T-2634

# AN ALGORITHM FOR THE SOLUTION OF A CLASS OF NONLINEAR RESOURCE ALLOCATION MODELS WHERE LEARNING EFFECTS ARE PRESENT

CLOSED RESERVE

ARTHUR LAKES LIBRARY
COLORADO SCHOOL of MINES
GOLDEN, COLORADO 80401

by

Charles E. Lienert

ProQuest Number: 10796206

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.

ProQuest 10796206

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-2634

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mineral Economics).

Golden, Colorado

Date: April 5, 1982

Signed: Charles E. Lienert
Charles E. Lienert

Approved: Robert E.D. Woolsey
Dr. R.E.D. Woolsey
Thesis Advisor

Golden, Colorado

Date: Apr 5, 1982

C. W. Berry
Dr. Charles W. Berry
Head, Mineral Economics Department

# ABSTRACT

An important problem that managers in mining, petroleum, manufacturing, and service industries must solve is the allocation of scarce resources, such as labor and machinery, to competing ends, such as the optimal product-mix (or service-mix) problem. These problems become considerably more difficult when they must be multitime and multidivisional and where learning effects are present and substantial. Learning effects on optimal resource allocation have been observed in many industries starting with the aircraft and petroleum industries during World War II. Incorporation of learning effects into resource allocation models leads to complex nonlinear models which, although possible to formulate, are mathematically intractable and, in general, unsolvable.

In this dissertation a new algorithm for solving a class of nonlinear models which include learning effects is developed. This algorithm uses simple transformations and bounds which reduce the nonlinear problem to a linear programming problem which is capable of solution using the simplex algorithm. This dissertation contains a computer program written for the DEC 1091 in FORTRAN which allows the user to easily implement this new

algorithm to solve allocation problems with learning effects in the minerals industries.

The new algorithm is applied to a series of test problems of the allocation of resources type with learning effects, and the results relative to other methods of solution are discussed.

TABLE OF CONTENTS

## LIST OF FIGURES

ACKNOWLEDGEMENT

This thesis is dedicated to Dr. Robert E.D. Woolsey, a true Renaissance man. Dr. Woolsey is the ultimate scholar, the ultimate teacher and, above all, the ultimate human being.

I can never begin to adequately express the gratitude that I owe to him. I will be eternally in his debt; when I totally lost faith in myself, he kept me going.

# CHAPTER ONE

# AN INTRODUCTION TO THE PROBLEM AND HISTORICAL SURVEY

## 1.1. The Standard Model and the Learning Model

The general product-mix linear programming model (henceforth referred to as the Standard Model) is of the form:

$$\text{Maximize} \quad Z = \sum_{j=1}^{n} f_j x_j \tag{1-1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m \tag{1-2}$$

$$\text{and} \quad x_j \geq 0, \quad j = 1, 2, \ldots, n \tag{1-3}$$

where $x_j$ = level of activity j ($j = 1, 2, \ldots, n$)

$f_j$ = unit "profit" from activity j

$Z$ = total "profit" from all activities

$a_{ij}$ = amount of resource i consumed by each unit of activity j

$b_i$ = amount of resource i available ($i = 1, 2, \ldots, m$)

This linear programming formulation is based on four assumptions - proportionality, additivity, divisibility, and certainty - as discussed by Hillier and Lieberman (1980). When learning effects are present and pronounced, the assumption of proportionality is violated. The proportionality assumption says that if only one of the n activities

is undertaken, say $x_k$, then in this situation "(1) the measure of effectiveness Z equals $f_k x_k$, and (2) the usage of each resource i equals $a_{ik}\ x_k$; that is, both quantities are directly proportional to the level of each activity k (k = 1, 2, ..., n) conducted by itself." (Hillier and Lieberman, 1980). Thus, the proportionality assumption rules out (1) any startup costs associated with initiating an activity, (2) any economies (or diseconomies) of scale, and (3) any learning effects.

If learning effects are present then we must incorporate two changes into the Standard Model to arrive at a more realistic model.

Change 1. As learning occurs, each unit of the jth activity (or product) produced may require fewer and fewer units of some of the resources. Thus, the technological coefficients $a_{ij}$ in the Standard Model must be modified to the form

$$a^*_{ij} = a_{ij}\ x_j^{\,b_{ij}} \qquad (1\text{-}4)$$

where

$a^*_{ij}$ = marginal rate of technical substitution of resource i for product j reflecting learning,

$a_{ij}$ and $x_j$ are as defined above, and

$b_{ij}$ = learning index of resource i in the production of product j

(Note: $b_{ij} = \log \phi_{ij}/\log 2$ where $\phi_{ij}$ = learning rate of resource i in the production of product j. This formula for $b_{ij}$ will be explained in Section 2 of this chapter.)

From studies in a number of industries (Yelle, 1979), $\phi_{ij}$ will typically be in the range $.70 < \phi_{ij} < 1$, and thus we can assume that $b_{ij}$ will be in the range $-.5 < b_{ij} < 0$.

Change 2. When learning effects are present, $f_j$, the marginal contribution to profit per unit of activity j may no longer be a constant, but may be an increasing function of the level of activity j. Thus, in the objective function, $f_j$ must be replaced by

$$f_j^* = P_j - V_{oj} - \sum_{i=1}^{m} V_i \, a_{ij}^* \qquad (1\text{-}5)$$

where

$P_j$ = the unit selling price of product j

$V_{oj}$ = the total unit variable cost not subject to learning required for each unit of product j

$V_i$ = the unit variable cost of resource i which is subject to learning effects and $a^*_{ij}$ is defined by equation (1-4).

Incorporating the learning effects as given by equations (1-4) and (1-5) into the Standard Model (equations (1-1) through (1-3)) and assuming that only the first g resources are subject to learning effects we can now write the general Learning Model as:

Maximize

$$Z = \sum_{j=1}^{n} \left[ (P_j - V_{oj})\, x_j - \sum_{i=1}^{g} V_i\, a_{ij}\, x_j^{1+b_{ij}} \right] \quad (1\text{-}6)$$

subject to

$$\sum_{j=1}^{n} a_{ij}\, x_j^{1+b_{ij}} \leq b_i, \quad i = 1, 2, \ldots, g \quad (1\text{-}7)$$

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i, \quad i = g+1, g+2, \ldots, m \quad (1\text{-}8)$$

and $x_j \geq 0, \quad j = 1, 2, \ldots, n \quad (1\text{-}9)$

The Learning Model is much more realistic than the Standard Model, but because the $b_{ij}$ are nonzero, whenever learning effects are present, the Learning Model is a non-linear programming problem in which the objective function is

convex and the learning functions in the constraints are concave. The convexity of the objective function can be established by taking the second derivative of a typical term in equation (1-6). For fixed j, the second derivative is given by $-(1+b_{ij})(b_{ij}) \sum_{i=1}^{g} V_i a_{ij} x_j^{b_{ij}-1}$, and, since $b_{ij}$ is usually in the range $-.5 < b_{ij} < 0$, this second derivative is positive for all $x_j$, and hence the term in the objective function is convex. Finally, because any finite sum of convex functions is convex, the objective learning function of the Learning Model (equation (1-6)) is convex. Similarly, it can be shown that the constraints (equation (1-7)) are concave. Thus, the Learning Model involves the maximization of a convex function over a nonconvex set, which at this time is certainly difficult, if not impossible, to solve.

## 1.2 Historical Survey of Learning Curve Theory

The learning curve phenomenon was first observed in manufacturing operations in 1925, but was not reported in literature until 1936 (Hirschmann, 1964). The basic theory of the learning curve is simple: a worker learns as he works and the more often he repeats an operation the more efficient he becomes, with the result that the direct labor per unit declines. What was not immediately recognized was that the rate of change in improvement is regular enough to be predictable.

The first definitive studies of the learning curve phenomenon were done in the aircraft industry, where the following

pattern for aircraft assembly was observed: the fourth plane required only 80 percent as much direct labor as the second; the eighth plane, only 80 percent as much as the fourth; the one hundredth, only 80 percent as much as the fiftieth; and so on. Thus, the amount of labor will reduce by 20 percent as the number of units doubles. This 80 percent learning curve is shown in Figure A. The 80 percent learning curve is a straight line when plotted on a double logarithmic chart, reflecting a constant rate of reduction (see Figure B).

Many geometric versions of the learning curve have been proposed besides the log-linear model, such as the plateau model, the Stanford-B model, the DeJong model, and the S model (Yelle, 1979), but the log-linear model has been, and still is, the most widely used and it will be the basis for the learning curve model in this thesis.

Mathematically, the log-linear learning curve model can be represented by the mathematical function

$$Y = KX^n$$

where

Y = the number of direct labor hours required to produce the Xth unit,

K = the number of direct labor hours required to produce the first unit,

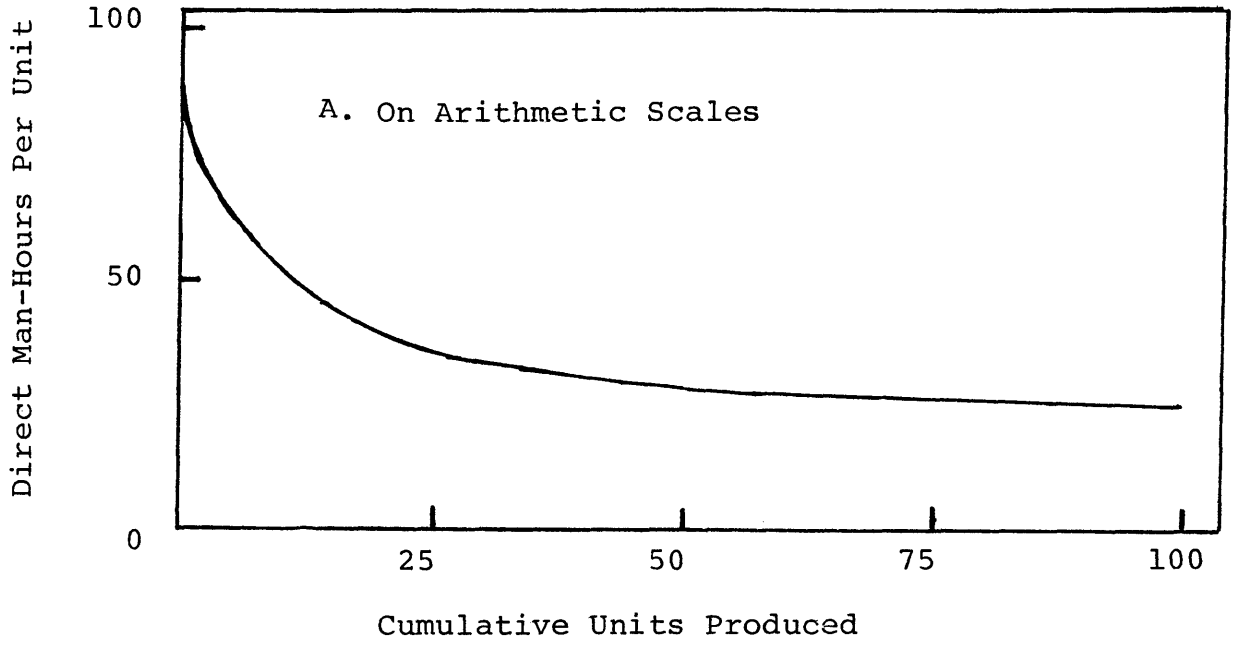
X = the cumulative number of units to be produced,

$n = \log \phi / \log 2$ = the learning index,

$\phi$ = the learning rate,

$1 - \phi$ = the progress ratio.

FIGURE A. THE 80% LEARNING CURVE PLOTTED WITH ARITHMETIC SCALES ON BOTH AXES



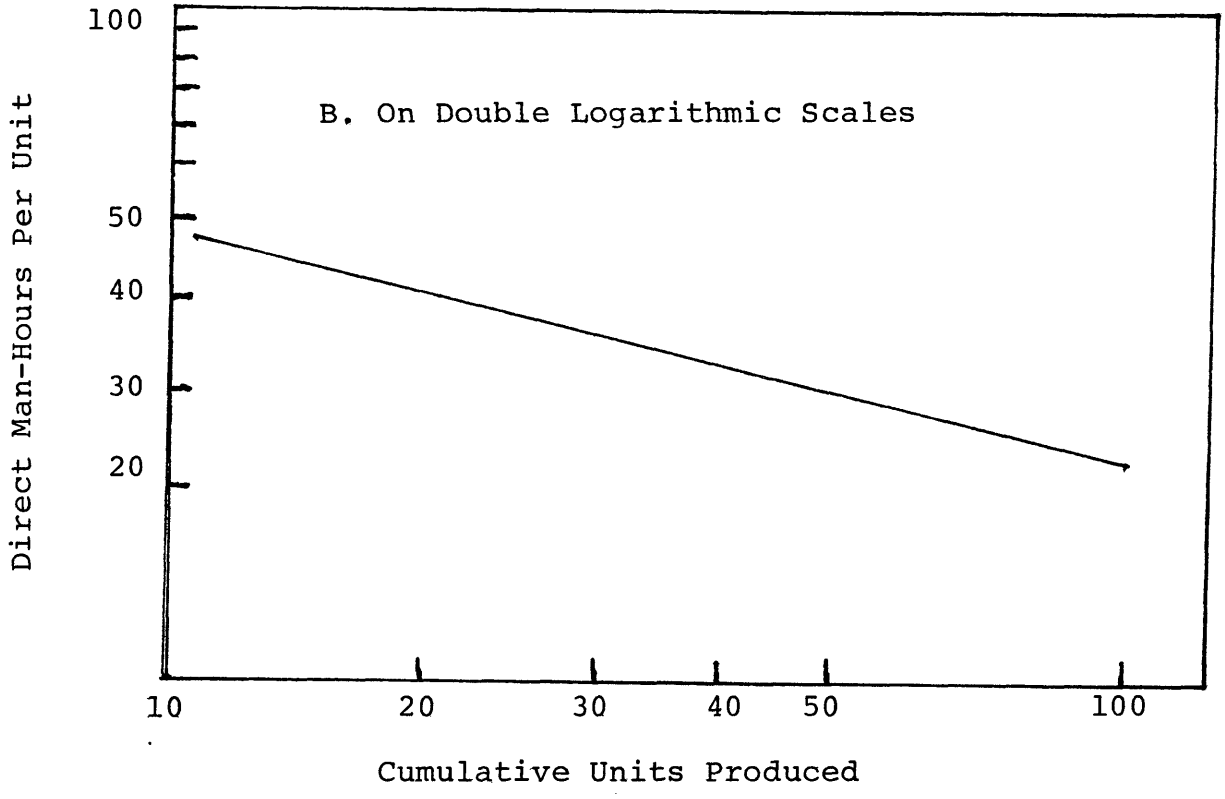



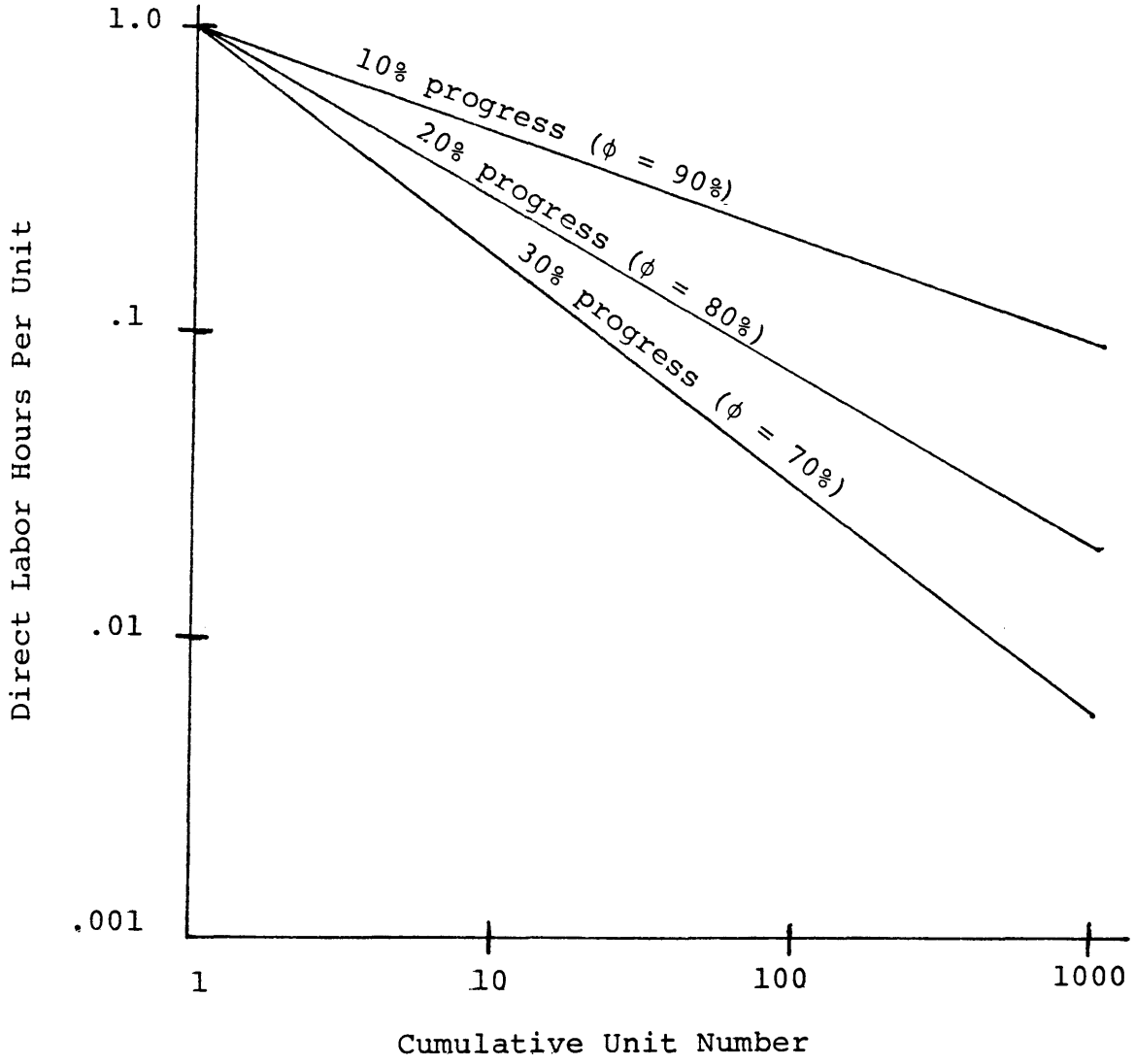

FIGURE B. THE 80% LEARNING CURVE PLOTTED ON A DOUBLE LOGARITHMIC SCALE

Some typical learning curves representing different learning rates are shown in Figure C. The steeper the slope the greater is the learning effect. Thus the 30 percent progress curve in Figure C represents operations with a higher degree of labor content than the 10 percent progress curve. Hirschmann (1964) points out that as the percentage of machine-paced labor increases the progress ratio decreases as is shown below:

| Machine-Paced Labor (as a percent of total labor) | Learning Rate | Progress Ratio |
|---|---|---|
| 25% | 80% | 20% |
| 50% | 85% | 15% |
| 75% | 90% | 10% |

FIGURE C

Typical Learning Curves All Requiring One Direct Labor Hour to Manufacture the First Unit (i.e., K = 1)



Since World War II the learning curve has become a standard management tool in the aircraft industry for forecasting labor inputs and for negotiating prices with the Department of Defense. Other industries which have characteristics similar to the aircraft industry that can profitably benefit from the application of the learning curve theory are electronics, home applicances, residential home construction, and shipbuilding (Andress, 1954).

In the early years of the development of the learning curve, it was widely believed that this theory was only applicable to operations with a high degree of direct labor content; or, in other words, paced by labor rather than by machine. If we take a macroscopic viewpoint, however, we should then include two categories of learning: labor learning and managerial or organizational learning. With this perspective, then, even petroleum refining, one of the least labor-intensive industries, can benefit from the application of learning curve theory. Hirschmann (1964) gives examples of significant learning effects in the operation of fluid catalytic cracking units, e.g., one unit which was one and one-half years old and had achieved 116 percent of design capacity, and another unit which was four years old and had achieved 125 percent of design

capacity. Hirschmann also points out that the learning curve theory is important in other aspects of petroleum refining, e.g., over a period of ten years the time necessary to put a fluid cracking unit on stream dropped to less than half the time initially required.

Although no applications of learning curve theory to the mining industry have been published to date, there are possibilities in such areas as: the scheduling of the operation of draglines and other major types of equipment, the allocation of labor inputs (e.g., miners) in the most efficient manner, the forecasting of output (tons of ore per shift), the scheduling and manpower requirements of maintenance, and the downtime required for maintenance operations. In addition, the learning curve affords a basis for comparing prices and costs and thus for estimating the length of the period of financial drain when expenditures exceed receipts. The main roadblock in applying the learning curve theory to the mining industry is the lack of effort which has been made to measure the necessary learning curve parameters.

Both the range of industries to which the learning curve theory is appropriate and the number of applications within any one firm have broadened as the research on learning curve theory has intensified (see Yelle, 1979,

and references therein). A problem to which the learning curve theory was not considered obviously applicable is the aggregate planning problem. This involves determining production rates and work force sizes for a succession of future time periods such that the organization's operating costs are minimized. Ebert (1976) has shown that learning curve analysis and aggregate planning are inherently related, and that any aggregate planning model must incorporate learning curve phenomena.

Yelle (1979) shows that the learning curve must be considered by the firm in solving such problems as capital budgeting, make-or-buy decisions, bidding on an order, evaluation of supplier quotes, and the determination of the economic order quantity.

# CHAPTER TWO

## A REVIEW OF PREVIOUS WORK

In this chapter three different methods of solving the Learning Model (equations (1-6) through (1-9)) will be discussed using a specific numerical problem as given in Liao (1979). The problem is to find $X_1$, $X_2$, and $X_3$, where $X_1$, $X_2$, $X_3 \geq 0$, such that

Maximize

$$Z(X_1, X_2, X_3) = (353X_1 - 320X_1^{.67807}) + (295X_2 - 256X_2^{.76553}) + (347X_3 - 288X_3^{.848}) \quad (2\text{-}1)$$

subject to

$$G_1(X_1,X_2,X_3) = 10X_1 + 15X_2 + 12X_3 \leq 14{,}400 \quad (2\text{-}2)$$

$$G_2(X_1,X_2,X_3) = 24X_1^{.67807} + 20X_2^{.76553} + 14X_3^{.848} \leq 5{,}920 \quad (2\text{-}3)$$

$$G_3(X_1,X_2,X_3) = 40X_1^{.67807} + 32X_2^{.76553} + 36X_3^{.848} \leq 13{,}026 \quad (2\text{-}4)$$

### 2.1. The Linear Programming Approach

The first approach to solving the problem represented by equations (2-1) through (2-4) is to ignore the learning

effects and instead to solve the Standard Model (equations (1-1) through (1-3) which for equations (2-1) through (2-4) becomes: find $X_1, X_2$, and $X_3$, where $X_1, X_2, X_3 \geq 0$, such that Maximize

$$Z = 33X_1 + 39X_2 + 59X_3 \qquad (2\text{-}5)$$

subject to

$$10X_1 + 15X_2 + 12X_3 \leq 14{,}400 \qquad (2\text{-}6)$$

$$24X_1 + 20X_2 + 14X_3 \leq 5{,}920 \qquad (2\text{-}7)$$

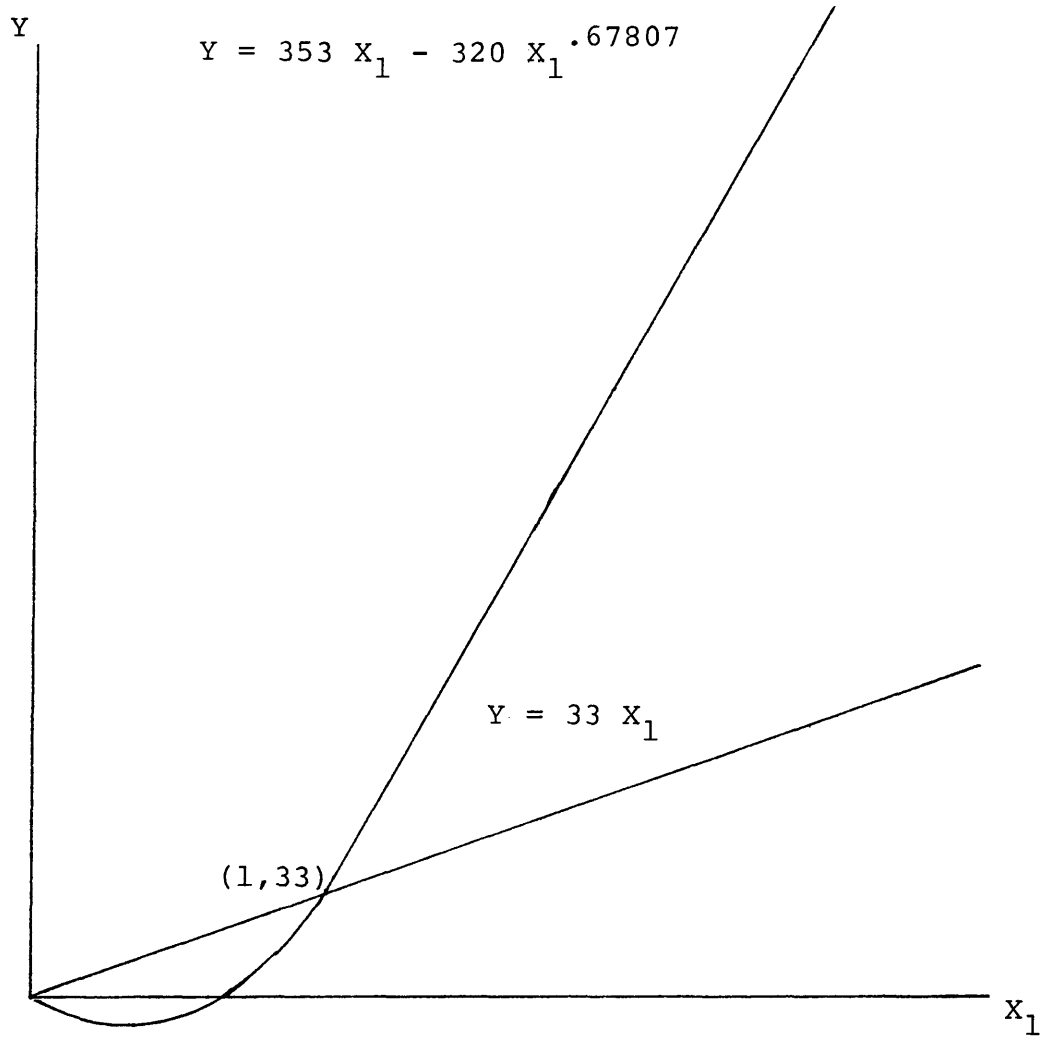
$$40X_1 + 32X_2 + 36X_3 \leq 13{,}026 \qquad (2\text{-}8)$$

Note that to transform equation (2-1) into equation (2-5) we ignore the learning effects and hence replace $X_1^{.67807}$ with $X_1$, and thus the first term in the objective function becomes $(353X_1 - 320X_1) = 33X_1$.

Liao (1979) shows the solution to equations (2-5) through (2-8) is $X_1 = X_2 = 0$, $X_3 = 361$, and $Z^* = 21{,}299$, which is far below the true optimum value of equations (2-1) through (2-4), which in Chapter Four will be shown to be $Z^* = 463{,}985.6$.

The linear programming solution of equations (2-5) through (2-8) gives feasible solutions to the Learning Model (equations (2-1) through (2-4)), but these feasible solutions are far from optimal. We can see why this is so if, for example, we plot the term in the objective function involving $X_1$ in equation (2-5), i.e., $33X_1$, on the same graph with the term involving $X_1$ in the objective function in equation (2-1), i.e., $353X_1 - 320X_1^{.67807}$, (see Figure D).

FIGURE D. PLOT OF $Y = 33\ X_1$ VERSUS $Y = 353\ X_1 - 320\ X_1^{.67807}$



We note that as $X_1$ increases the difference between the linear approximation and the actual objective function term increases. In conclusion, this linear approximation approach to the Learning Model assumes away the basic problem of properly accounting for learning effects. This approach yields consistently feasible but very suboptimal solutions to the Learning Model.

## 2.2. Lagrange Multiplier Approach

The second approach to solving the problem represented by equations (2-1) through (2-4) is the application of the method of Lagrange multipliers as proposed by Liao (1979). As is well known, using the method of Lagrange multipliers, the equations (2-1) through (2-4) are replaced by a new function $F(X_1,X_2,X_3,\lambda_1,\lambda_2,\lambda_3)$ given by

$$\begin{aligned} F(X_1,X_2,X_3,\lambda_1,\lambda_2,\lambda_3) = Z(X_1,X_2,X_3) &+ \lambda_1 G_1(X_1,X_2,X_3) \\ &+ \lambda_2 G_2(X_1,X_2,X_3) \\ &+ \lambda_3 G_3(X_1,X_2,X_3) \end{aligned}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are the Lagrange multipliers. In order to optimize F, F must be partially differentiated with respect to each of the six variables, $X_1$, $X_2$, $X_3$, $\lambda_1$, $\lambda_2$, $\lambda_3$, resulting in six partial derivative equations. Then the six equations are set equal to zero and solved simultaneously to obtain the solutions for $X_1$, $X_2$, $X_3$, $\lambda_1$, $\lambda_2$, $\lambda_3$, and Z.

This approach was shown by Reeves (1980) to lead to

an incorrect solution. Reeves, in the same article, points out that the method of Lagrange multipliers does not guarantee a global or even a local optimum. In addition, a correct application of Lagrange multipliers would require an application of the Kuhn-Tucker conditions, which would necessitate the solution of N simultaneous nonlinear equations in N unknowns. This problem has, at this point in time, no reliable closed-form solution.

## 2.3. The Reeves-Sweigart Approach

The third approach to solving the Learning Model was recently proposed by Reeves and Sweigart (1981). In Reeves' algorithm the Learning Model is replaced by a series of linear approximating models. A linear approximating outer envelope is constructed for each nonlinear term in both the objective function and the constraints subject to learning effects (see equations (1-6) and (1-7)). That is, each nonlinear term of the form $x_j^{1+b_{ij}}$ is replaced by

$$\left[ \frac{\mu_{1j}^{1+b_{ij}} - \ell_{1j}^{1+b_{ij}}}{\mu_{1j} - \ell_{1j}} \right] (x_j - \ell_{1j}) + \ell_{1j}^{1+b_{ij}} \qquad (2\text{-}9)$$

where $\ell_{1j}$ is a lower bound for $x_j$ ($j = 1, 2, \ldots, n$) and $\mu_{1j}$ is the maximum value of $x_j$ determined from all the

constraints (both those subject to and those not subject to learning effects). Equation (2-9) is just a linear equation in $x_j$ which agrees with each original nonlinear term at $\ell_{1j}$ and $\mu_{1j}$. The resulting approximating linear programming problem is solved using the simplex algorithm. As was shown in Reeves and Sweigart (1981), Reeves' algorithm will usually result in superoptimal, but infeasible, solutions to the original Learning Model.

The next step in Reeves' algorithm is to use a branch-and-bound process to approach optimality to the original problem. The difficulty with this portion of the algorithm is that branch-and-bound on a few variables is not expensive in terms of time or cost; but, unfortunately, as the number of variables subject to learning increases, branch-and-bound, although still theoretically possible, rapidly becomes computationally and practically infeasible.

Since the three methods currently available for solving the Learning Model are not suitable or adequate, another algorithm that is not dependent on either the solution of a system of nonlinear equations or the branch-and-bound procedure is required. Such an algorithm will be developed in Chapter Four.

# CHAPTER THREE

## FORMULATION OF THE PROBLEM AS A GEOMETRIC PROGRAM

(On first reading of this thesis, the reader may want to go directly to Chapter Four and return to this chapter at a later time.)

The general form of the Learning Model (equations (1-6) through (1-9)) discussed in Chapter One can be reformulated into the following equivalent form in order to facilitate the explanation of the geometric programming approach to solving the Learning Model:

$$\text{Maximize } Z = \sum_{j=1}^{n} \left[ a_j x_j - b_j x_j^{c_j} \right] \tag{3-1}$$

subject to

$$\sum_{j=1}^{n} d_{ij} x_j^{c_j} \leq r_i, \qquad i = 1, 2, \ldots, g \tag{3-2}$$

$$\sum_{j=1}^{n} e_{ij} x_j \geq r_i, \qquad i = g + 1, g + 2, \ldots, m \tag{3-3}$$

and

$$x_j \geq 0, \qquad j = 1, 2, \ldots, n \tag{3-4}$$

where

$$a_j = P_j - V_{oj}$$

$$b_j = \sum_{i=1}^{g} V_i a_{ij}$$

$c_j = 1 + b_{ij}$ where it is assumed that $b_{1j} = b_{2j} = \ldots = b_{gj}$

$r_i$ = original amount of resource i available (i = 1, 2, ..., m) (equivalent to the $b_i$'s of equations (1-7) and (1-8))

and the $a_{ij}$'s of equations (1-7) and (1-8) are redefined as follows:

$d_{ij}$ = coefficients of decision variables in constraints subject to learning

$e_{ij}$ = coefficients of decision variables in constraints <u>not</u> subject to learning

Equations (3-1) through (3-3) can be reformulated as a condensed geometric programming problem using the techniques discussed by Beightler and Phillips (1976) and Woolsey (1979) as follows:

$$\text{Minimize} \quad \phi = \frac{1}{Z} \tag{3-5}$$

subject to

$$Z \leq \sum_{j=1}^{n} a_j x_j - \sum_{j=1}^{n} b_j x_j^{c_j} \tag{3-6}$$

$$\sum_{j=1}^{n} d_{ij} x_j^{c_j} \leq r_i, \qquad i = 1, 2, \ldots, g \tag{3-7}$$

$$\text{and} \quad \sum_{j=1}^{n} e_{ij} x_j \leq r_i, \qquad i = g + 1, g + 2, \ldots, m \tag{3-8}$$

$$\text{and} \quad x_j \geq 0, \qquad j = 1, 2, \ldots, n \tag{3-9}$$

The first constraint (equation (3-6)) can be rewritten as

$$Z + \sum_{j=1}^{n} b_j x_j^{c_j} \leq \sum_{j=1}^{n} a_j x_j \tag{3-10}$$

The right-hand side of equation (3-10) can be condensed using the generalized arithmetic-geometric mean inequality (Beightler and Phillips, 1976) as follows:

$$\sum_{j=1}^{n} a_j x_j \geq \prod_{j=1}^{n} \left( \frac{a_j x_j}{\delta_j} \right)^{\delta_j} \tag{3-11}$$

where

the $\delta_j$'s are calculated from the formula

$$\delta_j = \frac{a_j \bar{x}_j}{\sum_{j=1}^{n} \bar{x}_j} \quad \text{and}$$

the $\bar{x}_j$ are initial choices of the $x_j$ which give a feasible solution to the constraint equations (3-6) through (3-9), and the $\delta_j$'s must also satisfy the condition

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

Thus, equation (3-10) can be rewritten as

$$Z\left[\prod_{j=1}^{n}\left(\frac{\delta_j}{a_j x_j}\right)^{\delta_j}\right] + \left[\prod_{j=1}^{n}\left(\frac{\delta_j}{a_j x_j}\right)^{\delta_j}\right] \sum_{j=1}^{n} b_j x_j^{c_j} \leq 1 \quad (3\text{-}12)$$

Finally, the condensed geometric programming formulation of the Learning Model (equations (3-1) through (3-4)) can be expressed as follows:

$$\text{minimize} \quad \phi = Z^{-1} \quad (3\text{-}13)$$

subject to

$$Z\left[\prod_{j=1}^{n}\left(\frac{\delta_j}{a_j x_j}\right)^{\delta_j}\right] + \left[\prod_{j=1}^{n}\left(\frac{\delta_j}{a_j x_j}\right)^{\delta_j}\right] \sum_{j=1}^{n} b_j x_j^{c_j} \leq 1 \quad (3\text{-}14)$$

$$\frac{1}{r_i} \sum_{j=1}^{n} d_{ij} x_j^{c_j} \leq 1, \quad i = 1, 2, \ldots, g \quad (3\text{-}15)$$

$$\frac{1}{r_i} \sum_{j=1}^{n} e_{ij} x_j \leq 1, \quad i = g + 1, \ldots, m \quad (3\text{-}16)$$

$$\text{and } x_j \geq 0, \quad j = 1, 2, \ldots, n \quad (3\text{-}17)$$

The condensed geometric programming problem (equations (3-13) through (3-16)) has n + 1 variables ( Z and $x_j$,

j = 1, 2, ..., n) and a total number of terms computed as follows:

| Equation Number | Number of Terms |
|---|---|
| (3-13) | 1 |
| (3-14) | n+1 |
| (3-15) and (3-16) | nm |

Hence the degree of difficulty (D.D.) of the condensed geometric programming problem which is given by the formula

D.D. = number of terms - number of variables - 1

becomes

D.D. = 1 + (n + 1) + mn - (n + 1) - 1 = mn

That is, the degree of difficulty of the condensed geometric programming formulation of the Learning Model will equal mn, i.e., the number of constraints in the problem times the number of decision variables.

Considering problems from the literature such as Liao (1979), (see Chapters Two and Five), Liao's example would have a degree of difficulty of nine. The problem given by Reeves and Sweigart (1981) (see Problem Two in Chapter Five) would have a degree of difficulty of sixteen.

There is, at this time, no reliable method of solution

for geometric programming problems of degree of difficulty greater than one (Woolsey, 1979). Therefore, it can be seen that a geometric programming approach to solving the general learning curve problem would be exceedingly difficult, and hence some other algorithm or reformulation will be necessary to solve the Learning Model. Such an algorithm will be presented in Chapter Four.

# CHAPTER FOUR

## DEFINITION OF THE NOLLE ALGORITHM

The general form of the Learning Model (equations (1-6) through (1-9)) discussed in Chapter One can be reformulated into the following equivalent form in order to facilitate the explanation of the proposed new algorithm: (This reformulation is the same as that given in Chapter Three in equations (3-1) through (3-4) and is reproduced here for the convenience of the reader who may have omitted Chapter Three on first reading.)

$$\text{Maximize} \quad Z = \sum_{j=1}^{n} \left[ a_j X_j - b_j X_j^{C_j} \right] \tag{4-1}$$

subject to

$$\sum_{j=1}^{n} d_{ij} X_j^{C_j} \leq r_i, \quad i = 1, 2, \ldots, g \tag{4-2}$$

$$\sum_{j=1}^{n} e_{ij} X_j \leq r_i, \quad i = g+1, g+2, \ldots, m \tag{4-3}$$

$$\text{and } X_j \geq 0, \qquad j = 1, 2, \ldots, n \tag{4-4}$$

where

$a_j = P_j - V_{oj}$

$b_j = \sum_{i=1}^{g} V_i a_{ij}$

$C_j = 1 + b_{ij}$ where it is assumed that

$b_{1j} = b_{2j} = \ldots = b_{gj}$

$r_i$ = original amount of resource i available ($1 = 1, 2, \ldots, m$) (equivalent to the $b_i$'s of equations (1-7) and (1-8))

and the $a_{ij}$'s of equations (1-7) and (1-8) are redefined as follows:

$d_{ij}$ = coefficients of decision variables in constraints subject to learning

$e_{ij}$ = coefficients of decision variables in constraints not subject to learning

The key idea in the new nonlinear learning effects (NOLLE) algorithm is to replace the linear terms both in the objective function and in the constraints not subject to learning with nonlinear terms of the general form $kX_j^C$ where k is a positive constant.

The procedure is as follows for each $X_j$ (j = 1, 2, ..., n):

Step 1: Find the upper bound on $X_j$ from the linear constraints; call this value $X_{jUBL}$.

Step 2: Find the upper bound on $X_j$ from the nonlinear constraints; call this value $X_{jUBNL}$.

Step 3: Find the least upper bound on $X_j$ from both the linear and nonlinear constraints; call this value $X_{jLUB}$.

Thus

$$X_{jLUB} = \text{minimum } (X_{jUBL}, X_{jUBNL})$$

The terms in the objective function given by equation (4-1) are now transformed as follows:

$$a_j X_j - b_j X_j^{C_j} = \left( a_j X_j^{1-C_j} - b_j \right) X_j^{C_j}$$

and then this term is approximated by

$$\left( a_j X_{jLUB}^{1-C_j} - b_j \right) X_j^{C_j}$$

where the coefficient of $X_j^{C_j}$ is now just a constant.

In a similar manner each term in the linear constraints given by equation (4-3) are transformed as follows:

$$e_{ij} X_j = \left( e_{ij} X_j^{1-C_j} \right) X_j^{C_j}$$

and then this term is approximated by

$$\left( e_{ij} X_{jLUB}^{1-C_j} \right) X_j^{C_j}$$

where, again, the coefficient of $X_j^{C_j}$ is just a constant.

Thus, the learning curve model can now be replaced by the Approximating Learning Curve Model (ALCM):

Maximize

$$Z = \sum_{j=1}^{n} \left( a_j X_{jLUB}^{1-C_j} - b_j \right) X_j^{C_j} \qquad (4\text{-}5)$$

subject to

$$\sum_{j=1}^{n} d_{ij} X_j^{C_j} \leq r_i, \quad i = 1, 2, \ldots, g \qquad (4\text{-}6)$$

$$\sum_{j=1}^{n} \left( e_{ij} \, X_{jLUB}^{1-C_j} \right) X_j^{C_j} \leq r_i, \quad i = g + 1, g + 2, \ldots, m \tag{4-7}$$

$$\text{and} \quad X_j \geq 0, \quad j = 1, 2, \ldots, n \tag{4-8}$$

Now, if we make the change of variables $Y_j = X_j^{C_j}$ in equations (4-5) through (4-8), the ALCM becomes a linear programming problem in the $Y_j$'s, and the resulting model can be solved using the simplex algorithm.

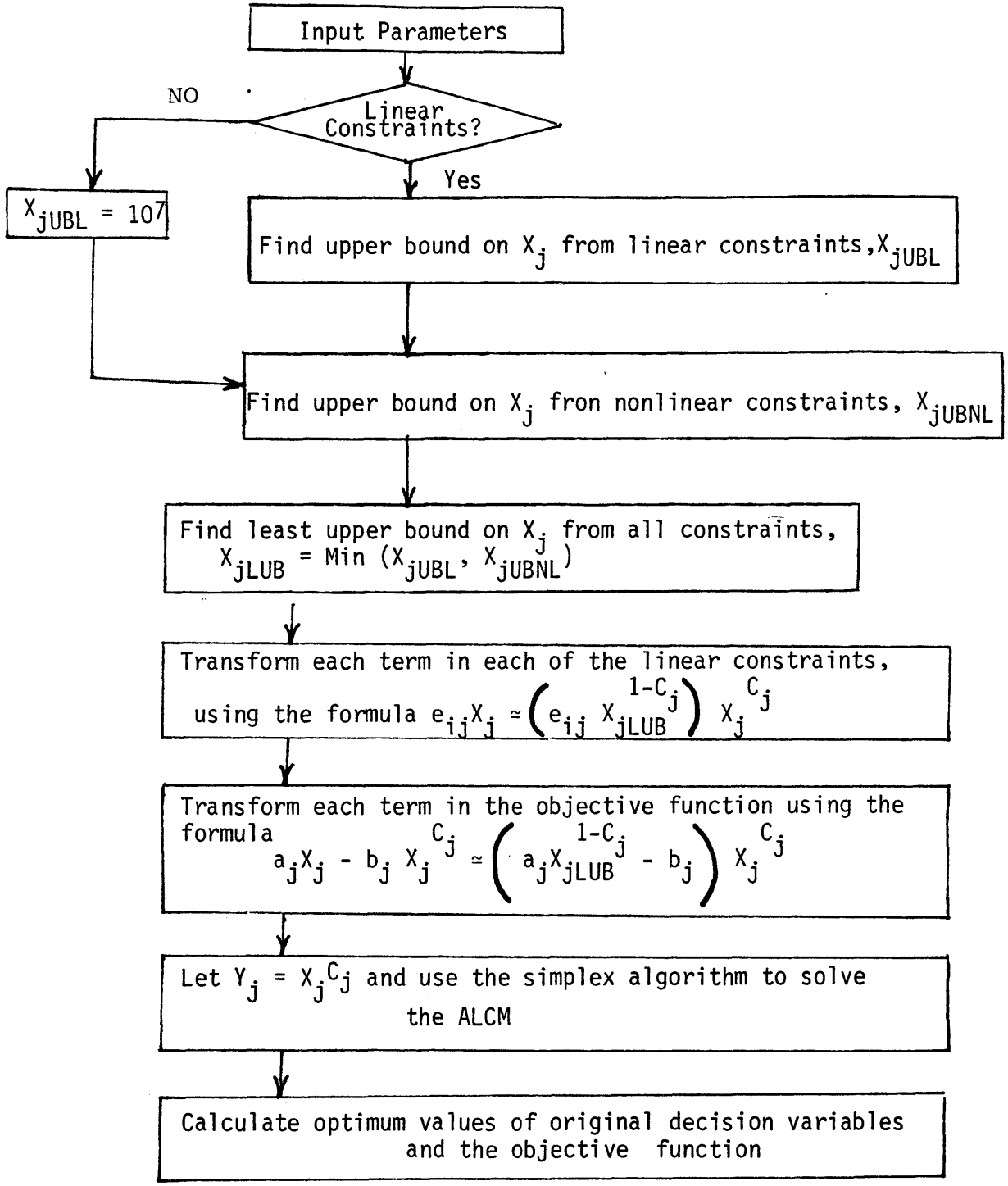
A flow chart for defining the NOLLE algorithm is shown in Figure E.

In conclusion, the NOLLE algorithm has several positive features:

First, the NOLLE algorithm quickly and easily gives feasible solutions to the original learning curve problem. These feasible solutions are either optimal or as close to optimal as would typically be required considering the inaccuracies and variability inherent in the parameter values characterizing the learning effect. As will be shown in Chapter Five, rarely does the NOLLE algorithm require more than three iterations of the simplex algorithm.

Second, the NOLLE algorithm avoids the two main difficulties in Reeves' method which (1) initially gives infeasible solutions and (2) requires a branch-and-bound procedure.

FIGURE E. FLOW CHART FOR THE NOLLE ALGORITHM



Third, the feasible region for the ALCM, which is determined by equations (4-6) through (4-8), is always a subset of the feasible region for the reformulated learning model (see equations (4-2) through (4-4)), and so the NOLLE algorithm will always give a good lower bound to the optimal solution to the reformulated learning problem (equations (4-1) through (4-4)), in those exceptional cases where the algorithm does not, in fact, converge to the optimal solution; and, in addition, at each iteration the NOLLE algorithm yields feasible solutions to the reformulated learning model.

These features of the NOLLE algorithm together with a geometrical comparison of how the feasible region is transformed by the NOLLE algorithm and by Reeves' method can be illustrated by considering a learning curve problem with the following constraints:

$$.157659\ X_1 + X_2 \leq 25 \tag{4-9}$$

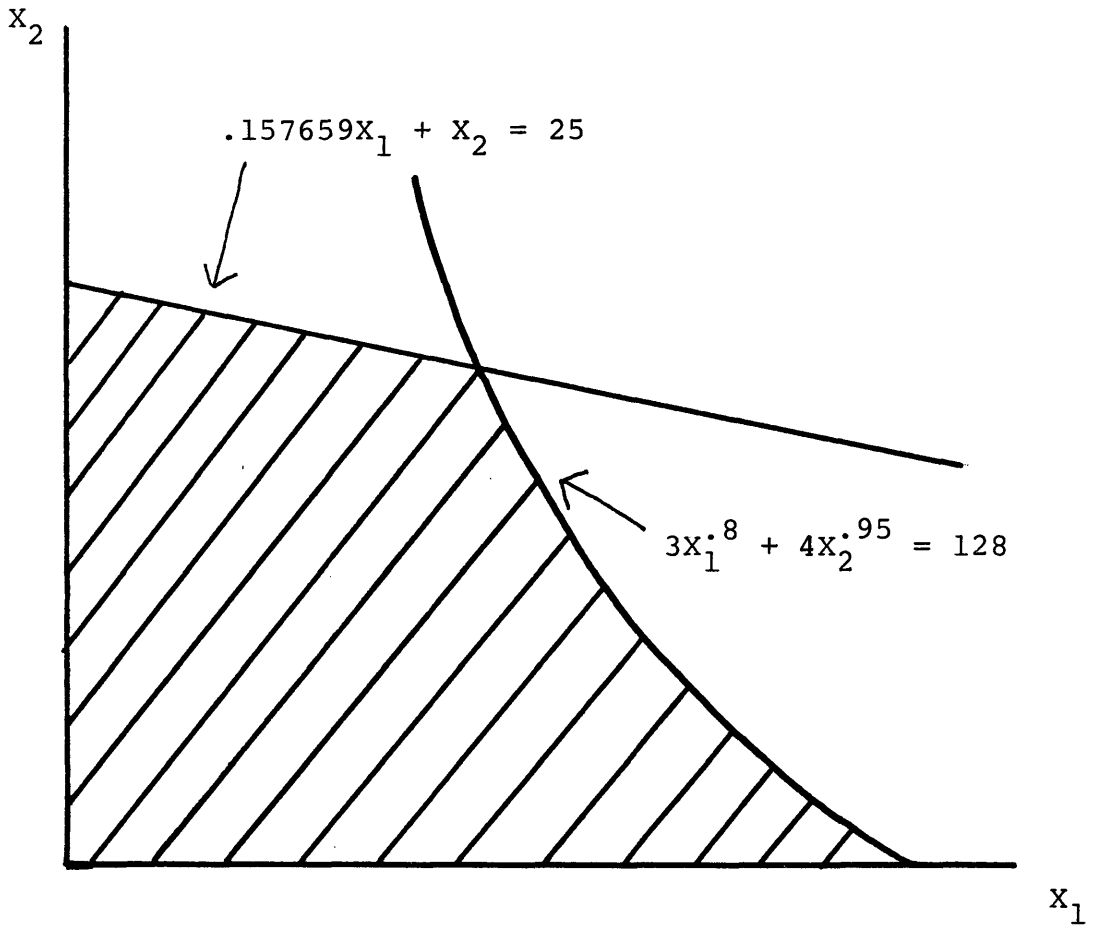
$$3X_1^{.8} + 4\ X_2^{.95} \leq 128 \tag{4-10}$$

$$X_1,\ X_2 \geq 0 \tag{4-11}$$

The graph of the feasible region determined by equations (4-9) through (4-11) is shown in Figure F.

Using the NOLLE algorithm and the notation introduced in this chapter, we have

FIGURE F. GRAPH OF FEASIBLE REGION DETERMINED BY EQUATIONS (4-9) THROUGH (4-11)



$X_{1UBL} = 158.57$, $X_{1UBNL} = 109.046$, and hence $X_{1LUB} = 109.046$. Also, $X_{2UBL} = 25$, $X_{2UBNL} = 38.403$, and hence $X_{2LUB} = 25$. Then, using the NOLLE algorithm, equation (4-9) is replaced by $.157659(109.046)^{1-.8} X_1^{.8} + (25)^{1-.95} X_2^{.95} \leq 25$ or $.4029X_1^{.8} + 1.1746X_2^{.95} \leq 25$.

Thus, the feasible region for the ALCM is given by:

$$.4029X_1^{.8} + 1.1746X_2^{.95} \leq 25 \qquad (4\text{-}12)$$

$$3X_1^{.8} + 4X_2^{.95} \leq 128 \qquad (4\text{-}13)$$

$$X_1, X_2 \geq 0 \qquad (4\text{-}14)$$

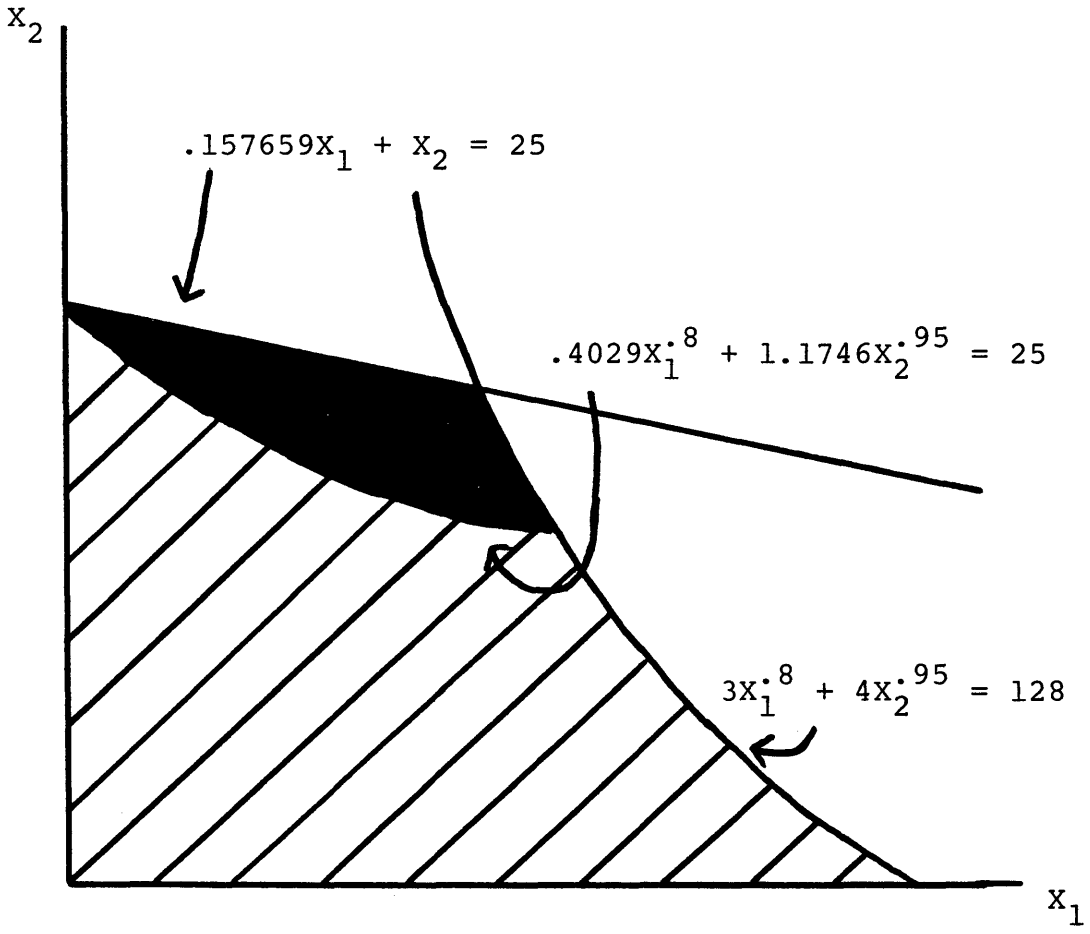
The graph of the feasible region determined by these three equations is shown in Figure G. The shaded region in Figure G is that part of the feasible region shown in Figure F which is not included in the feasible region determined by equations (4-12) through (4-14). Figure F illustrates that the feasible region for the ALCM (equations (4-5) through (4-8)) is a subset of the feasible region for the reformulated learning curve model (equations (4-1) through (4-4)).

Using Reeves' method, the lower and upper bounds on $X_1$ and $X_2$ are:

$$\ell_{11} = 0, \mu_{11} = 109.046$$

$$\ell_{12} = 0, \mu_{12} = 25$$

FIGURE G. GRAPH OF FEASIBLE REGION DETERMINED BY EQUATIONS (4-12) THROUGH (4-14)



Then, using equation (2-9), Reeves' method replaces equation (4-10) with

$$3\left[\frac{(109.046)^{.8}}{109.046}\right]X_1 + 4\left[\frac{(25)^{.95}}{25}\right]X_2 \leq 128$$

or $$1.1738X_1 + 3.4054X_2 \leq 128$$

Thus the feasible region to which Reeves' method is applied is given by:

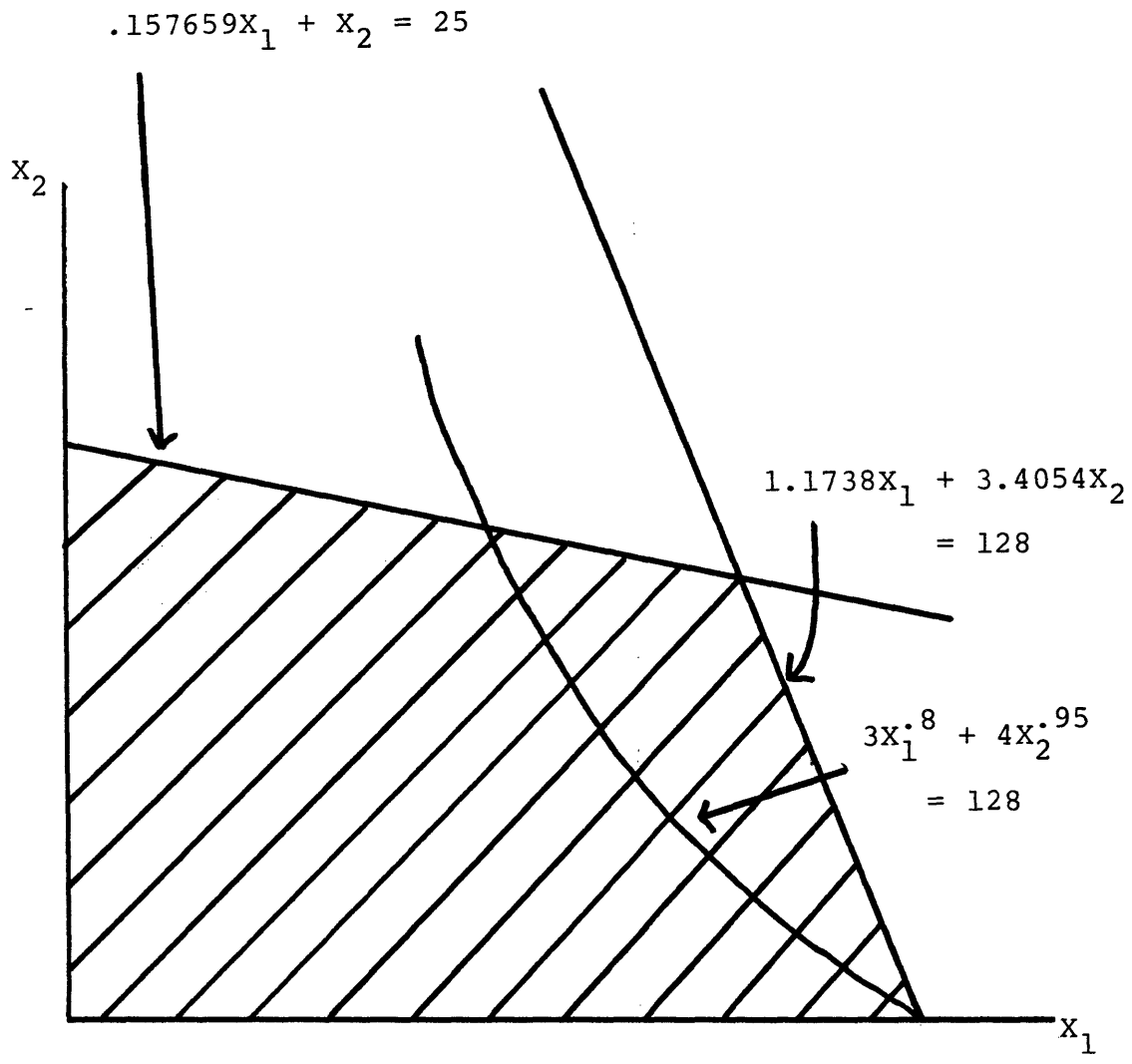
$$.157659X_1 + X_2 \leq 25 \qquad (4\text{-}15)$$

$$1.1738X_1 + 3.4054X_2 \leq 128 \qquad (4\text{-}16)$$

$$X_1, X_2 \geq 0 \qquad (4\text{-}17)$$

The graph of the feasible region determined by these three equations is shown in Figure H. Comparing Figures F and H, we see that Reeves' method enlarges the original feasible region so that this method yields infeasible solutions, and then the feasible region must be partitioned into a number of subregions, and then in each subregion a branch-and-bound procedure must be carried out to obtain feasible solutions to the original problem.

FIGURE G. GRAPH OF FEASIBLE REGION DETERMINED BY EQUATIONS (4-15) THROUGH (4-17)



A FORTRAN program written for a DEC 1091 for implementing the NOLLE algorithm appears in Appendix A, together with output from several sample runs. The computer program has considerable documentation included in it to facilitate any modifications that may be required to run the program on different computers.

In Chapter Five, several example problems illustrating the implementation of the NOLLE algorithm will be discussed in detail.

# CHAPTER FIVE

# APPLICATION OF THE NOLLE ALGORITHM TO A SET OF TEST PROBLEMS

The NOLLE algorithm proposed in Chapter Four is applied to a series of test problems from the literature and elsewhere. The first problem, from Liao (1979), was used to demonstrate the alternative approaches to solving the learning curve problem presented in Chapter Two:

## Problem One

Maximize

$$Z = (353X_1 - 320X_1^{.67807}) + (295X_2 - 256X_2^{.76553}) + (347X_3 - 288X_3^{.848}) \quad (5\text{-}1)$$

subject to

$$10X_1 + 15X_2 + 12X_3 \leq 14{,}400 \quad (5\text{-}2)$$

$$24X_1^{.67807} + 20X_2^{.76553} + 14X_3^{.848} \leq 5920 \quad (5\text{-}3)$$

$$40X_1^{.67807} + 32X_2^{.76553} + 36X_3^{.848} \leq 13{,}026 \quad (5\text{-}4)$$

and $X_1, X_2, X_3 \geq 0$.

Using the NOLLE algorithm developed in Chapter Four, this problem is transformed into

$$\text{Maximize} \quad Z = 3348.988X_1^{.67807} + 1219.962X_2^{.76553} + 709.5395X_3^{.848} \quad (5\text{-}5)$$

subject to

$$103.9373X_1^{.67807} + 75X_2^{.76553} + 34.497X_3^{.848} \leq 14,400 \quad (5\text{-}6)$$

$$24X_1^{.67807} + 20X_2^{.76553} + 14X_3^{.848} \leq 5920 \quad (5\text{-}7)$$

$$40X_1^{.67807} + 32X_2^{.76553} + 36X_3^{.848} \leq 13,026 \quad (5\text{-}8)$$

and $X_1, X_2, X_3 \geq 0$.

The coefficients of $X_1^{.67807}$ in equations (5-5) and (5-6) are obtained as shown below.

From equation (5-2), the upper bound on $X_1$ is 1440, obtained by setting $X_2$ and $X_3$ to zero. Thus, using the notation introduced in Chapter Four, $X_{1UBL} = 1440$. From equations (5-3) and (5-4), the upper bounds on $X_1$ are 3371.49 and 5078.55, respectively, and thus $X_{1UBNL} = 3371.49$. Consequently, $X_{1LUB} = 1440$.

Now the new coefficeint of $X_1$ in equation (5-5) is $(353 \cdot (1140)^{1-.67807} - 320) = 3348.988$, and the new coefficient of $X_1$ in equation (5-6) is $10(1440)^{1-.67807} = 103.9373$. The coefficients of $X_2$ and $X_3$ in equations (5-5) and (5-6) are obtained in a similar manner.

Returning to the equations (5-5) through (5-8), we see that this is a linear programming problem under the transformation of variables $Y_j = X_j^{C_j}$.

Solving this problem using the NOLLE algorithm we find the solution of equations (5-1) through (5-4) to be $X_1 = 1440$, $X_2 = X_3 = 0$, and $Z^* = 463,985.60$, which agrees

with the results given by Reeves (1980). See Appendix A for the printout of the solution to Problem One using the NOLLE algorithm.

The second problem is from Reeves and Sweigart (1981) and is reproduced below:

Problem Two

$$\text{Maximize } Z = \sum_{j=1}^{4} 500X_j - 309X_1^{.95} - 335X_2^{.8} - 201X_3^{.84} - 274X_4^{.9} \tag{5-9}$$

Subject to

$$18X_1^{.95} + 13X_2^{.8} + 9X_3^{.84} + 5X_4^{.9} \leq 859 \tag{5-10}$$

$$X_1^{.95} + 6X_2^{.8} + 15X_3^{.84} + X_4^{.9} \leq 547 \tag{5-11}$$

$$9X_1^{.95} + 16X_2^{.8} + 10X_3^{.84} + 18X_4^{.9} \leq 863 \tag{5-12}$$

$$9X_1^{.95} + 14X_2^{.8} + 3X_3^{.84} + 18X_4^{.9} \leq 663 \tag{5-13}$$

and $X_1, X_2, X_3, X_4 \geq 0$.

Reeves and Sweigart (1981) using their branch-and-bound linear approximation algorithm obtained the result $X_1 = 0$, $X_2 = 105.24$, $X_3 = 35.24$, $X_4 = 0$, and $Z^* = 52{,}341$.

Using the NOLLE algorithm the objective function of Problem Two is transformed into

$$\text{maximize } Z = 303.807X_1^{.95} + 976.6452X_2^{.8} + 790.914X_3^{.84} + 472.4432X_4^{.9} \tag{5-14}$$

and the constraints are unchanged, because in Problem Two,

all constraints are subject to learning effects.

The coefficient of $X_1^{.95}$ in equation (5-14) is obtained as shown below.

Since there are no linear constraints $X_{1UBL}$ is set arbitrarily to $10^6$. From equation (5-10) the upper bound on $X_1$ is 58.489, obtained by setting $X_2 = X_3 = X_4 = 0$ and solving $18X_1^{.95} = 859$ for $X_1$. Similarly, from equations (5-11) through (5-13) the upper bounds on $X_1$ are 762.24, 121.919, and 92.374, respectively, and thus $X_{1UBNL} = 58.489$. Consequently, $X_{1LUB} = 58.489$.

The new coefficient of $X_1$ in equation (5-14) is now obtained as follows:

$$(a_1 X_{1LUB}^{1-C_1} - b_1) = (500 \cdot (58.489)^{1-.95} - 309)$$
$$= 303.807$$

The coefficients of $X_2$, $X_3$, and $X_4$ in equation (5-14) are obtained in a similar manner.

Solving this problem using the NOLLE algorithm, we find the solution of Problem Two (equations (5-9) through (5-13)) to be $X_1 = 0$, $X_2 = 105.4201$, $X_3 = 35.08308$, $X_4 = 0$, and $Z^* = 52348.81$. The very slight differences in the answers are well within round-off error and can be accounted for by differences in word length of the machines used and differences in FORTRAN compilers. See Appendix A for the

printout of the solution to Problem Two using the NOLLE algorithm.

The remaining problems are from Woolsey (1981). The objective in running these problems was to determine the robustness of the NOLLE algorithm and to delineate the universe of problem types to which the NOLLE algorithm could be applied.

Problems Three, Four, and Five are similar to Problem Two in that all constraints are subject to learning, but the number of decision variables is increased. The only change in Problems Three, Four, and Five is in the learning exponents, i.e., the $C_j$'s in order to investigate how changes in the learning exponents will affect the optimum solution. In Problems Three, Four, and Five it is assumed that all decision variables must satisfy nonnegativity constraints.

Problem Three is as follows:

## Problem Three

$$\text{Maximize } Z = (600X_1 - 550X_1^{.6784}) + (700X_2 - 657X_2^{.7756}) + (895X_3 - 701X_3^{.8674}) + (658X_4 - 534X_4^{.6665}) + (784X_5 - 673X_5^{.7455}) + (785X_6 - 456X_6^{.8433}) + (902X_7 - 858X_7^{.7593})$$

subject to

$$60X_1^{.6784} + 78X_2^{.7756} + 92X_3^{8674} + 56X_4^{.6665} + 48X_5^{.7452} + 23X_6^{.8433} + 105X_7^{.7593} \leq 67,578 \quad \text{(Constraint 1)}$$

The second and third constraints are also subject to learning, and the coefficients and right-hand sides (RHS) of these $\leq$ constraints are as given below:

Coefficients

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | RHS |
|---|---|---|---|---|---|---|---|---|
| Constraint 2 | 62 | 34 | 67 | 105 | 102 | 67 | 89 | 56,667 |
| Constraint 3 | 104 | 107 | 109 | 102 | 98 | 95 | 45 | 34,589 |

Problems Four and Five are the same as Problem Three except that the learning exponents (the $C_j$'s) are changed as indicated below:

Learning Exponents

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
|---|---|---|---|---|---|---|---|
| Problem Four | .8867 | .8557 | .6774 | .8532 | .8711 | .6444 | .6345 |
| Problem Five | .8888 | .7543 | .7676 | .5478 | .7522 | .8211 | .7433 |

The solutions obtained by the algorithm of this paper are as follows (only decision variables with nonzero values are listed):

Solutions

Problem Three: $X_4 = 1373.833$, $X_7 = 3501.539$, $Z^* = 3,575,107$

Problem Four: $X_7 = 26250.26$ $Z^* = 23,131,439.1$

Problem Five: $X_4 = 41819.55$ $Z^* = 27,335,657$

In Problem Three Constraints 2 and 3 are tight. In Problem Four Constraint 2 is tight, and in Problem Five Constraint 3 is tight. Thus in all these problems the NOLLE algorithm converged to an optimum solution. The fact that different combinations of basic variables occur, and the values they assume are different, in the solutions listed above, indicate that the optimum solutions to these problems are sensitive to the values of the $C_j$'s, and thus considerable care must be taken in estimating the parameters of the learning curve. See Appendix A for the printout of the solutions to Problems Three, Four, and Five using the NOLLE algorithm.

As a rough test to determine the sensitivity of the optimal solution to changes in the $C_j$'s, Problem Three, as given above, was rerun changing only $C_4$. The results are as follows:

| $C_4$ | Values of Basic Variables |
|---|---|
| .5478 | $X_4 = 41819.55$ |
| .6665* | $X_4 = 1373.833$, $X_7 = 3501.539$ |
| .8532 | $X_1 = 675.427$, $X_7 = 4347.872$ |
| .95 | $X_1 = 675.427$, $X_7 = 4347.872$ |

*This is the original Problem Three presented above.

In all these cases at least one functional constraint was tight. As $C_4$ increases, $X_4$ is driven out of the basis, and $X_1$ and $X_7$ are forced into the basis with the values of both increasing until they reach limiting values. Thus changing only one parameter, $C_4$, in Problem Three affects not only the values of the basic variables, but also affects which of the original decision variables are basic in the optimum solution.

The remaining series of examples (Woolsey, 1981) are chosen in order to experiment with several changes in the learning curve problem:

I. The $C_j$'s in each problem are set equal to each other. This value, $\alpha$, changes with each problem.

II. Each problem contains both nonlinear constraints subject to learning and linear constraints not subject to learning (Problems Six through Eight have six nonlinear constraints and three linear constraints, and Problems Nine through Thirteen have four nonlinear constraints and three linear constraints.) All of the test problems of Reeves and Sweigart (1981)

consisted solely of nonlinear constraints subject to learning and did not contain a mixture of nonlinear and linear constraints.

III. Problems Six through Eight have nine decision variables and Problems Nine through Thirteen have eighteen decision variables.

Problems Six through Eight have the following form:

$$\text{maximize } Z = \sum_{j=1}^{9} (a_j X_j - b_j X_j^{\alpha})$$

$$\text{subject to } \sum_{j=1}^{9} d_{ij} X_j^{\alpha} \leq r_i, \quad i = 1, 2, \ldots, 6$$

$$\sum_{j=1}^{9} e_{ij} X_j \leq r_i, \quad i = 7, 8, 9$$

$$\text{and } X_j \geq 0 \quad j = 1, 2, \ldots, 9$$

where the values of the parameters are as listed below.

| j | $a_j$ | $b_j$ | Problem Number | α |
|---|---|---|---|---|
| 1 | 789 | 700 | Six | .6773 |
| 2 | 856 | 802 | Seven | .773 |
| 3 | 904 | 567 | Eight | .883 |
| 4 | 543 | 522 | | |
| 5 | 453 | 401 | | |
| 6 | 836 | 813 | | |
| 7 | 715 | 700 | | |
| 8 | 605 | 568 | | |
| 9 | 220 | 198 | | |

The coefficients ($d_{ij}$) and the right-hand sides ($r_i$) of the constraints subject to learning are:

| i | $d_{i1}$ | $d_{i2}$ | $d_{i3}$ | $d_{i4}$ | $d_{i5}$ | $d_{i6}$ | $d_{i7}$ | $d_{i8}$ | $d_{i9}$ | $r_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 23 | 45 | 46 | 78 | 34 | 93 | 65 | 83 | 49 | 1200 |
| 2 | 235 | 107 | 69 | 138 | 148 | 175 | 167 | 188 | 103 | 2100 |
| 3 | 78 | 67 | 93 | 76 | 75 | 73 | 92 | 70 | 84 | 16774 |
| 4 | 10 | 37 | 83 | 56 | 34 | 22 | 40 | 86 | 25 | 18698 |
| 5 | 56 | 67 | 33 | 50 | 93 | 86 | 73 | 42 | 59 | 12779 |
| 6 | 19 | 85 | 66 | 41 | 103 | 78 | 21 | 25 | 28 | 9777 |

The coefficients ($e_{ij}$) and the right-hand sides ($r_i$) of the constraints not subject to learning are:

| i | $e_{i1}$ | $e_{i2}$ | $e_{i3}$ | $e_{i4}$ | $e_{i5}$ | $e_{i6}$ | $e_{i7}$ | $e_{i8}$ | $e_{i9}$ | $r_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 678 | 957 | 453 | 733 | 692 | 224 | 533 | 213 | 201 | 17,968 |
| 8 | 204 | 455 | 782 | 291 | 678 | 453 | 600 | 439 | 601 | 21,458 |
| 9 | 208 | 200 | 199 | 199 | 567 | 378 | 598 | 512 | 704 | 104,578 |

The solutions obtained by the NOLLE algorithm for Problems Six through Eight are as follows (only decision variables with nonzero values are listed):

Solutions

Problem Six: $X_1 = 7.332763$, $X_3 = 12.02475$, $X_6 = 9.818807$

Problem Seven: $X_1 = 8.509071$, $X_3 = 23.44223$, $X_6 = .3614326$

Problem Eight: $X_1 = 4.436201$, $X_3 = 25.97293$

In all these problems at least one functional constraint was tight. As $\alpha$ increases from .6773 through .773 to .883 in Problems Six, Seven, and Eight, respectively, $X_6$ is driven out of the basis and the value of $X_3$ increases.

Problems Nine through Thirteen have the following form:

$$\text{maximize} \quad Z = \sum_{j=1}^{18} (a_j X_j - b_j X_j^{\alpha})$$

$$\text{subject to} \quad \sum_{j=1}^{18} d_{ij} X_j^{\alpha} \leq r_i, \qquad i = 1, 2, 3, 4$$

$$\sum_{j=1}^{18} e_{ij} X_j \leq r_i, \qquad i = 5, 6, 7$$

$$\text{and} \quad X_j \geq 0, \qquad j = 1, 2, \ldots, 18$$

where the values of the parameters are listed below.

| j | $a_j$ | $b_j$ |
|---|---|---|
| 1 | 1000 | 900 |
| 2 | 950 | 850 |
| 3 | 900 | 800 |
| 4 | 850 | 750 |
| 5 | 800 | 700 |
| 6 | 750 | 650 |
| 7 | 700 | 600 |
| 8 | 650 | 550 |
| 9 | 600 | 500 |
| 10 | 550 | 450 |
| 11 | 500 | 400 |
| 12 | 450 | 350 |
| 13 | 400 | 300 |
| 14 | 350 | 250 |
| 15 | 300 | 200 |
| 16 | 250 | 150 |
| 17 | 200 | 100 |
| 18 | 150 | 50 |

| Problem Number | $\alpha$ |
|---|---|
| 9 | .95 |
| 10 | .90 |
| 11 | .85 |
| 12 | .80 |
| 13 | .60 |

The coefficients ($d_{ij}$) and the right-hand sides ($r_i$) of the constraints subject to learning are:

| | i | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| $d_{i1}$ | 21 | 18 | 54 | 96 |
| $d_{i2}$ | 52 | 64 | 61 | 32 |
| $d_{i3}$ | 33 | 95 | 62 | 63 |
| $d_{i4}$ | 71 | 54 | 53 | 21 |
| $d_{i5}$ | 17 | 13 | 18 | 19 |
| $d_{i6}$ | 55 | 61 | 73 | 84 |
| $d_{i7}$ | 103 | 92 | 50 | 31 |
| $d_{i8}$ | 40 | 32 | 38 | 42 |
| $d_{i9}$ | 60 | 63 | 54 | 57 |
| $d_{i,10}$ | 90 | 87 | 77 | 65 |
| $d_{i,11}$ | 84 | 97 | 32 | 42 |
| $d_{i,12}$ | 36 | 37 | 40 | 45 |
| $d_{i,13}$ | 60 | 62 | 63 | 67 |
| $d_{i,14}$ | 30 | 32 | 17 | 25 |
| $d_{i,15}$ | 65 | 63 | 61 | 18 |
| $d_{i,16}$ | 24 | 27 | 23 | 16 |
| $d_{i,17}$ | 16 | 24 | 20 | 12 |
| $d_{i,18}$ | 32 | 40 | 43 | 47 |
| $r_i$ | 32,342 | 65,221 | 14,655 | 17,345 |

The coefficients ($e_{ij}$) and the right-hand sides ($r_i$) of the constraints not subject to learning are:

| | i | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| $e_{i1}$ | 401 | 321 | 651 |
| $e_{i2}$ | 453 | 352 | 811 |
| $e_{i3}$ | 307 | 145 | 215 |
| $e_{i4}$ | 236 | 166 | 327 |
| $e_{i5}$ | 501 | 173 | 921 |
| $e_{i6}$ | 602 | 81 | 803 |
| $e_{i7}$ | 713 | 191 | 601 |
| $e_{i8}$ | 231 | 211 | 603 |
| $e_{i9}$ | 445 | 166 | 531 |
| $e_{i,10}$ | 455 | 278 | 432 |
| $e_{i,11}$ | 473 | 433 | 635 |
| $e_{i,12}$ | 481 | 513 | 896 |
| $e_{i,13}$ | 507 | 164 | 732 |
| $e_{i,14}$ | 178 | 182 | 503 |
| $e_{i,15}$ | 188 | 173 | 107 |
| $e_{i,16}$ | 232 | 224 | 809 |
| $e_{i,17}$ | 327 | 325 | 932 |
| $e_{i,18}$ | 345 | 373 | 345 |
| $r_i$ | 17,624 | 18,642 | 102,327 |

Using the NOLLE algorithm the solution to Problems Nine through Thirteen was the same: $X_4 = 74.67797$, with at least one functional constraint tight in each problem.

It is interesting to compare the behavior of the solutions to Problems Six through Eight and Problems Nine through Thirteen. In Problems Six through Eight the basic variables in the optimal solution, and their values were directly dependent on the value of $\alpha$. In Problems Nine through Thirteen, the basic variable and its value in the optimal solution was independent of the value of $\alpha$. This interesting occurrence can be explained by noting that in Problems Six through Eight the nonlinear constraints are the binding constraints, while in Problems Nine through Thirteen the linear constraints are the binding constraints.

In all the examples tested the maximum number of decision variables in the optimum solution was three. Because of the complex geometry of the feasible region as determined by equations (4-2) and (4-3), and because of the convex objective function as given by equation (4-1) it seems likely that only in very rare cases will there be more than three of the original decision variables in the optimum solution.

In conclusion, the NOLLE algorithm seems to be robust and to converge very rapidly to an optimum solution under a range of conditions. If the critical constraints are those subject to learning then changes in the $C_j$ seem to produce major changes in the optimum solutions. If the constraints not subject to learning are dominant then, as would be expected, the optimum solution appears to be insensitive to the values of the $C_j$'s.

# CHAPTER SIX

# DISCUSSION OF THE RESULTS AND PROPOSALS FOR FURTHER STUDY

## 6.1. Discussion of the Results

In Chapter Five we applied the NOLLE algorithm to several sample problems involving learning effects and in every case arrived at feasible solutions in which at least one constraint was tight; but for such nonlinear programming problems we cannot conclude that the solutions represent global optima.

The general nonlinear programming problem is to find $\bar{x} = (x_1, x_2, \ldots, x_n)$ so as to

$$\text{maximize } f(\bar{x})$$

subject to

$$g_i(\bar{x}) \leq b_i, \quad i = 1, 2, \ldots, m$$

and

$$\bar{x} \geq \bar{0} .$$

The famous Kuhn-Tucker conditions that state necessary, but not sufficient, conditions for a solution to be optimal are given in the following theorem (Hillier and Lieberman, 1980):

> Theorem. Assume that $f(\bar{x})$, $g_1(\bar{x})$, $g_2(\bar{x})$, ..., $g_m(\bar{x})$ are differentiable functions satisfying certain regularity conditions. then

$$\bar{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$$

can be an optimal solution for the general nonlinear programming problem only if there exist m numbers $U_1, U_2, \ldots, U_m$, such that all of the following conditions are satisfied:

$$\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} U_i \frac{\partial g_i}{\partial x_j} \leq 0 \qquad (6\text{-}1)$$

$$x_j^* \left(\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} U_i \frac{\partial g_i}{\partial x_j}\right) = 0 \qquad (6\text{-}2)$$

(Equations (6-1) and (6-2) are evaluated at $x_j = x_j^*$, for $j = 1, 2, \ldots, n$)

$$g_i(\bar{x}^*) - b_i \leq 0, \quad i = 1, 2, \ldots, m \qquad (6\text{-}3)$$

$$U_i (g_i(\bar{x}^*) - b_i) = 0, \quad i = 1, 2, \ldots, m \qquad (6\text{-}4)$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n \qquad (6\text{-}5)$$

$$U_i \geq 0, \quad i = 1, 2, \ldots, m. \qquad (6\text{-}6)$$

The $U_i$'s are analogous to the dual variables of linear programming, and they have a similar economic interpretation, i.e., they represent the shadow price (or opportunity cost) of the ith resource (Chiang, 1974). In fact, the $U_i$'s are generalized Lagrange multipliers.

The Kuhn-Tucker conditions will now be applied to several of the problems solved in Chapter Five to show that they satisfy the necessary conditions for an optimal solution.

From Problem One of Chapter Five we have that $f(x_1, x_2, x_3)$ is given by equation (5-1), and thus

$$\frac{\partial f}{\partial x_1} = 353 - 216.9824\ x_1^{-.32193}$$

Also, $g_1$, $g_2$, and $g_3$ are given by equations (5-2), (5-3), and (5-4), respectively, and thus

$$\frac{\partial g_1}{\partial x_1} = 10, \quad \frac{\partial g_1}{\partial x_2} = 15, \quad \frac{\partial g_1}{\partial x_3} = 12$$

$$\frac{\partial g_2}{\partial x_1} = 16.27368 x_1^{-.32193}$$

$$\frac{\partial g_3}{\partial x_1} = 27.1228\ X_1^{-.32193}$$

Using the NOLLE algorithm, the solution obtained in Chapter Five for Problem One was $x_1^* = 1440$, and $x_2$ and $x_3$ were nonbasic variables with values of zero.

Equation (6-1) becomes for j = 1:

$$332.1237 - 10U_1 - 1.5657U_2 - 2.6095U_3 \leq 0$$

Equation (6-2) becomes for j = 1:

$$1440(332.1237 - 10U_1 - 1.5657U_2 - 2.6095U_3) = 0$$

Equation (6-3) becomes:

for j = 1: $14400 - 14400 \leq 0$

for j = 2: $3325.08 - 5920 \leq 0$

for j = 3: $5541.8 - 13026 \leq 0$

Equation (6-4) becomes:

for j = 1: $U_1 \cdot 0 = 0$

for j = 2: $U_2(-2594.92) = 0$

for j = 3: $U_3(-7484.2) = 0$

From Equation (6-4) we have $U_2 = U_3 = 0$, and from Equation (6-2) we have $U_1 = 33.21237$. Thus the solution to Problem One determined by the NOLLE algorithm satisfies the Kuhn-Tucker first-order (i.e., necessary) conditions. To check for sufficiency would require the formation of a bordered Hessian and then calculating the values of its bordered principal minors.

Next, we apply the Kuhn-Tucker conditions to Problem Two of Chapter Five given by Equations (5-9) through (5-13). Equation (5-9) gives f, and thus

$$\frac{\partial f}{\partial x_2} = 500 - 268x_2^{-.2}$$

$$\frac{\partial f}{\partial x_3} = 500 - 168.84x_3^{-.16}$$

Also $g_1$, $g_2$, $g_3$, $g_4$ are given by equations (5-10) through (5-13), respectively, and thus

$$\frac{\partial g_1}{\partial x_3} = 7.56x_3^{-.16}, \quad \frac{\partial g_1}{\partial x_2} = 10.4\ x_2^{-.2}$$

$$\frac{\partial g_2}{\partial x_3} = 12.6x_3^{-.16}, \quad \frac{\partial g_2}{\partial x_2} = 4.8x_2^{-.2}$$

$$\frac{\partial g_3}{\partial x_3} = 8.4x_3^{-.16}, \quad \frac{\partial g_3}{\partial x_2} = 12.8x_2^{-.2}$$

$$\frac{\partial g_4}{\partial x_3} = 2.52x_3^{-.16}, \quad \frac{\partial g_4}{\partial x_2} = 11.2x_2^{-.2}$$

Using the NOLLE algorithm the solution obtained in Chapter Five for Problem Two was $x_2^* = 105.4201$, $x_3^* = 35.08308$ with $x_1$ and $x_4$ nonbasic variables with values of zero.

Equation (6-1) becomes:

for j = 2: $394.4277 - 4.0968U_1 - 1.891U_2 - 5.042U_3 - 4.412U_3 \leq 0$

for j = 3: $404.4436 - 4.2786U_1 - 7.1311U_2 - 4.7540U_3 - 1.4262U_4 \leq 0$

Equation (6-2) becomes:

for j = 2: $105.4201\ (394.4277 - 4.0968U_1 - 1.891U_2 - 5.042U_3 - 4.412U_4) = 0$

for j = 3: $35.08308\ (404.4436 - 4.2786U_1 - 7.1311U_2 - 4.7540U_3 - 1.4262U_4) = 0$

Equation (6-3) becomes:

for j = 1: $718.561 - 859 \leq 0$

for j = 2: $547 - 547 \leq 0$

for j = 3: $863 - 863 \leq 0$

for j = 4: $640.96 - 663 \leq 0$

Equation (6-4) becomes:

for j = 1: $U_1(-140.44) = 0$

for j = 2: $U_2(0) = 0$

for j = 3: $U_3(0) = 0$

for j = 4: $U_4(-22.04) = 0$

From Equation (6-4) we have $U_1 = U_4 = 0$, and from Equation (6-2) we have

$394.4277 - 1.891U_2 - 5.042U_3 = 0$ and

$404.4436 - 7.1311U_2 - 4.7540U_3 = 0$.

Solving these two linear equations simultaneously we have $U_2 = 6.085$ and $U_3 = 75.946$. Thus all the Kuhn-Tucker conditions are satisfied and the solution obtained by the

NOLLE algorithm does satisfy the necessary conditions for an optimal solution.

Finally, we want to consider some of the economic implications of the problems solved in Chapter Five.

From the computer output for Problem One of Chapter Five appearing in Appendix A, the optimal values of the dual decision variables are $W_1^* = 32.22$, $W_2^* = 0$, and $W_3^* = 0$. These values are the shadow prices (or opportunity costs) associated with the three resource constraints, respectively. These dual decision variables tell us that we should be willing to pay \$0.00 to purchase additional units of the second and third resources. This is completely logical because at this time we are not using all of the currently available units of these resources.

The value of $W_1^*$ measures the rate of change of the value of the objective function for each additional unit of the first resource (so long as the current set of basic variables remain unchanged). The value of $W_1^*$ also represents the maximum unit price we would be willing to pay for additional units of the first resource. If the shadow price is greater than the actual unit cost, the allocation should be increased until this relationship no longer holds.

In addition, the final optimal tableau of Problem One appearing in Appendix A, tells us how to modify the production scheduling if we obtain one additional unit of resource one. We should use this additional resource to produce .0096212 more units of product one.

From the computer output for Problem Two of Chapter Five appearing in Appendix A, the optimal values of the dual decision variables are $W_1^* = 0$, $W_2^* = 71.069$, $W_3^* = 505680$, and $W_4^* = 0$. $W_1^* = 0$ and $W_4^* = 0$ imply that we should not purchase any additional units of resources one and four because, in fact, we are not using all of the currently available units of these resources.

The value for $W_3^*$ has considerable economic significance because of its large magnitude. Recall that $W_3^*$ measures the rate of change of the value of the objective function for each additional unit of the third resource (so long as the current set of basic variables remain unchanged). Because of the large value of $W_3^*$ strong consideration should be given to increasing the allocation of the third resource so long as the actual unit cost remains less than the shadow price.

Again, from the optimal tableau for Problem Two appearing in Appendix A, we can ascertain how the firm's production schedule should be altered if one additional

unit of resource three is made available. The firm should produce 230.63 more units of product three and 144.08 fewer units of product two. Since the firm is currently only producing 105.4201 units of product two this implies that the allocation of resource three cannot be increased by a full unit without changing the current set of basic variables.

Thus, in applying the NOLLE algorithm the firm will find not only an optimal solution or near optimal solution to its learning curve problem, but it can derive important economic information about the opportunity costs associated with the various resources, and how the firm's production planning should be altered if additional units of the resources should become available.

## 6.2. Proposals for Further Study

As is frequently the case in a research project, more questions arise during the course of the investigation than are answered. There are a number of additional areas in which further study would be beneficial.

First, in some learning situations the log-linear model of the learning curve, i.e., $Y = KX^n$, may not be the most appropriate, but instead either the S model or the DeJong model may provide a better fit to the data.

At this time, no algorithm exists which will solve the general learning curve problem for either of these two models.

Second, the aggregate planning problem mentioned in Chapter One can be stated as follows (Ebert, 1976):

minimize

$$Z_T = \sum_{t=1}^{n} C_1 WD_t + C_2 (WD_t - WD_{t-1})^2 + C_3 (P_t - C_4 WD_t)^2 + C_5 P_t - C_6 WD_t + C_7 (I_t - C_8)^2$$

subject to

$$I_t = I_{t-1} + P_t - S_t, \qquad t = 1, 2, \ldots, n$$

where the decision variables are the work force ($WD_t$) and the production rate ($P_t$). Numerical values for the coefficients $C_1, C_2, \ldots, C_8$ are statistically estimated from accounting data. $C_4$, the productivity factor (units of output per man-month) is no longer a constant in situations where the learning curve applies, but must be treated as a variable. The aggregate mix problem is typically solved by the Hooke-Jeeves direct-search approach assuming $C_4$ is a constant. The deficiencies in the Hooke-Jeeves algorithm are well known. Treating $C_4$ as a constant leads to feasible, but far from economically efficient, solutions.

A challenging problem for further study would be to solve the aggregate planning problem with learning effects included using algorithms which do not have the major defects of the Hooke-Jeeves algorithm.

Third, heuristic studies seem to indicate that lot sizes obtained from the traditional EOQ formulas can be made more economically efficient by considering learning effects (Yelle, 1979). Devising an algorithm to solve the economic order quantity problem which incorporates learning effects appears to be a particularly fruitful area for further research.

A fourth area of research would be the generalization of the NOLLE algorithm to include equality constraints and $\geq$ constraints.

A fifth area for additional study is to relax the assumption made on page 25 that $b_{1j} = b_{2j} = \ldots = b_{gj}$, and hence to develop a generalized NOLLE algorithm to handle a wider class of learning situations. This author is researching this, and it is hoped the results will be published in a journal.

A sixth area that merits additional study is to attempt to combine both surrogation and condensation in a geometric programming approach to solve the general learning curve problem. This author is also researching this approach and hopes to publish the results.

# BIBLIOGRAPHY

Abernathy, William J. and Wayne, Kenneth, 1974, "Limits of the Learning Curve," Harvard Business Review, Vol. 52, No. 5, pp. 109-119.

Andress, Frank J., 1954, "The Learning Curve as a Production Tool," Harvard Business Review, Vol. 32, No.1, pp. 87-97.

Beightler, Charles and Phillips, Don T., 1976, Applied Geometric Programming, John Wiley and Sons, New York.

Chiang, Alpha C., 1974, Fundamental Methods of Mathematical Economics, Second Edition, McGraw-Hill, New York.

Ebert, Ronald J., 1976, "Aggregate Planning with Learning Curve Productivity," Management Science, Vol. 23, No. 2, (October), pp. 171-182.

Hesse, Rick and Woolsey, Gene, 1980, Applied Management Science, Science Research Associates, Inc., Chicago.

Hillier, Frederick S. and Lieberman, Gerald J., 1980, Introduction to Operations Research, Third Edition, Holden-Day, San Francisco.

Hirschmann, Winfred B., 1964, "Profit from the Learning Curve," Harvard Business Review, Vol. 42, No. 1, pp. 125-139.

Liao, Woody M., 1979, "Effects of Learning on Resource Allocation Decisions," Decision Sciences, Vol. 10, No. 1, pp. 116-125.

Reeves, Gary R., 1980, "A Note on the Effects of Learning on Resource Allocation Decisions," Decision Sciences, Vol. 11, No. 1, pp. 169-170.

Reeves, Gary R. and Sweigart, James R., 1981, "Product-Mix Models When Learning Effects are Present," Management Sciences, Vol. 27, No. 2, (February), pp. 204-212.

Woolsey, Robert E.D. and Swanson, Huntington S., 1975, Operations Research for Immediate Application: A Quick and Dirty Manual, Harper and Row, New York.

BIBLIOGRAPHY (continued)

Woolsey, Robert E.D., 1979, Private Communication.

Woolsey, Robert E.D., 1981, Private Communications.

Yelle, Louis E., 1979, "The Learning Curve: Historical Review and Comprehensive Survey," Decision Sciences, Vol. 10, No. 2, pp. 302-328.

# APPENDIX A

## A.1 A Computer Program in FORTRAN for Implementing the NOLLE Algorithm

```
C     Program: LNGCRV.FOR -- Learning curve optimization
C
C     See Chapter 4 for a detailed description of the algorithim used
C     in this program and for a flowchart of the program.
C
C     variables: K       is the quantity M-G
C                N       is the number of products
C                N1      one plus N
C                G       is the number of scarce resources subject
C                        to learning
C                M       is the total number of constraints
C                A,B,C   input vectors containing the objective function
C                        coefficients where each term of the objective
C                        function is of the form A(J)*X(J)-B(J)*X(J)**C(J).
C                        NOTE: insert zeros for missing values.
C                D       input vector containing the technological
C                        coefficients of the constraints subject to
C                        learning in row order with zeros inserted
C                        for any missing coefficients.
C                E       input vector containing the technological
C                        coefficients of the constraints subject to
C                        learning in row order with zeros inserted
C                        for any missing coefficients.
C                RN      input vector containing the right
C                        hand sides of the constraints subject
C                        to learning.
C                RL      input vector containing the right
C                        hand sides of the constraints not subject
C                        to learning.
C                LB      array containing the linear bounds
C                NB      array containing the nonlinear bounds
C                BB      array containing the least of LB and NB
C                BV      array containing subscripts to !!!!!!!!!!
C                P       array that stores the rest of the tableau
C                P0      array containing the 0th row of the tableau
C                X0      array containing the optimal X values
C
C     The subroutines HOME and CLEAR0 contain the nontransportable
C     features of this program. Calls to them may be commented away
C     if they cause any problems. Thier functions are as follows:
C     HOME:   clears the crt screen and places the cursor in the
C             upper left hand corner.
C     CLEAR0: on the CSM DECsystem10 a user may type control O to
C             cause tty output to be thrown away so that the
C             program won't have to wait for the tty. CLEAR0 clears
C             the control O condition if it has occurred.
C
C     There are two other nontransportables:
C     the OPEN statment in the disk input section,
C     and the PARAMETER statment before the DIMENSION statements.
C     If the later is a problem, remove it and put constants in the
C     DIMENSION statments.


      parameter dim=30

      dimension a(dim),b(dim),c(dim),d(dim,dim),e(dim,dim)
      dimension rn(dim),rl(dim)
      real lb(dim),nb(dim)
      integer bv(dim)
```

```
      dimension bb(dim),p(dim,dim),p0(dim),xo(dim)

      integer g
      integer flag,out,dsk
      common out,flag
      double precision filnam

      data flag,in,out,dsk /0,4,4,10/

C     This program optimizes problems of the classical product-mix
C     type certain of the scarce resources are subject to learning
C     effects.
C     X(i), i=1,...,n are the decision varibles.
C     The first G constraints are the nonlinear constraints subject to
C     learning, and the last (M-G) constraints are the linear
C     constraints not subject to learning.
C     It is acceptable for G to equal M, i.e. all constraints
C     are subject to learning.

 5    format(t27,'Learning curve optimization',//,
     &          ' This program optimizes',//,5x,'J=N',12x,'I=G',/,
     &          ' MAX:SUM(A(I)*X(I))-SUM(B(I)*X(I)**C(I))',/,5X,'J=1',
     &          12x,'I=1',//,' Subject to:',//,' J=N',/,
     &          ' SUM(D(I,J)*X(J)**C(J)).LE.R(I), I=1,...,G',/,' J=1',
     &          //,' And',//,' J=N',/,
     &          ' SUM(E(I,J)*X(J).LE.R(I), I=G+1,...,M',/,' J=1',///)
10    format(' Would you like to enter data from a file or the TTY?')
15    format(a1)
20    format(' Enter number of products:')
25    format(3g)
30    format(' Enter number of scarce resources s.t. learning:')
35    format(' Enter total number of constraints:')
40    format(' Enter',i3,' lines of "A(I),B(I),C(I)"')
45    format(' Enter',i3,' groups of:',i3,
     &          ' lines of "D(I,J)" and one line of "RN(I)"')
50    format(' Enter',i3,' groups of:',i3,
     &          ' lines of "E(I,J)" and one line of "RL(I)"')
55    format(' What is the name of the data file?')
60    format(a10)
65    format(' Establish linear bounds')
70    format(' Linear bound on X(',i2,')=',f)
75    format(//,' Establish nonlinear bounds')
80    format(' Nonlinear bound on X(',i2,')=',f)
85    format(' Least upper bound on X(',i2,')=',f)
90    format(/,' Convert linear constraints')
95    format(' E(',i2, ',', i2, ') is now',f)
100   format(/,' Convert objective function')
105   format(' Old A(',i2,')-B(',i2,') is now',f)
110   format(/,' Solving with LP',/)
115   format(5(4x,1pe11.4))
120   format(' X(',i2,') enters basis',//,' X(',i2,') leaves basis')
125   format(/,' Basis',/)
130   format(' X(',i2,') =',f)
135   format(/,' Optimum solution')
140   format(' X(',i2,') =',f)
145   format(/,' Optimal value of objective function:',g)
150   format(//,' Would you like to enter another problem?')

      call home
```

```
        write(out,5)
        call clearo
 200    write(out,10)
        read(in,15) answer
        if(answer.eq.'F'.or.answer.eq.'f') goto 230
        if(answer.ne.'T'.and.answer.ne.'t') goto 200
C
C    This section will read data interactivly from the TTY
C
        call home
        write(out,20)
        read(in,25) n
        write(out,30)
        read(in,25) g
        write(out,35)
        read(in,25) m
        write(out,40) n
        do 205 i=1,n
 205    read(in,25) a(i),b(i),c(i)
        write(out,45) g,n
        do 215 i=1,g
        do 210 j=1,n
 210    read(in,25) d(i,j)
 215    read(in,25) rn(i)
        k=m-g
        if(k.eq.0) goto 265
        write(out,50) k,n
        do 225 i=1,k
        do 220 j=1,n
 220    read(in,25) e(i,j)
 225    read(in,25) rl(i)
        goto 265
C
C    This next section reads the input data from a file on disk
C
 230    write(out,55)
        read(in,60) filnam
        open(unit=dsk,file=filnam,access='seqin')
        read(dsk,25) n,g,m
        read(dsk,25) (a(i),b(i),c(i),i=1,n)
        do 240 i=1,g
        do 235 j=1,n
 235    read(dsk,25) d(i,j)
 240    read(dsk,25) rn(i)
        k=m-g
        if(k.eq.0) goto 260
        do 250 i=1,k
        do 245 j=1,n
 245    read(dsk,25) e(i,j)
 250    read(dsk,25) rl(i)
 260    close(unit=dsk)
C
C    The next section will establish linear bounds
C
 265    call home
        write(out,65)
        if(k.eq.0) goto 280
        do 275 i=1,n
        lb(i)=1E6
        do 270 j=1,k
```

```
        if(e(j,i).eq.0) goto 270
        h=r1(j)/e(j,i)
        if(h.lt.lb(i)) lb(i)=h
 270    continue
 275    write(out,70) i,lb(i)
        goto 290
C
C    This next section will set the nonlinear bounds
C
 280    do 285 i=1,n
 285    lb(i)=1E8
        write(out,75)
 290    do 300 i=1,n
        nb(i)=1E7
        do 295 j=1,g
        if(d(j,i).eq.0) goto 295
        sm=(rn(j)/d(j,i))**(1./c(i))
        if(sm.lt.nb(i)) nb(i)=sm
 295    continue
 300    write(out,80) i,nb(i)
C
C    This will select the least upper bound
C
        write(out,25)
        write(out,25)
        do 305 i=1,n
        bb(i)=lb(i)
        if(nb(i).lt.lb(I)) bb(i)=nb(i)
 305    write(out,85) i,bb(i)
C
C    This converts the linear constraints
C
        if(k.eq.0) goto 312
        write(out,90)
        do 310 i=1,n
        do 310 j=1,k
        p(j,i)=e(j,i)*bb(i)**(1.-c(i))
 310    write(out,95) j,i,p(j,i)
C
C     This will convert the objective function
C
 312    write(out,100)
        do 315 i=1,n
        p0(i)=a(i)*bb(i)**(1.-c(i))-b(i)
        write(out,105) i,i,p0(i)
 315    p0(i)=-p0(i)
C
C     Simplex algorithim applied to the modified and transformed
C     problem. [See Hillier and Lieberman text for basic explaination
C     of the algorithim.]
C
        write(out,110)
        n1=n+m+1
        do 317 i = 1, m
        n3 = n+1
        n2= m+n
        do 317 j = n3, n2
317     p(i, j) = 0
        do 318 i = 1, m
318     p(i, n+i) = 1
```

```
        if(k.eq.0) goto 325
        do 320 j=1,k
 320    p(j,N1)=r1(j)
 325    do 330 j=1,g
        do 330 i=1,n
 330    p(k+j,i)=d(j,i)
        do 335 j=1,g
 335    p(k+j,n1)=rn(j)
        do 340 j=1,m
 340    bv(j)=n+j
        write(out,115) (p0(i),i=1,n1)
        do 350 j=1,m
 350    write(out,115) (p(j,i),i=1,n1)
C
C     Find min col
C
 355    sm=0.
        n2 = n+m
        do 360 j=1,n2
        if(p0(j).ge.sm) goto 360
        sm=p0(j)
        jo=j
 360    continue
        if(sm.eq.0.) goto 405
C
C    Find min row
C
        sm=1E7
        do 365 i=1,m
        if(p(i,jo).le.0.) goto 365
        h=p(i,n1)/p(i,jo)
        if(h.ge.sm) goto 365
        sm=h
        io=i
 365    continue
        i=io+n
        write(out,120) jo,bv(io)
        bv(io)=jo
C
C     Pivot routine
C
        do 370 i=1,n1
        if(i.eq.jo) goto 370
        p0(i)=p0(i)-p0(jo)*p(io,i)/p(io,jo)
 370    continue
        do 380 j=1,m
        if(j.eq.io) goto 380
        do 375 i=1,n1
        if(i.eq.jo) goto 375
        p(j,i)=p(j,i)-p(j,jo)*p(io,i)/p(io,jo)
 375    continue
 380    continue
        pv=p(io,jo)
        do 385 i=1,n1
 385    p(io,i)=p(io,i)/pv
        p0(jo)=0.
        do 390 j=1,m
 390    p(j,jo)=0.
        p(io,jo)=1.
        write(out,125)
```

```
        do 395 j=1,M
 395    write(out,130) bv(j),p(j,n1)
C
C     Output the LP matrix
C
        write(out,115) (p0(i),i=1,n1)
        do 400 j=1,m
 400    write(out,115) (p(j,i),i=1,n1)
        goto 355
C
C     Output the optimal solution
C
        call clearo
 405    write(out,135)
        do 407 j=1,n
        xo(j)=0.
 407    CONTINUE
        DO 410 J=1,M
        if(bv(j).gt.n) goto 410
        xo(bv(j))=p(j,n1)**(1./c(bv(j)))
 410    continue
        do 415 j=1,n
 415    write(out,140) j,xo(j)
C
C     Output the objective function
C
        sm=0.
        do 420 j=1,n
 420    sm=sm+a(j)*xo(j)-b(j)*xo(j)**c(j)
        write(out,145) sm
C
C    See if the user wants to do another problem
C
 425    write(out,150)
        read(in,15) answer
        if(answer.eq.'Y'.or.answer.eq.'y') goto 200
        if(answer.ne.'N'.and.answer.ne.'n') goto 425
        callexit
        end



C
C     This routine will clear the crt screen and place the cursor
C     in the 'home' position for that particular terminal.
C
C     Variables:
C       atty - array containing the names of terminal types that this
C              routine understands. If you would like to add some more
C              don't forget to increase its size.
C       out  - the output device number. It must be defined by the
C              calling routine.
C       flag - is used so that in subsequent calls, the routine ITTYPE
C              won't have to be called. (if your tty type changes
C              during execution, set flag back to zero in the calling
C              routine to make HOME rethink what to do.)
C
        subroutine home

        implicit integer (b-z)
```

```
      common out,flag
      dimension atty(3)
      data atty /4hVT52, 3hH19, 1h /

      if(flag) 10,20,30

10    return

20    flag=ittype(atty,0)
      if(flag.gt.0) goto 30
      flag=-1
      return

30    write(out,40)
40    format(' HJ',$)
      return

      end
```

## A.2.1 PROBLEM ONE

```
                    Learning curve optimization

This program optimizes

    J=N             I=G
MAX:SUM(A(I)*X(I))-SUM(B(I)*X(I)**C(I))
    J=1             I=1

Subject to:

J=N
SUM(D(I,J)*X(J)**C(J)).LE.R(I), I=1,...,G
J=1

And

J=N
SUM(E(I,J)*X(J).LE.R(I), I=G+1,...,M
J=1


What is the name of the data file?
TEST

Establish linear bounds
Linear bound on X( 1)=   1440.0000000
Linear bound on X( 2)=    960.0000000
Linear bound on X( 3)=   1200.0000000
Nonlinear bound on X( 1)=   3371.4930000
Nonlinear bound on X( 2)=   1691.2590000
Nonlinear bound on X( 3)=   1040.1820000


Least upper bound on X( 1)=   1440.0000000
Least upper bound on X( 2)=    960.0000000
Least upper bound on X( 3)=   1040.1820000

Convert linear constraints
E( 1, 1) is now    103.9373000
E( 1, 2) is now     75.0489100
E( 1, 3) is now     34.4970400

Convert objective function
Old A( 1)-B( 1) is now   3348.9880000
Old A( 2)-B( 2) is now   1219.9620000
Old A( 3)-B( 3) is now    709.5395000

Solving with LP

   -3.3490E+03    -1.2200E+03    -7.0954E+02     0.0000E+00     0.0000E+00
    0.0000E+00     0.0000E+00
    1.0394E+02     7.5049E+01     3.4497E+01     1.0000E+00     0.0000E+00
    0.0000E+00     1.4400E+04
    2.4000E+01     2.0000E+01     1.4000E+01     0.0000E+00     1.0000E+00
    0.0000E+00     5.9200E+03
    4.0000E+01     3.2000E+01     3.6000E+01     0.0000E+00     0.0000E+00
    1.0000E+00     1.3026E+04
X( 1) enters basis
X( 4) leaves basis
```

```
Basis

X( 1) =    138.5450000
X( 5) =   2594.9200000
X( 6) =   7484.2000000
    0.0000E+00     1.1982E+03     4.0200E+02     3.2221E+01     0.0000E+00
    0.0000E+00     4.6399E+05
    1.0000E+00     7.2206E-01     3.3190E-01     9.6212E-03     0.0000E+00
    0.0000E+00     1.3855E+02
    0.0000E+00     2.6706E+00     6.0343E+00    -2.3091E-01     1.0000E+00
    0.0000E+00     2.5949E+03
    0.0000E+00     3.1176E+00     2.2724E+01    -3.8485E-01     0.0000E+00
    1.0000E+00     7.4842E+03

Optimum solution
X( 1) =   1440.0000000
X( 2) =      0.0000000
X( 3) =      0.0000000

Optimal value of objective function: 463985.6000000
```

## A.2.2 PROBLEM TWO

```
What is the name of the data file?
TEST2

Establish linear bounds


Establish nonlinear bounds
Nonlinear bound on X( 1)=      58.4890200
Nonlinear bound on X( 2)=     124.2315000
Nonlinear bound on X( 3)=      72.3435900
Nonlinear bound on X( 4)=      54.9879800


Least upper bound on X( 1)=      58.4890200
Least upper bound on X( 2)=     124.2315000
Least upper bound on X( 3)=      72.3435900
Least upper bound on X( 4)=      54.9879800

Convert objective function
Old A( 1)-B( 1) is now     303.8070000
Old A( 2)-B( 2) is now     976.6452000
Old A( 3)-B( 3) is now     790.9140000
Old A( 4)-B( 4) is now     472.4432000

Solving with LP

   -3.0381E+02    -9.7665E+02    -7.9091E+02    -4.7244E+02     0.0000E+00
    0.0000E+00     4.6399E+05     0.0000E+00     0.0000E+00
    1.8000E+01     1.3000E+01     9.0000E+00     5.0000E+00     1.0000E+00
    0.0000E+00     0.0000E+00     0.0000E+00     8.5900E+02
    1.0000E+00     6.0000E+00     1.5000E+01     1.0000E+00     0.0000E+00
    1.0000E+00     0.0000E+00     0.0000E+00     5.4700E+02
    9.0000E+00     1.6000E+01     1.0000E+01     1.8000E+01     0.0000E+00
    0.0000E+00     1.0000E+00     0.0000E+00     8.6300E+02
    9.0000E+00     1.4000E+01     3.0000E+00     1.8000E+01     0.0000E+00
    0.0000E+00     0.0000E+00     1.0000E+00     6.6300E+02
X( 2) enters basis
X( 8) leaves basis

Basis

X( 5) =     243.3571000
X( 6) =     262.8571000
X( 7) =     105.2857000
X( 2) =      47.3571400
    3.2404E+02     0.0000E+00    -5.8163E+02     7.8324E+02     0.0000E+00
    0.0000E+00     4.6399E+05     6.9760E+01     4.6251E+04
    9.6429E+00     0.0000E+00     6.2143E+00    -1.1714E+01     1.0000E+00
    0.0000E+00     0.0000E+00    -9.2857E-01     2.4336E+02
   -2.8571E+00     0.0000E+00     1.3714E+01    -6.7143E+00     0.0000E+00
    1.0000E+00     0.0000E+00    -4.2857E-01     2.6286E+02
   -1.2857E+00     0.0000E+00     6.5714E+00    -2.5714E+00     0.0000E+00
    0.0000E+00     1.0000E+00    -1.1429E+00     1.0529E+02
    6.4286E-01     1.0000E+00     2.1429E-01     1.2857E+00     0.0000E+00
    0.0000E+00     0.0000E+00     7.1429E-02     4.7357E+01
X( 3) enters basis
X( 7) leaves basis

Basis
```

```
X( 5) =    143.7935000
X( 6) =     43.1304400
X( 3) =     16.0217400
X( 2) =     43.9239100
    2.1024E+02     0.0000E+00     0.0000E+00     5.5565E+02     0.0000E+00
    0.0000E+00     4.6407E+05    -3.1393E+01     5.5570E+04
    1.0859E+01     0.0000E+00     0.0000E+00    -9.2826E+00     1.0000E+00
    0.0000E+00    -9.4565E-01     1.5217E-01     1.4379E+02
   -1.7391E-01     0.0000E+00     0.0000E+00    -1.3478E+00     0.0000E+00
    1.0000E+00    -2.0870E+00     1.9565E+00     4.3130E+01
   -1.9565E-01     0.0000E+00     1.0000E+00    -3.9130E-01     0.0000E+00
    0.0000E+00     1.5217E-01    -1.7391E-01     1.6022E+01
    6.8478E-01     1.0000E+00     0.0000E+00     1.3696E+00     0.0000E+00
    0.0000E+00    -3.2609E-02     1.0870E-01     4.3924E+01
X( 8) enters basis
X( 6) leaves basis

Basis

X( 5) =    140.4389000
X( 8) =     22.0444500
X( 3) =     19.8555600
X( 2) =     41.5277800
    2.0745E+02     0.0000E+00     0.0000E+00     5.3402E+02     0.0000E+00
    1.6045E+01     4.6404E+05     0.0000E+00     5.6262E+04
    1.0872E+01     0.0000E+00     0.0000E+00    -9.1778E+00     1.0000E+00
   -7.7778E-02    -7.8333E-01     0.0000E+00     1.4044E+02
   -8.8889E-02     0.0000E+00     0.0000E+00    -6.8889E-01     0.0000E+00
    5.1111E-01    -1.0667E+00     1.0000E+00     2.2044E+01
   -2.1111E-01     0.0000E+00     1.0000E+00    -5.1111E-01     0.0000E+00
    8.8889E-02    -3.3333E-02     0.0000E+00     1.9856E+01
    6.9444E-01     1.0000E+00     0.0000E+00     1.4444E+00     0.0000E+00
   -5.5556E-02     8.3333E-02     0.0000E+00     4.1528E+01

Optimum solution
X( 1) =      0.0000000
X( 2) =    105.4201000
X( 3) =     35.0830800
X( 4) =      0.0000000

Optimal value of objective function:  52348.8100000
```

## A.2.3 PROBLEM THREE

```
What is the name of the data file?
BAD1

Establish linear bounds


Establish nonlinear bounds
Nonlinear bound on X( 1)=   5239.3620000
Nonlinear bound on X( 2)=   1726.8160000
Nonlinear bound on X( 3)=    768.0061000
Nonlinear bound on X( 4)=   6285.4580000
Nonlinear bound on X( 5)=   2633.3720000
Nonlinear bound on X( 6)=   1092.9990000
Nonlinear bound on X( 7)=   4929.4250000


Least upper bound on X( 1)=   5239.3620000
Least upper bound on X( 2)=   1726.8160000
Least upper bound on X( 3)=    768.0061000
Least upper bound on X( 4)=   6285.4580000
Least upper bound on X( 5)=   2633.3720000
Least upper bound on X( 6)=   1092.9990000
Least upper bound on X( 7)=   4929.4250000

Convert objective function
Old A( 1)-B( 1) is now   8874.7790000
Old A( 2)-B( 2) is now   3071.5160000
Old A( 3)-B( 3) is now   1458.8440000
Old A( 4)-B( 4) is now  11627.0600000
Old A( 5)-B( 5) is now   5159.6050000
Old A( 6)-B( 6) is now   1893.7490000
Old A( 7)-B( 7) is now   6125.3310000

Solving with LP

   -8.8748E+03    -3.0715E+03    -1.4588E+03    -1.1627E+04    -5.1596E+03
   -1.8937E+03    -6.1253E+03     0.0000E+00     5.6262E+04     0.0000E+00
    0.0000E+00
    6.0000E+01     7.8000E+01     9.2000E+01     5.6000E+01     4.8000E+01
    2.3000E+01     1.0500E+02     1.0000E+00     0.0000E+00     0.0000E+00
    6.7578E+04
    6.2000E+01     3.4000E+01     6.7000E+01     1.0500E+02     1.0200E+02
    6.7000E+01     8.9000E+01     0.0000E+00     1.0000E+00     0.0000E+00
    5.6667E+04
    1.0400E+02     1.0700E+02     1.0900E+02     1.0200E+02     9.8000E+01
    9.5000E+01     4.5000E+01     0.0000E+00     0.0000E+00     1.0000E+00
    3.4689E+04
X( 4) enters basis
X(10) leaves basis

Basis

X( 8) =  48533.0600000
X( 9) =  20957.7400000
X( 4) =    340.0882000
    2.9803E+03     9.1255E+03     1.0966E+04     0.0000E+00     6.0115E+03
    8.9354E+03    -9.9575E+02     0.0000E+00     5.6262E+04     1.1399E+02
    3.9542E+06
    2.9020E+00     1.9255E+01     3.2157E+01     0.0000E+00    -5.8039E+00
   -2.9157E+01     8.0294E+01     1.0000E+00     0.0000E+00    -5.4902E-01
```

```
    4.8533E+04
   -4.5059E+01    -7.6147E+01    -4.5206E+01     0.0000E+00     1.1176E+00
   -3.0794E+01     4.2676E+01     0.0000E+00     1.0000E+00    -1.0294E+00
    2.0958E+04
    1.0196E+00     1.0490E+00     1.0686E+00     1.0000E+00     9.6078E-01
    9.3137E-01     4.4118E-01     0.0000E+00     0.0000E+00     9.8039E-03
    3.4009E+02
X( 7) enters basis
X( 9) leaves basis

Basis

X( 8) =   9101.8960000
X( 7) =    491.0841000
X( 4) =    123.4335000
    1.9289E+03     7.3488E+03     9.9114E+03     0.0000E+00     6.0376E+03
    8.2169E+03     0.0000E+00     0.0000E+00     5.6285E+04     8.9972E+01
    4.4432E+06
    8.7678E+01     1.6252E+02     1.1721E+02     0.0000E+00    -7.9067E+00
    2.8781E+01     0.0000E+00     1.0000E+00    -1.8815E+00     1.3878E+00
    9.1019E+03
   -1.0558E+00    -1.7843E+00    -1.0593E+00     0.0000E+00     2.6189E-02
   -7.2157E-01     1.0000E+00     0.0000E+00     2.3432E-02    -2.4121E-02
    4.9108E+02
    1.4854E+00     1.8362E+00     1.5360E+00     1.0000E+00     9.4923E-01
    1.2497E+00     0.0000E+00     0.0000E+00    -1.0338E-02     2.0446E-02
    1.2343E+02

Optimum solution
X( 1) =      0.0000000
X( 2) =      0.0000000
X( 3) =      0.0000000
X( 4) =   1373.8330000
X( 5) =      0.0000000
X( 6) =      0.0000000
X( 7) =   3501.5390000

Optimal value of objective function:3575107.0000000
```

## A.2.4 PROBLEM FOUR

```
What is the name of the data file?
BAD3

Establish linear bounds


Establish nonlinear bounds
Nonlinear bound on X( 1)=     700.7457000
Nonlinear bound on X( 2)=     859.4394000
Nonlinear bound on X( 3)=    4950.6680000
Nonlinear bound on X( 4)=     927.1981000
Nonlinear bound on X( 5)=     843.6181000
Nonlinear bound on X( 6)=    9473.6640000
Nonlinear bound on X( 7)=   26250.2600000


Least upper bound on X( 1)=     700.7457000
Least upper bound on X( 2)=     859.4394000
Least upper bound on X( 3)=    4950.6680000
Least upper bound on X( 4)=     927.1981000
Least upper bound on X( 5)=     843.6181000
Least upper bound on X( 6)=    9473.6640000
Least upper bound on X( 7)=   26250.2600000

Convert objective function
Old A( 1)-B( 1) is now     710.5302000
Old A( 2)-B( 2) is now    1198.6890000
Old A( 3)-B( 3) is now   13221.6400000
Old A( 4)-B( 4) is now    1259.9350000
Old A( 5)-B( 5) is now    1195.5140000
Old A( 6)-B( 6) is now   19910.6400000
Old A( 7)-B( 7) is now   36329.7500000

Solving with LP

   -7.1053E+02    -1.1987E+03    -1.3222E+04    -1.2599E+03    -1.1955E+03
   -1.9911E+04    -3.6330E+04     0.0000E+00     0.0000E+00     0.0000E+00
    0.0000E+00
    6.0000E+01     7.8000E+01     9.2000E+01     5.6000E+01     4.8000E+01
    2.3000E+01     1.0500E+02     1.0000E+00     0.0000E+00     0.0000E+00
    6.7578E+04
    6.2000E+01     3.4000E+01     6.7000E+01     1.0500E+02     1.0200E+02
    6.7000E+01     8.9000E+01     0.0000E+00     1.0000E+00     0.0000E+00
    5.6667E+04
    1.0400E+02     1.0700E+02     1.0900E+02     1.0200E+02     9.8000E+01
    9.5000E+01     4.5000E+01     0.0000E+00     0.0000E+00     1.0000E+00
    3.4689E+04
X( 7) enters basis
X( 9) leaves basis

Basis

X( 8) =     723.6738000
X( 7) =     636.7079000
X(10) =    6037.1460000
    2.4598E+04     1.2680E+04     1.4128E+04     4.1601E+04     4.0441E+04
    7.4387E+03     0.0000E+00     0.0000E+00     4.0820E+02     0.0000E+00
    2.3131E+07
   -1.3146E+01     3.7888E+01     1.2955E+01    -6.7876E+01    -7.2337E+01
   -5.6045E+01     0.0000E+00     1.0000E+00    -1.1798E+00     0.0000E+00
```

```
    7.2367E+02
    6.9663E-01      3.8202E-01      7.5281E-01      1.1798E+00      1.1461E+00
    7.5281E-01      1.0000E+00      0.0000E+00      1.1236E-02      0.0000E+00
    6.3671E+02
    7.2652E+01      8.9809E+01      7.5124E+01      4.8910E+01      4.6427E+01
    6.1124E+01      0.0000E+00      0.0000E+00     -5.0562E-01      1.0000E+00
    6.0371E+03

Optimum solution
X( 1) =       0.0000000
X( 2) =       0.0000000
X( 3) =       0.0000000
X( 4) =       0.0000000
X( 5) =       0.0000000
X( 6) =       0.0000000
X( 7) =   26250.2600000

Optimal value of objective function:  0.2313143E+08
```

## A.2.5 PROBLEM FIVE

```
What is the name of the data file?
BAD4

Establish linear bounds


Establish nonlinear bounds
Nonlinear bound on X( 1)=    689.9810000
Nonlinear bound on X( 2)=   2131.3740000
Nonlinear bound on X( 3)=   1821.8270000
Nonlinear bound on X( 4)=  41819.5500000
Nonlinear bound on X( 5)=   2447.2640000
Nonlinear bound on X( 6)=   1320.6110000
Nonlinear bound on X( 7)=   5919.5140000


Least upper bound on X( 1)=    689.9810000
Least upper bound on X( 2)=   2131.3740000
Least upper bound on X( 3)=   1821.8270000
Least upper bound on X( 4)=  41819.5500000
Least upper bound on X( 5)=   2447.2640000
Least upper bound on X( 6)=   1320.6110000
Least upper bound on X( 7)=   5919.5140000

Convert objective function
Old A( 1)-B( 1) is now    691.1662000
Old A( 2)-B( 2) is now   3945.0330000
Old A( 3)-B( 3) is now   4422.4780000
Old A( 4)-B( 4) is now  80378.1300000
Old A( 5)-B( 5) is now   4747.3980000
Old A( 6)-B( 6) is now   2383.0710000
Old A( 7)-B( 7) is now   7527.9520000

Solving with LP

   -6.9117E+02    -3.9450E+03    -4.4225E+03    -8.0378E+04    -4.7474E+03
   -2.3831E+03    -7.5280E+03     0.0000E+00     4.0820E+02     0.0000E+00
    2.3131E+07
    6.0000E+01     7.8000E+01     9.2000E+01     5.6000E+01     4.8000E+01
    2.3000E+01     1.0500E+02     1.0000E+00     0.0000E+00     0.0000E+00
    6.7578E+04
    6.2000E+01     3.4000E+01     6.7000E+01     1.0500E+02     1.0200E+02
    6.7000E+01     8.9000E+01     0.0000E+00     1.0000E+00     0.0000E+00
    5.6667E+04
    1.0400E+02     1.0700E+02     1.0900E+02     1.0200E+02     9.8000E+01
    9.5000E+01     4.5000E+01     0.0000E+00     0.0000E+00     1.0000E+00
    3.4689E+04
X( 4) enters basis
X(10) leaves basis

Basis

X( 8) =  48533.0600000
X( 9) =  20957.7400000
X( 4) =    340.0882000
    8.1263E+04     8.0373E+04     8.1472E+04     0.0000E+00     7.2479E+04
    7.2479E+04     2.7933E+04     0.0000E+00     4.0820E+02     7.8802E+02
    5.0467E+07
   2.9020E+00     1.9255E+01     3.2157E+01     0.0000E+00    -5.8039E+00
   -2.9157E+01     8.0294E+01     1.0000E+00     0.0000E+00    -5.4902E-01
```

```
 4.8533E+04
-4.5059E+01    -7.6147E+01    -4.5206E+01     0.0000E+00     1.1176E+00
-3.0794E+01     4.2676E+01     0.0000E+00     1.0000E+00    -1.0294E+00
 2.0958E+04
 1.0196E+00     1.0490E+00     1.0686E+00     1.0000E+00     9.6078E-01
 9.3137E-01     4.4118E-01     0.0000E+00     0.0000E+00     9.8039E-03
 3.4009E+02

Optimum solution
X( 1) =       0.0000000
X( 2) =       0.0000000
X( 3) =       0.0000000
X( 4) =   41819.5500000
X( 5) =       0.0000000
X( 6) =       0.0000000
X( 7) =       0.0000000

Optimal value of objective function:  0.2733566E+08
```