A TSO PRESENTATION OF A DECOMPOSITION TECHNIQUE FOR SOLVING LARGE-SCALE MULTIDIVISIONAL LINEAR PROGRAMMING PROBLEMS

By

WILLIAM ARTHUR SENTERS

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1975

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
December, 1978

A TSO PRESENTATION OF A DECOMPOSITION TECHNIQUE

FOR SOLVING LARGE-SCALE MULTIDIVISIONAL

LINEAR PROGRAMMING PROBLEMS

Thesis Approved:

Donald W. Grace
Thesis Adviser

G. E. Hedrick

J. Scott Turner

Norman N. Durham
Dean of Graduate College

PREFACE

This study was concerned with the development of an interactive program designed to aid the student in learning the decomposition technique. The primary objective is to give the student an opportunity to learn the concepts of decomposition at his own rate and at the time he chooses. The program allows the student to visualize how a computer algorithm goes about solving such a problem.

I would like to express my appreciation to my advisors, Dr. George Hedrick and Dr. Donald Grace for their assistance and encouragement through the years, and to Dr. Billy Thornton for giving me a solid start in the field of operations research. Appreciation is also expressed to Dr. Scott Turner for giving his time and encouragement in being a committee member. I express special gratitude to my wife, Deborah, for her love and understanding and for being nearby when needed.

TABLE OF CONTENTS

NOMENCLATURE

A_k	matrix of division k coefficients for corporate constraints
B_k	matrix of coefficients for divisional constraints
$B^{-1}_{1;m}$	matrix consisting of the first m columns of B^{-1}
B^{-1}_{m+j}	the $(m+j)^{th}$ column of B^{-1}
$\overline{b}_k$	vector of right hand side of division k constraints
$\overline{C}_k$	vector of relative cost factors of division k
C_B	vector of the objective coefficients associated with the basic variables
m	the number of corporate or linking constraints
λ^j_k	weights on the j^{th} extreme point of division k
π	simplex multipliers
S_k	solution space for the k^{th} division
X^j_k	j^{th} extreme point of the k^{th} division
$\overline{x}_k$	set of variables in division k

CHAPTER I

INTRODUCTION

In recent years the business world has turned to mathematical programming for a scientific approach to decision making. This is the process of representing a particular real life competitive situation in terms of an operations research mathematical model. The model usually consists of an objective function of variables which are subject to a number of functional constraints each representing a limitation of the organization. These limitations are usually of the form of limits on production, demands, manpower, machine hours, natural resources, and also social responsibilities such as standards on pollution and safety. It is common practice, and will be followed in this report, to refer to all constraints as constraints on limited resources. When these constraints and the objective function can be represented in linear statements, the process is simplified into linear programming. A simple technique for linear programming is the Simplex Method. For a small independent business, mathematical programming can be a simple task of incorporating the Simplex Method without resorting to special techniques.

However, in today's world if an organization wants to operate at an optimum level and expand, it cannot perceive itself as being independent from its environment. In other words, it must realize its organizational and social dependencies. In order for an organization to operate as a finely tuned machine it must operate at a level where the limits on its resources are approached but not reached. As a result strict new constraints are introduced into the mathematical model of the problem. As one can imagine, the model, accurately stated, could grow in the number of constraints to such a size as to create another problem in itself - this problem being that the great number of constraints makes for inefficient use of computer time and space.

At this point a company has two alternatives. It could reduce the number of constraints, hence reducing the accuracy of the model, or it could incorporate one of the many techniques that have been developed to alleviate this problem and still keep an accurate model of the situation. Techniques such as generalized upper bounding, revised simplex, and decomposition provide an effective way of solving a large problem with a special structure with reasonable expenditure of computer time and space.

When some or all of the variables can be divided into groups such that the sum of the variables in each group must not exceed a specified upper bound, a generalized upper bound technique can be invoked.

A technique developed specifically for use with digital computers is the Revised Simplex Method, whereby many of the data can be stored on external devices, making it possible to solve large problems on small computers.

The scope of this report will center on decomposition which is a technique for solving multidivisional types of problems. Many texts and reports have been written on this algorithm, but not enough programs have been written for the use of students to receive hands on experience. This report is aimed at developing an interactive program designed to allow the student to study the decomposition principles at his own level of detail. The student can cover the material quickly and briefly or request that a detailed explanation be given for a specific area. He may also review certain areas of trouble or return to take the session over as often as he wishes. The report develops a generalized decomposition program that can be used as a tutorial supplement to a theoretical presentation and give the advanced student a feel of how the algorithm can be used and interpreted. Hopefully, it will result in a better understanding and a more efficient use of the principle of decomposition and linear programming as a whole.

CHAPTER II

THE REVISED SIMPLEX METHOD

There are several factors affecting how long the general Simplex Method will require to solve a linear programming problem. Two of the most important factors are the number of constraints and the number of variables in a problem. If n is the number of variables and m the number of constraints, then the maximum number of iterations possible will be the value of $(m + n)!/(m!n!)$.

The general model for linear programming in matrix form is:

Minimize $\quad Z = \overline{C} \cdot \overline{X}$

subject to: $\quad A \cdot \overline{X} = \overline{b}$

and $\quad x_j >= 0$

where $\overline{C}$ is the row vector of the relative cost factors

$$\overline{C} = | C_1, C_2, \ldots, C_n |,$$

$\overline{X}$ and $\overline{b}$ are all column vectors such that

$$\overline{X} = \begin{vmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{vmatrix}, \quad \overline{b} = \begin{vmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{vmatrix}$$

and A is the coefficient matrix

$$A = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{vmatrix}$$

Recall that the general Simplex Method began with the entire mxn matrix A, m being the number of constraints and n being the number of original variables plus slack variables. At each iteration the entire mxn matrix was updated and stored. Of the n variables only m basic variables were in the solution.

The Revised Simplex Method was designed to compute only the information that is currently needed at each iteration and store it in a more compact form by comprising an mxm basis matrix of the columns corresponding to these m basic variables.

Let P_j be the j^{th} column of the coefficient matrix A.

$$P_j = \begin{vmatrix} a_{1j} \\ a_{2j} \\ \cdot \\ \cdot \\ \cdot \\ a_{mj} \end{vmatrix}$$

The model can now be restated as

$$\text{Minimize} \quad Z = \sum_{j=1}^{n} C_j \cdot x_j$$

$$\text{subject to:} \quad \sum_{j=1}^{n} P_j \cdot x_j = \overline{b}$$

$$\text{and} \quad x_j >= 0$$

With the Revised Simplex Method the updating operation does not need to be performed on the entire A matrix, which is mxn with n the number of original variables plus slack variables. Instead, let B be an mxm matrix comprised of the basic columns P_j.

$$\text{i.e.} \quad B = | P_1, P_2, \ldots, P_m |$$

Only B^{-1} need be updated by the pivot operations. The Revised Simplex Method therefore solves for a set of m equations in m unknowns (basic variables). This set of equations can be denoted by $B \cdot X_B = \overline{b}$ where X_B is the vector of basic variables so that the basic solution is $X_B = B^{-1} \cdot \overline{b}$.

Given the basic matrix B, the linear combination that expresses any other vector P_j is determined by computing the vector $P_j' = B^{-1} \cdot P_j$ which becomes the j^{th} column of the current iteration. The value of the objective function for a basic solution can now be written as $Z = C_B \cdot P_j' = C_B \cdot B^{-1} \cdot P_j$. Letting $\overline{0}$ be the vector of objective coefficients for the slack variables, C_B is the subset of the vector $| \overline{C}, \overline{0} |$ containing the values of the objective coefficients associated with the basic variables.

To avoid computing P_j' for all Z_j's a vector of pricing or simplex multipliers is derived by $\pi = C_B \cdot B^{-1}$. A vector P_j not in the basis is "priced out" by computing $Z_j = \pi \cdot P_j = C_B \cdot B^{-1} \cdot P_j$. Thus, the P_j can be stored on external devices and brought into core memory only as needed.

It should be remembered that the vectors $\overline{C}$ and P_j were recorded in the original data. The C_B vector needed to compute π is a row vector formed from $\overline{C}$. All that is needed to form C_B correctly is to keep track of which variables are in the current basis.

At each iteration the only relevant pieces of information are: 1) C', the vector of cost factors $C_j - Z_j$ or $C_j - \pi P_j$ relative to the current iteration. 2) the elements of the updated column P'_j where $P'_j = B^{-1} \cdot P_j$. 3) and the values of the basic variables X_B where $X_B = B^{-1} \cdot \bar{b}$.

Using the above information and formulas, let us derive a summary of the Revised Simplex Method.

Step 0 - Given:

A - coefficient matrix

$\bar{b}$ - right hand side

$\bar{C}$ - coefficients of objective function

Initialize matrix B as the columns associated with the initial basic variables (usually slack variables requiring B to be initialized as an identity matrix). Form C_B and B^{-1} as stated above, compute π as $C_B \cdot B^{-1}$, compute $C' = C_j - Z_j = C_j - \pi P_j$.

Step 1 - Determine the entering basic variable. Find C_s = min element of C' where s is the index for the entering variable.

Step 2 - Optimization test

If $C_s >= 0$ stop.

If $C_s < 0$ compute the updated column $P'_s = B^{-1} \cdot P_s$ and the new simplex multipliers $\pi = C_B \cdot B^{-1}$ and the new cost factors $C' = \bar{C} - \pi P_s$.

Step 3 - Determine the leaving basic variable.

$$\text{If } P'_s = \begin{vmatrix} a'_{1s} \\ a'_{2s} \\ \cdot \\ \cdot \\ \cdot \\ a_{ms} \end{vmatrix} \quad \text{find r as the}$$

index for the variable being removed from the basis by finding min b_i/a'_{is} for $a'_{is} > 0$.

Step 4 - Update the basic solution. Derive new B^{-1} and set $X_B = B^{-1} \cdot \overline{b}$.

Return to Step 1.

In Step 4, B^{-1} could be derived each time by using a standard computer routine for inverting a matrix. However, since B and B^{-1} change by only one vector from one iteration to the next, it is much more efficient to derive the new B^{-1} (denote it by B^{-1}_{new}) from the B^{-1} at the preceding iteration (denote it by B^{-1}_{old}). To do this, let x_k be the entering basic variable, a'_{ik} be the coefficient of x_k (these coefficients are determined in Step 2), and r be the index of the column in the preceding basis that is being replaced. The new B^{-1} can now be expressed in matrix notation as $B^{-1}_{new} = E \cdot B^{-1}_{old}$ where the matrix E is an elementary matrix, i.e., an identity matrix except that its r^{th} column is replaced by the vector

$$\eta = \begin{vmatrix} \eta_1 \\ \eta_2 \\ \cdot \\ \cdot \\ \cdot \\ \eta_m \end{vmatrix} \qquad \text{where } \eta_i = \begin{cases} \dfrac{-a'_{ik}}{a'_{rk}}, & \text{if } i \neq r \\[2ex] \dfrac{1}{a'_{rk}}, & \text{if } i = r \end{cases}$$

E can be written as

$$E = \begin{vmatrix} 1 & 0 & \cdots & -a'_{1k}/a'_{rk} & \cdots & 0 \\ 0 & 1 & \cdots & -a'_{2k}/a'_{rk} & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & 1/a'_{rk} & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & -a'_{mk}/a'_{rk} & \cdots & 1 \end{vmatrix}$$

Of course, in the actual coding of this method the entire E matrix does not need to be physically built. Only the η vector need be computed, which could save considerable storage.

If ℓ is allowed to represent the ℓ^{th} iteration then in general the inverse of the ℓ^{th} basis can be obtained from

$$B_{\ell}^{-1} = E_{\ell} \cdot E_{\ell-1} \cdot \cdot \cdot E_1 \cdot E_0 \cdot B_0^{-1}$$

Until now it was assumed that the matrix B contained no artificial variables or negative slack variables and was therefore equal to the identity matrix at the beginning of the procedure.

If "=" and/or ">=" constraints are included in the model, artificial and negative slack variables must be added as in the regular Simplex Method. The procedure has to begin with a basis consisting of an identity matrix that corresponds to either real or artificial vectors. A two-phase approach can then be used. If the procedure starts with artificial vectors, a basic feasible solution must be determined by Phase I, of which the computation is not included in this report. Phase I can be interpreted as minimizing the sum of the artificial variables over the feasible region. If a feasible solution is attainable, the artificial variables can be driven to zero.

Once an initial basic feasible solution is found Phase II solves for optimality by the Revised Simplex Method. If the constraints are all "<=" Phase I may be bypassed.

CHAPTER III

THE DECOMPOSITION ALGORITHM

Angular Structure

There has been a tremendous increase in the division of labor and segmentation of management responsibilities in organizations recently. There is also a tendency for the different divisions of an organization to become independent of the organization as a whole with their own goals and restrictions. This lends itself to a special class of problems called multidivisional, to which most large problems belong. Their special feature is that

> the problem is almost decomposable into separate problems, where each division is concerned only with optimizing its own operation. However, some overall coordination is required in order to best divide certain organizational resources among the divisions (4, p.142).

Decomposition ideas and methods are as old as linear programming (6). But the first workable decomposition algorithm was introduced by Dantzig and Wolfe in 1959 (3). The basic algorithm that this report will refer to is quite simple, at least for those familiar with the mathematics of linear programming and the Revised Simplex Method.

The Decomposition Method can be thought of as having each division solve its own subproblem and send its proposed solution to a central coordinator who can coordinate the proposals from all the divisions, impose the corporate viewpoint, and find the optimal solution for the overall organization. This is accomplished, not by explicitly imposing the corporate constraints on the divisions, but by "economic pressure" in the form of adjustments to the divisions' profit or cost coefficients to reflect their use of corporate resources. Therefore, we can reformulate the model in an angular structure as follows:

$$\text{Minimize} \quad \sum_{k=1}^{n} \bar{c}_k \cdot \bar{x}_k$$

$$\text{subject to:} \quad \begin{vmatrix} A_1 & A_2 & \cdots & A_n \\ B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_n \end{vmatrix} \cdot \begin{vmatrix} \bar{x}_0 \\ \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_n \end{vmatrix} <= \begin{vmatrix} \bar{b}_0 \\ \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_n \end{vmatrix}$$

where the A_j, $j = 1,2, \ldots, n$ are matrices that represent the corporate (linking) constraints. These constraints link the divisions by making them share the organizational resources available. And the B_j, $j = 1,2, \ldots, n$ represent the divisional constraints of each division. (0 are null matrices).

Formulation of the Model

At this point let us assume that the set of feasible solutions for each division is bounded. The solution space for each division is bounded by the constraint equations on the divisional resources. These equations define a "flat" geometrical shape (called a hyperplane) in n-dimensional space analogous to the line in two-dimensional space and the plane in three-dimensional space. The simultaneous solution of two constraint equations defines an extreme point. And since we are restricted to linear models the set of points X_k such that $X_k >= 0$ and $B_k \cdot X_k = \overline{b}_k$ constitute a convex set with a finite number of extreme points. Therefore, under the assumption that the set is bounded, any point in the set can be represented as a convex combination of the extreme points of the set.

Consider the solution space for k^{th} division; call it S_k. i.e. $S_k = \{X_k \mid B_k \cdot X_k <= \overline{b}_k \text{ and } X_k >= 0\}$. Any point in S_k can be represented as a (convex combination) weighted average of the extreme points of S_k. (Let $X_k^j = j^{th}$ e.p. of division k). Then

$X_k^* = \sum_j \lambda_k^j \cdot X_k^j$ is any feasible point of the k^{th} division, where

$\lambda_k^j = 0$ and $\sum_j \lambda_k^j = 1$. Therefore, this equation for X_k^* and the so-called "normalizing" or "convexity" constraints on the λ_k^j provide a way of representing the feasible solutions to division k without using any of the original constraints. Hence, the overall problem can now be reformulated with far fewer constraints as

$$\text{Maximize } Z = \sum_{k=1}^{n} \sum_{j=1}^{n} (\overline{c}_k \cdot X_k^j) \; \lambda_k^j$$

$$\text{subject to: } \sum_{k=1}^{n} \sum_{j=1}^{n} (A_k \cdot X_k^j) \; \lambda_k^j = \overline{b}_0$$

$$\sum_{j=1}^{n} \lambda_k^j = 1$$

$$\text{and } \lambda_k^j >= 0, \; k = 1, 2, \ldots, n$$

This formulation is completely equivalent to the one given earlier. However, since it has fewer constraints, it should be solvable with much less computational effort. It also has as many columns as the solution space S has extreme points, which may be thousands. This fact does not matter much if the Revised Simplex Method is used, as the columns to enter the basis are generated only as they are needed.

Development of Algorithm

Recall that with the Revised Simplex Method the vector of simplex multipliers ($\pi = C_B \cdot B^{-1}$) is used in computing the relative cost coefficients. During decomposition π needs to be partitioned as $(\overline{\pi}_1, \pi_0)$ with $\overline{\pi}_1$ associated with the reformulated division constraints and π_0 associated with the convexity constraints. Let m denote the number of corporate (linking) constraints. Let $(B^{-1})_{1;m}$ be the matrix consisting of the first m columns of B^{-1}, and let $(B^{-1})_j$ be the vector consisting of the j^{th} column of B^{-1}. Then $\overline{\pi}_1 = C_B \cdot (B^{-1})_{1;m}$ a vector and $\pi_0 = C_B \cdot (B^{-1})_{m+j}$ a scalar.

As in the regular Simplex Method, it must be determined whether or not the current feasible solution can be improved by pricing out vector P_j, a vector of A. Vector P_j is priced out as in the Revised Simplex Method by $\overline{\pi}_1 \cdot P_j - \overline{c}_j$.

The usual simplex criterion asks that we find

$$\min f_j = (\overline{\pi}_1 \cdot A_j - \overline{c}_j)\, \overline{x}_j + \pi_0$$

It should be noticed that the above equation is independent of the scalar π_0.

Therefore, the first step at each iteration requires solving n (number of divisions) linear programming problems of the type that follows.

Minimize $(\bar{\pi}_1 \cdot A_j - \bar{c}_j)\ \bar{x}_j$

subject to: $B_j \cdot \bar{x}_j = \bar{b}_j$

and $\bar{x}_j >= 0$

<u>Step 1</u> - Using the simplex multipliers $\bar{\pi}_1$ solve the division sub-problems as above obtaining solutions and optimal objective values Z_i.

<u>Step 2</u> - Compute the min $Z_i + \pi_0 = f_j$

<u>Step 3</u> - Stopping rule

If $f_j >= 0$ the optimal solution can now be calculated. By letting $X_k^j = j^{th}$ extreme point of division k and λ_k^j the weights on these extreme points the optimal solution can be calculated as $\sum_j (\lambda_k^j \cdot X_k^j)$ for every division k where the X_k^j's are the extreme points of the solution space corresponding to the λ_k^j in the basis of the corporate problem. This calculation results in a vector for each division, each vector consisting of the number of elements as there are variables for that division.

Stop.

<u>Step 4</u> - If $f_j < 0$ form the column to enter the basis as

$$P'_j = \left| \frac{A_j \cdot \bar{x}_j}{II} \right|$$

where II is an n component vector with a one in position j and zeroes elsewhere and A_j is the matrix of coefficients of the corporate constraints for division j.

Step 5 - For the Revised Simplex Method to determine the leaving basic variable it is necessary to calculate the current coefficients and right hand side as $B^{-1} \cdot P_j'$ and $B^{-1} \cdot b'$.

$$b' \text{ being the vector of } \left| \begin{array}{c} \bar{b}_0 \\ \hline 1 \end{array} \right| \text{ where 1 is an n component}$$

vector of all 1's.

Step 6 - Obtain a new basis inverse.

Obtain new simplex multipliers.

Go back to Step 1 and repeat.

CHAPTER IV

DISPLAY DEVICES

Time-Sharing Option Terminals

Any visual display device that can be used as a time-sharing option (TSO) terminal can be used to execute this decomposition presentation. Most TSO terminals differ only in the way the data is entered and displayed. Therefore, a basic understanding of the terminal being used will be helpful, much like one should know how to operate a typewriter before he can learn to type.

Three common devices used with TSO are the IBM 3277, Decwriter, and Decscope. General and brief instructions for each are included in this chapter. There are many models of each and detailed instructions might differ among them.

IBM 3277

An IBM 3277 is a device that consists of a screen to display output much like a television screen. Instead of displaying one line at a time, it can display a number of lines at one time. This is referred to as a page. The size of the page may differ with each model but the most common page is 22 lines long. The user has some control over when information is displayed. To enter information into the system, the IBM 3277 utilizes a keyboard. Data entered through the keyboard is also displayed on the screen. A cursor indicates where on the screen information will be displayed. To enter a command or answer a question, the user types the command or answer on the keyboard and depresses the 'ENTER' key. To retype any portion of the line he depresses the backspace (←) key. However, any mistakes must be corrected before the 'ENTER' key is depressed.

The program has one peculiarity when 3277 units are used: at times part of a page will be displayed at one time and the rest of the page on the next screen. In order to prevent this from happening, the 'CLEAR' key should be depressed before entering a command. The 'CLEAR' key will clear the screen and bring the cursor to the top, then the entire next page can be displayed.

Decscope

A Decscope is similar to an IBM 3277. It too has a keyboard and screen with a cursor. To enter data into the system via a decscope the user types the command on the keyboard and depresses the 'RETURN' key. However instead of displaying a page at a time, the decscope writes only one line at a time, then spaces it up. As the information reaches the top of the screen, it is lost. Again, any pertinent information should be recorded for future reference as it is lost upon leaving the screen. There is no possibility of only half a page appearing on the screen at a time; therefore, to continue the session it is not necessary to clear the screen before displaying the next page.

Decwriter

The Decwriter is a simple typewriter-type terminal with a keyboard for input and a hard copy printer for output. There are various models of Decwriters varying in the kind of printing mechanism, the speed of printing, and a number of other aspects. The Decwriter is similar to the Decscope in that only a line at a time is printed. To enter data the user depresses the carriage return after the data are typed. To learn the details of operating a particular model one should read the operations manual of that model.

Because the Decwriter uses a mechanical printing device rather than an electronic display device it is slower than the IBM 3277. However, it does allow the user to maintain a hard copy of the session for future reference.

CHAPTER V

PROGRAM DESCRIPTION

The program written in connection with this study is designed to convey basic ideas about decomposition. This chapter describes the function of the program, its limitations and some of the problems encountered. The program was developed to be used on a TSO (time sharing) system. Most TSO terminals have a typewriter-like keyboard to enter data. The features of each keyboard vary from terminal to terminal.

The program consists of three major TSO data sets working together to accomplish the desired results. They are named DECOMP, PAGE, and JESSE.

Data Set DECOMP

A command procedure is a TSO data set of prearranged executable sequence of commands with a description qualifier of 'CLIST'. The data set DECOMP is a command procedure or 'CLIST' created to control the processing of the overall program.

DECOMP is divided into two parts, Part 1 and Part 2. Part 1 controls the display of the pages of the text. The text begins by giving the background of decomposition. It follows with a description of the technique and gradually leads the student through the theory of the algorithm and an example.

The program is interactive in that the user can proceed, not only at his own rate, but to whatever degree of detail he wishes. The program is designed to take the student through a general approach to decomposition. He may request further instruction on any topic, as needed. The user must read the information and answer questions based on what he has just learned. The program will immediately tell the user if he has answered correctly or incorrectly, and either allow him to proceed or to review the information and attempt to answer the question again. There are several places where the user can stop and start over at the beginning if he feels it is necessary or reread previous pages.

At the conclusion of Part 1 the user has three choices. He can go through Part 1 again, terminate the session at that point, or enter Part 2.

Part 2 lets the user enter his own data to be run through a decomposition program named JESSE. Part 2 may be entered as often as needed to run more than one problem.

For greater detail on input and output, consult the User's Guide, Appendix A.

Data Set PAGE

A 'DATA' type data set contains any unformatted upper case data of any type. PAGE is a 'DATA' data set that contains all the pages of the text for Part 1 of the program. The command procedure DECOMP determines when these pages will be displayed. Each page explains ideas and gives instructions to the user prompting his response to questions. Briefly, DECOMP controls the interaction between the responses from the user and the text in PAGE.

The pages begin by indicating the assumptions made about the student's background in L.P. and Revised Simplex and gives an introduction to the operations of the program. It then continues with the background of decomposition and a development of the technique.

It then concludes with a step-by-step procedure to solve a decomposition problem and gives an example of the procedure.

Data Set JESSE

JESSE is a data set containing a Fortran program that executes a decomposition algorithm. It is used exclusively in Part 2 of the overall program. It allows the student to input the necessary coefficients to a decomposition problem. As it solves the problem, intermediate results are printed to allow the student to follow the progress of the algorithm.

For greater detail on input and output of Part 2, consult the User's Guide, Appendix A.

Limitations of Part 1

The user has the option of reviewing certain information but the information to be reviewed is not at the discretion of the user. The reviewed pages are predefined by the control program. The information contained in the program is the only information available to the user. Unlike classroom instruction where some personalized instruction is available and questions may be asked, programmed instruction limits the amount of feedback from the student. When using a terminal other than a Decwriter, the user should take notes to which he can refer later.

Limitations of Part 2

The actual program that performs the decomposition algorithm contains a few limitations on the type of problem that can be solved. The problem must have no more than 20 subdivisions and no more than 20 constraints each. It must have no more than 20 corporate constraints and all constraints must be "less than or equal to" inequalities.

Further Study

The possibilities of refinement of the presentation seem unlimited. A more sophisticated interactive program could be written to include more questions and examples and even keep a score to judge the student's progress.

Further study could also be done to incorporate a graphical representation of the decomposition concepts, as done by Adams (1) for basic Linear Programming.

Part 2 could be further developed to include problems with "greater than or equal to" inequalities. Part 2 was written in Fortran which limits its generality. A program which lent itself to variable dimensioning would require fewer limitations on the size of the problem.

CHAPTER VI

SUMMARY AND CONCLUSIONS

This report describes a method whereby the concepts of decomposition can be presented interactively using a time-sharing option (TSO) terminal. The first chapter is an introduction to the report. It discusses linear programming and leads into the large scale linear programming problem. Chapter II reviews the Revised Simplex Method. Chapter III describes a way of solving a large-scale problem. It covers the decomposition method and the formulation of the decomposition model. It then presents a six step decomposition algorithm. Chapter IV discusses three common time-sharing terminals that can be used to execute the program, along with their differences that may cause some difficulty in operation. Chapter V describes the function and internal operation of the program, its limitations, and some of the problems encountered. It is assumed that the user has some knowledge of linear programming, especially of the Revised Simplex Method.

Appendix A is a User's Guide of detailed instructions on the operations of the program. Appendix B gives instructions for storing and changing the program, allowing for changes to the tutorial text. Appendix C is an example of a short session that a user might execute. Appendix D contains the logic block diagrams of the control data set and the decomposition program. And Appendixes E and F are listings of the tutorial text and the control data set.

Any organization that has access to a TSO system has access to a valuable educational tool. A training program can benefit greatly by using the interactive capabilities of the system for pedagogical purposes. With a system of this kind the educational process is not subject to the inconsistent performance of an instructor. More time can be spent in preparing the sessions, which may be prepared by many educators, therefore achieving a more efficient presentation. Many times students contribute greatly to a particular subject during a lab or seminar. These contributions which would otherwise be lost, can be incorporated in the programmed instructions for the benefit of future classes. In effect it eliminates the human error factor from classroom instruction. However, it should be remembered that programmed instruction as discussed in this report is a tool of education and is not meant to replace classroom instruction. Such a tool is meant to give supplemental aid to the student, thus allowing the teacher more time to give individual attention.

BIBLIOGRAPHY

(1) Adams, Marsha Lynn. "Tutorial and Visual Display of the Basic Concepts of Two Dimensional Linear Programming." (Unpub. M.S. report, Oklahoma State University, 1975.)

(2) Dantzig, George B. Linear Programming and Extensions. Princeton, New Jersey: Princeton University Press, 1963.

(3) Dantzig, George B., and Wolfe, Philip. "Decomposition Principle for Linear Programs." Operations Research. Volume 8, Jan., 1960.

(4) Hillier, Frederick S., and Lieberman, Gerald J. Operations Research. 2nd Edition, Holden-Day Inc., 1974.

(5) Lasdon, Leon S. Optimization Theory for Large Systems. New York: The MacMillan Company, 1970.

(6) Orchard-Hays, William. Advanced Linear Programming Computing Techniques. McGraw-Hill Book Company, 1968.

(7) Wagner, G. R. "Conversational Linear Programming for Experiential Learning." Engineering Education. Volume 62, No. 7 (April, 1972), pp. 824-826.

(8) Zacharov, B., Davies, B. W., and Chau, A. Y. C. "Island - An Interactive Graphics System for Mathematical Analysis." The Computer Journal. Volume 17, No. 2 (May, 1974), pp. 104-112.

APPENDIXES

APPENDIX A

USER'S GUIDE

DECOMP can be executed on any system containing the TSO time-sharing options. The user must first acquire a TSO user ID from his computer center and find out the ID number under which the program is stored. Once this is done he can log on and begin the session in the following manner.

STEPS

1. To log on any TSO terminal, use the keyboard to key in the command LOGON aaaaaaa. Where aaaaaaa is the TSO user ID number.

2. Depress the ENTER or RETURN key. (If the number is invalid, the message INVALID PROJECT NUMBER will appear. If this happens, try Step 1 again. If the situation persists, consult the computer center about the ID number.

3. If the number is password protected, you will be asked for the password. Key in the password.

4. Depress the ENTER or RETURN key. (If the password is invalid the message INVALID PASSWORD FOR USER ID aaaaaaa will appear. If this happens, go to Step 1 again. If it happens again consult the computer center.)

5. To begin the session, the user should key in EXEC 'TSO.bbbbbbb DECOMP.CLIST'. Where bbbbbbb is the TSO ID number the program is stored under at the installation.

6. Depress the ENTER or RETURN key. The program will now begin executing.

Executing DECOMP

The first screen of information is general instructions on using the program. Detailed instructions are given at each step when a response is required by the user.

1. After reading the instructions thoroughly, depress ENTER or the carriage return. (If an IBM 3277 is used, the program generates a message to the user to clear the screen before each enter.)

2. The first page of the text will then appear. It explains that the session is in two parts, the tutorial text and an executable program. A choice is given as to which part to execute, enter the appropriate response. You will then be prompted for your name.

3. Assuming the student wishes to step through the text, the second page will appear, outlining the main context of the text, along with additional instructions.

4. Pages 3 and 4 give an introduction into multidivisional problems and explain the angular structure of their constraints. Again press ENTER or RETURN after each page.

5. Page 5 presents the first question of the session. The message ANS = will be printed allowing the answer to be entered immediately following the message. Three chances will be given to answer the question correctly. Upon request the program will return to Page 3 for a review.

6. Page 6 explains the answer to Question #1.

7. A general description of decomposition is then presented followed by Question #2. Again three chances are given to enter the correct answer and an explanation of the answer is given.

8. Page 9 defines a multidivisional problem and the constraints needed for reformulation. If the student wishes he can view Page 10 for an in depth study of the reformulation.

9. Otherwise he can continue to Page 11 and Question #3 on

reformulation. Again the message ANS = is printed to prompt a response. If all three chances are used he will be given the opportunity to review Page 10.

10. Page 12 discusses why the simplex multipliers need to be partitioned for decomposition. Here the student is given the opportunity to continue discussion at a more detailed level by requesting to view Page 13.

11. Or the student can continue to Page 14 and be given Question #4 on why the simplex multipliers are partitioned. The format for the response is similar to Question #2 and #3. And a chance is given to request a review of Page 13.

12. At this point the student may choose to see the six step simple algorithm or continue to Part 2. The steps are presented on two pages and at the end of the second page the student may choose to see an example of how the algorithm works.

Execution of the Example

The student is taken through the entire execution of an actual problem. Intermediate results are given at each step to help the student visualize the process taken at that point by the use of the interactive capabilities of the program. Further explanation and actual computations are available to the student at critical steps.

The first page of the example defines the problem to be solved. For efficiency purposes a problem was chosen from Hillier & Lieberman (4). The problem consists of two divisions of no more than two constraints each and two variables each. All matrices and vectors are singled out for clarity. Unless the user is using a decwriter he should copy this information down for future reference before depressing ENTER to continue.

Execution of Part 2

Part 2 is a Fortran program that uses the decomposition algorithm mentioned in Part 1 to solve a problem who's data is entered through the terminal. All the data to be entered will be asked for by appropriate prompting messages.

The first bit of information to be entered is a title to the problem. After the title is entered messages will be displayed asking for the number of divisions and the number of corporate constraints. These values should be entered as integer numbers without a decimal point. The right hand side of the corporate constraints will be asked for next. These will be read with a Fortran format of F5.2, which means the first value should be entered with a length of no more than 5 digits with the decimal point typed and no more than 2 digits to the right of the decimal point. Insignificant zeroes to the left or right of the decimal point do not need to be entered. The enter key should be depressed after each value is typed. This results in entering one value per line until all values are entered. The rest of the data is entered in four steps for each division as follows:

1. Prompting message - 'Type # constraints and # variables for Division 1'

   Response - Enter 2 integer values, one per line.
2. Prompting message - 'Type x coefficients of the objective function for Division 1'

   Response - Enter x number of real values with the decimal point as described above, one per line.

3. Prompting message - 'Type x coefficients of Division 1 constraints'

   Response - Again enter x number of real values, one per line.

4. Prompting message - 'Type x coefficients of the RHS of Division 1'

   Response - again enter x number of real values, one per line.

The preceding steps will be repeated for each division. When all data are entered a matrix representation of the program will be displayed giving the user a chance to view the data and then the opportunity to reenter the data if necessary.

Once the data are entered correctly the program solves the problem using the decomposition method mentioned in Part 1, giving intermediate results at each iteration.

APPENDIX B

INSTRUCTIONS FOR STORING AND CHANGING PROGRAMS

The sequential data sets that make up this program are stored at the Oklahoma State University TSO library under the TSO user identification number of U16300A. Their full qualification is as follows:

'TSO.U16300A.DECOMP.CLIST'
'TSO.U16300A.PAGE.DATA'
'TSO.U16300A.JESSE.FORT'

To store the programs under a personal identification number, a simple copy command on TSO of the form COPY 'TSO.U16300A.name.type' 'TSO.aaaaaaa.name.type' is all that is needed (where aaaaaaa is the personal identification number of the user).

Once the user has stored the data sets a few changes must be made. At present DECOMP, which controls the flow of the program, lists the tutorial text by its fully qualified name. The full qualification must be changed to the user's ID as follows.

STEPS

It is assumed the user has logged on his own TSO ID and copied the data sets.

1. With TSO in the READY mode, edit the CLIST by the command E DECOMP.CLIST.

2. Once in the EDIT mode, enter the following command: C 10 50000 /U16300A/aaaaaaa/ALL (where aaaaaaa is the user's ID number). This command changes all occurances of a fully qualified data set name to the user's ID.

3. Get out of the edit mode by entering END S.

Before the programs can be executed, one other change must be made. DECOMP calls an object module of the fortran program. Assuming the user has acquired a copy of the fortran source program JESSE.FORT he must now create an object module as follows.

With TSO in the READY mode, compile the fortran program by entering the command FORT JESSE. This compile will create an object module and the program will be ready to execute.

If any changes are made to either the CLIST DECOMP or the tutorial text PAGE, caution must be exercised. There is a close relationship between these two data sets and a similar relationship must be present after any changes are made.

Changes may also be made to the source program JESSE, although this program can be altered as you would any program written in a high level programming language. Each time the program is altered, a new object module must be created as above.

APPENDIX C

SAMPLE OF A SHORT SESSION

```
exec decomp
THIS PROGRAM IS DESIGNED TO OPERATE ON ANY TSO TERMINAL.
IT IS INTERACTIVE, MEANING THE USER WILL BE PROMPTED FOR A
RESPONSE.  A DECWRITER IS PREFERRED SINCE YOU CAN MAINTAIN
A HARDCOPY OF THE SESSION AND REFER TO IT AT ANY TIME.
HOWEVER, DECSCOPES AND IBM 3277'S CAN ALSO BE USED.
THE OPERATION OF A DECSCOPE AND DECWRITER IS SLIGHTLY
DIFFERENT THAN A 3277. IF YOU ARE USING A DECSCOPE OR
DECWRITER, AFTER TYPING A RESPONSE PRESS THE RETURN KEY.
HOWEVER, WITH THE 3277 YOU MUST CLEAR THE SCREEN FIRST
THEN ENTER YOUR RESPONSE.  THE INSTRUCTIONS DURING A
SESSION ASSUME YOU ARE USING AN IBM 3277.
   IF YOU ARE USING A 3277 OR SIMILAR TERMINAL ENTER  CRT

 PAGE.DATA
                          A TSO PRESENTATION OF THE
                           DECOMPOSITION TECHNIQUE
                            OF LINEAR PROGRAMMING


    THIS PRESENTATION IS DESIGNED TO GIVE THE ADVANCED STUDENT A
 BETTER UNDERSTANDING OF DECOMPOSITION.  IT IS DIVIDED INTO TWO PARTS.

    PART 1.  A TUTORIAL TEXT THAT TAKES THE STUDENT THROUGH THE
             DEVELOPMENT OF DECOMPOSITION.  IT IS ASSUMED THE STUDENT
             HAS A THOROUGH UNDERSTANDING OF LP AND REVISED SIMPLEX.

    PART 2.  AN EXECUTABLE PROGRAM THAT LETS YOU ENTER YOUR OWN DATA
             TO BE RUN AND GIVES YOU INTERMEDIATE RESULTS TO ALLOW
             YOU TO MONITOR ITS PROGRESS.

    IF YOU WOULD LIKE TO STEP THROUGH PART 1 ENTER YES.

    IF YOU WANT TO RUN DATA ENTER NO.
yes
TYPE IN YOUR NAME AND HIT ENTER.
bill
 PAGE.DATA
                        DECOMPOSITION

    THIS IS A DEVELOPMENT OF THE DECOMPOSITION TECHNIQUE OF
 LINEAR PROGRAMMING.  IT IS ASSUMED THE STUDENT'S BACKGROUND INCLUDES
 A THOROUGH UNDERSTANDING OF LINEAR PROGRAMMING AND REVISED SIMPLEX.
 THE TEXT WILL COVER:
   1. MULTIDIVISIONAL PROBLEMS
   2. THEIR ANGULAR STRUCTURE
   3. THE DECOMPOSITION APPROACH - THEORY
   4. A DECOMPOSITION ALGORITHM
   5. AN EXAMPLE


  EVERY SO OFTEN A QUESTION WILL BE ASKED OF YOU.  TYPE IN THE ANSWER
  AND PRESS ENTER.
  IF AT ANYTIME YOU WANT TO TERMINATE PART 1 AND GO TO PART 2
  TYPE IN STOP AND PRESS ENTER.
  (PRESS CLEAR AND ENTER TO CONTINUE)
```

PAGE.DATA

DECOMPOSITION IS A TECHNIQUE USED FOR SOLVING PROBLEMS HAVING A SPECIAL STRUCTURE. THESE PROBLEMS ARE CALLED MULTIDIVISIONAL AND THEIR NAME HINTS AT THE TYPE OF STRUCTURE USED, MULTIDIVISIONAL. HENCE, THEY ARE PROBLEMS THAT ENCOMPASS SEVERAL DIVISIONS. THEREFORE, THE PROBLEMS ARE ALMOST DECOMPOSABLE INTO SEPARATE PROBLEMS, WHERE EACH DIVISION IS CONCERNED ONLY WITH OPTIMIZING IT'S OWN OPERATION. HOWEVER, SOME OVERALL COORDINATION IS REQUIRED IN ORDER TO BEST DIVIDE CERTAIN ORGANIZATIONAL RESOURCES AMONG THE DIVISIONS.

IF YOU WERE TO LOOK AT A TABLE OF CONSTRAINT COEFFICIENTS FOR THIS TYPE OF PROBLEM YOU WOULD FIND THAT THE CONSTRAINTS FOR EACH DIVISION COULD BE GROUPED TOGETHER IN A BLOCK FORMING AN ANGULAR STRUCTURE.

THE NEXT PAGE EXPLAINS THE ANGULAR STRUCTURE OF MULTIDIVISIONAL PROBLEMS AND GIVES AN EXAMPLE.
(PRESS CLEAR AND ENTER TO CONTINUE OR TYPE STOP TO TERMINATE)

PAGE.DATA
TABLE OF CONSTRAINT COEFFICIENTS FOR MULTIDIVISIONAL PROBLEMS.

```
 --------------
|              |          CORPORATE CONSTRAINTS ON
|______________|          ORGANIZATIONAL RESOURCES

 --
|  |                      CONSTRAINTS ON RESOURCES AVAILABLE ONLY TO
|__|                              DIVISION 1
     --
    |  |
    |__| .                         DIVISION 2
          .
           . --
            |  |
            |__|                   LAST DIVISION
```

EACH SMALLER BLOCK CONTAINS THE COEFFICIENTS OF THE CONSTRAINTS FOR ONE DIVISION. THE LONG BLOCK AT THE TOP CONTAINS THE COEFFICIENTS OF THE CORPORATE CONSTRAINTS FOR THE MASTER PROBLEM (THE PROBLEM OF COORDINATING THE ACTIVITIES OF THE DIVISIONS).

```
PAGE.DATA

   QUESTION #1  :
               WHAT TYPE OF SPECIAL PROBLEM WAS THE DECOMPOSITION
                  METHOD DEVELOPED FOR?

ANS =multidivisional
VERY GOOD BILL
 PAGE.DATA


   THE CORRECT ANSWER IS MULTIDIVISIONAL

        THOSE PROBLEMS WHERE THE MAJORITY OF THE CONSTRAINTS
 CAN BE SEPARATED INTO GROUPS ACCORDING TO THE
 RESOURCES AVAILABLE.


   TO LEARN HOW THE DECOMPOSITION METHOD SOLVES THESE SPECIAL
 STRUCTURED PROBLEMS PRESS ENTER TO GO TO THE NEXT PAGE.

     (OR TYPE STOP TO TERMINATE)



 PAGE.DATA


   THE BASIC APPROACH IS TO REFORMULATE THE PROBLEM IN A WAY THAT
 GREATLY REDUCES THE NUMBER OF FUNCTIONAL CONSTRAINTS AND THEN TO
 APPLY THE REVISED SIMPLEX.  THIS VERSION OF THE SIMPLEX METHOD CAN
 BE THOUGHT OF AS HAVING EACH DIVISION SOLVE ITS OWN SUBPROBLEM AND
 SENDING ITS PROPOSAL TO THE MASTER PROBLEM.

   IF THESE PROPOSALS VIOLATE THE CORPORATE CONSTRAINTS THE
 DECOMPOSITION TECHNIQUE WILL EVALUATE THAT VIOLATION AND CALCULATE
 PENALTIES FOR EACH OF THE DIVISIONS IN ORDER TO FORCE THEIR SOLUTIONS
 TOWARD A CORPORATE OPTIMUM.  IN THIS WAY WE CAN COORDINATE THE
 PROPOSALS FROM ALL THE DIVISIONS TO FIND THE OPTIMAL SOLUTION FOR
 THE OVERALL ORGANIZATION.

 PRESS CLEAR AND ENTER FOR QUESTION #2 OR STOP TO TERMINATE.
```

```
 PAGE.DATA

    QUESTION #2:

         YOU ARE IN CHARGE OF BUDGETING A LARGE CORPORATION
         AND EACH PLANT MANAGER SENDS YOU PROPOSED BUDGET
         REQUIREMENTS FOR HIS PLANT.  BUT, AS IS USUALLY
         THE CASE, YOU CANNOT MEET ALL THE REQUIREMENTS.
         AS BUDGETING DIRECTOR YOUR NEXT STEP IS TO:

         A.   DETERMINE YOURSELF WHAT THE PLANT BUDGETS SHOULD BE.
         B.   CALCULATE SOME KIND OF PENALTY FOR EACH PLANT TO
              FORCE THEM TO COME UP WITH AN AGREEABLE PROPOSAL.
         C.   TEAR UP THE PROPOSALS AND HAVE THEM START OVER.
         D.   RUN THE CORPORATE BUDGET AS A WHOLE USING
              REVISED SIMPLEX. THEN SEND EACH PLANT ITS BUDGET.

ANS =d
SORRY BILL, D IS AN INCORRECT ANSWER.
TRY AGAIN, YOU HAVE 2 MORE CHANCES.
ANS =b
VERY GOOD BILL
THE CORRECT ANSWER IS B.
YOU WOULD EVALUATE THE VIOLATIONS AND CALCULATE PENALTIES.
BUT HOW?
HIT ENTER.
 PAGE.DATA
    LET'S DEFINE A PROBLEM WITH N DIVISIONS AS SUCH:

                      N
    MAXIMIZE          SUM  (C(I)*X(I))
                      I=1

    SUBJECT TO:       A(1)  A(2) ... A(N)     X(0)       B(0)
                      A(N+1)                  X(1)       B(1)
                            A(N+2)            X(2)       B(2)
                                   .           .     =    .
                                    .          .          .
                                     .         .          .
                                      A(2N)   X(N)       B(N)
    WHERE THE B,C,X'S ARE VECTORS AND A'S ARE MATRICES.
    CONSIDER THE SOLUTION SPACE FOR DIVISION K; CALL IT S(K). ANY POINT
 IN S(K) CAN BE REPRESENTED AS A WEIGHTED AVG. OF THE EXTREME POINTS
 OF S(K).
    LET X(J,K) = EP(J) OF DIVISION K AND L(J,K) IT'S WEIGHT.
    I.E.  ANY FEASIBLE POINT X(*,K) = SUM ON J OF (L(J,K)*X(J,K)) FOR
  SOME COMBINATION OF THE L(J,K) SUCH THAT 0 <= L(J,K) <= 1 AND THE
  SUM ON J OF ALL L(J,K) IS EQUAL TO 1.
IF YOU WOULD LIKE TO SEE THE PROBLEM REFORMULATED
BY USING THESE CONSTRAINTS ENTER YES.
yes
```

PAGE.DATA

THIS EQUATION FOR X(*,K) AND THE CONSTRAINTS ON THE L(J,K) PROVIDE A METHOD FOR REPRESENTING THE FEASIBLE SOLUTIONS TO DIVISION K WITHOUT USING ANY OF THE ORIGINAL CONSTRAINTS. HENCE THE OVERALL PROBLEM CAN NOW BE REFORMULATED WITH FAR FEWER CONSTRAINTS AS

```
                    N
MAXIMIZE           SUM  SUM  L(J,K)(C(K)*X(J,K))
                   K=1   J

SUBJECT TO:
                    N
                   SUM  SUM  L(J,K)(A(K)*X(J,K))
                   K=1   J

           AND     SUM  L(J,K) = 1
                    J
```

STUDY THIS REFORMULATION OF THE MASTER PROBLEM FOR AWHILE. THE SYMBOLISM MIGHT BE CONFUSING. THE FIRST SUMMATION (ON K) REFERS TO THE DIVISIONS. THE SECOND SUMMATION (ON J) REFERS TO THE EXTREME POINTS WITHIN EACH DIVISION.

PAGE.DATA

QUESTION #3 :

IN THE REFORMULATION OF THE MASTER PROBLEM
WHAT DO THE L(J,K)'S STAND FOR ?

A. CONSTRAINT COEFFICIENTS
B. SIMPLEX MULTIPLIERS
C. EXTREME POINTS IN THE SOLUTION
D. RESPECTIVE WEIGHTS ON THE EXTREME POINTS

ANS =d
VERY GOOD BILL
THE CORRECT ANSWER IS D
PAGE.DATA

SINCE THIS REFORMULATION HAS FAR FEWER CONSTRAINTS IT SHOULD BE SOLVABLE WITH MUCH LESS COMPUTATIONAL EFFORT. AT FIRST GLANCE IT WOULD SEEM THAT ALL THE EXTREME POINTS (X(J,K)) NEED BE IDENTIFIED. A TEDIOUS TASK TO SAY THE LEAST. FORTUNATELY, IT IS NOT NECESSARY TO DO THIS WHEN USING THE REVISED SIMPLEX METHOD. ALL THAT IS REQUIRED IS THAT THE SIMPLEX MULTIPLIERS (PI) BE PARTITIONED SO THAT YOU CALCULATE ONLY WHAT IS NEEDED.

DO YOU WANT TO LEARN HOW THIS IS DONE IN MORE DETAIL?yes

```
PAGE.DATA
   RECALL THAT WITH REVISED SIMPLEX THE VECTOR OF SIMPLEX MULTIPLIERS
 (PI = CB * BI) IS USED IN COMPUTING THE RELATIVE COST COEFFICIENTS
 (BI = B INVERSE).  DURING DECOMPOSITION PI NEEDS TO BE PARTITIONED
 AS (PI1,PI0).  LET NLC DENOTE THE NUMBER OF CORPORATE (LINKING)
 CONSTRAINTS.  LET  BI(1;NLC) BE THE MATRIX CONSISTING OF THE FIRST
 NLC COLUMNS OF BI, AND LET BI(J) BE THE VECTOR CONSISTING OF THE JTH
 COLUMN OF BI.  THEN PI1 = CB * BI(1;NLC) AND PI0 = CB * BI(NLC + J).

    THE USUAL SIMPLEX CRITERION ASKS THAT WE FIND
        MIN F(J) = (PI1 * A(J) - C(J)) X(J) + PI0

    THEREFORE, THE FIRST STEP AT EACH ITERATION REQUIRES SOLVING
 N (NUMBER OF DIVISIONS) LP PROBLEMS OF THE TYPE

        MIN  (PI1 * A(J) - C(J)) X(J) + PI0

        SUBJECT TO  A(N+J) * X(J) <= B(J)

                    X(J) >= 0

PRESS ENTER TO CONTINUE OR TYPE STOP TO TERMINATE.
 PAGE.DATA

          QUESTION #4 :
              WHY ARE THE SIMPLEX MULTIPLIERS, PI,
        PARTITIONED INTO PI1 AND PI0 ?

        A.   TO SAVE COMPUTATIONAL EFFORT
        B.   TO DISTINGUISH BETWEEN THE SIMPLEX MULTIPLIERS
             OF EACH DIVISION.
        C.   TO COMPUTE EACH RELATIVE COST COEFFICIENT
        D.   SO THAT IT IS NOT NECESSARY TO IDENTIFY ALL EXTREME POINTS.


ANS =a
VERY GOOD BILL
THE CORRECT ANSWER IS A.
B IS AN M*M MATRIX,
BUT TO CALCULATE PI1 AND PI0 YOU NEED ONLY
NLC+1 COLUMNS OF B.
WOULD YOU LIKE TO SEE A SIMPLE ALGORITHM AND EXAMPLE?yes
 PAGE.DATA

                     STEP BY STEP ALGORITHM

 STEP 1.     USING THE SIMPLEX MULTIPLIERS PI1 SOLVE THE DIVISION
             SUBPROBLEMS AS ABOVE OBTAINING SOLUTIONS X(I)
             AND OPTIMAL OBJECTIVE VALUES Z(I).

 STEP 2.     COMPUTE MIN Z(I) + PI0 = F(J)

 STEP 3.     STOPPING RULE
             IF F(J) >= 0 THE OPTIMAL SOLUTION IS SUM L(J)*X(J)
             WHERE THE X(J)'S ARE THE EXTREME POINTS OF THE SOLUTION
             SPACE CORRESPONDING TO BASIC L(J).
             REMEMBER, L(J)'S ARE THE RESPECTIVE WEIGHTS ON THESE POINTS
             AND ARE COMPUTED ONLY UPON TERMINATION OF THE PROBLEM
             BY THE FINAL B INVERSE TIMES THE ORIGINAL RHS.

             STOP

   (PRESS CLEAR AND ENTER TO CONTINUE)
```

PAGE.DATA

STEP 4. IF F(J) < 0 FORM THE COLUMN TO ENTER THE BASIS AS

```
           | A(J)*X(J) |
      A' = |-----------|
           |     I     |
```

WHERE I IS AN N COMPONENT VECTOR WITH A ONE IN POSITION J AND ZEROES ELSEWHERE.

STEP 5. FOR THE REVISED SIMPLEX METHOD TO NOW DETERMINE THE LEAVING BASIC VARIABLE IT IS NECESSARY TO CALCULATE THE CURRENT COEFFICIENTS AND RHS AS BI*A' AND BI*B'.

```
B' BEING THE VECTOR OF | B(0) |
                       |------|
                       |  1   |
```

WHERE 1 IS AN N COMPONENT VECTOR OF ALL 1'S.

STEP 6. OBTAIN A NEW BASIS INVERSE.
OBTAIN NEW SIMPLEX MULTIPLIERS.
GO BACK TO STEP 1 AND REPEAT.

WOULD YOU LIKE TO SEE AN EXAMPLE OF THIS ALGORITHM?yes

PAGE.DATA

FOR AN EXAMPLE, CONSIDER THIS PROBLEM WITH 2 DIVISIONS

```
MAXIMIZE  Z = 4X(1) + 6X(2) + 8Y(1) + 5Y(2)
  S.T.
               X(1) + 3X(2) + 2Y(1) + 4Y(2) <= 20
              2X(1) + 3X(2) + 6Y(1) + 4Y(2) <= 25
               X(1) +  X(2)                 <=  5
               X(1) + 2X(2)                 <=  8
                              4Y(1) + 3Y(2) <= 12
     AND  X(J),Y(J) >= 0

A(1)=| 1 3 |   A(2)=| 2 4 |   A(3)=| 1 1 |   A(4)=| 4 3 |
     | 2 3 |        | 6 4 |        | 1 2 |

C(1)=| 4 6 |   C(2)=| 8 5 |   B(0)=| 20 |   B(1)=| 5 |   B(2)=| 12 |
                                   | 25 |        | 8 |
AND X = X(1),X(2) AND Y = Y(1),Y(2)
```

COPY THE ABOVE DOWN FOR FUTURE REFERENCE

PRESS ENTER.

PAGE.DATA

THE REFORMULATED MASTER PROBLEM REQUIRES ONLY 4 CONSTRAINTS 2 FOR THE CORPORATE CONSTRAINTS AND 1 CONSTRAINT FOR EACH DIVISION THAT REQUIRES THE SUM OF THE WEIGHTS ADD UP TO 1. (ON A LARGE PROBLEM THIS WOULD BE A SIGNIFICANT SAVINGS)

FOR THE INITIAL BASIC FEASIBLE SOLUTION :

```
    | 1 0 0 0 |               | 20 |
B = | 0 1 0 0 | = BI     B' = | 25 |     CB = (0,0,0,0)
    | 0 0 1 0 |               |  1 |
    | 0 0 0 1 |               |  1 |
```

WHERE B' IS THE RHS OF THE REFORMULATED MASTER PROBLEM.

HIT ENTER.

```
 PAGE.DATA

STEP 1.     USING THE SIMPLEX MULTIPLIERS PI1 SOLVE THE DIVISION PROBLEMS
       REMEMBER PI = CB * BI
       INITIALLY CB=(0,0,0,0) & BI=I=B, SO PI=(0,0,0,0) & PI1=(0,0)
       THE SOLUTIONS ARE :  X(1) = 2 , X(2) = 3 , AND Z(1) = -26
                            Y(1) = 3 , Y(2) = 0 , AND Z(2) = -24
DO YOU WANT TO SEE HOW THE SOLUTION IS COMPUTED?yes
 PAGE.DATA
   SOLVE DIVISION #1 :
       MIN Z(1) = (PI1 * A(1) - C(1))X      | OR  MIN  (-4,-6)X
       S.T.                                 |     S.T.
                   A(3)X <= B(1)            |          | 1 1 | X(1) <= 5
                                            |          | 1 2 | X(2) <= 8
       THE SOLUTION IS X(1)=2, X(2)=3 AND Z(1)=-26

   SOLVE DIVISION #2 :
       MIN Z(2) = (PI1 * A(2) - C(2))Y      | OR  MIN  (-8,-5)Y
       S.T.                                 |     S.T.
                   A(4)Y <= B(2)            |          | 4 3 | Y <= 12

       THE SOLUTION IS Y(1)=3, Y(2)=0 AND Z(2)=-24

 PAGE.DATA

STEP 2.     FIND THE MINIMUM OF Z(N) + PI0 = F
            REMEMBER PI0 = CB * COLUMN(NLC + N) OF BI
            N BEING THE NUMBER OF THE DIVISION.
            THEREFORE PI0 DIFFERS ACCORDING TO THE DIVISION.
       F = MIN = -26 THEREFORE THE WEIGHTS(PENALTY) ON
         E.P. (2,3) OF DIVISION 1 ENTERS THE BASIS
DO YOU NEED HELP?yes
 PAGE.DATA
   SOLVE:
       Z(1) + PI0 = -26 + 0 = -26

       Z(2) + PI0 = -24 + 0 = -24

 PAGE.DATA

STEP 3.     STOPPING RULE.  IF F IS >= 0 IT IS AN OPTIMAL SOLUTION. STOP.

       F = -26 THEREFORE WE MUST CONTINUE.
HIT ENTER.
 PAGE.DATA
                                                               |11|
STEP 4.     GENERATE THE COLUMN TO ENTER THE BASIS AS :  A' =  |13|
                                                               | 1|
                                                               | 0|
IF YOU NEED HELP TO GENERATE THE COLUMN ENTER YES.yes
 PAGE.DATA
           | A(1)*E.P.|     | 1 3 | | 2 |     | 11 |
     A' =  |-----------|  =  | 2 3 |*| 3 |  =  | 13 |  =  A'
           |     I     |         1           |  1 |
                                 0           |  0 |
HIT ENTER.
```

```
 PAGE.DATA
STEP 5.     DETERMINE THE LEAVING BASIC VARIABLE.  PROCEED IN THE USUAL
            WAY TO CALCULATE THE CURRENT COEFFICIENTS AND THE RHS.

               | 11 |              | 20 |   THE RATIOS ARE :
   BI * A' =   | 13 |  ,  BI * B' = | 25 |   20/11 , 25/13 , 1/1
               |  1 |              |  1 |
               |  0 |              |  1 |

   THE MINIMUM RATIO IS 1 (THE THIRD ROW). R = 3.
   THUS THE NEW VALUES OF CB ARE (0,0,26,0)
   THE EXTREME POINTS IN THE BASIS ARE : (_,_) (_,_) (2,3) (_,_)
HIT ENTER.
 PAGE.DATA
STEP 6.     OBTAIN A NEW BASIS INVERSE AND NEW SIMPLEX MULTIPLIERS.
                                            | 1 0 |
   PI1 = CB * BI(1;2) = (0,0,26,0) *        | 0 1 | = (0,0) = PI1
                                            | 0 0 |
                                            | 0 0 |
THERE ARE MANY WAYS TO FIND AN INVERSE.
WOULD YOU LIKE TO SEE AN EASY ONE?yes
 PAGE.DATA

   BI' = E * BI WHERE E IS AN IDENTITY MATRIX EXCEPT THAT IT'S KTH
   COLUMN IS REPLACED BY THE VECTOR M WHERE
                                                        | 1 0 -11 0 |
       | -A'(I,K)/A'(R,K),IF I¬=R |                     | 0 1 -13 0 |
   M = |                          | THEREFORE E =       | 0 0   1 0 |
       | 1/A'(R,K),IF I=R         |                     | 0 0   0 1 |

   SO         | 1 0 -11 0 |
      BI' =   | 0 1 -13 0 |
              | 0 0   1 0 |
              | 0 0   0 1 |
HIT ENTER.
 PAGE.DATA
*** ITERATION 1

STEP 1.  RESOLVE THE DIVISION PROBLEMS.
   SOLUTION FOR DIVISION #1 :  X(1) = 2    X(2) = 3    Z(1) = -26
   SOLUTION FOR DIVISION #2 :  Y(1) = 3    Y(2) = 0    Z(2) = -24
DO YOU NEED MORE INFORMATION?yes
 PAGE.DATA
   COMPUTE THE OBJ. COEFFICIENTS FOR DIVISION 1
         (0 0) * |1 3| - (4,6)  =  (-4 -6)
                 |2 3|


   COMPUTE THE OBJ. COEFFICIENTS FOR DIVISION 2
         (0 0) * |2 4| - (8,5)  = (-8 -5)
                 |6 4|
HIT ENTER.
```

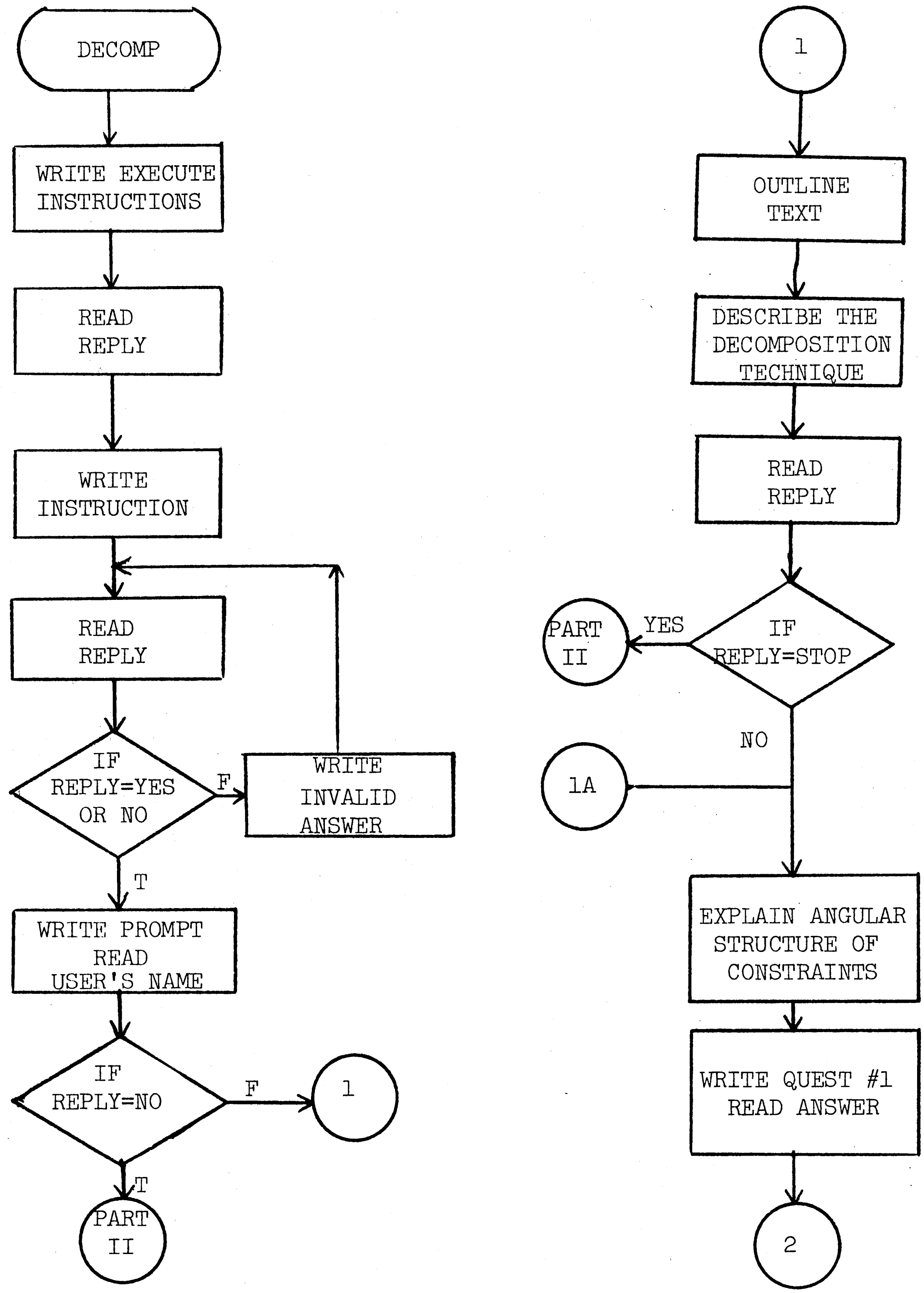
```
PAGE.DATA

STEP 2.
                                                |-11|
   PI0 = CB * BI(3) = (0,0,26,0) * |-13| = 26
                                                | 1 |
   Z(1) + 26  =  0                              | 0 |

                                                |0|
   PI0 = CB * BI(4) = (0,0,26,0) * |0| = 0
                                                |0|
   Z(2) + 0  = -24                              |1|

   F = -24  THERFORE WEIGHTS ON E.P. (3,0) OF DIVISION 2 ENTERS BASIS


STEP 3.     STOPPING RULE.

   F = -24
DO WE STOP OR CONTINUE?continue
HIT ENTER.
 PAGE.DATA

STEP 4.
         |A(2) * E.P.|    |2 4|   |3|    | 6|
   A' = |-----------| = |6 4| * |0| = |18|
         |     I     |          0        | 0|
                                 1        | 1|


STEP 5.
                |1 0 -11 0|   | 6|   | 6|
   BI * A' = |0 1 -13 0| * |18| = |18|
                |0 0  1  0|   | 0|   | 0|   THE RATIOS ARE :
                |0 0  0  1|   | 1|   | 1|
                                              9/6 , 12/18 , 1/0 , 1/1
                |1 0 -11 0|   |20|   | 9|
   BI * B' = |0 1 -13 0| * |25| = |12|
                |0 0  1  0|   | 1|   | 1|
                |0 0  0  1|   | 1|   | 1|

   THE MINIMUM RATIO IS 12/18. R = 2.  CB = (0,24,26,0)
   THE EXTREME POINTS IN THE BASIS ARE : (_,_) (3,0) (2,3) (_,_)


 PAGE.DATA

STEP 6:
   B INVERSE  |1 -1/3  -20/3  0|        PI1 = (0,4/3)
              |0  1/18 -13/18 0|
              |0   0      1   0|
              |0 -1/18  13/18 1|

WORK THIS YOURSELF TO BE SURE YOU KNOW HOW IT IS DONE.
DO YOU WANT TO START OVER AT THE BEGINNING OF THE EXAMPLE?
no
```

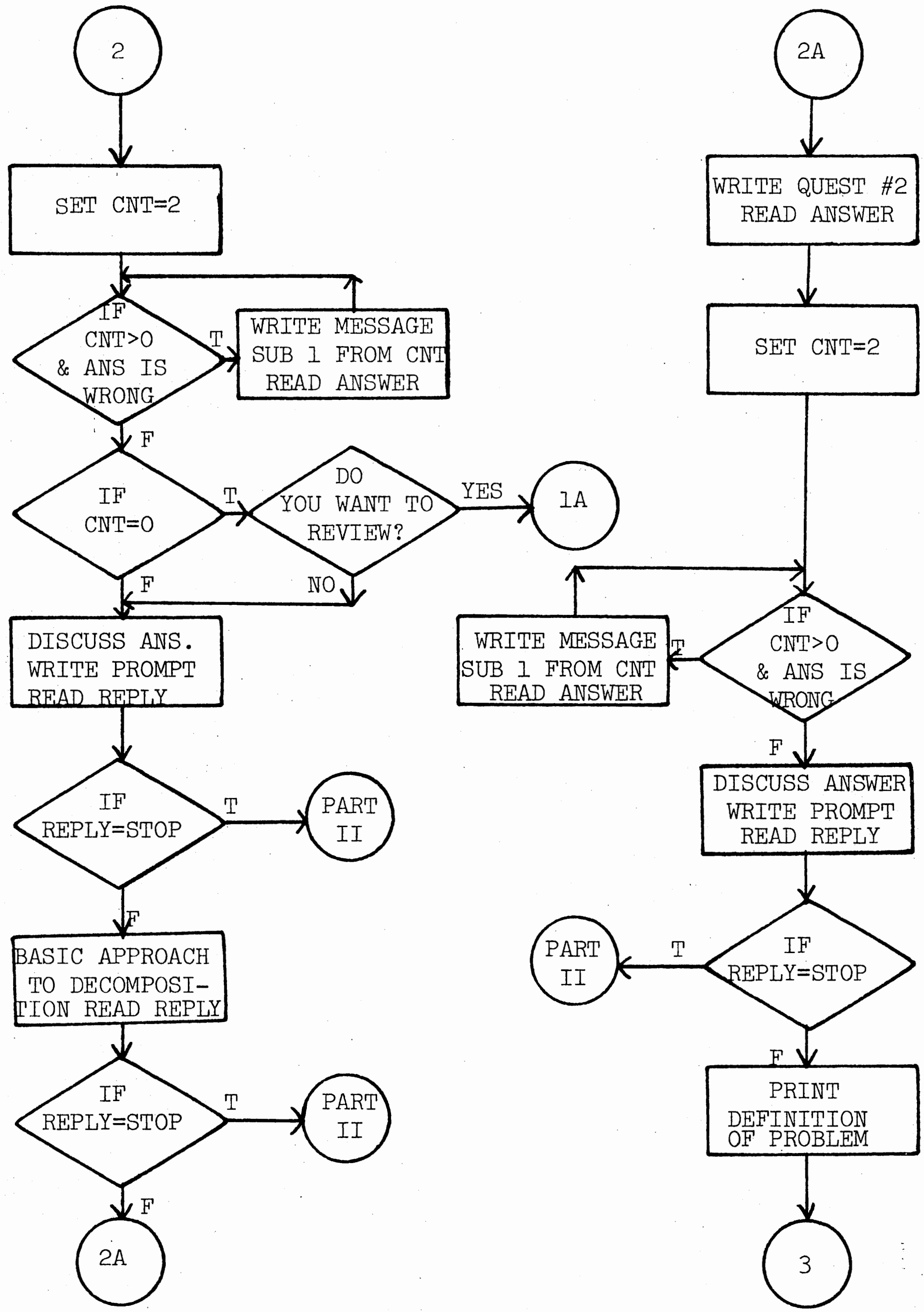
```
 PAGE.DATA
       QUESTION #5 :
           IF                |1 -1/3   -20/3  0|
              B INVERSE =    |0   1/18 -13/18 0|   PI1 = (0,4/3)
                             |0   0       1   0|
                             |0 -1/18   13/18 1|

                 |1 3|           |2 4|
           A(1) = |2 3|   A(2) = |6 4|    C(1) = (4 6) AND C(2) = (8 5)

           COMPUTE THE COEFFICIENTS OF THE OBJECTIVE FUNCTION
             FOR DIVISION 1 AND DIVISION 2 RESPECTIVELY.

           A.   (0,0) AND (0,0)
           B.   (-4,-6) AND (-8,-5)
           C.   (-4/3,-2) AND (0,1/3)
           D.   (-16/3,-1) AND (4,-2/3)

ANS =c
VERY GOOD BILL
THE CORRECT ANSWER IS C.
 PI1 * A(I) - C(I)
HIT ENTER.
 PAGE.DATA
*** ITERATION 2

STEP 1.
   DIVISION 1 OBJ. COEFFICIENTS
          (0,4/3) * |1 3| - (4,6)   =  (-4/3,-2)
                    |2 3|

   SOLUTION :  X(1) = 2  X(2) = 3   Z(1) = -26/3


   DIVISION 2 OBJ. COEFFICIENTS
          (0,4/3) * |2 4| - (8,5)   =  (0,1/3)
                    |6 4|

   SOLUTION :  Y(1) = 0  Y(2) = 0   Z(2) = 0


HIT ENTER.
 PAGE.DATA

STEP 2.
                         |-20/3 |
   PI0 = (0,24,26,0) *   |-13/18|   =  26/3
                         |   1  |
                         | 13/18|

   Z(1) + 26/3  =  0
                         |0|
   PI0 = (0,24,26,0) *   |0| = 0
                         |0|
                         |1|

   Z(2) + 0  =  0

  F = 0

STEP 3.    STOPPING RULE .

   F IS >= 0  THIS IS AN OPTIMAL SOLUTION.
HIT ENTER.
```

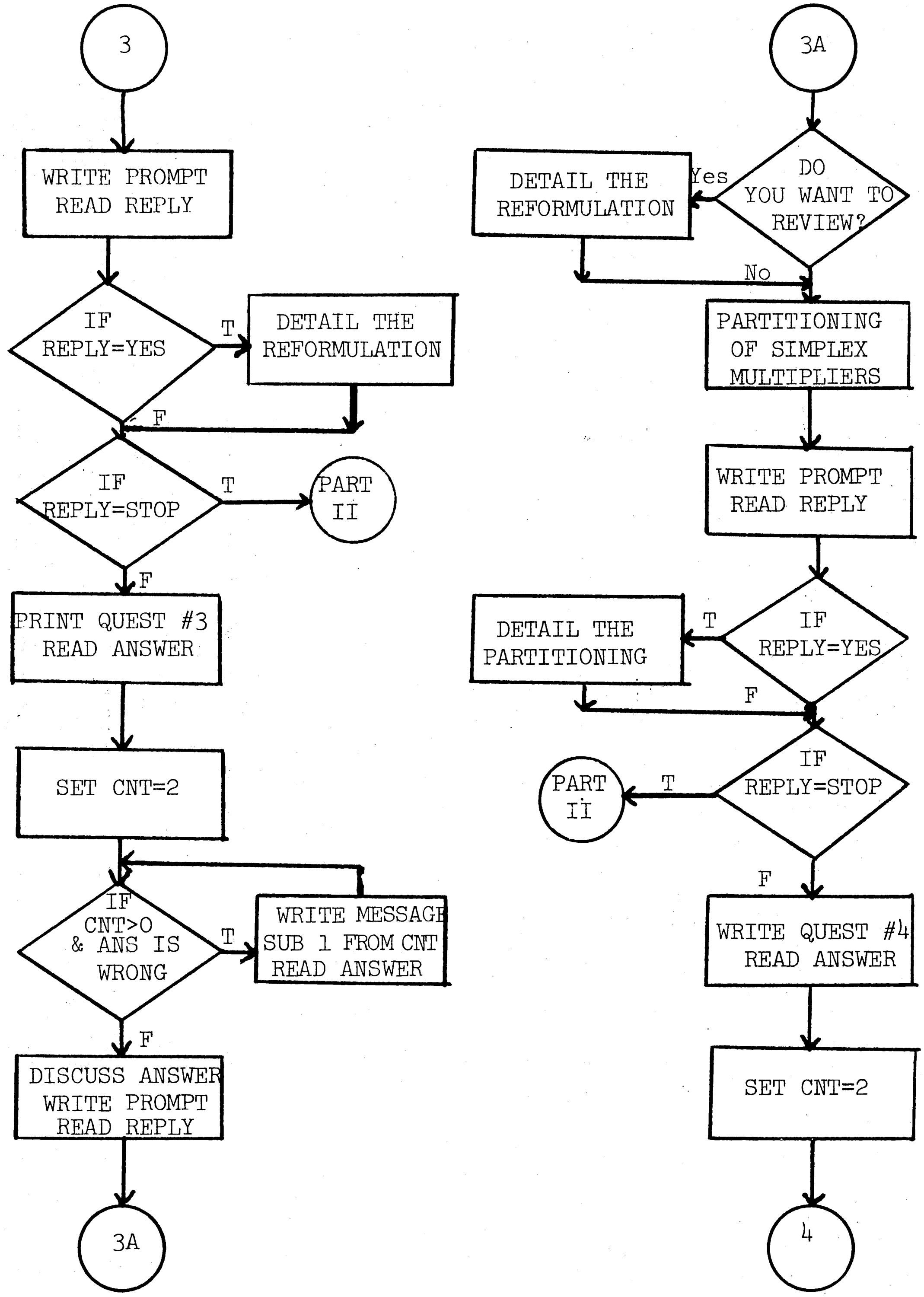
```
PAGE.DATA
   THE EXTREME POINTS IN THE BASIS ARE : (_,_) (3,0) (2,3) (_,_)
      THE WEIGHTS ON THESE POINTS ARE :  5 , 2/3 , 1 , 1/3
          COMPUTED BY B INVERSE * ORIGINAL RHS

 X = SUM  L(J) * X(J) = 1 * (2,3) = (2,3) = X(1),X(2)

 Y = SUM  L(J) * Y(J) = 2/3 * (3,0) = (2,0) = Y(1),Y(2)

 THUS, AN OPTIMAL SOLUTION FOR THIS PROBLEM IS
          X(1) = 2 , X(2) = 3
          Y(1) = 2 , Y(2) = 0

 Z = 4*2 + 6*3 + 8*2 + 5*0 = 42



HIT ENTER.
IF YOU HAVE DATA YOU WANT TO RUN AS A PROGRAM ENTER YES.
no
TO END THE SESSION ENTER LOGOFF
READY
```

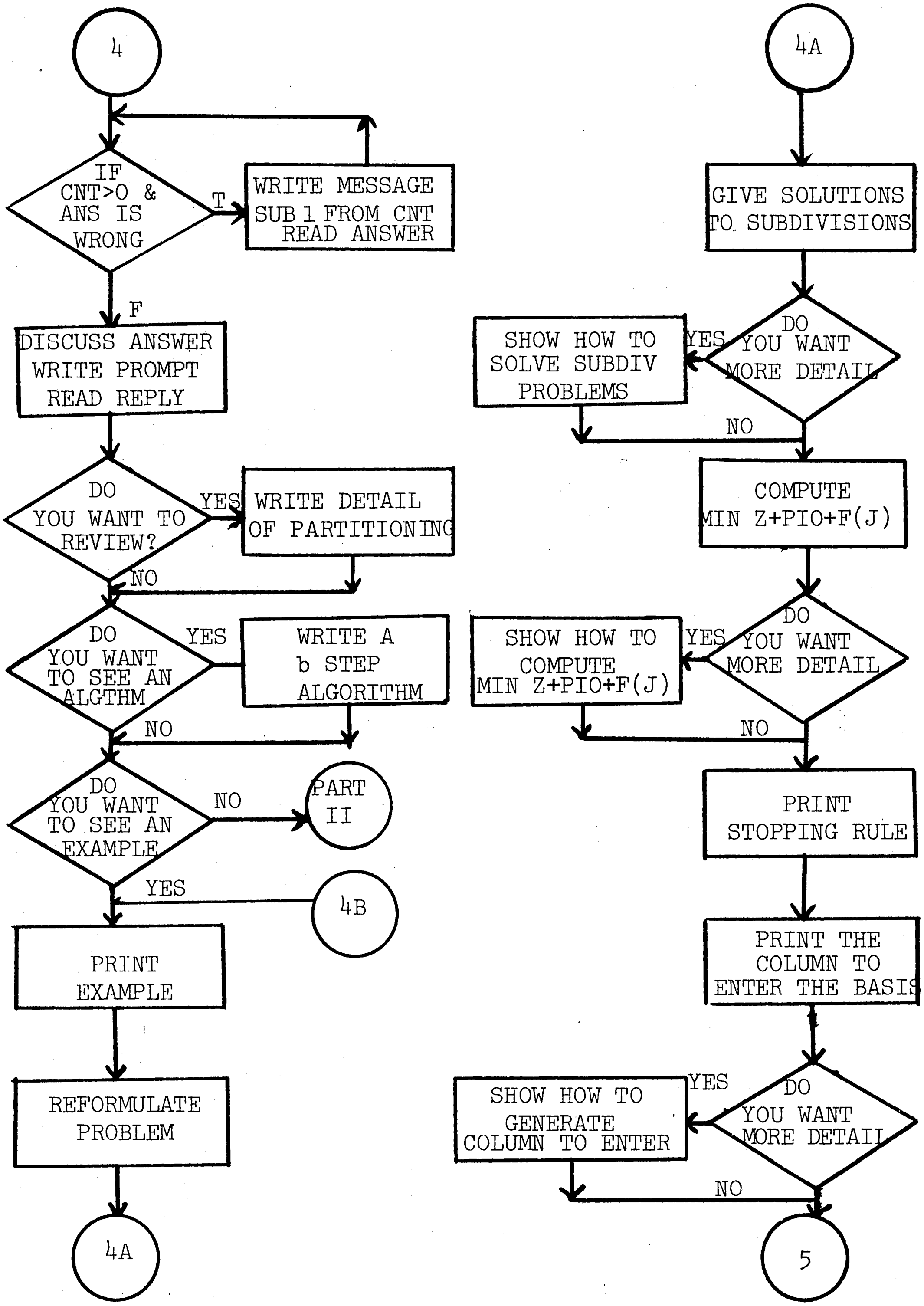
APPENDIX D

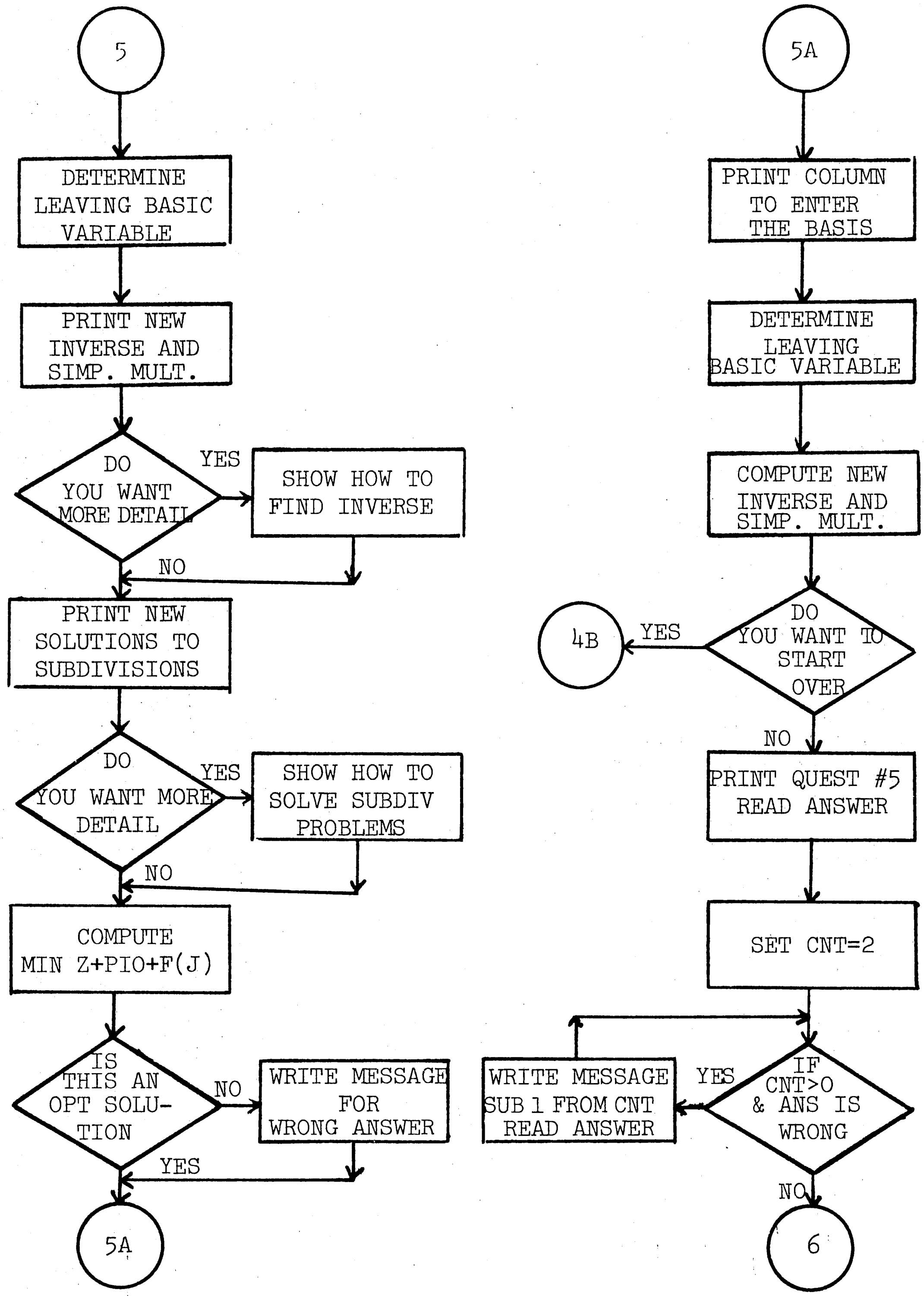
LOGIC BLOCK DIAGRAMS

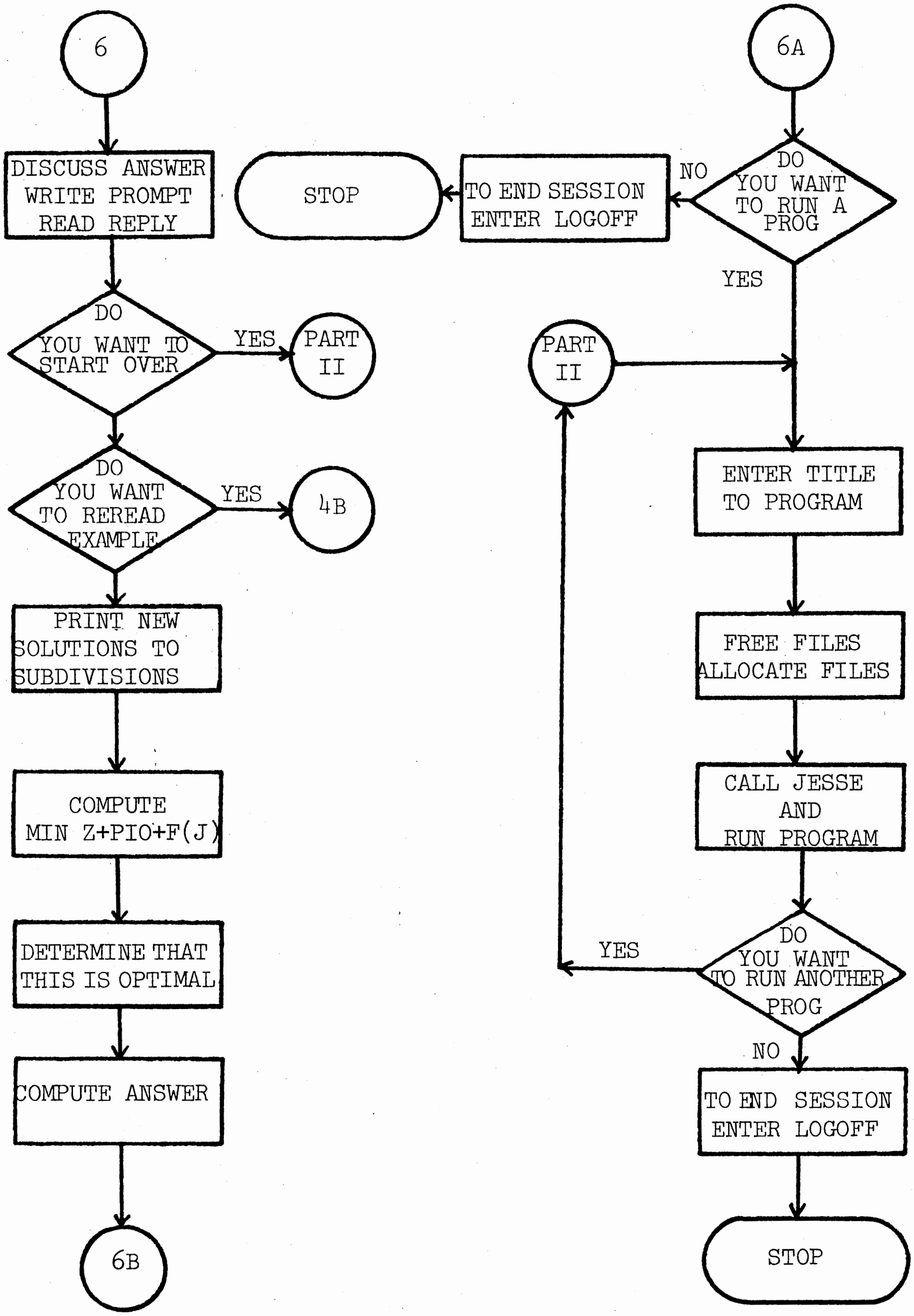

DECOMP
WRITE EXECUTE INSTRUCTIONS
READ REPLY
WRITE INSTRUCTION
READ REPLY
IF REPLY=YES OR NO
F
WRITE INVALID ANSWER
T
WRITE PROMPT READ USER'S NAME
IF REPLY=NO
F
1
T
PART II
1
OUTLINE TEXT
DESCRIBE THE DECOMPOSITION TECHNIQUE
READ REPLY
IF REPLY=STOP
YES
PART II
NO
1A
EXPLAIN ANGULAR STRUCTURE OF CONSTRAINTS
WRITE QUEST #1 READ ANSWER
2


2
SET CNT=2
IF CNT>0 & ANS IS WRONG
T
WRITE MESSAGE SUB 1 FROM CNT READ ANSWER
F
IF CNT=0
T
DO YOU WANT TO REVIEW?
YES
1A
NO
F
DISCUSS ANS. WRITE PROMPT READ REPLY
IF REPLY=STOP
T
PART II
F
BASIC APPROACH TO DECOMPOSITION READ REPLY
IF REPLY=STOP
T
PART II
F
2A
2A
WRITE QUEST #2 READ ANSWER
SET CNT=2
IF CNT>0 & ANS IS WRONG
T
WRITE MESSAGE SUB 1 FROM CNT READ ANSWER
F
DISCUSS ANSWER WRITE PROMPT READ REPLY
IF REPLY=STOP
T
PART II
F
PRINT DEFINITION OF PROBLEM
3


3
WRITE PROMPT READ REPLY
IF REPLY=YES
T
DETAIL THE REFORMULATION
F
IF REPLY=STOP
T
PART II
F
PRINT QUEST #3 READ ANSWER
SET CNT=2
IF CNT>0 & ANS IS WRONG
T
WRITE MESSAGE SUB 1 FROM CNT READ ANSWER
F
DISCUSS ANSWER WRITE PROMPT READ REPLY
3A
3A
DO YOU WANT TO REVIEW?
Yes
DETAIL THE REFORMULATION
No
PARTITIONING OF SIMPLEX MULTIPLIERS
WRITE PROMPT READ REPLY
IF REPLY=YES
T
DETAIL THE PARTITIONING
F
IF REPLY=STOP
T
PART II
F
WRITE QUEST #4 READ ANSWER
SET CNT=2
4


4
IF CNT>0 & ANS IS WRONG
T
WRITE MESSAGE SUB 1 FROM CNT READ ANSWER
F
DISCUSS ANSWER WRITE PROMPT READ REPLY
DO YOU WANT TO REVIEW?
YES
WRITE DETAIL OF PARTITIONING
NO
DO YOU WANT TO SEE AN ALGTHM
YES
WRITE A b STEP ALGORITHM
NO
DO YOU WANT TO SEE AN EXAMPLE
NO
PART II
YES
4B
PRINT EXAMPLE
REFORMULATE PROBLEM
4A
4A
GIVE SOLUTIONS TO SUBDIVISIONS
DO YOU WANT MORE DETAIL
YES
SHOW HOW TO SOLVE SUBDIV PROBLEMS
NO
COMPUTE MIN Z+PIO+F(J)
DO YOU WANT MORE DETAIL
YES
SHOW HOW TO COMPUTE MIN Z+PIO+F(J)
NO
PRINT STOPPING RULE
PRINT THE COLUMN TO ENTER THE BASIS
DO YOU WANT MORE DETAIL
YES
SHOW HOW TO GENERATE COLUMN TO ENTER
NO
5


5
DETERMINE LEAVING BASIC VARIABLE
PRINT NEW INVERSE AND SIMP. MULT.
DO YOU WANT MORE DETAIL
YES
SHOW HOW TO FIND INVERSE
NO
PRINT NEW SOLUTIONS TO SUBDIVISIONS
DO YOU WANT MORE DETAIL
YES
SHOW HOW TO SOLVE SUBDIV PROBLEMS
NO
COMPUTE MIN Z+PIO+F(J)
IS THIS AN OPT SOLU- TION
NO
WRITE MESSAGE FOR WRONG ANSWER
YES
5A
5A
PRINT COLUMN TO ENTER THE BASIS
DETERMINE LEAVING BASIC VARIABLE
COMPUTE NEW INVERSE AND SIMP. MULT.
DO YOU WANT TO START OVER
YES
4B
NO
PRINT QUEST #5 READ ANSWER
SET CNT=2
IF CNT>0 & ANS IS WRONG
YES
WRITE MESSAGE SUB 1 FROM CNT READ ANSWER
NO
6


6
DISCUSS ANSWER
WRITE PROMPT
READ REPLY
DO YOU WANT TO START OVER
YES
PART II
DO YOU WANT TO REREAD EXAMPLE
YES
4B
PRINT NEW SOLUTIONS TO SUBDIVISIONS
COMPUTE MIN Z+PIO+F(J)
DETERMINE THAT THIS IS OPTIMAL
COMPUTE ANSWER
6B
6A
DO YOU WANT TO RUN A PROG
NO
TO END SESSION ENTER LOGOFF
STOP
YES
PART II
ENTER TITLE TO PROGRAM
FREE FILES ALLOCATE FILES
CALL JESSE AND RUN PROGRAM
DO YOU WANT TO RUN ANOTHER PROG
YES
NO
TO END SESSION ENTER LOGOFF
STOP

APPENDIX E

LISTING OF THE TUTORIAL TEXT

PAGE 1

A TSC PRESENTATION OF THE
DECOMPOSITION TECHNIQUE
OF LINEAR PROGRAMMING

THIS PRESENTATION IS DESIGNED TO GIVE THE ADVANCED STUDENT A BETTER UNDERSTANDING OF DECOMPOSITION. IT IS DIVIDED INTO TWO PARTS.

PART 1. A TUTORIAL TEXT THAT TAKES THE STUDENT THROUGH THE DEVELOPMENT OF DECOMPOSITION. IT IS ASSUMED THE STUDENT HAS A THOROUGH UNDERSTANDING OF LP AND REVISED SIMPLEX.

PART 2. AN EXECUTABLE PROGRAM THAT LETS YOU ENTER YOUR OWN DATA TO BE RUN AND GIVES YOU INTERMEDIATE RESULTS TO ALLOW YOU TO MONITOR ITS PROGRESS.

IF YOU WOULD LIKE TO STEP THROUGH PART 1 ENTER YES.

IF YOU WANT TO RUN DATA ENTER NO.

PAGE 2

DECOMPOSITION

THIS IS A DEVELOPMENT OF THE DECOMPOSITION TECHNIQUE OF LINEAR PROGRAMMING. IT IS ASSUMED THE STUDENT'S BACKGROUND INCLUDES A THOROUGH UNDERSTANDING OF LINEAR PROGRAMMING AND REVISED SIMPLEX. THE TEXT WILL COVER:

1. MULTIDIVISIONAL PROBLEMS
2. THEIR ANGULAR STRUCTURE
3. THE DECOMPOSITION APPROACH - THEORY
4. A DECOMPOSITION ALGORITHM
5. AN EXAMPLE

EVERY SO OFTEN A QUESTION WILL BE ASKED OF YOU. TYPE IN THE ANSWER AND PRESS ENTER.
IF AT ANYTIME YOU WANT TO TERMINATE PART 1 AND GO TO PART 2 TYPE IN STOP AND PRESS ENTER.
(PRESS CLEAR AND ENTER TO CONTINUE)

PAGE 3

DECOMPOSITION IS A TECHNIQUE USED FOR SOLVING PROBLEMS HAVING A SPECIAL STRUCTURE. THESE PROBLEMS ARE CALLED MULTIDIVISIONAL AND THEIR NAME HINTS AT THE TYPE OF STRUCTURE USED, MULTIDIVISIONAL. HENCE, THEY ARE PROBLEMS THAT ENCOMPASS SEVERAL DIVISIONS. THEREFORE, THE PROBLEMS ARE ALMOST DECOMPOSABLE INTO SEPARATE PROBLEMS, WHERE EACH DIVISION IS CONCERNED ONLY WITH OPTIMIZING IT'S OWN OPERATION. HOWEVER, SOME OVERALL COORDINATION IS REQUIRED IN ORDER TO BEST DIVIDE CERTAIN ORGANIZATIONAL RESOURCES AMONG THE DIVISIONS.

IF YOU WERE TO LOOK AT A TABLE OF CONSTRAINT COEFFICIENTS FOR THIS TYPE OF PROBLEM YOU WOULD FIND THAT THE CONSTRAINTS FOR EACH DIVISION COULD BE GROUPED TOGETHER IN A BLOCK FORMING AN ANGULAR STRUCTURE.

THE NEXT PAGE EXPLAINS THE ANGULAR STRUCTURE OF MULTIDIVISIONAL PROBLEMS AND GIVES AN EXAMPLE.
(PRESS CLEAR AND ENTER TO CONTINUE OR TYPE STOP TO TERMINATE)

PAGE 4

TABLE OF CONSTRAINT COEFFICIENTS FOR MULTIDIVISIONAL PROBLEMS.

```
 ______________
|              |          CORPORATE CONSTRAINTS ON
|______________|          ORGANIZATIONAL RESOURCES
 _
|_|                       CONSTRAINTS ON RESOURCES AVAILABLE ONLY
                                  DIVISION 1
     _
    |_| .                         DIVISION 2
         .
          . _
           |_|                    LAST DIVISION
```

EACH SMALLER BLOCK CONTAINS THE COEFFICIENTS OF THE CONSTRAINTS FOR ONE DIVISION. THE LONG BLOCK AT THE TOP CONTAINS THE COEFFICIENTS OF THE CORPORATE CONSTRAINTS FOR THE MASTER PROBLEM (THE PROBLEM OF COORDINATING THE ACTIVITIES OF THE DIVISIONS).

PAGE 5

QUESTION #1 :
WHAT TYPE OF SPECIAL PROBLEM WAS THE DECOMPOSITION METHOD DEVELOPED FOR?

PAGE 6

THE CORRECT ANSWER IS MULTIDIVISIONAL

THOSE PROBLEMS WHERE THE MAJORITY OF THE CONSTRAINTS CAN BE SEPARATED INTO GROUPS ACCORDING TO THE RESOURCES AVAILABLE.

TO LEARN HOW THE DECOMPOSITION METHOD SOLVES THESE SPECIAL STRUCTURED PROBLEMS PRESS ENTER TO GO TO THE NEXT PAGE.

(OR TYPE STOP TO TERMINATE)

PAGE 7

THE BASIC APPROACH IS TO REFORMULATE THE PROBLEM IN A WAY THAT GREATLY REDUCES THE NUMBER OF FUNCTIONAL CONSTRAINTS AND THEN TO APPLY THE REVISED SIMPLEX. THIS VERSION OF THE SIMPLEX METHOD CAN BE THOUGHT OF AS HAVING EACH DIVISION SOLVE ITS OWN SUBPROBLEM AND SENDING ITS PROPOSAL TO THE MASTER PROBLEM.

IF THESE PROPOSALS VIOLATE THE CORPORATE CONSTRAINTS THE DECOMPOSITION TECHNIQUE WILL EVALUATE THAT VIOLATION AND CALCULATE PENALTIES FOR EACH OF THE DIVISIONS IN ORDER TO FORCE THEIR SOLUTIONS TOWARD A CORPORATE OPTIMUM. IN THIS WAY WE CAN COORDINATE THE PROPOSALS FROM ALL THE DIVISIONS TO FIND THE OPTIMAL SOLUTION FOR THE OVERALL ORGANIZATION.

PRESS CLEAR AND ENTER FOR QUESTION #2 OR STOP TO TERMINATE.

PAGE 8

QUESTION #2:

YOU ARE IN CHARGE OF BUDGETING A LARGE CORPORATION AND EACH PLANT MANAGER SENDS YOU PROPOSED BUDGET REQUIREMENTS FOR HIS PLANT. BUT, AS IS USUALLY THE CASE, YOU CANNOT MEET ALL THE REQUIREMENTS. AS BUDGETING DIRECTOR YOUR NEXT STEP IS TO:

A. DETERMINE YOURSELF WHAT THE PLANT BUDGETS SHOULD BE.
B. CALCULATE SOME KIND OF PENALTY FOR EACH PLANT TO FORCE THEM TO COME UP WITH AN AGREEABLE PROPOSAL.
C. TEAR UP THE PROPOSALS AND HAVE THEM START OVER.
D. RUN THE CORPORATE BUDGET AS A WHOLE USING REVISED SIMPLEX. THEN SEND EACH PLANT ITS BUDGET.

PAGE 9

LET'S DEFINE A PROBLEM WITH N DIVISIONS AS SUCH:

```
                     N
MAXIMIZE            SUM  (C(I)*X(I))
                    I=1

SUBJECT TO:         A(1)  A(2) ... A(N)      X(0)        B(0)
                    A(N+1)                   X(1)        B(1)
                          A(N+2)             X(2)        B(2)
                                .             .    =      .
                                 .            .           .
                                  .           .           .
                                   A(2N)     X(N)        B(N)
```

WHERE THE B,C,X'S ARE VECTORS AND A'S ARE MATRICES.
CONSIDER THE SOLUTION SPACE FOR DIVISION K; CALL IT S(K). ANY POINT IN S(K) CAN BE REPRESENTED AS A WEIGHTED AVG. OF THE EXTREME POINTS OF S(K).
LET X(J,K) = EP(J) OF DIVISION K AND L(J,K) IT'S WEIGHT.
I.E. ANY FEASIBLE POINT X(*,K) = SUM ON J OF (L(J,K)*X(J,K)) FOR SOME COMBINATION OF THE L(J,K) SUCH THAT 0 <= L(J,K) <= 1 AND THE SUM ON J OF ALL L(J,K) IS EQUAL TO 1.

PAGE 10

THIS EQUATION FOR X(*,K) AND THE CONSTRAINTS ON THE L(J,K) PROVIDE A METHOD FOR REPRESENTING THE FEASIBLE SOLUTIONS TO DIVISION K WITHOUT USING ANY OF THE ORIGINAL CONSTRAINTS. HENCE THE OVERALL PROBLEM CAN NOW BE REFORMULATED WITH FAR FEWER CONSTRAINTS AS

```
                 N
MAXIMIZE        SUM  SUM  L(J,K)(C(K)*X(J,K))
                K=1   J

SUBJECT TO:
                 N
                SUM  SUM  L(J,K)(A(K)*X(J,K))
                K=1   J

          AND   SUM  L(J,K) = 1
                 J
```

STUDY THIS REFORMULATION OF THE MASTER PROBLEM FOR AWHILE. THE SYMBOLISM MIGHT BE CONFUSING. THE FIRST SUMMATION (ON K) REFERS TO THE DIVISIONS. THE SECOND SUMMATION (ON J) REFERS TO THE EXTREME POINTS WITHIN EACH DIVISION.

PAGE 11

QUESTION #3 :
IN THE REFORMULATION OF THE MASTER PROBLEM
WHAT DO THE L(J,K)'S STAND FOR ?

A. CONSTRAINT COEFFICIENTS
B. SIMPLEX MULTIPLIERS
C. EXTREME POINTS IN THE SOLUTION
D. RESPECTIVE WEIGHTS ON THE EXTREME POINTS

PAGE 12

SINCE THIS REFORMULATION HAS FAR FEWER CONSTRAINTS IT SHOULD BE SOLVABLE WITH MUCH LESS COMPUTATIONAL EFFORT. AT FIRST GLANCE IT WOULD SEEM THAT ALL THE EXTREME POINTS (X(J,K)) NEED BE IDENTIFIED. A TEDIOUS TASK TO SAY THE LEAST. FORTUNATELY, IT IS NOT NECESSARY TO DO THIS WHEN USING THE REVISED SIMPLEX METHOD. ALL THAT IS REQUIRED IS THAT THE SIMPLEX MULTIPLIERS (PI) BE PARTITIONED SO THAT YOU CALCULATE ONLY WHAT IS NEEDED.

PAGE 11

QUESTION #3 :
IN THE REFORMULATION OF THE MASTER PROBLEM
WHAT DO THE L(J,K)'S STAND FOR ?

A. CONSTRAINT COEFFICIENTS
B. SIMPLEX MULTIPLIERS
C. EXTREME POINTS IN THE SOLUTION
D. RESPECTIVE WEIGHTS ON THE EXTREME POINTS

PAGE 12

SINCE THIS REFORMULATION HAS FAR FEWER CONSTRAINTS IT SHOULD BE SOLVABLE WITH MUCH LESS COMPUTATIONAL EFFORT. AT FIRST GLANCE IT WOULD SEEM THAT ALL THE EXTREME POINTS (X(J,K)) NEED BE IDENTIFIED. A TEDIOUS TASK TO SAY THE LEAST. FORTUNATELY, IT IS NOT NECESSARY TO DO THIS WHEN USING THE REVISED SIMPLEX METHOD. ALL THAT IS REQUIRED IS THAT THE SIMPLEX MULTIPLIERS (PI) BE PARTITIONED SO THAT YOU CALCULATE ONLY WHAT IS NEEDED.

PAGE 11

QUESTION #3 :
IN THE REFORMULATION OF THE MASTER PROBLEM
WHAT DO THE L(J,K)'S STAND FOR ?

A. CONSTRAINT COEFFICIENTS
B. SIMPLEX MULTIPLIERS
C. EXTREME POINTS IN THE SOLUTION
D. RESPECTIVE WEIGHTS ON THE EXTREME POINTS

PAGE 12

SINCE THIS REFORMULATION HAS FAR FEWER CONSTRAINTS IT SHOULD BE SOLVABLE WITH MUCH LESS COMPUTATIONAL EFFORT. AT FIRST GLANCE IT WOULD SEEM THAT ALL THE EXTREME POINTS (X(J,K)) NEED BE IDENTIFIED. A TEDIOUS TASK TO SAY THE LEAST. FORTUNATELY, IT IS NOT NECESSARY TO DO THIS WHEN USING THE REVISED SIMPLEX METHOD. ALL THAT IS REQUIRED IS THAT THE SIMPLEX MULTIPLIERS (PI) BE PARTITIONED SO THAT YOU CALCULATE ONLY WHAT IS NEEDED.

PAGE 17

FOR AN EXAMPLE, CONSIDER THIS PROBLEM WITH 2 DIVISIONS

```
MAXIMIZE  Z = 4X(1) + 6X(2) + 8Y(1) + 5Y(2)
  S.T.
                X(1) + 3X(2) + 2Y(1) + 4Y(2) <= 20
               2X(1) + 3X(2) + 6Y(1) + 4Y(2) <= 25
                X(1) +  X(2)                 <=  5
                X(1) + 2X(2)                 <=  8
                               4Y(1) + 3Y(2) <= 12
    AND   X(J),Y(J) >= 0

A(1)=| 1 3 |   A(2)=| 2 4 |   A(3)=| 1 1 |   A(4)=| 4 3 |
     | 2 3 |        | 6 4 |        | 1 2 |

C(1)=| 4 6 |   C(2)=| 8 5 |   B(0)=| 20 |   B(1)=| 5 |   B(2)=| 12 |
                                   | 25 |        | 8 |
AND X = X(1),X(2) AND Y = Y(1),Y(2)
```

COPY THE ABOVE DOWN FOR FUTURE REFERENCE

PAGE 18

THE REFORMULATED MASTER PROBLEM REQUIRES ONLY 4 CONSTRAINTS
2 FOR THE CORPORATE CONSTRAINTS AND 1 CONSTRAINT FOR EACH
DIVISION THAT REQUIRES THE SUM OF THE WEIGHTS ADD UP TO 1.
(ON A LARGE PROBLEM THIS WOULD BE A SIGNIFICANT SAVINGS)

FOR THE INITIAL BASIC FEASIBLE SOLUTION :

```
    | 1 0 0 0 |                | 20 |
B = | 0 1 0 0 | = BI     B' =  | 25 |     CB = (0,0,0,0)
    | 0 0 1 0 |                |  1 |
    | 0 0 0 1 |                |  1 |
```

WHERE B' IS THE RHS OF THE REFORMULATED MASTER PROBLEM.

PAGE 19

```
STEP 1.     USING THE SIMPLEX MULTIPLIERS PI1 SOLVE THE DIVISION PROBLEMS
       REMEMBER PI = CB * BI
       INITIALLY CB=(0,0,0,0) & BI=I=B, SO PI=(0,0,0,0) & PI1=(0,0)
       THE SOLUTICNS ARE :  X(1) = 2 , X(2) = 3 , AND Z(1) = -26
                            Y(1) = 3 , Y(2) = 0 , AND Z(2) = -24
    SOLVE DIVISION #1 :
       MIN Z(1) = (PI1 * A(1) - C(1))X      | OR  MIN  (-4,-6)X
       S.T.                                 |     S.T.
                  A(3)X <= B(1)             |          | 1 1 | X(1) <= 5
                                            |          | 1 2 | X(2) <= 8
       THE SOLUTICN IS X(1)=2, X(2)=3 AND Z(1)=-26

    SOLVE DIVISION #2 :
       MIN Z(2) = (PI1 * A(2) - C(2))Y      | OR  MIN  (-8,-5)Y
       S.T.                                 |     S.T.
                  A(4)Y <= B(2)             |          | 4 3 | Y <= 12

       THE SOLUTICN IS Y(1)=3, Y(2)=0 AND Z(2)=-24
```

PAGE 20

```
STEP 2.     FIND THE MINIMUM OF Z(N) + PI0 = F
            REMEMBER PI0 = CB * COLUMN(NLC + N) CF BI
            N BEING THE NUMBER OF THE DIVISICN.
            THEREFCRE PI0 DIFFERS ACCORDING TO THE DIVISION.
       F = MIN = -26 THEREFORE THE WEIGHTS(PENALTY) ON
        E.P. (2,3) OF DIVISION 1 ENTERS THE BASIS
    SOLVE:
       Z(1) + PI0 = -26 + 0 = -26

       Z(2) + PI0 = -24 + 0 = -24

STEP 3.     STOPPING RULE. IF F IS >= 0 IT IS AN OPTIMAL SOLUTION. STOP.

       F = -26 THEREFORE WE MUST CONTINUE.
```

```
PAGE 21


                                                                            |11|
STEP 4.     GENERATE THE COLUMN TO ENTER THE BASIS AS :  A' = |13|
                                                                            | 1|
                                                                            | 0|
              | A(1)*E.P.|      | 1 3 | | 2 |     | 11 |
      A' =    |----------|  =   | 2 3 |*| 3 |  =  | 13 |  =  A'
              |    I     |              1         |  1 |
                                        0         |  0 |

STEP 5.     DETERMINE THE LEAVING BASIC VARIABLE.  PROCEED IN THE USUAL
            WAY TO CALCULATE THE CURRENT COEFFICIENTS AND THE RHS.

                  | 11 |                 | 20 |   THE RATIOS ARE :
   BI * A'   =    | 13 |  ,  BI * B' =   | 25 |   20/11 , 25/13 , 1/1
                  |  1 |                 |  1 |
                  |  0 |                 |  1 |

   THE MINIMUM RATIO IS 1 (THE THIRD ROW). R = 3.
   THUS THE NEW VALUES OF CB ARE (0,0,26,0)
   THE EXTREME POINTS IN THE BASIS ARE : (_,_) (_,_) (2,3) (_,_)


PAGE 22


STEP 6.     OBTAIN A NEW BASIS INVERSE AND NEW SIMPLEX MULTIPLIERS.
                                                  | 1 0 |
   PI1 = CB * BI(1;2) = (0,0,26,0) *              | 0 1 | = (0,0) = PI1
                                                  | 0 0 |
                                                  | 0 0 |

   BI' = E * BI WHERE E IS AN IDENTITY MATRIX EXCEPT THAT IT'S KTH
   COLUMN IS REPLACED BY THE VECTOR M WHERE
                                                          | 1 0 -11 0 |
        | -A'(I,K)/A'(R,K),IF I¬=R |                      | 0 1 -13 0 |
   M =  |                          | THEREFORE E =        | 0 0   1 0 |
        | 1/A'(R,K),IF I=R         |                      | 0 0   0 1 |

   SO          | 1 0 -11 0 |
       BI' =   | 0 1 -13 0 |
               | 0 0   1 0 |
               | 0 0   0 1 |
```

```
PAGE 23



*** ITERATION 1

STEP 1.  RESOLVE THE DIVISION PROBLEMS.
   SOLUTION FOR DIVISION #1 :  X(1) = 2     X(2) = 3      Z(1) = -26
   SOLUTION FOR DIVISION #2 :  Y(1) = 3     Y(2) = 0      Z(2) = -24

   COMPUTE THE OBJ. COEFFICIENTS FOR DIVISION 1
          (0 0) * |1 3| - (4,6)  =  (-4 -6)
                  |2 3|

   COMPUTE THE OBJ. COEFFICIENTS FOR DIVISION 2
          (0 0) * |2 4| - (8,5)  =  (-8 -5)
                  |6 4|




PAGE 24



STEP 2.
                                          |-11|
   PIO = CB * BI(3) = (0,0,26,0) * |-13| = 26
                                          | 1 |
   Z(1) + 26  =  0                        | 0 |

                                          |0|
   PIO = CB * BI(4) = (0,0,26,0) * |0| = 0
                                          |0|
   Z(2) + 0  = -24                        |1|

   F = -24  THERFCRE WEIGHTS CN E.P. (3,0) OF DIVISION 2 ENTERS BASIS


STEP 3.    STOPPING RULE.

   F = -24
```

PAGE 25

```
STEP 4.
         |A(2)  * E.P.|     |2 4|    |3|    | 6|
   A' =  |------------|  =  |6 4| *  |0|  = |18|
         |      I     |           0         | 0|
                                  1         | 1|

STEP 5.
                |1 0 -11 0|    | 6|    | 6|
   BI  * A' =   |0 1 -13 0| *  |18| =  |18|
                |0 0   1 0|    | 0|    | 0|     THE RATIOS ARE :
                |0 0   0 1|    | 1|    | 1|
                                                9/6 , 12/18 , 1/0 , 1/1
                |1 0 -11 0|    |20|    | 9|
   BI  * B' =   |0 1 -13 0| *  |25| =  |12|
                |0 0   1 0|    | 1|    | 1|
                |0 0   0 1|    | 1|    | 1|

   THE MINIMUM RATIO IS 12/18. R = 2.  CB = (0,24,26,0)
   THE EXTREME POINTS IN THE BASIS ARE : (_,_) (3,0) (2,3) (_,_)
```

PAGE 26

```
STEP 6:
   B INVERSE   |1 -1/3   -20/3  0|          PI1 = (0,4/3)
               |0  1/18 -13/18  0|
               |0   0       1   0|
               |0 -1/18  13/18  1|
```

PAGE 27

```
QUESTION #5 :
    IF                   |1 -1/3   -20/3  0|
       B INVERSE =       |0  1/18 -13/18  0|   PI1 = (0,4/3)
                         |0   0      1    0|
                         |0 -1/18  13/18  1|

          |1 3|              |2 4|
    A(1) = |2 3|    A(2) = |6 4|    C(1) = (4 6) AND C(2) = (8 5)

    COMPUTE THE COEFFICIENTS OF THE OBJECTIVE FUNCTION
      FOR DIVISION 1 AND DIVISION 2 RESPECTIVELY.

    A.   (0,0) AND (0,0)
    B.   (-4,-6) AND (-8,-5)
    C.   (-4/3,-2) AND (0,1/3)
    D.   (-16/3,-1) AND (4,-2/3)
```

PAGE 28

```
*** ITERATION 2

STEP 1.
   DIVISION 1 OBJ. COEFFICIENTS
          (0,4/3) * |1 3| - (4,6)    =   (-4/3,-2)
                    |2 3|

   SOLUTION :  X(1) = 2   X(2) = 3    Z(1) = -26/3


   DIVISION 2 OBJ. COEFFICIENTS
          (0,4/3) * |2 4| - (8,5)    =   (0,1/3)
                    |6 4|

   SOLUTION :  Y(1) = 0   Y(2) = 0    Z(2) = 0
```

```
PAGE 29


STEP 2.
                               |-20/3  |
   PI0 = (0,24,26,0) *  |-13/18|   =  26/3
                               |    1    |
                               | 13/18|

   Z(1) + 26/3   =   0
                               |0|
   PI0 = (0,24,26,0) *  |0| = 0
                               |0|
                               |1|

   Z(2) + 0   =   0

  F = 0

STEP 3.     STOPPING RULE .

   F IS >= 0   THIS IS AN OPTIMAL SOLUTION.



  PAGE 30


     THE EXTREME POINTS IN THE BASIS ARE : (_,_) (3,0) (2,3) (_,_)
        THE WEIGHTS ON THESE POINTS ARE :  5 , 2/3 , 1 , 1/3
            COMPUTED BY B INVERSE * ORIGINAL RHS

  X = SUM  L(J) * X(J) = 1 * (2,3) = (2,3) = X(1),X(2)

  Y = SUM  L(J) * Y(J) = 2/3 * (3,0) = (2,0) = Y(1),Y(2)

  THUS, AN OPTIMAL SOLUTION FOR THIS PROBLEM IS
            X(1) = 2 , X(2) = 3
            Y(1) = 2 , Y(2) = 0

  Z = 4*2 + 6*3 + 8*2 + 5*0 = 42
```

APPENDIX F

LISTING OF THE CONTROL PROGRAM

```
                  /*    GENERAL INSTRUCTIONS    */
CONTROL PROMPT MAIN
WRITE THIS PROGRAM IS DESIGNED TO OPERATE ON ANY TSO TERMINAL.
WRITE IT IS INTERACTIVE, MEANING THE USER WILL BE PROMPTED FOR A
WRITE RESPONSE.  A DECWRITER IS PREFERRED SINCE YOU CAN MAINTAIN
WRITE A HARDCOPY OF THE SESSION AND REFER TO IT AT ANY TIME.
WRITE HOWEVER, DECSCOPES AND IBM 3277'S CAN ALSO BE USED.
WRITE THE OPERATION OF A DECSCOPE AND DECWRITER IS SLIGHTLY
WRITE DIFFERENT THAN A 3277. IF YOU ARE USING A DECSCOPE OR
WRITE DECWRITER, AFTER TYPING A RESPONSE PRESS THE RETURN KEY.
WRITE HOWEVER, WITH THE 3277 YOU MUST CLEAR THE SCREEN FIRST
WRITE THEN ENTER YOUR RESPONSE.  THE INSTRUCTIONS DURING A
WRITE SESSION ASSUME YOU ARE USING AN IBM 3277.
                  /*    BEGINNING OF PROGRAM    */
PART1: +
WRITE    IF YOU ARE USING A 3277 OR ANYTHING SIMILAR ENTER - CRT
READ &TERM
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND ENTER
  READ &REPLY
END
L 'TSO.U16300A.PAGE.DATA' 10 190 SNUM
READ &ANS
DO WHILE (&ANS ¬= YES) AND (&ANS ¬= NO)
  WRITE INVALID ANSWER &ANS - REENTER
  READ &ANS
END
IF &TERM = CRT THEN WRITENR PRESS CLEAR THEN
WRITE TYPE IN YOUR NAME AND HIT ENTER.
                  /*    READ STUDENT'S NAME    */
READ &NAME
IF &ANS = NO THEN GOTO PART2
                  /*    INTRODUCTION    */
L 'TSO.U16300A.PAGE.DATA' 250 430 SNUM
READ &ANS
LBL3: +
L 'TSO.U16300A.PAGE.DATA' 470 650 SNUM
READ &ANS
IF (&ANS = STOP) THEN GOTO LBL1
                  /*    ANGULAR STRUCTURE    */
L 'TSO.U16300A.PAGE.DATA' 690 880 SNUM
READ &ANS
                  /*    PRINT QUESTION #1    */
L 'TSO.U16300A.PAGE.DATA' 990 1050 SNUM
WRITENR ANS =
READ &ANS
SET &CNT = 2
DO WHILE (&CNT > 0) AND (&ANS ¬= MULTIDIVISIONAL)
```

```
  WRITE WRONG, TRY AGAIN &NAME.
  WRITE AND WATCH FOR SPELLING OR TRY A SIMILAR WORD
  WRITENR ANS =
  READ &ANS
  SET &CNT = &CNT - 1
END

               /*   GIVE A CHANCE TO REREAD PREVIOUS PAGE   */

IF &ANS = MULTIDIVISIONAL THEN WRITE VERY GOOD &NAME
IF &CNT = 0 THEN +
DO
  WRITE WOULD YOU LIKE TO REREAD THE PREVIOUS PAGE &NAME?
  READ &ANS
  IF &ANS = YES THEN +
  DO
    IF &TERM ¬= CRT THEN GOTO LBL3
    WRITENR PRESS CLEAR AND ENTER.
    READ &REPLY
    GOTO LBL3
  END
END
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
END

              /*   ANSWER TO QUESTION #1   */

L 'TSO.U16300A.PAGE.DATA' 1140 1320 SNUM
READ &ANS
IF (&ANS = STOP) THEN GOTO LBL1

              /*   INTRODUCTION TO REFORMULATION   */

SET &CNT = 2
L 'TSO.U16300A.PAGE.DATA' 1340 1500 SNUM
READ &ANS
IF (&ANS = STOP) THEN GOTO LBL1

               /*   PRINT QUESTION #2   */

L 'TSO.U16300A.PAGE.DATA' 1580 1730 SNUM
WRITENR ANS =
READ &ANS
DO WHILE (&ANS ¬= B) AND (&CNT > 0)
  WRITE SORRY &NAME, &ANS IS AN INCORRECT ANSWER.
  WRITE TRY AGAIN, YOU HAVE &CNT MORE CHANCES.
  IF &TERM = CRT THEN +
  DO
    WRITENR PRESS CLEAR AND HIT ENTER.
    READ &REPLY
    L 'TSO.U16300A.PAGE.DATA' 1580 1730 SNUM
  END
  WRITENR ANS =
  SET &CNT = &CNT - 1
  READ &ANS
END
```

```
                 /*   ANSWER TO QUESTION #2   */

IF &ANS = B THEN WRITE VERY GOOD &NAME.
WRITE THE CORRECT ANSWER IS B.
WRITE YOU WOULD EVALUATE THE VIOLATIONS & CALCULATE PENALTIES.
WRITE BUT HOW?
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &ANS
IF (&ANS = STOP) THEN GOTO LBL1

                 /*   DEFINE A GENERAL PROBLEM   */

L 'TSO.U16300A.PAGE.DATA' 1770 1970 SNUM
WRITE IF YOU WOULD LIKE TO SEE THE PROBLEM REFORMULATED
WRITE BY USING THESE CONSTRAINTS ENTER YES.
READ &ANS
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
END

                 /*   REFORMULATION IN MORE DETAIL   */

IF (&ANS = YES) THEN +
DO
  L 'TSO.U16300A.PAGE.DATA' 2000 2200 SNUM
  READ &ANS
END
IF (&ANS = STOP) THEN GOTO LBL1

                 /*   PRINT QUESTION #3   */

L 'TSO.U16300A.PAGE.DATA' 2290 2400 SNUM
WRITENR ANS =
READ &ANS
SET &CNT = 2
DO WHILE (&CNT > 0) AND (&ANS ¬= D)
  WRITE SORRY &NAME, &ANS IS AN INCORRECT ANSWER
  WRITE TRY AGAIN, YOU HAVE &CNT MORE CHANCES.
  IF &TERM = CRT THEN +
  DO
    WRITENR PRESS CLEAR AND HIT ENTER.
    READ &REPLY
    L 'TSO.U16300A.PAGE.DATA' 2290 2400 SNUM
  END
  WRITENR ANS =
  READ &ANS
  SET &CNT = &CNT - 1
END

                 /*   ANSWER TO QUESTION #3   */

IF &ANS = D THEN WRITE VERY GOOD &NAME
WRITE THE CORRECT ANSWER IS D
IF (&TERM = CRT) THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
```

```
END

          /*   GIVE A CHANCE TO REREAD THE PREVIOUS PAGE   */

IF (&CNT = 0) AND (&ANS ¬= D) THEN +
DO
  WRITE WOULD YOU LIKE TO REREAD THE PREVIOUS PAGE?
  READ &ANS
  IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
  WRITENR HIT ENTER.
  READ &REPLY
  IF &ANS = YES THEN L 'TSO.U16300A.PAGE.DATA' 2000 2200 SNUM
  READ &REPLY
END

               /*   PARTITIONING OF SIMPLEX MULTIPLIERS   */

L 'TSO.U16300A.PAGE.DATA' 2480 2660 SNUM
WRITENR DO YOU WANT TO LEARN HOW THIS IS DONE IN MORE DETAIL?
READ &ANS

               /*   PARTITIONING  N MORE DETAIL   */

IF (&ANS = YES) THEN +
DO
  L 'TSO.U16300A.PAGE.DATA' 2670 2860 SNUM
  WRITENR PRESS ENTER TO CONTINUE OR TYPE STOP TO TERMINATE.
  READ &ANS
END
IF (&ANS = STOP) THEN GOTO LBL1

               /*   PRINT QUESTION #4   */

L 'TSO.U16300A.PAGE.DATA' 2930 3040 SNUM
WRITENR ANS =
READ &ANS
SET &CNT = 2
DO WHILE (&CNT > 0) AND (&ANS ¬= A)
  WRITE SORRY &NAME, &ANS IS AN INCORRECT ANSWER.
  WRITE TRY AGAIN, YOU HAVE &CNT MORE CHANCES.
  IF &TERM = CRT THEN +
  DO
    WRITENR PRESS CLEAR AND ENTER.
    READ &REPLY
    L 'TSO.U16300A.PAGE.DATA' 2930 3040 SNUM
  END
  WRITENR ANS =
  READ &ANS
  SET &CNT = &CNT - 1
END

               /*   ANSWER TO QUESTION #4   */

IF &ANS = A THEN WRITE VERY GOOD &NAME
WRITE THE CORRECT ANSWER IS A.
WRITE B IS AN M*M MATRIX,
WRITE BUT TO CALCULATE PI1 AND PI0 YOU NEED ONLY
WRITE NLC+1 COLUMNS OF B.

          /*   GIVE A CHANCE TO REREAD THE PREVIOUS PAGE   */
```

```
IF (&CNT = 0) AND (&ANS ¬= A) THEN +
DO
  WRITE WOULD YOU LIKE TO REREAD THE PREVIOUS PAGE?
  READ &ANS
  IF (&ANS = YES) THEN +
  DO
    IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
    WRITENR HIT ENTER.
    READ &REPLY
    L 'TSO.U16300A.PAGE.DATA' 2670 2870 SNUM
    READ &REPLY
  END
END
WRITENR WOULD YOU LIKE TO SEE A SIMPLE ALGORITHM AND EXAMPLE?
READ &ANS
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
END
IF (&ANS ¬= YES) THEN GOTO LBL1

              /*   BEGINNING OF ALGORITHM   */

L 'TSO.U16300A.PAGE.DATA' 3110 3300 SNUM
READ &ANS
L 'TSO.U16300A.PAGE.DATA' 3330 3520 SNUM
WRITENR WOULD YOU LIKE TO SEE AN EXAMPLE OF THIS ALGORITHM?
READ &ANS
IF &ANS = NO THEN GOTO LBL1

              /*   PRINT THE EXAMPLE   */

LBL2: +
L 'TSO.U16300A.PAGE.DATA' 3550 3740 SNUM
WRITENR PRESS ENTER.
READ &REPLY
L 'TSO.U16300A.PAGE.DATA' 3810 3940 SNUM

                      /*   INITIALIZE   */

IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                      /*   ** ITERATION 0    */
                      /*   STEP 1            */

L 'TSO.U16300A.PAGE.DATA' 3990 4040 SNUM
WRITENR DO YOU WANT TO SEE HOW THE SOLUTION IS COMPUTED?
READ &ANS
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
END

                      /*   STEP 1 IN MORE DETAIL   */
```

```
IF &ANS = YES THEN +
DO
  L 'TSO.U16300A.PAGE.DATA' 4050 4170 SNUM
  READ &ANS
END

                        /*   STEP 2    */

L 'TSO.U16300A.PAGE.DATA' 4220 4280 SNUM
WRITENR DO YOU NEED HELP?
READ &ANS

                        /*   STEP 2 IN MORE DETAIL    */

IF (&ANS = YES) THEN L 'TSO.U16300A.PAGE.DATA' 4290 4330 SNUM

                        /*   STEP 3    */

L 'TSO.U16300A.PAGE.DATA' 4330 4360 SNUM
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                        /*   STEP 4    */

L 'TSO.U16300A.PAGE.DATA' 4430 4460 SNUM
WRITENR IF YOU NEED HELP TO GENERATE THE COLUMN ENTER YES.
READ &ANS

                        /*   STEP 4 IN MORE DETAIL    */

IF (&ANS = YES) THEN L 'TSO.U16300A.PAGE.DATA' 4470 4500 SNUM
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                        /*   STEP 5    */

L 'TSO.U16300A.PAGE.DATA' 4520 4620 SNUM
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                        /*   STEP 6    */

L 'TSO.U16300A.PAGE.DATA' 4660 4700 SNUM
WRITE THERE ARE MANY WAYS TO FIND AN INVERSE.
WRITENR WOULD YOU LIKE TO SEE AN EASY ONE?
READ &ANS

                        /*   EASY WAY TO FIND AN INVERSE    */

IF (&ANS = YES) THEN L 'TSO.U16300A.PAGE.DATA' 4710 4820 SNUM
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                        /*   ** ITERATION 1    */
                        /*   STEP 1            */

```

```
L 'TSO.U16300A.PAGE.DATA' 4890 4930 SNUM
WRITENR DO YOU NEED MORE INFORMATION?
READ &ANS

                         /*   STEP 1 IN MORE DETAIL   */

IF (&ANS = YES) THEN L 'TSO.U16300A.PAGE.DATA' 4950 5020 SNUM
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                         /*   STEPS 2,3   */

L 'TSO.U16300A.PAGE.DATA' 5090 5280 SNUM
WRITENR DO WE STOP OR CONTINUE?
READ &ANS
IF (&ANS = STOP) THEN WRITE NO.  -24 < 0. WE MUST CONTINUE.
IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
WRITENR HIT ENTER.
READ &REPLY

                         /*   STEPS 4,5   */

L 'TSO.U16300A.PAGE.DATA' 5300 5510 SNUM
READ &REPLY

                         /*   STEP 6   */

L 'TSO.U16300A.PAGE.DATA' 5540 5600 SNUM
WRITE WORK THIS YOURSELF TO BE SURE YOU KNOW HOW IT IS DONE.
WRITE DO YOU WANT TO START OVER AT THE BEGINNING OF THE EXAMPLE?
READ &ANS

                    /*   AT THIS POINT YOU CAN START OVER   */

IF (&ANS = YES) THEN +
DO
  IF &TERM = CRT THEN +
  DO
    WRITENR PRESS CLEAR AND HIT ENTER.
    READ &REPLY
  END
  GOTO LBL2
END
IF &TERM = CRT THEN +
DO
  WRITENR PRESS CLEAR AND HIT ENTER.
  READ &REPLY
END

                         /*   PRINT QUESTION #5   */

L 'TSO.U16300A.PAGE.DATA' 5740 5900 SNUM
WRITENR ANS =
READ &ANS
SET &CNT = 2
DO WHILE (&CNT > 0) AND (&ANS ¬= C)
  WRITE SORRY &NAME, &ANS IS AN INCORRECT ANSWER.
  WRITE TRY AGAIN, YOU HAVE &CNT MORE CHANCES.
  IF &TERM = CRT THEN +
```

```
    DO
      WRITENR PRESS CLEAR AND HIT ENTER.
      READ &REPLY
      L 'TSO.U16300A.PAGE.DATA' 5740 5900 SNUM
    END
    WRITENR ANS =
    READ &ANS
    SET &CNT = &CNT - 1
  END

                          /*   ANSWER TO QUESTION #5   */

  IF &ANS = C THEN WRITE VERY GOOD &NAME.
  WRITE THE CORRECT ANSWER IS C.
  WRITE  PI1 * A(I) - C(I)

          /*   GIVE A CHANCE TO REREAD THE PREVIOUS PAGE   */

  IF &CNT = 0 THEN +
  DO
    WRITE DO YOU THINK YOU SHOULD START OVER AT THE BEGINNING?
    READ &ANS
    IF &ANS = YES THEN GOTO PART1
    WRITE DON'T YOU THINK YOU SHOULD AT LEAST REREAD THE EXAMPLE?
    READ &ANS
    IF &ANS = YES THEN GOTO LBL2
    WRITE OK, LET'S CONTINUE.  WE'RE ALMOST DONE.
  END
  IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
  WRITENR HIT ENTER.
  READ &REPLY

                          /*   ** ITERATION 2   */
                          /*   STEP 1           */

  L 'TSO.U16300A.PAGE.DATA' 5980 6140 SNUM
  IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
  WRITENR HIT ENTER.
  READ &REPLY

                          /*   STEPS 2,3  OPTIMAL SOLUTION   */

  L 'TSO.U16300A.PAGE.DATA' 6180 6370 SNUM
  IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
  WRITENR HIT ENTER.
  READ &REPLY

                          /*   COMPUTE THE ANSWER   */

  L 'TSO.U16300A.PAGE.DATA' 6400 6580 SNUM
  IF &TERM = CRT THEN WRITENR PRESS CLEAR AND
  WRITENR HIT ENTER.
  READ &REPLY

                          /*   END OF EXAMPLE AND PART1   */

  LBL1: +
  WRITE IF YOU HAVE DATA YOU WANT TO RUN AS A PROGRAM ENTER YES.
  READ &ANS
  IF (&ANS ¬= YES) THEN GOTO FIN
```

```
  /*********************   PART 2 - RUN A PROBLEM   **********/

PART2: +
WRITE ENTER A TITLE TO YOUR PROBLEM
READ &TITLE
FREE FILE(FT05F001,FT06F001)
ALLOC DA(*) FI(FT05F001) SHR
ALLOC DA(*) FI(FT06F001) SHR
LCADGO JESSE.OBJ FORTLIB
WRITE &NAME, IS THERE ANOTHER PROBLEM YOU WANT TO RUN?
READ &ANS
FREE FILE(FT05F001,FT06F001)
IF (&ANS = YES) THEN GOTO PART2
WRITE DO YOU WANT TO GO THROUGH PART1 AGAIN?
READ &ANS
IF (&ANS = YES) THEN GOTO PART1
FIN: +
WRITE TO END THE SESSION ENTER LOGOFF
END
```

VITA - 2

William Arthur Senters

Candidate for the Degree of

Master of Science

Thesis: A TSO PRESENTATION OF A DECOMPOSITION TECHNIQUE FOR SOLVING LARGE-SCALE MULTIDIVISIONAL LINEAR PROGRAMMING PROBLEMS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Midwest City, Oklahoma, July 8, 1948, the son of Mr. and Mrs. Charles D. Senters.

Education: Graduated from Collegio del Espiritu Santo, San Juan, Puerto Rico, in May, 1966; attended Eastern New Mexico University, Portales, New Mexico, from Sept., 1970 to May, 1972; attended Chapman College, Orange, California, from Jan., 1973 to Dec., 1973; received Bachelor of Science degree from Oklahoma State University at Stillwater, Oklahoma, in May, 1975, with a major in Business Administration; completed requirements for a Master of Science degree at Oklahoma State University in May, 1978.

Professional Experience: Member of the United States Air Force from Dec., 1969 to Dec., 1973; graduate teacher of 'Introduction to Data Processing' in the College of Business at Oklahoma State University, Stillwater, Oklahoma from Sept., 1976 to May, 1977. Applications programmer, Texaco, Inc., Houston, Texas, January, 1978 to the present.