 | 7    | 2    |
| S6               | 0.100000E+01 | 8    | 3    |

| X1               | 0.100000E+01 | 9    | 1    |
| X2               | 0.100000E+01 | 10   | 2    |
| X3               | 0.100000E+01 | 11   | 3    |
| X4               | 0.100000E+01 | 12   | 4    |

| S1               | 0.100000E+01 | 1     |    |
| S2               | 0.100000E+01 | 2     |    |
| S3               | 0.100000E+01 | 3     |    |
| S4               | 0.100000E+01 | 4     |    |

| S5               | 0.100000E+01 | 5     |    |
| S6               | 0.100000E+01 | 6     |    |

SOLVE
### Partial Solution

<table>
<thead>
<tr>
<th>Constraint Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0.5111 Times Solution 1 of Subproblem 3:</strong></td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>$-0.000000E\ 00$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$0.100000E\ 01$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>$0.500000E\ 02$</td>
</tr>
<tr>
<td>$X_6$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$0.450000E\ 02$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td><strong>1.0000 Times Solution 1 of Subproblem 1:</strong></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>$0.100000E\ 01$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$-0.000000E\ 00$</td>
</tr>
<tr>
<td>$X_1$</td>
<td>$0.200000E\ 01$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$0.800000E\ 01$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td><strong>0.4889 Times Solution 2 of Subproblem 3:</strong></td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>$-0.199995E\ -01$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$0.301005E\ -08$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td>$X_6$</td>
<td>$0.500000E\ 01$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$0.250000E\ 02$</td>
</tr>
<tr>
<td><strong>1.0000 Times Solution 4 of Subproblem 2:</strong></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>$0.400000E\ 01$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$0.000000E\ 00$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$0.120000E\ 02$</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$0.000000E\ 00$</td>
</tr>
</tbody>
</table>
DECOMPOSITION ALGORITHM TEST - TAHA PAGE 282, PROBLEM 8-21

**FINAL SOLUTION = 0.156000E 03**

<table>
<thead>
<tr>
<th>CONSTRT</th>
<th>VARIABLE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>-0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>0.100000E 01</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>0.400000E 01</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>-0.977753E-02</td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>0.511111E 00</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>0.200000E 01</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>0.800000E 01</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td>0.120000E 02</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>0.000000E 00</td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td>0.255556E 02</td>
<td></td>
</tr>
<tr>
<td>X6</td>
<td>0.244444E 01</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>0.230000E 02</td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>0.122222E 02</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

COMPUTER LISTING

The attached computer listing beginning on the next page was run on an IBM 360 located at the University of South Florida through a HASP link with Florida Technological University. The WATFIV compiler was used to facilitate debugging and to provide a shorter listing. The use of class H and WATFIV would permit about 120,000 bytes of allocatable memory. Using the FORTRAN IV H compiler for most efficient object code produces an object module which can be called without recompiling. For instructional purposes, the amount of allocatable memory defined in the program was limited to 48,000 bytes. The resulting object module will execute in class A.
$JOB
C LEVEL 1 - ENTRIES=MAIN
1 DO 10 I=1,1000
2 CALL INITI8
3 CALL INPUT
4 CALL DCOMP
5 CALL OUTPUT
6 10 CONTINUE
7 CALL EXIT
8 STOP
9 END
10 SUBROUTINE LEVEL2
C LEVEL 2 - ENTRIES=INITI8,INPUT,DCOMP,OUTPUT
11 COMMON NWORK,EPS,BIGM,ISEL,ISTOP,IINT,ISGL,IDBL,ITOTAL,REMARK(15)
C
12 ENTRY INITI8
13 NWORK=4
14 EPS=1.E-6
15 BIGM=1.E50
16 ISEL=0
17 ISTOP=0
18 IINT=4
19 ISGL=4
20 IDBL=8
21 ITOTAL=48000
22 RETURN
23 ENTRY INPUT
24 CALL SIZEIN
25 CALL DIVIDE
26 CALL CONSIN
27 CALL VARSIN
28 CALL TEKSIN
29 RETURN
30 ENTRY DCOMP
31 CALL FIXSUB
32 CALL FIXRMP
33 CALL OPTALL
34 RETURN
35 ENTRY OUTPUT
36 CALL PUTOPT
37 RETURN
38 END
39 SUBROUTINE LEVEL3
C ENTRY SIZEIN,DIVIDE,CONSIN,VARSIN,TEKSIN,
FIXSUB,FIXRMP,OPTALL,PUTOPT
C
40 COMMON NWORK,EPS,BIGM,ISEL,ISTOP,IINT,ISGL,IDBL,ITOTAL,REMARK(15)
41 COMMON/INDEX/I,J,K,L,M,N
42 COMMON/POINT/LC1,LC2,LC3,LC4,LC5,LC6,LV1,LV2
LS1,LS2,LS3,LS4,LR1,LR2,LR3
43 COMMON/RANGE/MLOW<100>,MHIGH<100>,NLOW<100>
1,NHIGH<100>,M0,N0,MLO,MHI,NLO,NHI
44 COMMON/SIZE/MTOT,NTOT,MC,NV,MS,NS,MT,NT
ENTRY SIZEIN

READ PAGE HEADING

READ 1003,REMARK

PRINT 2003,REMARK

READ NUMBER OF SUBPROBLEMS

READ 1001,LS

IF(LS.LE.0) CALL EXIT

PRINT 2001

READ SIZE OF TIE-IN MATRIX

READ 1002,M,N

IF(M.LE.0) M=1

IF(N.LT.0) N=0

MT=M+2

NT=N+1

MR=2+M+LS

NR=1+N+LS

MHOLD(100)=MR

NHOLD(100)=NR

MSUM=M

NSUM=N

MLOW(100)=1

NLOW(100)=1

MHIGH(100)=MSUM

NHIGH(100)=NSUM

READ SIZES OF ALL SUBPROBLEMS

DO 10 L=1,LS

READ 1002,M,N

IF(M.LE.0) M=1

IF(N.LE.0) N=1

MS=M+2

NS=N+1

NOW=MS*IINT+MS*NS*ISGL+MS*MS*IDBL+MT*NS*ISGL

LEFT=LEFT-NOW

PRINT 2002,L,M,N,NOW,LEFT

MHOLD(L)=MS

NHOLD(L)=NS

MSUM=MSUM+M

NSUM=NSUM+N

MLOW(L)=MSUM-(M-1)

NLOW(L)=NSUM-(N-1)

MHIGH(L)=MSUM

NHIGH(L)=NSUM

10 CONTINUE

READ SIZE OF OVERALL PROBLEM

READ 1002,MC,NV

MTOT=MC

NTOT=NV

NOW=MR*IINT+MR*NR*ISGL+MR*MR*IDBL

1 +MC*(IDBL+ISGL+IINT+IINT+ISGL+ISGL)

2 +NV*(IDBL+ISGL+IINT+IINT+ISGL+ISGL)

LEFT=LEFT-NOW

PRINT 2004,NOW,LEFT
31

96  RETURN
97  1001 FORMAT(I2)
98  1002 FORMAT(I2)
99  1003 FORMAT(I2)
100  2001 FORMAT(*8,15X,"CORE REQUIREMENTS AND SIZES/
1  " ,15X,"SUBPROB CONSTRAINTS"
2 ,"VARIABLES CORE: REQD LEFT" ))
101  2002 FORMAT(*15X,I4,10X,I4,10X,I4,1I5,1I8)
102  2003 FORMAT(*15X,15X,15A4)
103  2004 FORMAT(*15X,"OTHER",25X,I15,1I8)
C
104  ENTRY DIVIDE
105  LEFT=ITOTAL
C  ALLOCATE CONSTRAINT DATA - NAME, RHS, SUB,
C  LOC, SEL, SOL
106  CALL ALLOC8(MC,IDBL,LEFT,LC1)
107  CALL ZEROQ(DQ(LC1),MC,1)
108  CALL ALLOC8(MC,ISGL,LEFT,LC2)
109  CALL ZEROQ(SQ(LC2),MC,1)
110  CALL ALLOC8(MC,IINT,LEFT,LC3)
111  CALL ZEROIQ(IO(LC3),MC,1)
112  CALL ALLOC8(MC,IINT,LEFT,LC4)
113  CALL ZEROIQ(IO(LC4),MC,1)
114  CALL ALLOC8(MC,ISGL,LEFT,LC5)
115  CALL ZEROQ(SQ(LC5),MC,1)
116  CALL ALLOC8(MC,ISGL,LEFT,LC6)
117  CALL ZEROQ(SQ(LC6),MC,1)
C  ALLOCATE VARIABLE DATA - NAME, OFC, SUB,
C  LOC, SEL, SOL
118  CALL ALLOC8(NV,IDBL,LEFT,LV1)
119  CALL ZEROQ(DQ(LV1),NV,1)
120  CALL ALLOC8(NV,ISGL,LEFT,LV2)
121  CALL ZEROQ(SQ(LV2),NV,1)
122  CALL ALLOC8(NV,IINT,LEFT,LV3)
123  CALL ZEROIQ(IO(LV3),NV,1)
124  CALL ALLOC8(NV,IINT,LEFT,LV4)
125  CALL ZEROIQ(IO(LV4),NV,1)
126  CALL ALLOC8(NV,ISGL,LEFT,LV5)
127  CALL ZEROQ(SQ(LV5),NV,1)
128  CALL ALLOC8(NV,ISGL,LEFT,LV6)
129  CALL ZEROQ(SQ(LV6),NV,1)
C  ALLOCATE SPACE FOR RESTRICTED MASTER PROGRAM
130  CALL ALLOC8(MR,IINT,LEFT,LR1)
131  CALL ZEROIQ(IO(LR1),MR,1)
132  CALL ALLOC8(MR*NR,ISGL,LEFT,LR2)
133  CALL ZEROQ(SQ(LR2),MR,MR)
134  CALL ALLOC8(MR*MR,IDBL,LEFT,LR3)
135  CALL ZEROQ(DQ(LR3),MR,MR)
C  ALLOCATE SPACE FOR SUBPROBLEMS
136  DO 20 L=1,LS
137  CALL ALLOC8(MS,IINT,LEFT,LS1)
138  CALL ZEROIQ(IO(LS1),MS,1)
139  CALL ALLOC8(MS*NS,ISGL,LEFT,LS2)
140  CALL ZEROQ(SQ(LS2),MS,NS)
141  CALL ALLOC8(MS*MS,IDBL,LEFT,LS3)
142  CALL ZEROQ(DQ(LS3),MS,MS)
143  CALL ALLOC8(MT*NS,ISGL,LEFT,LS4)
144  CALL ZEROQ(SQ(LS4),MT,NS)
145  LHOLD(1,L)=LS1
146  LHOLD(2,L)=LS2
LHOLD(3,L)=LS3
LHOLD(4,L)=LS4
20 CONTINUE
RETURN
ENTRY CONSIN
PRINT 2006,REMARK
CALL FILLC(DQ(LC1),SQ(LC2),IQ(LC3),IQ(LC4))
RETURN
2006 FORMAT('1',15X,'15A4/
  * 10X,'CONSTRT VARIABLE',10X,
  1 'VALUE CABS CREL VABS VREL LSUB'//)
ENTRY VARSIN
CALL FILLV(DQ(LV1),SQ(LV2),IQ(LV3),IQ(LV4))
RETURN
ENTRY TEKSIN
30 READ 1009,CNAME,VNAME,TECH
DATA SOLVE/'SOLVE'/
IF(CNAME.EQ.SOLVE) GO TO 45
CALL MATCH(DQ(LC1),IQ(LC3),IQ(LC4))
*DQ(LV1),IQ(LV3),IQ(LV4),CNAME,VNAME)
IF(L.LT.0) GO TO 40
IF(L.NE.0) CALL LOC8
CALL FILLM(SQ(LS2),SQ(LS4),SQ(LR2),TECH)
PRINT 2009,CNAME,VNAME,TECH,I,M,J,N,K
GO TO 30
40 PRINT 2009,CNAME,VNAME,TECH
GO TO 30
45 PRINT 2009,CNAME
RETURN
1009 FORMAT(A8,2X,A8,2X,E12.0)
2009 FORMAT('1',15X,'15X,A8,2X,A8,2X,15.6,SI5)
ENTRY FIXSUB
DO 70 L=1,LS
CALL LOC8
CALL SUBOUT(SQ(LS2),DQ(LS3),SQ(LS4),SQ(LC2))
CALL PHASE(1,L,MS,NS,IQ(LS1),SQ(LS2),DQ(LS3),W)
IF(W.GT.EPS) GO TO 50
IF(W.EQ.-BIGM) GO TO 60
CALL PHASE(2,L,MS,NS,IQ(LS1),SQ(LS2),DQ(LS3),Z)
IF(Z.EQ.-BIGM) GO TO 60
CALL EVAL(IQ(LS1),SQ(LS2),DQ(LS3),SQ(LS4))
GO TO 70
50 ISTOP=1
PRINT 2011,L
CALL DUMP(MS,NS,IQ(LS1),SQ(LS2),DQ(LS3))
CALL DUMPT(MT,NS,IQ(LR1),SQ(LS4))
GO TO 70
60 ISTOP=1
PRINT 2012,L
CALL DUMP(MS,NS,IQ(LS1),SQ(LS2),DQ(LS3))
CALL DUMPT(MT,NS,IQ(LR1),SQ(LS4))
CONTINUE
IF(ISTOP.EQ.1) CALL EXIT
RETURN
ENTRY FIXRMP

K IS PHASE NUMBER FOR RESTRICTED MASTER

K=1

DO 80 L=1,LS

CALL LOC8

CALL FORCE(SQ(LS4),SQ(LR2),SQ(LC5),SQ(LV5))

80 CONTINUE

CALL RMPOUT(SQ(LR2),DO(LR3),SQ(LC2),SQ(LV2))

CALL PHASE(1,100,MR,NR,IQ(LR1),SQ(LR2)

1,DO(LR3),OUT)

RETURN

ENTRY OPTALL

DO 105 K=1,2

OUTMIN=BIGM

90 LBEST=0

RHOMIN=-EPS

DO 100 L=1,LS

CALL LOC8

CALL CHANGE(IQ(LS1),SQ(LS2),DO(LS3),SQ(LS4)

1,DO(LR3))

CALL PHASE(2*L,MS,NS,IQ(LS1),SQ(LS2),DO(LS3),Z)

IF(Z.EQ.-BIGM) GO TO 110

CALL EVAL(IQ(LS1),SQ(LS2),DO(LS3),SQ(LS4)

1,DO(LR3),SQ(LC5),SQ(LV5),RHO)

CALL SELECT(SQ(LS4),SQ(LR2),DO(LR3)

1,SO(LC5),SQ(LV5))

IF(RHO.GE.RHOMIN) GO TO 100

LBEST=L

RHOMIN=RHO

100 CONTINUE

IF(LBEST.EQ.0) GO TO 103

CALL PHASE(K,100,MR,NR,IQ(LR1),SQ(LR2)

1,DO(LR3),OUT)

IF(OUT.EQ.-BIGM) GO TO 110

IF(OUTMIN-EPS.LT.OUT) GO TO 103

OUTMIN=OUT

GO TO 90

103 IF(K.EQ.1.AND.OUT.GT.EPS) GO TO 120

CONTINUE

RETURN

110 PRINT 2013,L,ISEL

CALL DUMP(MR,NR,IQ(LR1),SQ(LR2),DO(LR3))

CALL DUMP(MS,NS,IQ(LS1),SQ(LS2),DO(LS3))

CALL DUMPT(MT,NS,IQ(LR1),SQ(LS4))

CALL EXIT

RETURN

120 PRINT 2014

CALL DUMP(MR,NR,IQ(LR1),SQ(LR2),DO(LR3))

CALL EXIT

RETURN

2013 FORMAT(1*15X,'UNBOUNDED SOL. L=',I3,' ISEL=',I4,'')

2014 FORMAT(1*15X,'INFEASIBLE RESTRICTED MASTER.')
ENTRY PUTOPT
REWIND NWORK
CALL PUTSEL(IQ(LR1), SQ(LR2), DQ(LC1), SQ(LC5)
1, SQ(LC6), DQ(LV1), SQ(LV5), SQ(LV6)
CALL PUTANS(IQ(LR1), SQ(LR2), DQ(LR3), DQ(LC1)
1, SQ(LC6), DQ(LV1), SQ(LV6))
RETURN
END

SUBROUTINE LEVEL4(C1, C2, C3, C4, C5, C6, V1, V2
1, V3, V4, V5, V6, IS, SB, T, IR, RB, CNAME, VNAME)
ENTRY...
C entry
C force, select, rmputout, change, putsel, putans
C COMMON NWORK, EPS, BIGM, ISEL, ISTOP, IINT, ISGL, IDBL
C ITOTAL, REMARK(15)
C COMMON/INDEX/I, J, K, L, M, N
C COMMON/RANGE/MLOW(100), MHIGH(100), NLOW(100)
C NHIGH(100), M0, N0, ML0, MH1, NLO, NHI
C COMMON/INDEX/MTOT, NTOT, MC, NV, NS, MS, MT, NT
C MR, NR, MWRITE(100), NWWRITE(100), LS
C DOUBLE PRECISION C1(MC), V1(NV), CNAME, VNAME
C REAL C2(MC), C5(MC), C6(MC)
C REAL V2(NV), V5(NV), V6(NV)
C INTEGER C3(MC), C4(MC), V3(NV), V4(NV)
C INTEGER IS(MS), IR(MR)
C REAL S(MS, NS), T(MT, NS), R(MR, NR)
C DOUBLE PRECISION SB(MS, MS), RB(MR, MR), SUM
C
ENTRY FILLC(C1, C2, C3, C4)
L=0
MAX=MHIGH(100)
M=0
DO 20 I=1, MTOT
IF (I .LE. MAX) GO TO 10
L=L+1
MAX=MHIGH(L)
M=0
10 M=M+1
READ 1007, CNAME, RHS
IF (RHS .LT. 0.) RHS=0.
PRINT 2007, CNAME, RHS, I, M, L
C1(I)=CNAME
C2(I)=RHS
C3(I)=L
C4(I)=M
20 CONTINUE
RETURN

ENTRY FILLV(V1, V2, V3, V4)
L=0
MAX=MHIGH(100)
N=0
DO 30 J=1, NTOT
IF (J .LE. MAX) GO TO 30
L=L+1
MAX=MHIGH(L)
N=0
N = N + 1
READ 1008, VNAME, OFC
PRINT 2008, VNAME, OFC, J, N, L
V1(J) = VNAME
V2(J) = OFC
V3(J) = L
V4(J) = N
40 CONTINUE
RETURN
1008 FORMAT (10X, A8, 2X, E12.0)
2008 FORMAT (•15X, 1QX, A8, 2X, E15.6, 1OX, 3I5)
ENTRY MATCH(C1, C3, C4, V1, V3, V4, CNAME, VNAME)
DO 50 I = 1, MTOT
IF (CNAME.EQ. C1(I)) GO TO 60
50 CONTINUE
GO TO 90
60 DO 70 J = 1, NTOT
IF (VNAME.EQ. V1(J)) GO TO 80
70 CONTINUE
GO TO 90
80 K = C3(I)
L = V3(J)
IF (K.NE.0.AND.K.NE.L) GO TO 90
M = C4(I)
N = V4(J)
RETURN
90 L = -1
RETURN
ENTRY FILLM(S, T, R, TECH)
IF (K.EQ.0) GO TO 100
S(M+2, N+1) = TECH
RETURN
100 IF (L.EQ.0) GO TO 110
T(M+2, N+1) = TECH
RETURN
110 R(M+2, N+1) = TECH
RETURN
ENTRY SUBOUT(S, SB, T, C2, V2)
C INITIALIZE RIGHT HAND SIDES
DO 120 I = 3, MS
120 S(I, 1) = C2(I-2) + M0
C INITIALIZE OBJECTIVE FUNCTION COEFFICIENTS
DO 140 J = 2, NS
140 T(2, J) = -V2(J-1) + M0
C INITIALIZE IDENTITY MATRIX DIAGONAL
DO 150 KK = 1, MS
150 SB(KK, KK) = 1.0
C INITIALIZE ARTIFICIAL FUNCTION COEFFICIENTS
DO 170 J = 1, NS
170 SUM = 0.
DO 160 I = 3, MS
160 SUM = SUM + S(I, J)
170 DO 190 J = 2, NS
190 SUM = 0.
180 SUM = SUM + T(I, J)
367 190 $T(1,J)=-\text{SUM}$

C USE T ART FOR FIRST S OBJECTIVE FUNCTION
348 DO 195 J=2,NS
349 195 S(2,J)=$T(1,J)$
350 RETURN
C
351 ENTRY EVAL(IS,S,SB,T,RB,C5,V5,RHO)
C UNSCRAMBLE X SOLUTION
352 DO 200 J=NL0,NHI
353 200 V5(J)=0.
354 DO 210 I=3,MS
355 J=IS(I)
356 IF(J.NE.0) V5((J-1)*N0)=S(I,1)
357 210 CONTINUE
358 DO 215 I=3,MS
359 215 C5((I-2)+MO)=SB(2,I)
C COMPUTE T*X AND INSERT IN FIRST COL OF T
360 DO 230 I=1,MT
361 230 SUM=0.
362 DO 240 J=2,NS
363 240 SUM=SUM+$T(J,J)*V5(J-1)+NO$)
364 230 $T(1,1)=-\text{SUM}$
365 SUM=0.
366 DO 235 I=3,MT
367 235 SUM=SUM+$T(I,1)$
368 $T(1,1)=-\text{SUM}$
C EVALUATE ZJ-CJ FOR SUBPROBLEM
369 SUM=0.
370 SUM=RB(K,1)*($T(1,1)-1.$)
371 DO 240 KK=2,MT
372 240 SUM=SUM+RB(K,KK)*$T(KK,1)$
373 RHO=SUM+RB(K,MT+L)
374 RETURN
C
375 ENTRY FORCE(T,R,C5,V5)
376 R(1,NT+L)=$T(1,1)-1.$
377 DO 250 I=2,MT
378 250 R(I,NT+L)=$T(I,1)$
379 R(MT+L,NT+L)=1.
380 IF(L.LT.LS) RETURN
381 ISEL=ISEL+1
382 WRITE(NWORK) C5,V5
383 RETURN
C
384 ENTRY SELECT(T,R,RB,C5,V5)
C SELECT SUBPROBLEM TO ENTER RESTRICTED MASTER
385 DO 370 I=1,MR
386 SUM=0.
387 SUM=RB(I,1)*($T(1,1)-1.$)
388 DO 360 KK=2,MT
389 360 SUM=SUM+RB(I,KK)*$T(KK,1)$
390 370 R(I,NT+L)=SUM+RB(I,MT+L)
391 IF(L.LT.LS) RETURN
392 ISEL=ISEL+1
393 WRITE(NWORK) C5,V5
394 RETURN
C
395 ENTRY RMPOUT(R,RB,C2,V2)
C INITIALIZE RIGHT HAND SIDES
396 MO=MLOW(100)-1
DO 410 I=3,MT
410 R(I,1)=C2(((I-2)+M0)
MTP1=MT+1
DO 420 I=MTP1,MR
420 R(I,1)=1.

C INITIALIZE OBJECTIVE FUNCTION COEFFICIENTS
IF(NT.EQ.1) GO TO 435
N0=NLOW(100)-1
DO 430 J=2,NT
430 R(2,J)=-V2((J-1)+N0)
435 CONTINUE

C INITIALIZE IDENTITY MATRIX DIAGONAL
DO 440 K=1,MR
440 RB(K,K)=1.DO

C INITIALIZE ARTIFICIAL FUNCTION COEFFICIENTS
DO 480 J=1,NR
SUM=0.
480 R(1,J)=SUM
RETURN

C ENTRY CHANGE(IS,S,SB,T,RB)
DO 730 J=2,NS
730 SUM=0.
DO 710 KK=1,MT
710 SUM=SUM+RB(K,KK)*T(KK,J)
S(2,J)=SUM

C SET ARTIFICIAL OFC'S TO ZERO
DO 740 J=3,MS
740 SB(2,J)=0.

C RESET BASIC OFC'S TO ZERO
S(2,1)=0.
DO 770 I=3,MS
770 CONTINUE
RETURN

C ENTRY PUTSEL(IR,R,C1,C5,C6,V1,V5,V6)
PRINT 2014,REMARK
DO 850 IREAD=1,ISEL
850 READ(NWORK) C5,V5
LDIFF=(IREAD-1)*LS
DO 850 IROW=3,MR
850 NEGSEL=IR(IROW)
IF(NEGSEL.GE.0) GO TO 850
OFC=S(2,J)
IF(ABS(OFC).LT.EPS) GO TO 850
DO 750 J=1,NS
750 S(2,J)=S(2,J)-OFC*S(I,J)
DO 760 J=3,MS
760 SB(2,J)=SB(2,J)-OFC*SB(I,J)
770 CONTINUE
RETURN

C
DO 840 J=NLO,NHI
V6(J)=V6(J)+XLAMB*V5(J)
PRINT 2015,XLAMB,IREAD,L
PRINT 2018,(C1(I),C5(I),I=MLO,MHI)
PRINT 2017,(V1(J),V5(J),J=NLO,NHI)
850 CONTINUE
RETURN
C
ENTRY PUTANS(IR,R,RP,C1,C6,V1,V6)
PRINT 2019,REMARK,R(2,1)
M0=MLOW(100)-1
DO 855 I=3,MT
C6((I-2)+M0)=RP(I,2)
N0=NLOW(100)-1
DO 860 I=3,MR
J=IR(I)
IF(J.LE.0) GO TO 860
V6((J-1)+N0)=R(I,1)
860 CONTINUE
PRINT 2018,(CL,(C6,I),I=1,M)
PRINT 2017,(V1,J),V6(J),J=1,N)
RETURN
2014 FORMAT('1',15X,15A4//16X,'PARTIAL SOLUTION'/1-testing variable value')
2015 FORMAT('0',15X,F6.4!* TIMES SOLUTION',1X,14'
1-testing variable value')
2017 FORMAT('1',15X,10X,A8,E17.6)
2018 FORMAT('1',15X,8,10X,E17.6)
2019 FORMAT('1',15X,15A4/
1-'FINAL SOLUTION =',E19.6//1-testing variable value')
END
SUBROUTINE LEVELSCIQ,SQ,OQ, MROWS,NCOLS)
ENTRY ALLOC8,LOC8,ZER0IQ,ZEROSQ,ZERODQ
COMMON/INDEX/I,J,K,L,M,N
COMMON/POINT/LC1,LC2,LC3,LC4,LC5,LC6,LV1,LV2
1,LV3,LV4,LVS,LV6,LS1,LS2,LS3,LS4,LR1,LR2,LR3
2,LR4,LHOLD(4,100)
COMMON/RANGE/MLow(100),MHIGH(100),NLOW(100)
1,NHIGH(100),M0,N0,MLO,MHI,NLO,NHI
COMMON/Size/MTOT,NTOT,MC,NS,MS,NT,MT
1,M0,M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15,M16,M17,M18,M19,M20,M21,M22,M23,M24,M25,M26,M27,M28,M29,M30,M31,M32,M33,M34,M35,M36,M37,M38,M39,M40,M41,M42,M43,M44,M45,M46,M47,M48,M49,M50,M51,M52,M53,M54,M55,M56,M57,M58,M59,M60,M61,M62,M63,M64,M65,M66,M67,M68,M69,M70,M71,M72,M73,M74,M75,M76,M77,M78,M79,M80,M81,M82,M83,M84,M85,M86,M87,M88,M89,M90,M91,M92,M93,M94,M95,M96,M97,M98,M99,M100,LS
INTEGER IQ(MROWS,NCOLS)
REAL SQ*4(MROWS,NCOLS)
REAL DQ*8(MROWS,NCOLS)
C
ENTRY ALLOC8(NVALS,NSIZE,NMAX,LOC)
NEED=NVALS*NSIZE
LOC=(NMAX-NEED)/NSIZE+1
NMAX=(LOC-1)*NSIZE
IF(NMAX.GE.0) RETURN
PRINT 2000
CALL EXIT
RETURN
2000 FORMAT('0',15X,'INSUFFICIENT MEMORY.'
1,' JOB ABANDONED.')
C
ENTRY LOC8
LS1=LHOLD(1,L)
ENTRY ZEROIQ(IQ,MROWS,NCOLS)
DO 10 J=1,NCOLS
DO 10 I=1,MROWS
10 IQ(I,J)=0
RETURN

ENTRY ZEROSQ(SQ,MROWS,NCOLS)
DO 20 J=1,NCOLS
DO 20 I=1,MROWS
20 SQ(I,J)=0.0
RETURN

ENTRY ZERODQ(DQ,MROWS,NCOLS)
DO 30 J=1,NCOLS
DO 30 I=1,MROWS
30 DQ(I,J)=0.0
RETURN

SUBROUTINE LEVEL6(M,N,IA,A,AB)
LEVEL 6 - ENTRIES=PHASE,DUMP
COMMON NWORK,EPS,BIGM,ISEL,ISTOP,IINT,ISGL,IDBL
1,ITOTAL,REMARK(15)
COMMON/SIZE/MTOT,NTOT,MC,NV,MS,NS,MT,NT
L,M,N,MHOLD(100),NHOLD(100),LS
INTEGER IA(M)
REAL A(M,N)
DOUBLE PRECISION AB(M,M),DABS,BEST,WORST
DOUBLE PRECISION AKJ,AIC,ARC,ARJ,BI,RATIO

ENTRY PHASE(K,L,M,N,IA,A,AB,OUT)
FIND PIVOT COLUMN
1 JCOL=0
BEST=-EPS
DO 10 J=2,N
AKJ=A(K,J)
IF(AKJ.GE.BEST) GO TO 10
JCOL=J
BEST=AKJ
10 CONTINUE
IF(JCOL.EQ.0) GO TO 800
FIND PIVOT ROW
IF(K.EQ.1) GO TO 30
ATTEMPT TO PIVOT OUT ARTIFICIAL
DO 20 I=3,M
IF(IA(I).NE.0) GO TO 20
AIC=A(I,JCOL)
IF((DABS(AIC).LT.EPS) GO TO 20

IROW=I

GO TO 50

20 CONTINUE

IROW=0

WORST=BIGM

DO 40 I=3,M

AIC=A(I,JCOL)

IF(AIC.LT.EPS) GO TO 40

BI=A(I,1)

RATIO=BI/AIC

IF(RATIO.GE.WORST) GO TO 40

-- IROW=I

WORST=RATIO

40 CONTINUE

IF(IROW.EQ.0) GO TO 900

IA(IROW)=JCOL

C USE NEGATIVE LAMBDA SELECTION IF APPLICABLE

NEGSEL=-((ISEL-1)*LS-(JCOL-NT)

C TRANSFORM TABLEAU

ARC=A(IROW,JCOL)

DO 80 I=K,M

IF(I.EQ.IROW) GO TO 80

AIC=A(I,JCOL)

IF(DABS(AIC).LT.EPS) GO TO 75

DO 60 J=1,N

ARJ=A(IROW,J)

AIJ=A(I,J)

A(I,J)=AIJ-AIC*ARJ/ARC

60 CONTINUE

DO 70 J=1,M

ARJ=AB(IROW,J)

AIJ=AB(I,J)

AB(I,J)=AIJ-AIC*ARJ/ARC

70 CONTINUE

75 A(I,JCOL)=0.

80 CONTINUE

DO 90 J=1,N

A(IROW,J)=A(IROW,J)/ARC

90 CONTINUE

DO 100 J=1,M

A(IROW,J)=A(IROW,J)/ARC

100 CONTINUE

105 A(IROW,JCOL)=1.

GO TO 1

OUT=-A(K,1)

RETURN

C ENTRY DUMP(M,N,IA,A,AB)

PRINT 2098

DO 1000 I=1,M

IHOLD=IA(I)

IF(IHOLD.GT.0) IHOLD=IHOLD-1

IF(I.LT.3) IHOLD=999999

1000 PRINT 2099,IHOLD,(A(I,J),J=1,N)

PRINT 2096

DO 1050 I=1,M
IHOLD=IA(I)
IF (IHOLD .GT. 0) IHOLD=IHOLD-1
IF (I .LT. 3) IHOLD=999999
1050 PRINT 2099, IHOLD, (AB(I,J), J=1,M)
RETURN
ENTRY DUMPT(M,N,IA,A)
PRINT 2097
DO 1100 I=1,M
1100 PRINT 2099, IA(I), (A(I,J), J=1,N)
RETURN
2096 FORMAT( '0', 15X, ' VREL, M X M INVERSE')
2097 FORMAT( '0', 15X, ' VREL, (T(I,J), J=1,N)')
2098 FORMAT( '0', 15X, ' VREL, (A(I,J), J=1,N)')
2099 FORMAT( '0', 15X, ' I5', (1X, F10.3)/
          1 ( ' ', 15X, ' I5', (1X, F10.3))
END
$ENTRY
BIBLIOGRAPHY


