THEORY AND APPLICATIONS OF A NONCONVEX PROGRAMMING ALGORITHM

BY

ERDOĞAN GÜRMAN

BSME-ITÜ-1974

Submitted to the Faculty of Engineering
in Partial Fulfillment of the Requirements

for the Degree of
Master of Science
in
Industrial Engineering

Boğaziçi University

Nov.1977

- E R R A T A -

Page	Line	Should Read
52	5	S.T. $P_{ikt} \leq \sum_{\zeta=0}^{t} y_{ik\zeta}$
55	15	the first term in the summation $\sum_{i=1}^{n} \psi_i^k$
56	1	... over $G \cap C^k$
	2	(same)
58	9	$f_i(x_i) = FX_i + B(x_i)^{E_i}$ if $X_i > 0$
	10	$f_i(x_i) = 0$ if $x_i = 0$
	11	where FX_i is the fixed charge, B_i coefficient and $E_i \leq 1$ ...
60	10	11-20 MXSIZE (110) Minimum number of ...

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to those who have helped me during the development of this study, especially to my thesis advisor Prof. Gündüz Ulusoy for his invaluable advice, guiding and encouragement and providing me with necessary research material and to Mrs. Nazan Erkmen for her skillful typing the manuscript.

Erdoğan Güfman

ABSTRACT

In this thesis, a solution technique to solve a class of nonlinear programming problems is presented. The problem considered is the minimization of separable concave functions and linear functions over linear polyhedra. A branch-and-bound algorithm for identifying an optimal solution is described; it is equivalent to the solution of a finite sequence of linear programming problems. Computational results are cited for a computer code developed implementing the algorithm. The algorithm is applied to dynamic capacity expansion problem considering single plant-single commodity, multi-plant-single commodity, and multi-plant-multi commodity cases.

ÖZET

Bu tezde, belli bir grup doğrusal olmıyan programlama problemlerini çözen bir algoritma takdim edilmektedir. İncelenen problem ayrılabilir iç bükey fonksiyonlar ve doğrusal fonksiyonlardan oluşan bir amaç fonksiyonunun doğrusal kısıtların oluşturduğu bir kapalı ve sınırlı set üzerinde enküçüklenmesidir. Eniyi çözümü bulmak için bir dal-düğüm algoritması tanımlanmakta ve uygulanmaktadır. Bu algoritma sonlu sayıda bir dizi doğrusal programlama problemlerine eşdeğerdir. Algoritmayı uygulamak için geliştirilen bir bilgisayar programı ile ilgili sayısal sonuçlar verilmektedir. Algoritma tek fabrika, tek tip ürün-çok-fabrika-tek tip ürün, ve çok-fabrika-çok tip ürün durumları için dinamik kapasite arttırım problemlerine uygulanmaktadır.

CONTENTS

CHAPTER I

INTRODUCTION

In the past, solution techniques of optimization problems were primarily based upon the assumption of linearity. Since linear programming (LP) has found large application fields to bring solutions to industrial and economical problems, the theory and solution techniques of LP has been completely developed. Today, highly efficient computer program packages are available to LP users to solve large-scale LP problems. On the other hand, there are several classes of problems which are nonlinear but can be solved using LP as an approximating technique by applying successivel Such a class results from the formulation of problems with economies of scale.

Economies of scale are empirically significant in a number of major industries. Decreasing unit costs are encountered in the process of creating additional capacity and in manpower use in industries such as petroleum, chemicals, petrochemicals, steel, cement, and aluminum (1,2,3). Empirical studies of plant

construction costs have demonstrated that in these industries the elasticity of total cost with respect to plant size is constant over a wide range, and frequently has a valve between 0.6 and 0.8. Therefore doubling plant size increases costs between sixty and eighty percent. Similarly distribution cost of goods between regions display economies of scale in certain cases and again, decreasing unit costs are encountered in distribution systems. Recently, minimum cost models, focusing on optimal plant size, location and construction timing in a single industry with economies of scale have been of considerable interest (4,5). Plant location type of problems which can be included in the above class of problems require the consideration of fixed charges (6,7). Mathematical models which are expected to cover the above features lead to the consideration of nonconvex cost functions, thus requiring the development of new solution techniques.

In this study, a general algorithm which can solve the problems having the above features will be presented. An objective function consisting of separable conceva functions and linear functions including fixed charges if necessary, will be minimized over linear polyhedra. Here, a branch-and-bound algorithm will be applied to obtain the solution. The algorithm is mainly based upon soland's (7) simplified algorithm for minimizing separable concave fucntions over linear polyhedra.

The general principles of branch-and bound solution technique are covered in the second chapter. In the third chapter, the problem is defined and the algorithm is presented. An example

problem is given to demonstrate the algorithm. In the fourth chapter, development of a computer program written in FORTRAN IV and some computational results are introduced. Applications of the algorithm to dynamic expansion problems considering production, inventory, and distribution appear in the fifth chapter. The last chapter covers the conclusion and extensions. A user's manual of the computer program used in this thesis is given in the appendix together with a source listing of the program.

CHAPTER 2

REVIEW OF BRANCH-AND-BOUND METHOD

2.1 Introduction

Among the most general approaches to the solution of constrained optimization problems is that of 'branching and bounding'. Branch and bound is intelligently structured search of the space of all feasible solutions. Most commonly, the space of all feasible solutions is repeatedly partitioned into smaller and smaller subsets, and a lower bound (LB) and an upper bound (UB) is determined for the cost of the solutions within each subset. After each partitioning, those subsets with an LB that exceeds the minimum UB of all subsets are excluded from all further partitionings. The partitioning contiues until a feasible solution (ZUP) is found, the cost of which is no greater than the LB for any subset.

2.2 Essential Features of Branching - and - Bounding

There are many optimization problems for which 'direct' methods of solution do not exist, or are inefficient. Typical examples are problems with a nonconvex objective function or

nonconvex constraints or problems with some or all variables constrained to take on discrete values. Branch and bound technique makes it possible to solve such problems by applying existing methods of solution to easy problems.

Suppose we are confronted with the following problem (P_o)

p_o : Minimize $f^o(\underline{x})$

subject to $g_i^o(\underline{x}) \geq 0 \quad i = 1,2,..,m$

$$\underline{x} \in X^o$$

where X^o denotes the domain of optimization and $\underline{x}$ denotes a vector $(x_1, x_2, \ldots, x_n)$. In accordance with the general terminology, an $\underline{x}$ vector which satisfies the constraints and lies within the domain of optimization is called a feasible solution and a feasible solution which yields the minimum value of $f^o(\underline{x})$ is called optimal solution.

One way to solve such a problem is to solve a related easy problem and show that this solution is also a solution to the original problem (P_o). Suppose, in the case of minimization, p_o is replaced by problem p_1 which bounds p_o, in the sense that following bounding property is satisfied :

There exists at least one optimal solution

$\underline{x}^o$ of p_o, which is feasible for p_1 and $f^1(\underline{x}^o) \leq f^o(\underline{x}^o)$

where f_1 is the objective function of P_1.

Assume x^1 is the optimal solution to P_1, if $\underline{x}^1$ satisfies the following optimality conditions : Optimality Condition 1(OC1): $\underline{x}^1$ is a feasible solution to P_o Optimality Condition 2(OC2): $f^1(\underline{x}^1)=f^o(\underline{x}^1)$

the $\underline{x}^1$ is an optimal solution to P_o.

If (OC1) and (OC2) are not satisfied then one proceeds as follows : Generally, the easiest way is to replace P_i by a set of problems P {(2),(3), ...} that similarly bound P_o in the sense that they jointly satisfy the following generalized bounding property:

Generalized Bounding Property : There exists at least one optimal solution $\underline{X}^o$ of P_o, such that it is feasible for at least one problem $(j)\varepsilon P$, and $f^j(\underline{x}^o)\leq f^o(\underline{x}^o)$

Branching : Suppose we have the optimal solution $\underline{x}^j$ to each problem $(j)\varepsilon P$. Let $\underline{x}^k$ be such that

$$f^k(\underline{x}^k)=\min_{(j)\varepsilon P} f^j(\underline{x}^j)$$

In general, P represents the set of problems that have not been replaced (i.e., set of problems that have not yet been branched from).

It is obvious that $\underline{x}^k$ is optimal solution to P_o if (OC1) and (OC2) are satisfied (with $\underline{x}^k$ replacing $\underline{x}^1$ in their statement.) If (OC1) and (OC2) are not satisfied, then problem (k) is replaced by a set of problems $P^{(k)}$. In addition to requiring that the new set of bounding problems $(P-\{k\}) \cup P^{(k)}$ satisfy the generalized bounding property, we may impose the following "weak convergence" condition.

> For each problem $(j) \in P^{(k)}$, either $\underline{x}^k$ is infeasible for (j) or $f^j(\underline{x}^k) > f^k(\underline{x}^k)$

and the "strong convergence" condition

> For each problem $(j) \in P^{(k)}$, and each feasible solution $\underline{x}$ to problem (k), either $\underline{x}$ is infeasible for (j) or $f^j(\underline{x}) > f^k(\underline{x})$.

These conditions, however, are not sufficient to end up with an optimal solution to P_o with a finite amount of computation. They do represent minimal conditions to be imposed in order that some progress toward a final solution can be made.

A tree is usually used to represent the branching-and-bounding. Each node of the tree corresponds to a problem (j) as follows :

Minimize $f^j(\underline{x})$

$$g_i^j(\underline{x}) \geq 0 \quad i = 1,2,..,m_j$$

$$\underline{x} \in X^j$$

The problems that replace problem (j) in the bounding set are pointed by branches directed outward from node(j). Thus set P at any intermediate point in the calculations are designated by the nodes of the tree.

The total amount of computation is related to the number as well as the complexity of the problems created in the fully developed tree. The amount of storage requirement is also related to the maximum number of nodes.

To reduce the cardinality of the set P, some observations can be made. In the course of calculations, various feasible solutions to P_o are discovered. Suppose x is the least costly solution (with respect to f^o) at any intermediate stage and (j) is a bounding problem in P with optimal solution $\underline{x}^j \neq \hat{\underline{X}}$. If $f^j(\underline{x}^j) > f^o(\underline{x})$, it is obvious that a better solution cannot be obtained by proceeding from node (j),then problem (j) can be removed from set P without affecting the optimality condition 2. This process is called 'fathoming."

A lower bound $f^j(\underline{x}^j)$ is associated with each node (j) of the tree, and a node (j) with $f^j(\underline{x}^j) < f^o(\underline{x})$ is called 'active' and the others are called 'fathomed'. The branching-and-bounding concludes when every intermediate node is fathomed.

CHAPTER 3

A L G O R I T H M

3.1. Introduction

There are different approaches to solve the problem of globally minimizing o concave function over a linear polyhedron. Some of them use branch-and-bound techniques and some others employ extreme point techniques. Since Tui's fundamental work(8), this problem has been considered by a number of authors. Some authors (7,9,10) emphasized branch-and-bound techniques, some others(11,12,13) emphasized extreme point ranking techniques.

Most of the approaches require the separability of the objective function (7,9,10). However, there are some recent approaches in which separability of the objective function is not a rqquirement.(14)

According to Lawler and Wood(15), branch-and-bound methods appear to be the best way to obtain globally optimal solutions to nonlinear programming problems in which a nonconvex cost function

is to be minimized. In this study, this approach will be adopted.

3.2. Convex Envelopes

Prior to developing the branch-and-bound technique for the problem, the concept of 'convex envelope' is to be emphasized.

Let f be some function defined over a compact set C. Let Ψ be the convex envelope of f takenover C.

Definition

i) $\Psi(\underline{x})$ is convex over C.

ii) $\Psi(\underline{x}) \leq f(\underline{x})$ for all $\underline{x} \in C$.

iii) If $g(\underline{x})$ is any convex function over C, and if $g(\underline{x}) \leq f(\underline{x})$ for all $\underline{x} \in C$, then $\Psi(\underline{x}) \geq g(\underline{x})$ for all $\underline{x} \in C$.

A single dimensional example of convex envelope is given in Fig. 3.1.

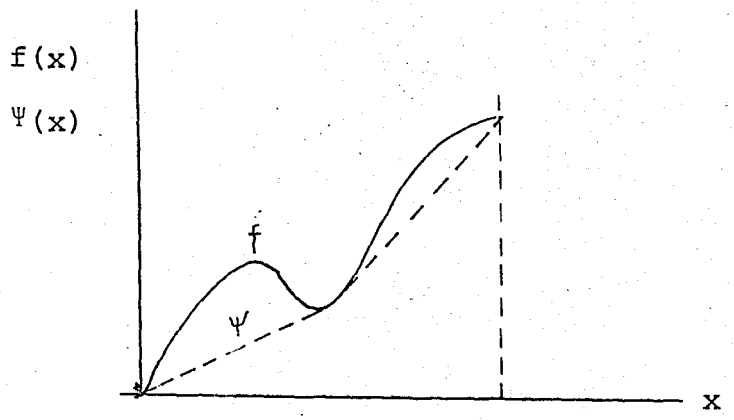


Fig. 3.1. A nonconvex function and its convex envelope

Intuitively, the convex envelope of f taken over C is the "highest" convex function which fits below f. We will be dealing with functions $f(\underline{x})=\sum_{i=1}^{n} f_i(x_i)$ which are separable and set C which may be described by simple inequalities, e.g., $C=\{\underline{x} \mid \underline{l} \leq \underline{x} \leq \underline{L}\}$ where $\underline{l}$ and $\underline{L}$ are fixed vectors in E^n.

Theorem: (9) Convex envelope of a separable function f over a rectangular set C is equal to the sum of the convex envelopes of f_i taken over $C_i=\{x_i \mid l_i \leq x_i \leq L_i\}$.

Proof. Define a function r over E^n by means of the relation:

$$r(\underline{t}) = \max_{\underline{x} \in C} \{<\underline{x},\underline{t}> - f(\underline{x})\} .$$

$$\Psi(\underline{x}) = \sup_{\underline{t} \in E^n} \{<\underline{x},\underline{t}> - r(\underline{t})\}$$

If f is separable and $C=\{\underline{x} \mid \underline{l} \leq \underline{x} \leq \underline{L}\}$ we may write

$$\Psi(\underline{x}) = \sum_{i=1}^{n} \Psi_i(x_i) = \sum_{i=1}^{n} \sup_{t_i} \{x_i t_i - r_i(t_i)\}$$

where $$r(\underline{t}) = \sum_{i=1}^{n} r_i(t_i) = \sum_{i=1}^{n} \max_{l_i \leq x_i \leq L_i} \{x_i t_i - f_i(x_i)\}$$

In other words, the convex envelope of a separable function f over a rectangular set C is equal to the sum of the convex envelopes of the f_i taken over $C_i=\{x_i \mid l_i \leq x_i \leq L_i\}$.

This observation is especially useful if f_i is concave, for the convex envelope of a concave function of a single variable is that linear function passing through the endpoints of the given function. (See Fig. 3.2)

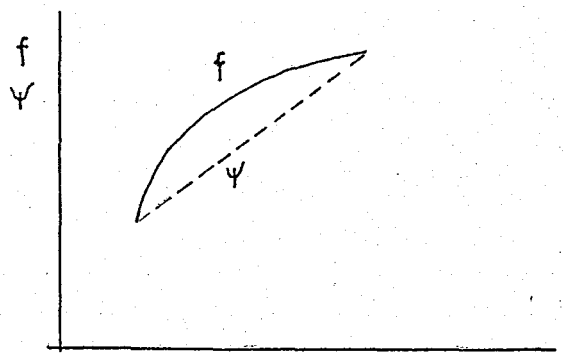


Fig. 3.2

In the material which follws, we take the convex envelopes of f over rectangular subsets of C and use the resulting information to obtain increasingly better approximations to f.

The following theorem is the heart of the solution method to the above cited problem.

Theorem : (Kleibohm (16)). If Ψ is the convex envelope of f taken over C, then a point $\underline{x}^o$ globally solving the minimization problem with concave objective function also minimizes Ψ over C.

Proof . Given $\Psi(\underline{x}) \leqslant f(\underline{x})$ $\underline{x} \in C$ (3.1)

we want to show that $\Psi(\underline{x}^o) = f(\underline{x}^o) \leqslant \Psi(\underline{x})$ $\underline{x}, \underline{x}^o \in C$ (3.2)

i. From (3.1) we know that $\Psi(\underline{x}^o) \leq f(\underline{x}^o)$ Assume

$$\Psi(\underline{x}^o) < f(\underline{x}^o) \tag{3.3}$$

with

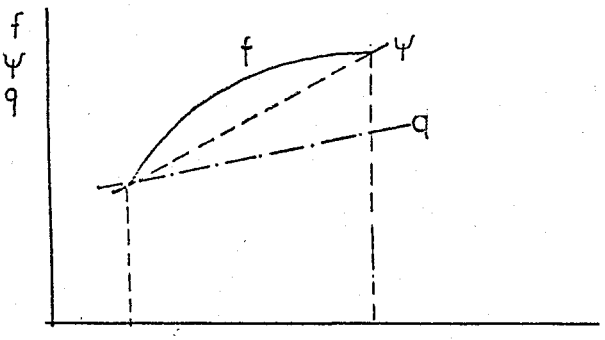
$$q(\underline{x}^o) = f(\underline{x}^o) \tag{3.4}$$

where $q(\underline{x})$ is any convex function defined over C and

$$q(\underline{x}) \leq f(\underline{x}) \tag{3.5}$$

From (3.3) and (3.4) $\Psi(\underline{x}^o) < q(\underline{x}^o)$. This contradicts the definition of convex envelope. Therefore, $\Psi(\underline{x}^o) < f(\underline{x}^o)$ is not possible, thus

$$\Psi(\underline{x}^o) = f(\underline{x}^o) .$$



ii. Assume $f(\underline{x}^o) > \Psi(\underline{x})$ for $\underline{x} \in C$ and let the convex function $q(x)$ be defined by (3.4) and (3.5). Then

$$f(\underline{x}^o) = q(\underline{x}^o) > \Psi(\underline{x}) \qquad \underline{x} \, , \, \underline{x}^o \in C$$

but we know that $q(\underline{x}) \leq \Psi(\underline{x})$, $\underline{x} \in C$ by definition of $\Psi(\underline{x})$ and therefore, this is a contradiction. Thus $f(\underline{x}^o) \leq \Psi(\underline{x})$ $\underline{x}, \underline{x}^o \in C$.

3.3 Problem Definition

As cited in Chapter 1, some optimization problems require the involvement of nonlinearities in model formulations. Existence of economics of scale leads us to a certain group of nonlinear programming problems and causes the relationships between costs and amounts of activities to become concave. A minimization problem with a concave objective function cannot be solved by a direct method because of its nonconvex nature. A model which will handle the above conditions will be a minimization problem as follows :

$$\text{Minimize } f(\underline{x}) = \sum_{i=1}^{n} f_i(x_i) + \sum_{i=n+1}^{N} g_i(x_i)$$

$$\text{subject to } \underline{X} \in G = \{ \underline{x} \mid A\underline{x} \leq \underline{b} \}$$

$$\underline{X} \in C = \{ \underline{x} \mid \underline{l} \leq \underline{x} \leq \underline{L} \}$$

where f_i is concave and g_i is linear over the finite interval $[l_i, L_i]$. It is well known that f takes on its minimum value at some extr

point of the linear polyhedron G∩C, so that the search for an optimal solution may be restricted to extreme points of G∩C.

Theorem: If objective function f is concave over linear polyhedron G∩C, then f achieves its minimum at some extreme point of liner polyhedron.

Proof : Let x^o be optimal but not an extreme point, then

$$\underline{x}^o = \sum_j \alpha_j \underline{x}^j$$

where $\underline{x}^j$ is the jth extreme point of G∩C and $\alpha_j > 0$ and $\sum_j \alpha_j = 1$.

By definition, if f is concave, then

$$f\left(\sum_j \alpha_j \underline{x}^j\right) \geq \sum_j \alpha_j f(\underline{x}^j)$$

since $\underline{x}^o$ minimizes f over G∩C

$$f(\underline{x}^o) \leq f(\underline{x}) \qquad \underline{x}, \underline{x}^o \in G \cap C$$

and since we have assumed that optimal solution is not on one of the extreme points

$$f(\underline{x}^o) < f(\underline{x}^j) \qquad \underline{x}^o, \underline{x}^j \in G \cap C$$

$$f(\underline{x}^o) = f\left(\sum_j \alpha_j \underline{x}^j\right) < \sum_j \alpha_j f(\underline{x}^j)$$

This is a contradiction.

Falk and Soland (9) presented an algorithm for separable nonconvex programming problems that may be applied to the above problem. For this problem, their method reduces to a finite sequence of LP problems. Soland (7) has developed a different version of the algorithm for 'a plant location problem' with an objective function composed only of the separable concave cost functions and a constraint set of the transportation type. He shows that the constraint set can remain the same at each subproblem (iteration). Based on this result, his algorithm exploits the special structure of the transportation problem when solving the subproblem after each partition. Rech and Barton (10) have developed a different version of the algorithm. Their problem contains an objective function which is composed of mostly linear cost functions and some separable nonconvex piecewise linear ones. They convert a distribution problem involving only one commodity into a 'minimal cost network flow' problem and because of the special structure of network flow problem they use the out-of - kilter algorithm for the solution of the subproblems. In their algorithm, the constraint set does not remain the same since the lower and upper bound constraints on nonconvex variables change from one iteration to another. The algorithm which will be presented here will be a combination of the algorithms developed by Rech and Barton (10), and Soland (7) and no restriction will be imposed on the structure of the constraint set with the

exception of linearity.

3.4 Essential Features of the Algorithm

As mentioned previously, the algorithm to be presented is a branch-and-bound algorithm. In the following, the set C and its subset will be referred to as a rectangle. Branching corresponds to partitioning the subset of solutions in a subrectangle $C^k \subset C$ into two new subsets of solutions in the two subrectangles C^p and C^q, where $C^p \cup C^q = C^k$. Bounding corresponds to the determination of a lower bound on the optimal solution value in a subrectangle C^k. As in all branch-and-bound solution methods, a tree will be used to represent the development of the problem. $N^1, N^2, N^3, \ldots$ represent the nodes of the tree where N^1 is the initial node. At stage k of the algorithm nodes N^{2k} and N^{2k+1} are created. Each node N^k represents a rectangle $C^k = \{ \underline{x} \mid \underline{l}^k \leq \underline{x} \leq \underline{L}^k \}$ with $\underline{l} \leq \underline{l}^k < \underline{L}^k \leq \underline{L}$. When N^k is selected for branching, N^p and N^q are branched from N^k and they respectively correspond to rectangles C^p and C^q such that $C^p \cup C^q = C^k$. An intermediate node is one which has not yet been branched from.

The algorithm generates a sequence of feasible solutions $\{ \underline{x}^k : k=1,2,3,\ldots \}$ where $\underline{x}^k$ is obtained at node N^k.

Let $ZUP^k = \min\{f(\underline{x}^j) : j=1,2, \ldots\}$ and $XBEST^k = \underline{x}^i$ such that $ZUP^k = f(\underline{x}^i)$. ZUP^k is an upper bound on the optimal solution value of the problem and $XBEST^k$ is the best solution found through node N^k.

Let $LB(N^k)$ be a lower bound on the optimal value of f. Now it is time to introduce convex envelopes. With $C^k = \{\underline{x} \mid \underline{l}^k \leq \underline{x} \leq \underline{L}^k\}$, let ψ_i^k be the linear function passing through the endpoints of the interval $[l_i^k, L_i^k]$ such that $\psi_i^k(l_i^k) = f_i(l_i^k)$ and $\psi_i^k(L_i^k) = f_i(L_i^k)$. Recall that g_i's are linear and obviously they overlap with their convex envelopes over C. For this reason g_i's are included in the algorithm without any changes. Since f_i is concave, the following facts are evident.

$$\psi_i^k(x_i) \leq f_i(x_i), \text{ if } x_i \in [l_i^k, L_i^k] \tag{3.6}$$

$$\psi_i^k(x_i) \geq f_i(x_i), \text{ if } x_i \in (l_i^k, L_i^k) \tag{3.7}$$

Let $\Psi^k(\underline{x}) = \sum_{i=1}^{n} \psi_i(x_i) + \sum_{i=n+1}^{N} g_i(x_i)$ it is obvious that

$$\Psi^k(\underline{x}) \leq f(\underline{x}) \quad \text{if} \quad \underline{x} \in C^k$$

Thus the minimum of Ψ^k over $G \cap C^k$ is less than or equal to the minimum of f over $G \cap C^k$. Ψ^k is bounded by the minimum of ψ^k over $G \cap C$ and $LB(N^k)$ is defined as follows :

$$LB(N^k) = \min\{\psi^k(\underline{x}) \mid \underline{x}\in G\cap C\} \leq \min\{\psi^k(\underline{x}) \mid \underline{x}\in G\cap C^k\}$$

Therefore, $LB(N^k)$ is the optimal value of the following linear programming problem, which is called P^k.

$$(P^k): \quad \text{Minimize } \psi^k(\underline{x}) = \sum_{i=1}^{n} \psi_i^k(x_i) + \sum_{i=n+1}^{N} g_i(x_i)$$

$$\text{Subject to} \quad \underline{x}\in G\cap C$$

at each iteration on $\underline{x}^k$ is obtained and the sequence $\{\underline{x}^k\}$ is the optimal solutions to Problems P^1, P^2, ...

Branching from an intermediate node, say node N^{k^*}, can be considered if and only if $LB(N^{k^*}) < ZUP^{2k+1}$ (the last numbered node presently is N^{2k+1}). $\underline{x}^{k^*}$ is the solution to problem P^{k^*} at node N^{k^*}. Determine the component i_b, for which the following difference is maximized. This difference corresponds to the error introduced by the convex approximation for each concave function. The difference is

$$f_i(x_i^{k^*}) - \psi_i^{k^*}(x_i^{k^*}) \qquad (i=1,2,\ldots,n) \qquad (3.8)$$

Divide the corresponding interval $[l_{ib}^{k^*}, L_{ib}^{k^*}]$ into two intervals $[l_{ib}^{k^*}, x_{ib}^{k^*}]$ and $[x_{ib}^{k^*}, L_{ib}^{k^*}]$. By branching from node N^{k^*}, nodes (i.e. subproblems) N^{2k+2} and N^{2k+3} are created. Although the convex envelopes are updated according to the new bounds at each node, constraint set is not updated to introduce the changes in lower and upper bounds. Therefore, some $x_i^{k^*}$ may occur outside of $(l_i^{k^*}, L_i^{k^*})$. What would happen if $x_i^{k^*} \notin (l_i^{k}, L_i^{k})$? The following theorem explains that at node N^{k^*} selected for branching $X_{ib}^{k^*}$ must fall in open interval $(l_{ib}^{k^*}, L_{ib}^{k^*})$.

Theorem : If $LB(N^{k^*}) < ZUP^{2k+1}$, then $X_{ib}^{k^*} \in (l_{ib}^{k}, L_{i_b}^{k})$

and $f_{i_b}(x_{i_b}^{k^*}) - \psi_{i_b}^{k^*}(x_{i_b}^{k^*}) > 0$.

Proof : We are given $LB(N^{k^*}) < ZUP^{2k+1} \leq f(\underline{x}^{k^*})$,

since $LB(N^{k}) = \psi^{k^*}(\underline{x}^{k^*})$

$\psi^{k^*}(\underline{x}^{k^*}) < f(\underline{x}^{k^*})$

$f(\underline{x}^{k^*}) - \psi^{k^*}(\underline{x}^{k^*}) > 0$

This implies at least one of the differences in (3.8) is positive and by (3.7) this can only happen if $x_{i_b}^{k^*} \in (l_{i_b}^{k^*}, L_{i_b}^{k^*})$.

What the theorem shows is that, although the solution point $X_i^{k^*}$ may lie outside the rectangle C^{k^*}, at least one component $x_{i_b}^{k^*}$ must lie in $(l_{i_b}^{k^*}, L_{i_b}^{k^*})$ if $LB(N^{k^*}) < ZUP^{2k+1}$.

Before the steps of the algorithm are given there is a final remark to be made. As mentioned in Chapter 2, branch-and-bound technique is an intelligently structured search of the space of all feasible solutions, and in order to reduce the number of nodes created some observations can be made. At any intermediate stage, let node N^j be a bounding problem in the set of intermediate nodes (i.e., node that has not been branched from). and let total number of nodes be m and $\underline{x}^j \neq XBEST^m$. According to section 2.2, if $\psi^j(\underline{x}^j) \geq f^o(XBEST^m)$, the N^j is to be fathomed and excluded from further consideration. However, this is not the case in this algorithm i.e., no such fathoming rule can be stated since the condition is automatically taken into account. That is, at the end of any stage, N^j (or any other node with the same property) may or may not be selected until the optimal solution has been obtained. If N^j is not selected for branching, then there will be no problem. If N^j is selected for branching, then optimality condition is satisfied since $LB(N^j) = \psi^j(x^j) \geq f^o(XBEST^1) = ZUP^1$, where 1 is the number of nodes created until node N^j has been selected for branching and $ZUP^1 \leq ZUP^m$.

Algorithm stops with optimal solution $XBEST^1$ and optimal value ZUP^1.

3.5 Steps of the Algorithm

1. k=1. Find $LB(N^1)$, ZUP^1, $XBEST^1$. Go to step 2.
2. Find k^* such that $LB(N^{k^*})$ is the minimum of $LB(N^k)$ over all intermediate nodes N^k. If $LB(N^{k^*}) \geq ZUP^{2k+1}$, $XBEST^{2k+1}$ is an optimal solution and ZUP^{2k+1} is the value of the objective function. Otherwise go to step 3.
3. k=k+1. Branch from node N^{k^*} to create nodes N^{2k} and N^{2k+1}. Find $LB(N^{2k})$, $LB(N^{2k+1})$, ZUP^{2k}, ZUP^{2k+1}. Go to step 2.

Steps of the algorithm given above comprise only the general framework of the algorithm. The details of the steps will be given in the flow chart for the computer program of the algorithm.

3.6 Convergence of the Algorithm

Theorem : The algorithm terminates at an optimal solution after a finite number of iterations.

Proof : The linear polyhedron $G \cap C$ possesses a finite number of extreme points and we assume that each point x^k that

solves P_k is one of these. Let E be this finite set of extreme points and let E_i, $i=1,2,\ldots,n$ be the set of projections of the elements of E on the i^{th} coordinate axis. Define $\bar{E}_i = E_i \cup \{l_i\} \cup \{L_i\}$ and $\bar{E} = \bar{E}_1 x \bar{E}_2 x \ldots x \bar{E}_{n_i}$; $\bar{E}$ is a finite set of points in C. The operation of the algorithm is such that every rectangle C^k is defined by an appropriate set of 2^n points of $\bar{E}$; i.e., 2^n extreme points of C^k are elements of $\bar{E}$. There are, therefore, only a finite number of possible rectangles C^k that may be obtained in the course of the algorithm. Since it generates two new rectangels at each stage, it can proceed for only a finite number of stages.

3.7 Example Problem

The following problem is to illustrate how the algorithm proceeds and converges to optimal solution.

Minimize $z = f_1(x_1) + f_2(x_2) + x_3$

Subject to $x_1 + 4x_2 + 2x_3 \geq 8$

$$3x_1 + 2x_2 \geq 6$$
$$0 \leq x_1 \leq 16$$
$$0 \leq x_2 \leq 9$$
$$0 \leq x_3 \leq 8$$

where $f_1(x_1) = 16+8\ x_1^{1/2}$ if $x_1 > 0$

0 if $x_1 = 0$

$f_2(x_2) = 9+3x_2$ if $x_2 > 0$

0 if $x_2 = 0$

Solution

Upper bound of the problem is set to ZUP ∞

Node 1

Convex envelopes $C_1^1 = \frac{48}{16} = 3$, $C_2^1 = \frac{36}{9} = 4$

Problem to be solved at node 1

$$\text{Min } Z=3x_1+4x_2+x_3$$

$$x_1+4x_2+2x_3 \geq 8$$
$$3x_1+2x_2 \geq 6$$

$$0 \leq x_1 \leq 16$$
$$0 \leq x_2 \leq 9$$
$$0 \leq x_3 \leq 8$$

Solution at node 1

$Z=ZL(1)=9 \quad x_1^1 = 2,\ x_2^1 = 0,\ x_3^1 = 3$

Value of the original objective function at X^1 is $ZU(1) = (16+8(2^{1/2})+0+3 = 30.3137$

Since $ZU(1)<ZUP$, current upper bound becomes $ZUP=ZU(1)=30.3137$ and $XBEST^1=\underline{x}^1$

Optimality test

Since ZUP ZL(1) this solution is not optimal, and we must continue with branching. Nodes 2 and 3 will be created. First branching variable is to be found.

a. $f_1(x_1^1) = 16+8(2^{1/2})=27.313$

$\Psi_1^1(x_1^1)=(3)(2)=6$

$f_1(x_1^1) - \Psi_1^1(x_1^1) = 21.313$

b. $f_2(x_2^1) = 0$

$\Psi_2^1(x_2^1)=0$

$f_2(x^1_2) - \Psi_2^1(x_2^1)=0$

$\therefore$ x_1 is the branching variable.

At nodes 2 and 3 only the slope of x_1 in the objective function will change, since the interval $[l_1^1, L_1^1] = [0,16]$ is divided into two intervals $[l_1^1, x_1^1] = [l_1^2, L_1^2] = [0,2]$ and $[x_1^1, L_1^1] = [l_1^3, L_1^3] = [2,16]$.

Node 2

Coefficient of x_1, in this approximate problem is the slope of the linear function whose endpoints are $[l_1^2, L_1^2] = [0,2]$ and $[f_1(l_1^2), f_1(L^2_1)] = [0,27.313]$ thus $c_1^2 = \frac{27.313}{2} = 13.656$

and other coefficients remain the same i.e., $c_2^2 = C_2^1 = 4$.

Attitudes to be added F=0.

Problem to be solved at node 2

$$\text{Min } Z = 13.656\, x_1 + 4x_2 + x_3$$

$$x_1 + 4x_2 + 2x_3 \geq 8$$

$$3x_1 + 2x_2 \geq 6$$

$$0 \leq x_1 \leq 16$$

$$0 \leq x_2 \leq 9$$

$$0 \leq x_3 \leq 8$$

Solution at node 2

$z=ZL(2)=12, \quad x_1^2=0, \quad x_2^2=3, \quad x_3^2=0$

Value of the original objective function at $\underline{x}^2$ is $ZU(2)=0+(9+3(3))+0=18$

Since $ZU(2)<ZUP$ current upper bound becomes $ZUP=ZU(2)=18$ and $XBEST^2=\underline{x}^2$.

Node 3

Coefficient of X_1 in this approximate problem is the slope of the linear function whose endpoints are $[l_1^3, L_1^3]=[2,16]$ and $[f_1(l_1^3), f_1(L_1^3)]=[27.313,48]$ thus $C_1^3=1.477$ and other coefficients remain the same as those in node 1,i.e., $C_2^3=C_2^1=4$. Altitude to be added (i.e., intercept of the function with the vertical axis) is F=24.35.

Problem to be solved at node 3

Min Z $1.477x_1+4x_2+x_3$

$$x_1+4x_2+2x_3 \geq 8$$
$$3x_1+2x_2 \geq 6$$

$0 \leq x_1 \leq 16$

$0 \leq x_2 \leq 9$

$0 \leq x_3 \leq 8$

Solution at node 3

$Z=5.995 \quad ZL(3)=Z+F\ 30.3137, \quad x_1^3 = 2, \quad x_2^3 = 0,$

$x_3^3 = 3$. Value of the original objective function at $\underline{x}^3$ is $ZU(3) = 30.317$. Since $ZU(3) > ZUP$ upper bound does not change, i.e., $ZUP=18 \quad XBEST^2 = \underline{X}^2$

Select the node from among nodes 2 and 3, with the minimum lower bound.

Lower bound of node 2 $ZL(2)=12$ and lower bound of node 3 $ZL(3) = 30.316$. Since $ZL(2) < ZL(3)$ node2is selected for branching.

Optimality Test

$ZUP=18 \neq ZL(2)=12$ therefore solution is not yet optimal and we must continue with branching from node 2.

Nodes 4 and 5 will be created. First branching variable at node 2 is to be found.

a. $f_1(x_1^2) = 0$

$\Psi_1^2(x_1^2) = 0$

$f_1(x_1^2) - \Psi_1^2(x_1^2) = 0$

b. $f_2(x_2^2) = 16$

$\Psi_2^2(x_2^2) = 12$

$f_2(x_2^2) - \Psi_2^2(x_2^2) = 4$

. . x_2 is the branching variable.

At nodes 4 and 5, only the coefficient of x_2 will be different from the coefficients of all variables at node 2. Interval $[l_2^2, L_2^2]$ is divided into two intervals. $[l_2^2, x_2^2] = [l_2^4, L_2^4] = [0,3]$ and $[x_2^5, L_2^5] = [l_2^5, L_2^5] = [3,9]$.

Node 4

Coefficient of x_2 in this approximate problem is the slope of linear function whose endpoints are $[l_2^4, L_2^4] = [0,3]$ and $[f_2(l_2^4), f_2(L_2^4)] = [0.18]$ thus $C_2^4 = 18/3 = 6$ and other coefficients are the same as those in node 2, i.e., $C_1^4 = C_1^2 = 13.656$ and altitude

to be added F=0

Problem to solved at node 4

$$\text{Min } Z=13.656\ X_1+6X_2+X_3$$

$$x_1+4x_2+2x_3 \geq 8$$

$$3x_1+2x_2 \geq 6$$

$$0 \leq x_1 \leq 16$$

$$0 \leq x_2 \leq 9$$

$$0 \leq x_3 \leq 8$$

Solution at node 4

$Z=ZL(4)=18 \quad x_1^4=0,\ x_2^4=3,\ x_3^4=0$ and value of original objective function at node 4.

$ZU(4)=0+(9+3(3))+0=18$. Since $ZU(4)=ZUP$ upper bound of the problem remains the same, i.e., ZUP=18 $XBEST^4=\underline{X}^2$ (or $\underline{X}^4$)

Node 5

Coefficient of x_2 in this approximate problem is the slope of the linear function whose endpoints are$[l_2^5,\ L_2^5]=[3,9]$and

$[f(l_2^5), f_2(L_2^5)] = [18, 36]$ thus $C_2^5 = 3$ and other coeeficients remain the same as those at node 2. Altitudes to be added to the objective function of LP solution is F=9.

Problem to be solved at node 5

$$\text{Min } Z = 13.656x_1 + 3x_2 + x_3$$

$$x_1 + 4x_2 + 2x_3 \geq 8$$

$$3x_1 + 2x_2 \geq 6$$

$$0 \leq x_1 \leq 16$$

$$0 \leq x_2 \leq 9$$

$$0 \leq x_3 \leq 8$$

Solution at node 5

Z=9 $ZL(5) = Z+F = 9+9 = 18$, $x_1^5 = 0$, $x_2^5 = 3$, $x_3^5 = 0$ and value of original objective function at $\underline{x}^5$ is $ZU(5) = 0+(9+3(3))+0 = 18$. Since $ZU(5) = ZUP$ upper bound of the problem remains the same i.e., ZUP=18 and $XBEST^5 = X^2$.

Select the node from among the intermediate nodes 3,4, and 5, with the minimum lower bound.

$ZL(4) = ZL(5) < ZL(3)$ therefore node 4 or 5 is selected. Let us select node 4.

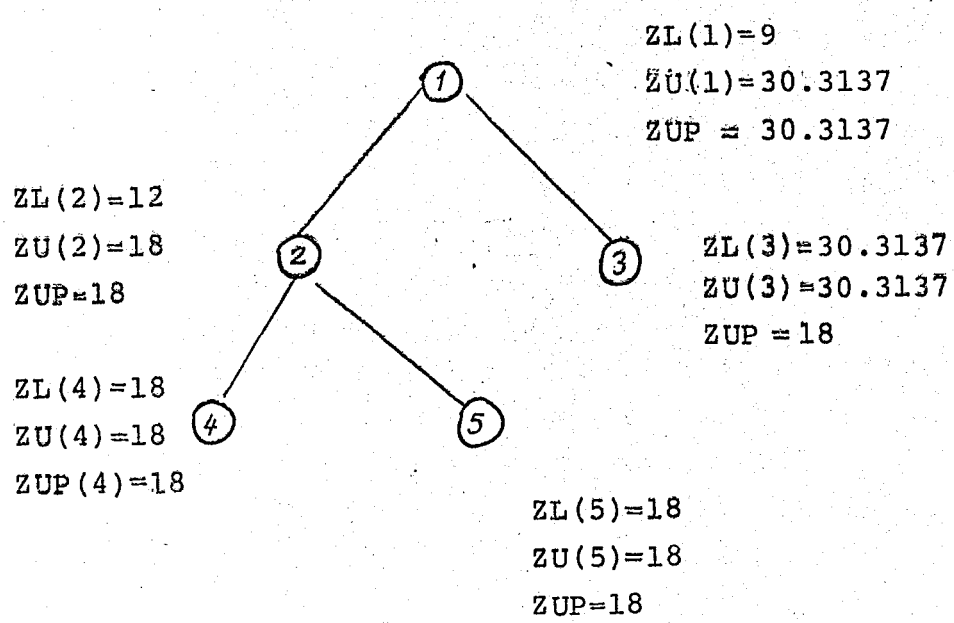
Optimality Test

ZUP=18=ZL(4)=18. Therefore optimal solution has been obtained. Algorithm terminates.

Optimal solution of the problem

ZUP=18 $x_1 = 0$, $x_2 = 3$, $x_3 = 0$

Tree representation of the problem is as follows :



CHAPTER 4

COMPUTER PROGRAM OF THE ALGORITHM

4.1 General

The program is composed of a main program, MAIN, and various subprograms which are called in the main program. One of the subprograms is SMPLEX, which is a FORTRAN IV code of Simplex algorithm. This subprogram has been developed by Land and Powell (17) , to solve LP problems. SMPLEX contains several subprograms. Some of the subprograms have been removed and some have been modified. DATA and IPRINT subroutines of SPLEX have been deleted and rewritten in consistence with the requirements of the problem intrudced in this study. Subroutine SPRINT of SMPLEX has been removed completely since it is unnecessary in the program. Some minor modifications have been made in subroutines IEXIT, COPY to match them to our program. A new subroutine, ARRGMT, has been writtten for some tolerances used in SMPLEX. As mentioned SMPLEX is one of the subprograms called in MAIN and its subroutines will not be explained here. However, the details of the subprograms of SMPLEX are available in (17) for those interested. In the following, we will briefly, refer to what happens in the main and subprograms developed in this thesis.

4.2 MAIN PROGRAM

Main program contains a number of subroutine subprograms. DATA is the first subroutine which reads in a problem for solution. After the execution of DATA. convex envelope of the objective function over set C is computed and subroutine SMPLEX is called. This subroutine solves problem 1(P_1) at node 1. If the solution to P_1 is infeasible or unbounded, the execution stops, this implies an error in the data input. If the solution to P_1 is optimal, execution proceeds and subroutine ACOBJF, which computes the value of the objective function of the original problem, is called. Optimality test is made, if solution is optimal control is transferred to subroutine SOLUTN to print out the results, otherwise subroutine BRVAR is called to find the branching variable at this node. Subroutines BND and CFBND are called to obtain the bounds and objective function coefficients to be used in the following two problems created. For each of the two new problems subroutine SMPLEX is called and then subroutine FXCH is called and associated altitudes are added to the value of the objective functions computed in SMPLEX for the problems considered. Value of the original objective function for the solution at this node is computed and if it is less than the upper bound of the problem (ZUP), this new value becomes the UB of the problem. Subroutine BRVAR is called to find the branching variable at this node. After this sequence has been completed for the two created problems, the intermediate node with minimum lower bound is looked for and an optimality test is made if solution is optimal control is transferred to subroutine SOLUTN, otherwise branching occurs from

the node with minimum LB and the above steps are repeated.

4.3 DATA

In appendix there isa list of input formats for the program. In this subroutine all necessary data for the problem are read in. Subroutine SMPLEX had a DATA subroutine, but this has been deleted nd a new DATA subroutine consistent with the problem has been written.

The first read statement reads MAXNN, maximum number of nodes to be involved to obtain a solution, PRECIS, percentage at which a 'acceptable solution' is obtained. The second read statement reads NFXOPI, an indicator whether there exists any fixed charge of the nonconvex variables, and IPROP2, an indicator whether the solutions to subproblems are printed. The third read statement reads NNVC, number of concave variables and NLV, number of linear variables in the problem. The fourth read statement reads values of esssentially used in subroutine SMPLEX. M is the number of constraints, N is the number of variables, ISBND is the number of upper bounded variables, ITRMAX is the maximum number of iterations allowed in SMPLEX to find the optimal solution to subproblem at a node, IRMAX is teh maximum of reinversions to 'clean up" the basis and improve the accuracy of the SMPLEX solution. After these read statementb, comes the reading of a few sets of cards. The sets are :

1. the fixed cahrges (FX) of concave variables (which may be omitted if NFXOP1=0)

2. the coefficients (D) of the concave variables
3. the exponents (E) of the concave variables
4. the coefficients (C) of linear variables following concave variables in order (i.e. C(NNVC+1) to C(N))
5. the signs of constraints and the values of right-hand sides
6. the upper bounds
7. the lower bounds of only concave variables (if lower bounds are different from zero)
7. the nonzero elements of constraint matrix, including lower bounded variables as sconstraints.

The details of data formats are given in Appendix.

4.4 ACOBJF (Value of the objective function)

This subroutine computes the value of the objective function for the solution to LP problem at the node considered. Values of variables are obtained after subroutine SMPLEX has been called and these are substituted in the original objective function (recall that all the solutions form LP problems are feasible to the original problem.) If this value of the objective function is less than the upper bound of the problem then this value becomes the new upper bound and values of variable are stored in array XBEST, otherwise the upper bound of the problem remains the same.

4.5 BRVAR (Branching Variable)

This subroutine finds the branching variable at each node,

to be used immediately or later for branching. After the LP problem at the node considered has been solved, the concave variable that will cause the biggest difference (thus the biggest eror in a sense) between the original objective function and objective function to LP is selected for branching. Branching variable at this node is stored in array JBRCAR and value of this variable is stored in array VALBRV.

4.6 BND

This subroutine determines the values of the limits (l and L) of the branching variable for two new problems created by branching. It also computes the coefficients of the branching variable in the objective functions of these two subproblems. Recall that the two problems are different from the problem at previous node in only the limits (l and L) and slope in objective functions of the branching variable.

4.7 CFBND

This subroutine determines the upper and lower limitss (l and L) and the coefficients of concave variables in the two new problems, with the exception of the branching variable. After the call of this subroutine all the data necessary to solve the subproblem at the node become ready. Then subroutine SMPLEX is called to solve the subproblem. Subroutine CFBND is called

only for the first problem created at each stage in that the data provided here remain the same for the second problem of the stage.

4.8 FXCH

This subroutine computes the total altitude to be added to the function value obtained by SMPLEX. When convex envelopes are drawn of the concave functions, their slopes are computed and used in LP problem at the node to find optimal solution. However, some of these convex envelopes, which are linear functions do not pass through the origin and intersect the vertical exis. Therefore, associated altitudes for the problem currently solved are to be computed. This value is found as F and added to the optimal solution coming from SIMPLAX, and lower bound (ZL) to the problem is obtained. See Fig. 4.1.

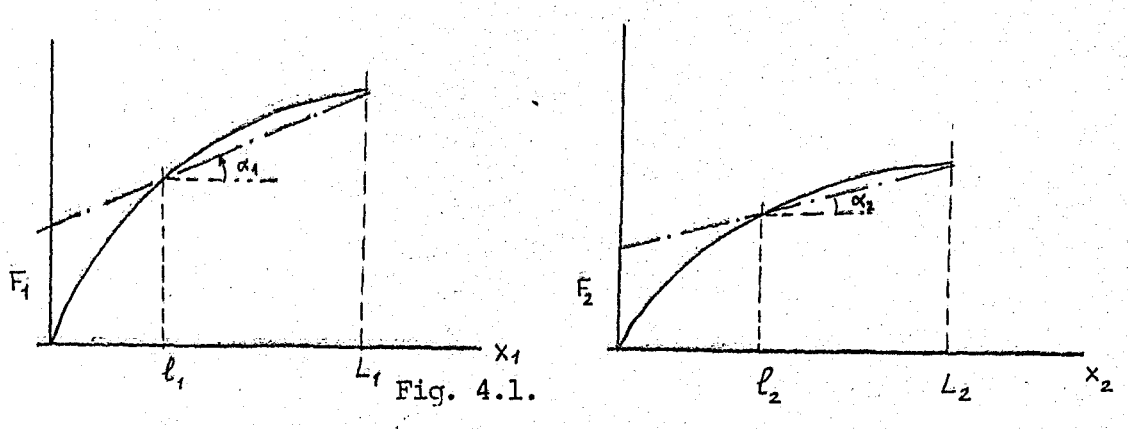


Fig. 4.1.

For a subproblem with two concave variables, where $l_1 \leq x_1 \leq L_1$ and $l_2 \leq x_2 \leq L_2$, $c_1 = \text{tg}\,\alpha_1$ $C_2 = \text{tg}\,\alpha_2$ and associated total altitude to be added to the objective function from SMPLEX is $F = F_1 + F_2$.

4.9 SLTND

As mentioned, when branching occurs two new problems are created (LP problems) and solved. After ZL's to each problem has been found, the intermediate node with minimum ZL is searched for. This subroutine finds the indermediate node with minimum ZL. As soon as this node has been found an optimality check(in the MAIN) is made. If solution is optimal subroutine SOLUTN is called, otherwise a new branching occurs.

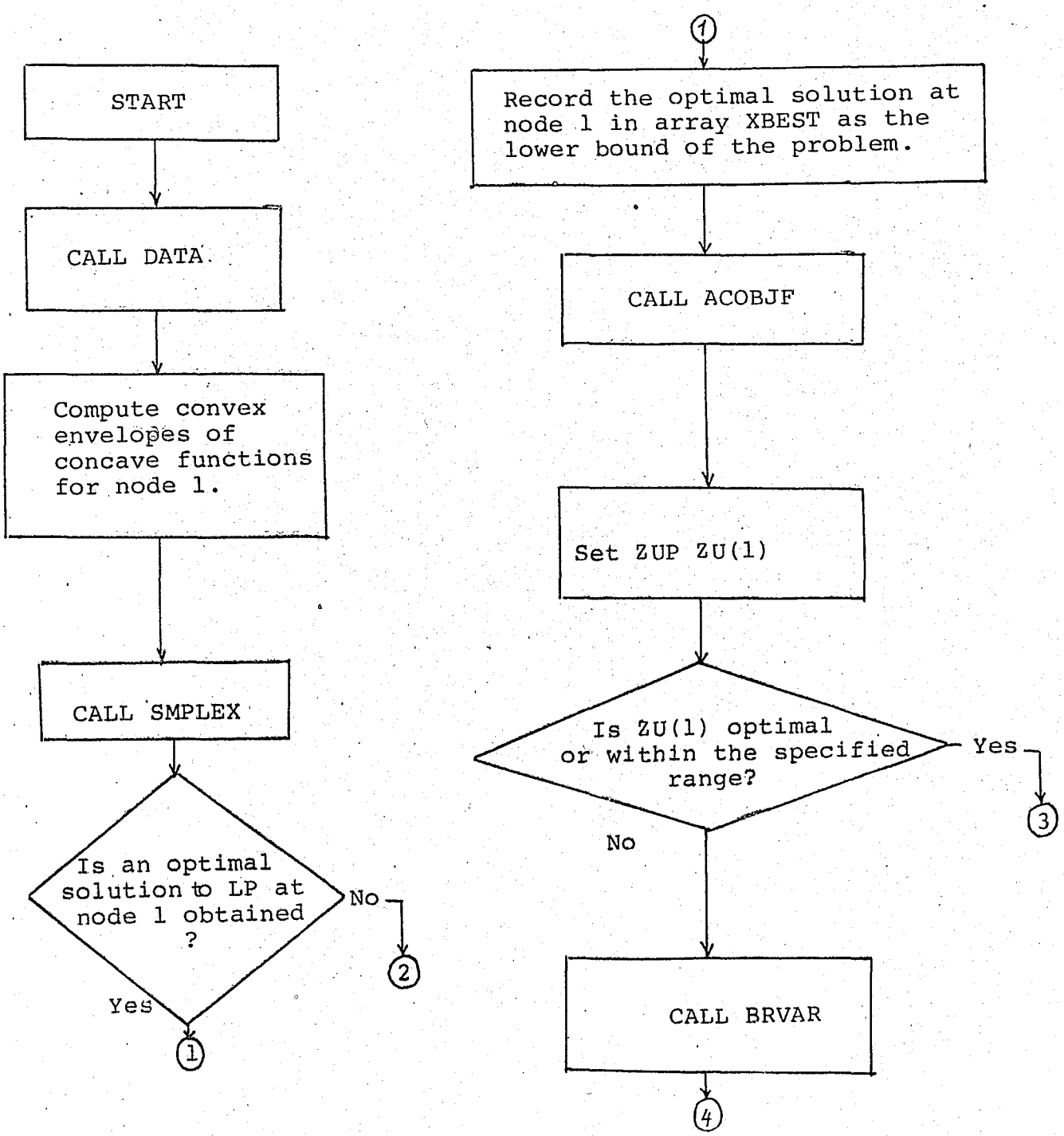
4.10 PRNT

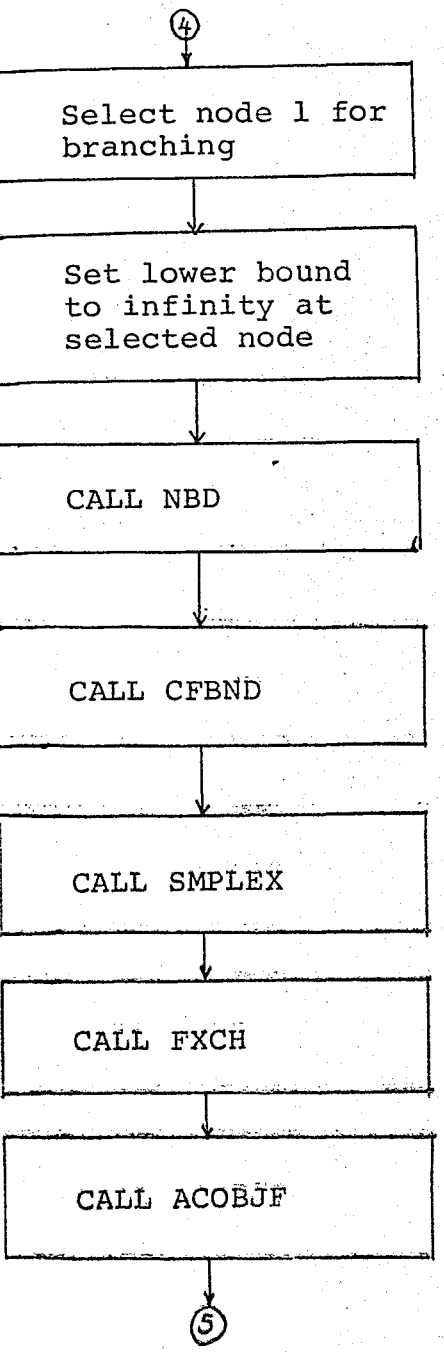
Call of this subroutine is optional. If the solution to each subproblem is desired, IPRO2 is set to 1 then the values of variables, optimal value, value of original objective function and the upper bound of the problem at each subproblem are printed.

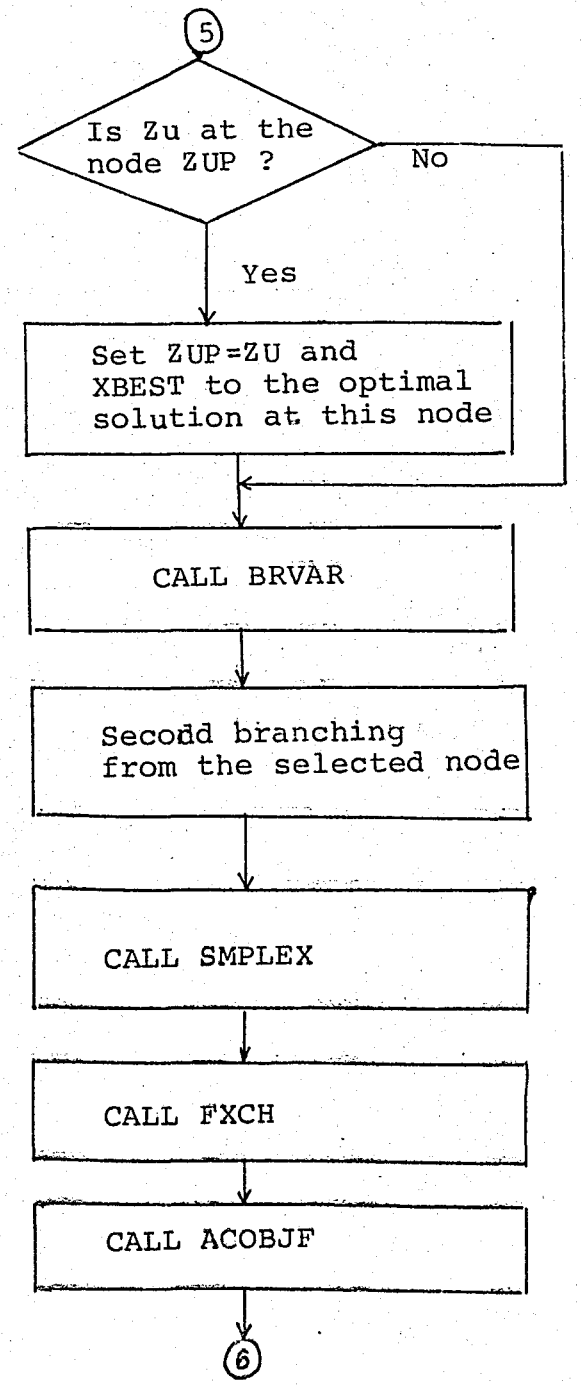
4.11 SOLUTN

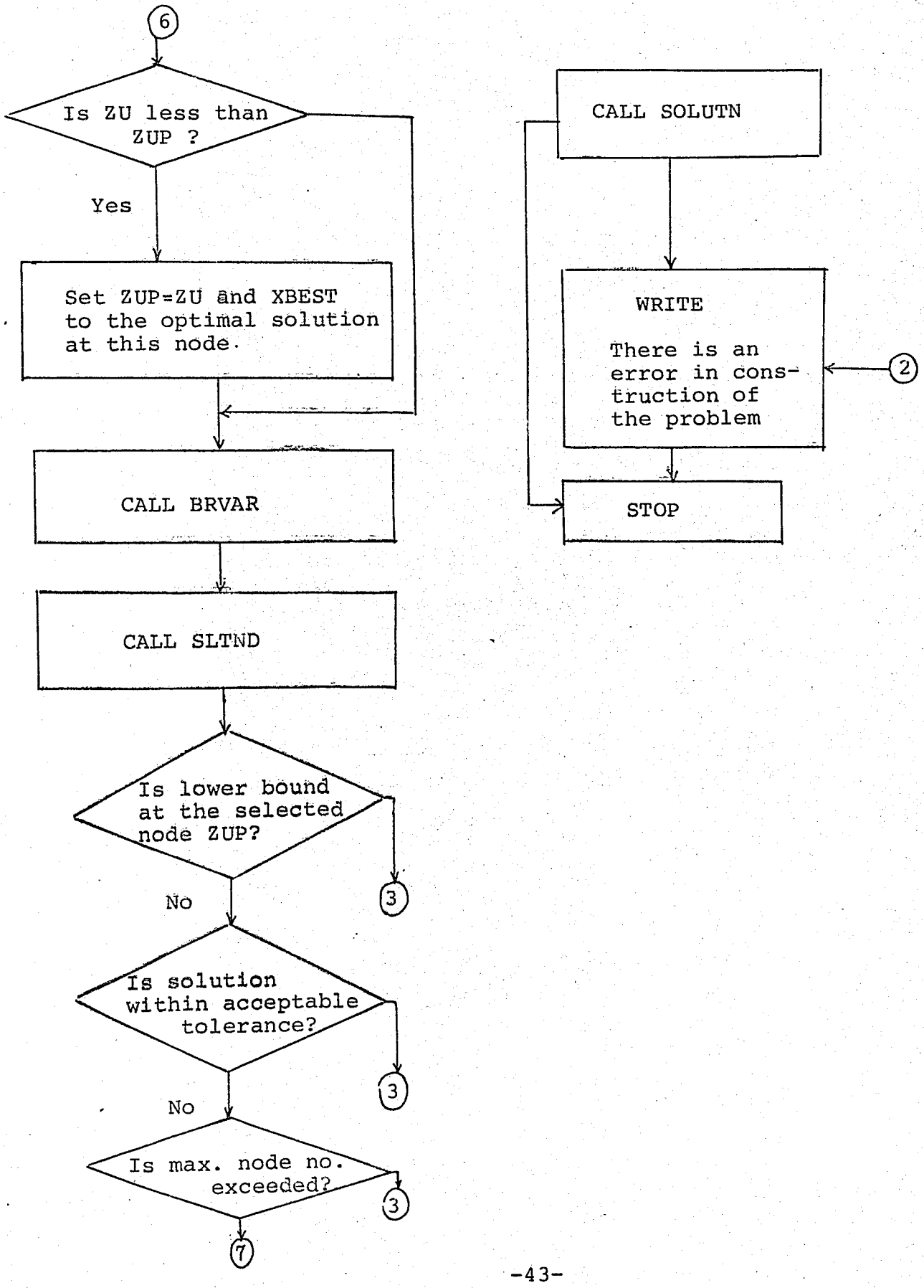
This subroutine is called when optimal solution is found or the solution is within acceptable range or the maximum (specified7 number of iterations is exceeded. When optimal solution is reached, optimal value of the variables and objective function are printed with a message of 'optimal solution.' If the solution is within acceptable range or the number of iterations exceeds a prespecified number, then the best available solution is printed out with a message of 'best solution'. In the case of optimal solution, number of iterations is also printed, in the ltatre cases, the number of iterations, the tolerance raito, and the difference between the upper and lower bounds of the problem are also printed.

4.12 FLOW CHART OF THE ALGORTIHM




4
Select node l for branching
Set lower bound to infinity at selected node
CALL NBD
CALL CFBND
CALL SMPLEX
CALL FXCH
CALL ACOBJF
5


5
Is Zu at the node ZUP ?
No
Yes
Set ZUP=ZU and XBEST to the optimal solution at this node
CALL BRVAR
Secodd branching from the selected node
CALL SMPLEX
CALL FXCH
CALL ACOBJF
6


6
Is ZU less than ZUP ?
Yes
Set ZUP=ZU and XBEST to the optimal solution at this node.
CALL BRVAR
CALL SLTND
Is lower bound at the selected node ZUP?
3
No
Is solution within acceptable tolerance?
3
No
Is max. node no. exceeded?
3
7
CALL SOLUTN
WRITE
There is an error in construction of the problem
2
STOP

4.13 Computational Considerations

Computer program of the algortihm yields an acceptable solution in that the algorithm is stopped when ZUP^{2k+1} is guaranteed to be within 100r percent of the optimal solution value. Here $r>0$ is specified by the user. Another criterion to stop the algorithm with an acceptable solution is the number of nodes to be created. When the number of nodes (subproblems) exceeds a prespecified value. algorithm is stopped and the 'best solution' up to this node is printed out with a message (TOLERANCE) stating how much percent of the optimal solution value is the 'best solution' within.

Computer program has been applied to various problems randomly selected, of the type mentioned in Section 3.3.

$$\text{Minimize } f(x) = \sum_{j=1}^{n} (K_j + d_j . x_j^{e_j}) + \sum_{j=n+1}^{N} c_j X_j \qquad e_j \leq 1$$

$$\text{Subject to } \sum_{j=1}^{N} a_{ij} X_j \leq b_i \qquad i=1,2,..,m$$

$$1_j \leq X_j \leq L_j \qquad j=1,2,...N$$

Various coefficients of these random problems lie in the ranges

$$0 \leq c_j \leq 200$$

$$0 \leq K_j \leq 55$$

$$-10 \leq a_{ij} \leq 25$$

$$0 \leq b_i \leq 230$$

$$0 \leq d_j \leq 500$$

$$250 \leq L_j \leq 1000$$

The sizes of the generated problems are given by $(n \times M) = (5 \times 20)$ (10×20). Same problems are solved by replacing K_j with $2K_j$ and $3K_j$ and by changing the value of L_j appropriately. Computational results are summarized in the following table. (UNIVAC 1106)*

---

* The runs in the table were performed on UNIVAC 1106 computer

Problem No.			L_j	CUP (sec)	r
1	$(m \times N) = (5 \times 20)$	K_j	250	4.27	$.17 \times 10^{-7}$
2			500	5.90	.0053
3			750	5.63	.0056
4		$2K_j$	250	5.28	$.96 \times 10^{-8}$
5			500	5.41	$.90 \times 10^{-8}$
6			750	5.29	.00029
7		$3K_j$	250	3.53	$.88 \times 10^{-8}$
8			500	3.52	$.88 \times 10^{-8}$
9			750	3.52	$.88 \times 10^{-8}$
10	$(m \times N) = (10 \times 20)$	K_j	250	29.28	$.17 \times 10^{-3}$
11			500	29.17	$.93 \times 10^{-3}$
12			1000	30.74	$.51 \times 10^{-3}$
13		$2K_j$	250	37.98	$.84 \times 10^{-3}$
14			500	44.03	0.021
15			1000	42.71	0.031
16		$3K_j$	250	28.07	$.48 \times 10^{-3}$
17			500	33.25	$.59 \times 10^{-3}$
18			1000	40.15	.014

CHAPTER 5

APPLICATIONS OF THE ALGORITHM

5.1 Introduction

Algorithm and computer program presented in the previous chapters can be used to solve various optimization problems. As has been mentioned, in some of the major industries capacity expansion costs are concave functions of the capacity created and production costs are concave functions of the amount produced. Such a problem can be handled and oslved by the algorithm presented. A dynamic (in time) problem will be considered and the present value of the costs involved will be minimized. First a signle plant-single commodity problem and then a multi plant-single commodity problem and finally, a multi-plant-multi commodity problem will be presented.

5.2 Problem Formulation

1. Single Plant - Single Commodity Problem

A Company has one plant, several warehouses and several demand centers. Goods produced in the plant are distributed to the

warehouses for storage and then distributed to the demand centers from them. Demand on goods increases over time and the capacity of the plant will have to be increased appropriately to meet increasing demand of each period. Shortages are not aloowed. This problem is to determine the optimal capacity expansion, production, inventory, and distribution program over time.

Mathematical formulation of the problem is as follows. Let y_t be the capacity added in period t, which is also available for use in the same period. Let P_t be amount of production in time period t (all production is shipped to the warehouses) and s_{jt} be the inventory level in warehouse j at the end of period t_1, and X_{ijt} be the amount shipped between nodes i and j in period t.

Define $Y_t(y_t)$ to be the cost of expanding capacity, $P_t(p_t)$ production cost, $S_{jt}(s_{jt})$ inventory cost in warehouse j, and $X_{ijt}(x_{ijt})$ cost of shipping X_{ijt} units in period t. Let d_{jt} be the demand at demand center k in period t. All cost values are discounted to their present values by some appropriate discount factors. Let $_t$ show this discount factor. Let T be the set of time periods, W be the set of warehouses and D be the set of demand centers and A be the set of arcs through the network. We may now state the problem as :

Choose all y_t, P_t, s_{jt}, X_{ijt} to

$$\text{Minimize} \quad \sum_{t\in T} \alpha_t \left[Y_t(y_t) + P_t(p_t) + \sum_{j\in w} S_{jt}(s_{jt}) + \sum_{(ij)\in A} X_{ijt}(x_{ijt}) \right]$$

Subject to $p_t \leq \sum_{\zeta=0}^{t} y_\zeta$ $\quad t \in T$

$P_t = \sum_{j \in W} X_{1jt}$ $\quad t \in T$ and node number of plant through the network is 1.

$s_{j,t-1} + \sum_i x_{ijt} - \sum_k X_{jkt} = s_{jt}$ $\quad t \in T$, $j \in W$ and i and k are nodes incident to j.

$\sum_{i \in W} X_{ijt} = d_{jt}$ $\quad j \in D$

Bounds on Variables

The cost functions $Y_t(y_t)$ and $P_t(p_t)$ are, to be more realistic, nonlinear concave functions. Cost functions $S_{jt}(s_{jt})$ and $X_{ijt}(x_{ijt})$ are linear. Therefore the problem can be solved by the algorithm presented in Chapter 3.

2. Multi plant - Single Commodity Problem

In this case, the problem is similar to the previous one

except that the company has several plants in which goods are produced.

Mathmematical formulation of the problem is as follows . Let p_{it} be the amount produced in plant i in period t (all production is shipped to the warehouses) y_{it} be capacity added in plant i in period t, which is also available for use in period t, s_{jt} be the level of inventory in warehouse j at the end of period t, and X_{ijt} be the amount shipped between nodes i and j in period t. Define associated cost functions $Y_{it}(y_{it})$.

$P_{it}(p_{it})$, $S_{jt}(s_{jt})$, $X_{ijt}(x_{ijt})$ similar to the previous problem. Let P be the set of plants, W be the set of warehouses, D be the set of demand centers, T be the set of time periods , and A be the set of ares through the network. α_t denotes the discount factor.

We may now state the problam as : Choose all y_{it}, p_{it}, s_{jt}, X_{ijt} to

$$\text{Minimize} \quad \sum_{t\in T} \alpha_t \Big[\sum_{i\in P} Y_{it}(y_{it}) + \sum_{i\in P} P_{it}(p_{it}) + \sum_{j\in W} S_{jt}(s_{jt}) + \sum_{(ij)\in A} X_{ijt}(x_{ijt}) \Big]$$

$$\text{subject to} \quad p_{it} \leq \sum_{\zeta=0}^{t} y_{i\zeta} \qquad i\in P,\ t\in T$$

$$p_{it} = \sum_{j\in W} X_{ijt} \qquad i\in P,\ t\in T$$

$$S_{j,t-1} + \sum_i x_{ijt} - \sum_k X_{jkt} = S_{jt} \qquad j \in w,\ t \in T \text{ and } i \text{ and } k \text{ are inodes incident to } j.$$

$$\sum_{i \in w} x_{ijt} = d_{jt} \qquad J \in D,\ t \in T$$

Bounds On Variables

3. Multi-plant - Multi Commodity Problem

This last case is the most general case. A company has several plants, warehouses, demand centers and several products are produced in plants. Mathematical formulation of the model is similar to the previous ones. Let P_{ikt} be the amount of product k produced in plant i in period t (all products are shipped to the warehouses), and y_{ikt} be the increase in the capacity of product k in plant i in period t, and S_{jkt} be the level of inventory of product k at the end of period t in warehouse j, and X_{ijkt} be the amount of product k shipped between regions i and j in period t. Associated cost function are $P_{ikt}(p_{ikt})$, $Y_{ikt}(y_{ikt})$, $S_{jkt}(s_{jkt})$, $X_{ijkt}(x_{ijkt})$. Let d_{jkt} be the demand on product k at demand center j in period t and $_t$ be the discount factor. In addition to the sets, P,W,D,T, and A defined in the previous problem, let K show the set of products

We may now state the problem as : Choose $p_{ikt}, y_{ikt}, S_{jkt}, X_{ijkt}$, to

$$\text{Min} \sum_{t\in T} \alpha_t \Big[\sum_{i\in P} \sum_{k\in K} \{ Y_{ikt}(y_{ikt}) + P_{ikt}(p_{ikt}) \} + \sum_{j\in w} \sum_{k\in K} S_{jkt}(s_{jkt}) + \sum_{k\in K} \sum_{(i,j)\in A} X_{ijkt}(X_{ijkt}) \Big]$$

$$\text{S.T.} \quad P_{ikt} \leq \sum_{\zeta=0}^{t} y_{ik\zeta} \qquad i\in P,\ k\in K,\ t\in T$$

$$P_{ikt} = \sum_{j\in w} X_{ijkt} \qquad i\in P,\ k\in K,\ t\in T$$

$$S_{j,k,t-1} + \sum_{i} X_{ijkt} - \sum_{q} x_{jqkt} = s_{j,k,t} \qquad j\in w,\ k\in K,\ t\in T$$

and i and q are nodes incident to j.

$$\sum_{i\in w} x_{ijkt} = d_{jkt} \qquad j\in D,\ k\in K,\ t\in T$$

Bounds on Variables

CHAPTER 6

6.1. CONCLUSIONS AND EXTENSIONS

In this study, we have presented a solution technique and developed its computer programs to solve a class of optimization problems which has a wide spectrum of applications. The computer program developed can be used as a package program when the data are provided according to the format specifications presented in Appendix A. The program can be used directly if the problem has the features mentioned previously. It can also be used as subprogram for optimization problems, requiring such a solution procedure for its subproblems. In this case, some minor modifications are to be made in subroutine DATA, to make the program handle the parameters of the problem.

The study can be extended to cover some different kinds of objective functions which are encountered in some real life problems. In communication and pipeline network constructions a different kind of nonconvex cost function is encountered. However, the objective function is still separable. This new problem is close to the one, for which a solution technique is offered in this study.

Cost function in the above cited real life problems is a nonconvex function of the amount of activity and may be represented as ing Fig. 6.1.

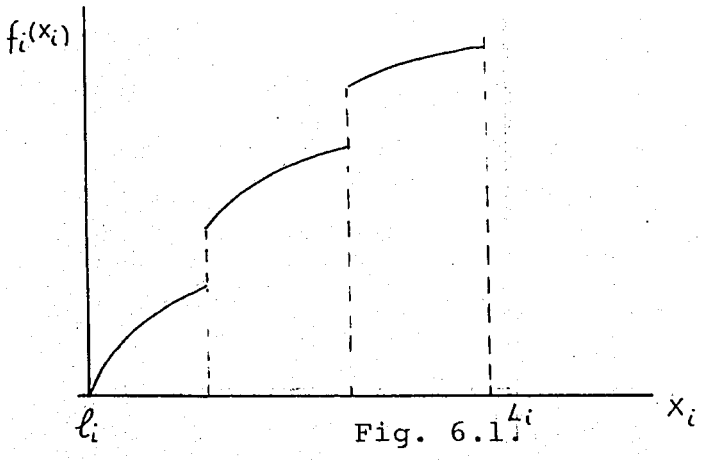


Fig. 6.1

Total cost function is a separable nonconvex cost function. This total cost function is minimized over a linear constraint set.

$$\text{Minimize} \quad f(\underline{x}) = \sum_{i=i}^{n} f_i(x_i) + \sum_{i=n+1}^{N} g_i(x_i)$$

$$\underline{X} \in G = \{\underline{x} \mid A\underline{x} \leqslant \underline{b}\}$$

$$\underline{X} \in C = \{\underline{x} \mid \underline{l} \leqslant \underline{x} \leqslant \underline{L}\}$$

For this new problem, similar theoretical development can be made but some modifications will be necessary. Again, in this problem

convex envelopes will be needed and they will be piecewise linear convex functions. Since the objective function is separable and each f_i is defined over $[l_i, L_i]$. then convex envelope of f is equal to the sums of the convex envelopes of f_i. Let ψ_i be the convex envelope of f_i. Inconsistent with the terminology used in the text, for subproblem k let ψ_i^k be the convex piecewise linear function passing through the endpoints of the interval $[l_i^k, L_i^k]$ then $\psi_i^k = c_i' x' + c_i'' x_i'' + \cdots + c_i^{(m)} x_i^{(m)}$ where $0 < c_i' < c_i'' < \quad < c_i^m$ and m is the number of the intervals to obtain a convex piecewise linear function (convex envelope) in $[l_i^k, L_i^k]$ for subproblem k. The following facts are evident :

$$\psi_i^k \leq f_i \qquad \text{if} \quad x_i \in [l_i^k, L_i^k]$$

$$\psi_i^k \geq f_i \qquad \text{if} \quad x_i \notin [l_i^k, L_i^k]$$

Let $$\psi^k(\underline{x}) = \sum_{i=1}^{n} \psi_i^k + \sum_{i=n+1}^{N} g_i(x_i)$$

It is obvious that

$$\psi^k(\underline{x}) \leq f(\underline{x}) \qquad \text{if } \underline{x} \in C^k$$

Thus the minimum of ψ^k over GnC^k is less than or equal to the minimum of f over GnC^k. In this case constraint set of the problem cannot remain the same since lower and upper bounds of the variables for each subproblem k will change. Let $H=\{\underline{x} \mid \underline{x} \leq \mu\}$ where μ_i is the lengthof the subinterval in $[l_i^k, L_i^k]$ over which pieces of the piecewise linear function of x_i are valid.

Then the lower bound on the optimal value of f at node k :

$$LB\ (N^k) = \min \{ \psi^k(\underline{x}) \mid x \in G \cap C \cap H \}$$

All the other considerations and the steps of the algorithm remain the same as in Chapter 3.

APPENDIX A

DATA INPUT FORMAT

The computer program can solve optimization problems of the following type. An optimal solution may not be obtained with the specified number of iterations (MAXNN), however, in this case the best solution up to this termination point is printed.

$$\text{Minimize } f(\underline{x}) = \sum_{i=1}^{n} f_i(x_i) + \sum_{i=n+1}^{N} g_i(x_i)$$

$$\underline{x} \varepsilon G = \{\underline{x} \mid A\underline{x} \leqslant \underline{b}\}$$

$$\underline{x} \varepsilon C = \{\underline{x} \mid \underline{l} \leqslant \underline{x} \leqslant \underline{L}\}$$

where f_i is a concave function of the following type.

$$f_i(x_i) = FX + B(X_i) \qquad \text{if } x_i > 0$$

$$\text{if } x_i = 0$$

where FX is the fixed charge, B coefficient and E$\leqslant$1 exponent of the variable. g_i is a linear function.

The subroutine DATA of the computer program has been written to handle concave functions of the above type. However, any separable concave function can be handled in the same way. To achieve this, subroutine DATA is to be altered to read the parameters, and the form of this new $f_i(x_i)$ is to replace the existing form in MAIN, ACOBJF, BF BND.

DATA INPUT FORMAT

Card 1 Col - 15 MAXNN(I5) Maximum number of nodes allowed to find the optimal or best possible solution. Its value is the dimension of ZL, JBRVAR,NPN, FC, BNDLW, BNDUP, CEE in COMMON/NCVXAL/. MAXNN is an odd number.

6-15 PRECIS (F10.3) Percentage at which an 'aaceptable solution' is obtained. Algorithm is stopped when ZUP is guaranteed to be within 100xPRECIS percent of the optimal solution.

Card 2 1 - 5 NFXØP1 (I5) Indicator whether all concave variabl do not have fixed charges.

0 None of the concave variables ha fixed charges. (Do not punch zer in reading fixed charges.)

1 Some or all of the concave variables have fixed charges. (Punch all of them).

6-10 IPRØP2 (I5) Indicator whether the solutions to subproblems at each node are printed or not.

0 Do not print solutions to subproblems.

1 Print solutions to subproblems.

Card 3	Col 1-5	NNCV(I5)	Number of concave variables. Its value is the dimension of FX,D,E, BL,BU,CE,BLW,BUP,FCVI in COMMON/NCVXAL/.
	6-10	NLV (I5)	Number of linear variables.
Card 4	1-10	MAXA (I10)	Maximum number of nonzero elements un constraint matrix. Its value is the dimension of AA,JCOL, in COMMON/AREF/.
	11-20	MXSIZE (I10)	Maximum number of variables and the constraints. Its value is the dimension of INV, XBASIS, XR,YBASIS, YR, and GR in COMMON.
Card 5	1-10	M (I10)	Number of constraints. Its valua is the dimension of the arrays B,G ISEFF, S, SLACK, Y in COMMON and the dimension of IROW in COMMON/ AREF/ is M+1.
	11-20	N (I10)	Number of variables (in SMPLEX), excluding slack variables. Its value is the dimension of the arrays BOUND, C, INBASE,PIV,X, and YAC in COMMON AND XBEST in COMMON/NCVX2/. N = NNCV+NLV.
	21-30	ISBND (I10)	Number of upper bounded variables.

	31-40 ITRMAX	(I10)	Maximum number of iterations permitted for solving on LP problem. If it is zero on the data card, it is set to 3(M+N+ISBND).
	41-50 IRMAX	(I10)	Maximum number of reinversions that are permitted in solving an LP problem.
ard 6	Col 1-10 FX(1)	(F10.0)	Fixed charge of the first concave variable.
	11-20 FX(12)	(F10.0)	
	.		
	.		
	.		
	71-80 FX(8)	(F10.0)	

Card 6' is repeated until fixed charges of all concave variables have been read in, up to eight per card. If all the fixed charges are zero, then by using NFXOP1 0 the sequence of 'Card 6' can be omitted.

ard 7	Col1-10 D(1)	(F10.2)	Coefficient of the first concave variable.
	11-20 D(2)	(F10.2)	
	.		
	.		
	.		
	71-80 D(8)	(F10.2)	

'Card 7' ib repeated until the coefficients of all concave variables, in the objective function have been specified, up to eight per cent.

Card 8	Col 1-10 E(1)	(F10.2)	Exponent of the first concave variable
	11-20 E(2)	(F10.2)	
	.		
	.		
	71-80 E(8)	(F10.2)	

(Card 8' is repeated until the exponents of all concave variables in the objective function, have been read in, up to eight per card.

'Card 9'	Col 1-10 C(NNCV 1)	(F10.2)	Coefficient of the first linear variable following the last concave variable in the objective function.
	11-20 C(NNCV 2)	(F10.2)	
	.		
	.		
	71-80 C (NNCV 8)	(F10.2)	

(Card 9' is repeated until the coefficients of all linear variables (following the concave variable in the objective function) have been read in, up to eight per card.

'Card 10' Col 1-2 S(1) (F2.0) sign of the first constraint
0 equality
1 less-than-or-equal
2 greater than or equal

9-10 B(1) (F8.0) Right-hand side of the first constraint.

-1-12 S(2) (F2.0)

13-20 B(2) (F8.0)

.
.
.

71-72 S(8)
73-80 B(8) (F8.0)

'Card 10' is repeated until the signs and RHS's of all constraints have been read in (i.e., until S(M) , B(M) up to eight per card. RHS's of lower bounded variables like ordinary constraints are specified here. Uppre bounded variables are not taken to be constarints, they are treated in 'Card 11'.

'Card 11' Col 1 - 3 J (I3) Number of upper bounded variable.
4-10 BOUND(J) (F7.0) Value of the upper bound on the jth variable.

11-13J (13)
14-20 BOUND(J) (F7.0)

.
.
.
.

71-73 J (I3)
74-80 BOUND (J) (F7.0)

'Card 11" is repeated until all upper bounds have been read in, up to eight per card. (Any that are not specified are assumed to have no upper bound.)

'Card 12' Col 1-10 9999999999 indicates the end of upper bounds.

'Card 13' Col 1-3 J (I3) Number of lower bounded variable (lower bound other than zero)

4-10 BL(J) (F 7.0) Value of the lower bound on the jth variable

⋮

71-73 J (I3)

74-80 BL(J) (F1.0)

'Card 13' is repeated until lower bounds have been read in, up to eight per card. (Any that are not specified are assumed to have zero lower bound).

'Card 14' Col-10 9999999999 indicates the end of reading lower bounds. (If there is no lower bounded variable only this card will be placed).

'Card 15' Col 5-10 I (15) number of a row of the A matrix.

11-13 J (I3) number of a column of the A matrix.

15-20A(I,J) (F6.0) a nonzero element of the A matrix.

21-23 J (I3)

25-30 A(I,J) (F6.0)

⋮

71-173 J (I3)

75-80A(I,J) (F6.0)

All elements on a card must belong to the same row but need not be in correct column order. The rows must be entered in correct row order. 'Card 15' is repeated until all nonzero elements of the A matrix have been specified, up to seven per card. Lower bounded variables are enterede as constraints and associated coefficient is specified as above.

'Card 16'	Col 1-10	9999999999	indicates the end of reading the elementsof A matrix.

APPENDIX B

THE LISTING OF THE COMPUTER PROGRAM

```
MAIN1
C       DATA ROUTINE
4111    FORMAT(////10X,,ALTER TOLERANCE(S) APPROPRIATELY IN SUBROUTINE ARR
       *GMT,)
        REAL INV
        INTEGER SIZE,SIZE1,XBASIS,YBASIS
        COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
       1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
       2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
       4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
       5  ITRMAX,M,MARK1,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
       6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
       7   YAM1NC
        COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
        COMMON/DAMN/MAXA
        COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
       *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
       *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
        COMMON/NCVX2/ MAXNN,PRECIS,NFXOP1,IPROP2,IRLV,NNCV,NLV,IN,
       *ACRNG,DIF,ZUP,XBEST(200)
        CF(FX,D,X,E)=FX+D*(X**E)
        CALL DATA
        IF(ISDONE.EQ.1) GO TO 8888
C       NODE 1
        DO 111 J=1,NNCV
        IF(BL(J)-0.0) 33,34,33
34      BLO=0.0
        GO TO 36
33      BLO=FX(J)+D(J)*(BL(J)**E(J))
36      BUB=FX(J)+D(J)*(BU(J)**E(J))
        CE(J)=(BLO-BUB)/(BL(J)-BU(J))
        C(J)=-CE(J)
111     FCV1(J)=BLO-(BLO-BUB)*BL(J)/(BL(J)-BU(J))
C       ALL PARAMETERS ARE READY TO USE IN SIMPLEX
        NN=1
        NNS=1
        IN=1
        CALL SMPLEX
C       THIS SUBROUTINE PROVIDES X(J) AND OBJ.
        IF(ISTATE.EQ.7) GO TO 7444
        IF(ISTATE.NE.1) GO TO 8888
        ZL(NN)=-OBJ
        F=0.0
        OBJCR=-OBJ
        CALL ACOBJF(NNCV,IRLV,NLV,ZU)
        IF(ZU .GT. ZUP) GO TO 223
        ZUP=ZU
        DO 422 J=1,N
422     XBEST(J)=X(J)
        IF(IPROP2.EQ.0) GO TO 14
        CALL PRNT(NN,OBJCR,F,ZL(NN),ZU,ZUP)
14      DIF=ZUP-ZL(1)
        ACRNG=DIF/ZL(1)
        INDEX=1
        IF(ZL(1).GE.ZUP) CALL SOLUTN(1,N,INDEX)
        IF(INDEX.EQ.10) GO TO 8888
        IF(ACRNG.LE.PRECIS) CALL SOLUTN(2,N,INDEX)
        IF(INDEX.EQ.10) GO TO 8888
C       SELECT BRANCHING VARIABLE
        DO 87 I=1,NNCV
        BLW(I)=BL(I)
87      BUP(I)=BU(I)
```

```
      CALL BRVAR (NN,NNCV,AMAX)
223   ZL(NNS)=1.0E+10
C     FIRST BRANCHING
      IN=IN+1
      NN=IN
      NPN(NN)=NNS
      NV=JBRVAR(NNS)
      CALL BND (NNS,NV,IN,NN)
      CALL CFBND(NNCV,NV,IN,NN)
      C(NV)=-CEE(NN)
      CALL SMPLEX
      IF(ISTATE.EQ.7) GO TO 7444
      IF(ISTATE.NE.1) GO TO 8888
      CALL FXCH (NN,IN,NNCV,F)
      ZL(NN)=-OBJ+F
      OBJCR=-OBJ
      CALL ACOBJF(NNCV,IRLV,NLV,ZU)
      IF(ZU.GE.ZUP) GO TO 1113
      ZUP=ZU
      DO 1111 J=1,N
1111  XBEST(J)=X(J)
C     BRANCHING VARIABLE AT THIS NODE FOR FUTURE USE
1113  BLW(NV)=BNDLW(NN)
      BUP(NV)=BNDUP(NN)
      IF(IPROP2.EQ.0) GO TO 1112
      CALL PRNT(NN,OBJCR,F,ZL(NN),ZU,ZUP)
1112  CALL BRVAR (NN,NNCV,AMAX)
C     SECOND BRANCHIN G
      IN=IN+1
      NN=IN
      NPN(NN)=NNS
      NV=JBRVAR(NNS)
      C(NV)=-CEE(NN)
      CALL SMPLEX
      IF(ISTATE.EQ.7) GO TO 7444
      IF(ISTATE.NE.1) GO TO 8888
      CALL FXCH (NN,IN,NNCV,F)
      ZL(NN)=-OBJ+F
      OBJCR=-OBJ
      CALL ACOBJF(NNCV,IRLV,NLV,ZU)
      IF(ZU.GE.ZUP) GO TO 3335
      ZUP=ZU
      DO 3333 J=1,N
3333  XBEST(J)=X(J)
C     BRANCHING AT THIS NODE FOR FUTURE USE
3335  BLW(NV)=BNDLW(NN)
      BUP(NV)=BNDUP(NN)
      IF(IPROP2.EQ.0) GO TO 3334
      CALL PRNT(NN,OBJCR,F,ZL(NN),ZU,ZUP)
3334  CALL BRVAR (NN,NNCV,AMAX)
C     SELECTION OF THE NODE FOR BRANCHING FROM AMONG THE INTERMEDIATE NODES
      CALL SLIND (IN,NNS)
C     OPTIMALITY,PRECISION,ANDMAX. ALLOWABLE #OF NODES TESTS
      DIF=ZUP-ZL(NNS)
      ACRNG=DIF/ZL(NNS)
      IF(ZL(NNS).GE.ZUP) CALL SOLUTN(1,N,INDEX)
      IF(INDEX.EQ.10) GO TO 8888
      IF(ACRNG.LE.PRECIS) CALL SOLUTN(2,N,INDEX)
      IF(INDEX.EQ.10) GO TO 8888
      IF(NN.GE.MAXNN) CALL SOLUTN(3,N,INDEX)
      IF(INDEX.EQ.10) GO TO 8888
      GO TO 223
```

```
 7444  WRITE(6,4111)
 8888  STOP
       END
C
C      SUBPROGRAMS
C      SUBROUTINE SUBPROGRAMS FOR THE CASE WITH FIXED CHARGES
C
URMAN.DATA
```

```
N:DATA
      SUBROUTINE DATA
 9000 FORMAT(8I10)
 9012 FORMAT(1H0,,ITRMAX=,,I5)
 9016 FORMAT(1H1)
 9020 FORMAT(1H0,,YOU ARE TRYING TO SOLVE A PROBLEM  WHICH IS TOO BIG FO
     1R THE PROGRAM.,/1X,,THE LARGEST PROBLEM HAS,,I5,,CONSTRAINTS AND,,
     215,,VARIABLES.,)
 9024 FORMAT(1H0,,IT IS NOT POSSIBLE TO HAVE A NEGATIVE NUMBER OF ROWS O
     1R COLUMNS.,)
 9050 FORMAT((8(I3,I1,F6.0)))
 9082 FORMAT(1H0,,YOU ARE TRYING TO READ ROW NO.,,I5,,WHICH IS MORE THAN
     1YOUR SPECIFIED NO. OF ROWS,,,I5,,.,)
 9084 FORMAT(1H0,,YOUR PREVIOUS ROW WAS,,I5,,AND YOU ARE NOW TRYING TO
     1READ ROW NO.,,I6,,.,/,IT IS ESSENTIAL THAT THE ROWS OF THE A MATRI
     2X BE ENTERED IN THE RIGHT ORDER.,)
 9070 FORMAT(1H0,,YOU HAVE SPECIFIED AN ELEMENT FOR VARIABLE NO.,,I5,, W
     1HICH IS GREATER THAN N,,I5,,.,/,IT HAS BEEN IGNORED.,)
 9096 FORMAT(1H0,,**** YOU HAVE SPECIFIED THAT THERE ARE,,I5,,ROWS AND Y
     1OU HAVE TRIED TO READ IN ELEMENTS OF A FOR,,I5,,ROWS,)
 9098 FORMAT(1H0,,******THERE IS A FATAL ERROR IN YOUR DATA INPUT******,
     1)
   51 FORMAT((8(F2.0,F8.0)))
   53 FORMAT(8F10.0)
   54 FORMAT(8F10.2)
   55 FORMAT((8(I3,F7.0)))
   57 FORMAT(I5,F10.3)
   58 FORMAT(3I5)
      REAL K3
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARK1,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      COMMON/NCVX2/ MAXNN,PRECIS,NFXOP1,IPROP2,IRLV,NNCV,NLV,IN,
     *ACRNG,DIF,ZUP,XBEST(200)
      DIMENSION KBND(8),BDVL(8),K1(8),K2(8),K3(8)
      READ(5,57) MAXNN,PRECIS
C     IF FX(J)=0 FOR ALL J THEN NFXOP1=0:THEN DO NOT READ FX(J).OTHERWISE
C     NFXOP1=1:IF IPROP2=0 RESULTS OF SUBPROBLEMS ARE NOT PRINTED
C     OTHERWISE IPROP2=1
      READ(5,58) NFXOP1,IPROP2
      ZUP=10.0E+10
C
      READ(5,58) NNCV,NLV
      IRLV=NNCV+1
C     PARAMETERS USED IN SIMPLEX ALGORITHM
      BIG=1.0E11
      SMALL=1.0E-9
      READ(5,58) MAXA,MXSIZE
      ISDONE=0
C     N:# OF VARIABLES IN LINEARIZED PROBLEM
```

```
C       IN LINEARIZED PROBLEM
        READ(5,9000) M,N,ISBND,ITRMAX,IRMAX
        MAXN=N
        MAXM=M
        IF(ITRMAX .LE. 0) GO TO 10
        WRITE(6,9012) ITRMAX
        GO TO 30
10      ITRMAX=3*(M+N+ISBND)
        WRITE(6,9012) ITRMAX
30      WRITE(6,9016)
        IF(M .LE. MAXM .AND. N .LE. MAXN .AND. ISBND .LE. MAXN) GO TO 35
        WRITE(6,9020) MAXM,MAXN
        ISDONE=1
        GO TO 610
35      IF(M.GE.0.AND.N.GE.0.AND.ISBND.GE.0) GO TO 40
        WRITE(6,9024)
        ISDONE=1
        GO TO 610
40      MNOW=1
        IROW(1)=1
        DO 50 J=1,N
        PIV(J)=0.0
50      BOUND(J)=-1.0
        DO 60 I=1,M
        B(I)=BIG
60      S(I)=1.0
C       READ THE PARAMETERS OF CONCAVE VARIABLES
        IF(NFXOP1.EQ.0) GO TO 11
        READ(5,53) (FX(J),J=1,NNCV)
        GO TO 13
11      DO 12 J=1,NNCV
12      FX(J)=0.0
13      READ(5,54) (D(J),J=1,NNCV)
        READ(5,54) (E(J),J=1,NNCV)
C       READ THE COEFFICIENTS OF LINEAR VARIABLES IN OBJ. FUNCT.
        IF(NLV.EQ.0) GO TO 697
        READ(5,54) (C(J),J=IRLV,N)
        DO 696 J=IRLV,N
696     C(J)=-C(J)
C       READ RHS'S AND SIGNS OF CONSTRAINTS
C       EQ: S(I)=0 , LE: S(I)=1 , GE: S(I)=2
697     READ(5,51) ((S(I),B(I)),I=1,M)
        DO 888 I=1,M
        IF(S(I) .EQ. 2.0) S(I)=-1.0
888     CONTINUE
C       READ ALL UPPER BOUNDED VARIABLES (LINEAR+CONCAVE)
1000    READ(5,55) ((KBND(I),BDVL(I)),I=1,8)
        IF(KBND(1) .EQ. 999) GO TO 1001
        DO 444 I=1,8
        J=KBND(I)
        IF(J .GT. NNCV) GO TO 77
        BU(J)=BDVL(I)
77      BOUND(J)=BDVL(I)
444     CONTINUE
        GO TO 1000
C       READ LOWER BOUNDS (ONLY CONCAVE VARIABLES AND NONZERO LOWER BOUNDS)
1001    DO 222 J=1,NNCV
222     BL(J)=0.0
1002    READ(5,55) ((KBND(I),BDVL(I)),I=1,8)
        IF(KBND(1) .EQ. 999) GO TO 1005
        DO 232 I=1,8
```

```
      BL(J)=BDVL(1)
232   CONTINUE
      GO TO 1002
C     READ THE CEFFICIENTS OF CONSTRAINTS MATRIX
1005  READ(5,9050) ((K1(JK),K2(JK),K3(JK)),JK=1,8)
      IF(K1(1) .EQ. 999) GO TO 600
500   I=K3(1)
510   IF(I .EQ. MNOW) GO TO 540
      IF(I .LE. M) GO TO 515
      WRITE(6,9082) I,M
      ISDONE=1
      GO TO 610
515   IF(I .GT. MNOW) GO TO 520
      WRITE(6,9084) MNOW,I
      ISDONE=1
      GO TO 610
520   CALL COPY
      IF(ISDONE .NE. 1) GO TO 525
      GO TO 610
525   DO 530 J=1,N
530   PIV(J)=0.0
      MNOW=MNOW+1
      GO TO 510
540   DO 550 JK=2,8
      J=K1(JK)
      IF(J .LE. 0) GO TO 550
      IF(J .LE. N) GO TO 545
      WRITE(6,9070) J,N
      GO TO 550
545   PIV(J)=K3(JK)
550   CONTINUE
      GO TO 1005
600   IF(ISDONE .EQ. 0) CALL COPY
      IF(ISDONE .EQ. 1) GO TO 610
      IF(MNOW .EQ. M) GO TO 610
      WRITE(6,9096) M,MNOW
      ISDONE=1
610   IF(ISDONE.EQ.1) WRITE(6,9098)
      RETURN
      END
URMAN.SOLUTN
```

```
AN.SOLUTN
      SUBROUTINE SOLUTN(IND,N,INDEX)
      COMMON/NCVX2/ MAXNN,PRECIS,NFXOP1,IPROP2,IRLV,NNCV,NLV,IN,
     *ACRNG,DIF,ZUP,XBEST(200)
 6029 FORMAT(10X,'OBJECTIVE=',E15.9/)
 6010 FORMAT(////5X,'TERMINATION BECAUSE TOLERANCE IS WITHIN ACCEPTABLE
     *RANGE',/5X,'NO. OF ITERATIONS=',I4/5X,'TOLERANCE=',1X,E15.9/5X,'DIF
     *FERENCE=',E15.9////40X,'****BEST SOLUTION****'////)
 6020 FORMAT(////5X,'TERMINATION BECAUSE MAX. ALLOWABLE NO. OF ITERATION
     *S OF THE ALGORITHM HAS BEEN EXCEEDED',/5X,'NO. OF ITERATIONS=',I4/5
     *X,'TOLERANCE=',1X,E15.9/5X,'DIFFERENCE=',E15.9////40X,'****BEST SO
     *LUTION****'///)
 6030 FORMAT(4(7X,'X',I4,'=',E15.9))
 6040 FORMAT(////40X,'****OPTIMAL SOLUTION****',/5X,'NUMBER OF NODES=',I4/
     *//)
      IF(IND.EQ.1) GO TO 7035
      IF(IND.EQ.2) GO TO 7010
      IF(IND.EQ.3) GO TO 7020
 7010 WRITE(6,6010) IN,ACRNG,DIF
      GO TO 7030
 7020 WRITE(6,6020) IN,ACRNG,DIF
      GO TO 7030
 7035 WRITE(6,6040) IN
 7030 WRITE(6,6029) ZUP
 7045 WRITE(6,6030) ((J,XBEST(J)),J=1,N)
      INDEX=10
      RETURN
      END
URMAN.ACOBJF
```

```
N.ACOBJF
      SUBROUTINE ACOBJF(NNCV,IRLV,NLV,ZU)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      ZU=0.
      DO 100 J=1,NNCV
      IF(X(J).EQ.0.0) GO TO 100
      ZU=ZU+FX(J)+D(J)*(X(J)**E(J))
  100 CONTINUE
      IF(NLV.EQ.0) GO TO 102
      DO 101 J=IRLV,N
      ZU=ZU+X(J)*(-C(J))
  101 CONTINUE
  102 RETURN
      END
C
URMAN.BRVAR
```

```
N.BRVAR
      SUBROUTINE BRVAR (NN,NNCV,AMAX)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      AMAX=-1000000.0
      DO 1 I=1,NNCV
      IF(BLW(I)-0.0) 88,89,88
89    BLWR=0.0
      GO TO 91
88    BLWR=FX(I)+D(I)*(BLW(I)**E(I))
91    BUPR=FX(I)+D(I)*(BUP(I)**E(I))
      C1=(X(I)-BLW(I))*(BLWR-BUPR)
      CV=BLWR+C1/(BLW(I)-BUP(I))
      IF(X(I)-0.0) 13,14,13
13    AMX=FX(I)+D(I)*(X(I)**E(I))-CV
      GO TO 15
14    AMX=-CV
15    IF(AMX-AMAX) 1,2,2
2     AMAX=AMX
      JBRVAR(NN)=I
      VALBRV(NN)=X(I)
1     CONTINUE
      RETURN
      END
C
GURMAN.BND
```

```
AN.BND
      SUBROUTINE BND (NNS,NV,IN,NN)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4 TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5 ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7 YAMINC
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      IF(NNS-1) 81,86,81
   81 JJJ=NNS
      DO 51 J=1,IN
      MPP=NPN(JJJ)
      IF(NV.EQ.JBRVAR(MPP)) GO TO 84
      JJJ=NPN(JJJ)
      IF(JJJ.EQ.1) GO TO 86
   51 CONTINUE
   84 BNDLW(NN)=BNDLW(JJJ)
      BNDUP(NN)=VALBRV(NNS)
      BNDLW(NN+1)=VALBRV(NNS)
      BNDUP(NN+1)=BNDUP(JJJ)
      GO TO 87
   86 BNDLW(NN)=BL(NV)
      BNDUP(NN)=VALBRV(NNS)
      BNDLW(NN+1)=VALBRV(NNS)
      BNDUP(NN+1)=BU(NV)
   87 DO 61 JK=1,2
      MM=NN+JK-1
      IF(BNDLW(MM)-0.0) 181,182,181
  182 BLL=0.0
      GO TO 191
  181 BLL=FX(NV)+D(NV)*(BNDLW(MM)**E(NV))
  191 BUU=FX(NV)+D(NV)*(BNDUP(MM)**E(NV))
      CEE(MM)=(BLL-BUU)/(BNDLW(MM)-BNDUP(MM))
   61 FC(MM)=BLL-(BLL-BUU)*(BNDLW(MM))/(BNDLW(MM)-BNDUP(MM))
      RETURN
      END
C
GURMAN,CEBND
```

```
AN.CFBND
      SUBROUTINE CFBND (NNCV,NV,IN,NN)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4 TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5 ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7 YAMINC
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      DO 52 KK=1,NNCV
      IF(KK.EQ.NV) GO TO 52
      C(KK)=-CE(KK)
      BLW(KK)=BL(KK)
      BUP(KK)=BU(KK)
      LLL=NPN(NN)
      DO 53 I=1,IN
      IF(LLL.EQ.1) GO TO 52
      MMM=NPN(LLL)
      IF(KK.NE.JBRVAR(MMM)) GO TO 8
      C(KK)=-CEE(LLL)
      BLW(KK)=BNDLW(LLL)
      BUP(KK)=BNDUP(LLL)
      GO TO 52
    8 LLL=NPN(LLL)
   53 CONTINUE
   52 CONTINUE
      RETURN
      END
GURMAN,EXCH
```

```
AN.FXCH
      SUBROUTINE FXCH (NN,IN,NNCV,F)
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
      F=0.0
      DO 5 KK=1,NNCV
      LLL=NN
      DO 9 I=1,IN
      IF(LLL.EQ.1) GO TO 10
      MNM=NPN(LLL)
      IF(JBRVAR(MNM).NE.KK) GO TO 8
      F=F+FC(LLL)
      GO TO 5
    8 LLL=NPN(LLL)
    9 CONTINUE
   10 F=F+FCV1(KK)
    5 CONTINUE
      RETURN
      END
GURMAN.SLTND
```

```
AN.SLTND
      SUBROUTINE SLTND (IN,NNS)
      COMMON/NCVXAL/FX(100),D(100),E(100),BL(100),BU(100),CE(100),
     *BLW(100),BUP(100),ZL(501),JBRVAR(501),VALBRV(501),
     *NPN(501),FC(501),BNDLW(501),BNDUP(501),CEE(501),FCV1(100)
C
      AMIN=ZL(1)
      DO 20 I=1,IN
      IF(ZL(I)-AMIN) 21,20,20
21    AMIN=ZL(I)
      NNS=I
20    CONTINUE
      RETURN
      END
C
GURMAN.SMPLEX
```

SMPLEX

```
      SUBROUTINE SMPLEX
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),P1V(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7    YAMINC
      CALL ARRGMT
      ISTATE=0
      CALL LP
      CALL IEXIT (ISTATE)
      RETURN
      END
C
RMAN.ARRGMT
```

```
:ARRGMT
 C
       SUBROUTINE ARRGMT
       REAL INV
       INTEGER SIZE,SIZE1,XBASIS,YBASIS
       COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
      1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
      2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
      4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
      5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
      6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
      7   YAMINC
       INREV=0
       IR=0
       ITR=0
       NEGINV=0
       NEGROW=0
       NEWX=0
       NEWY=0
       R=0.0
       SIZE=0
       ISBIG=0
       TOL(1)=1.0E-6
       TOL(2)=1.0E-4
       TOL(3)=1.0E-6
       TOL(4)=1.0E-5
       TOL(5)=1.0E-5
       TOL(6)=1.0E-5
       TOL(7)=1.0E-5
       TOL(8)=TOL(5)*10.0
       YAMINC=0.0
       RETURN
       END
 C
RMAN.PRNT
```

```
PRNT
C
      SUBROUTINE PRNT(NN,OBJCR,F,ZLOW,ZU,ZUP)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
15    FORMAT(///10X,,****NODE NUMBER=,,I4/)
44    FORMAT(14X,,OBJECTIVE OF SIMPLEX=,,E15.9/14X,,ALTITUDE TO BE ADDED
     *=,,E15.9/14X,,LOWER BOUND(,,I4,,)=,,3X,E15.9/14X,,UPPER BOUND(,,I4
     *,,)=,,3X,E15.9/14X,,MINIMUM UPPER BOUND=,,1X,E15.9//)
99    FORMAT(4(7X,,X,,I4,,=,,E15.9))
      WRITE(6,15) NN
      WRITE(6,44) OBJCR,F,NN,ZLOW,NN,ZU,ZUP
      WRITE(6,99) ((J,X(J)),J=1,N)
      RETURN
      END
RMAN.LP
```

```
LP
C
C
      SUBROUTINE LP
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),NBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      ITR=0
      IR=0
      IF(ISTATE.EQ.10) INREV=1
   10 CALL DOANLP
      IF(ISTATE.GT.3)GO TO 20
      CALL CHACC
      IF(ISTATE.NE.7) GO TO 20
      IF(IR.GE.IRMAX)GO TO 20
      CALL REVERT
      ISTATE=11
      GO TO 10
   20 RETURN
      END
RMAN.DOANLP
```

```
I:DOANLP
      SUBROUTINE DOANLP
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJR,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      IF(ISTATE.EQ.0)CALL FIRSTB
      IF(ISTATE.EQ.11)GO TO 20
      IF(ISTATE.EQ.12)GO TO 50
   10 CALL CHSLCK
      IF(ITR.LE.ITRMAX)GO TO 20
      ISTATE=5
      GO TO 80
   20 IF(NEGROW.EQ.0.AND.NEGINV.EQ.0)GO TO 40
      IF(NEGINV.NE.0)GO TO 30
      NEWY=NEGROW+SIZE
      DRIVER=1.0
      IF(SLACK(NEGROW).GT.0.0)DRIVER=-1.0
      NEGINV=SIZE1
      CALL ADDCON
      IF(ISTATE.EQ.4)GO TO 80
   30 CALL SEEKX
      IF(NEWX.NE.0)GO TO 50
      ISTATE=2
      GO TO 80
   40 CALL ISOPT
      IF(NEWX.NE.0)GO TO 50
      ISTATE=1
      GO TO 80
   50 CALL NEWVEC
      CALL SEEKY
      IF(NEWY.NE.0)GO TO 60
      ISTATE=3
      GO TO 80
   60 IF(NEWY.LE.SIZE)GO TO 70
      CALL ADDCON
      IF(ISTATE.EQ.4)GO TO 80
   70 CALL CHBSIS
      CALL REDUCE
      GO TO 10
   80 RETURN
      END
URMAN.A
```

```
:A
      FUNCTION A(I,J)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      ISTART=IROW(I)
      LAST=IROW(I+1)-1
      A=0.0
C
      DO 1 LOOK=ISTART,LAST
      JHERE=JCOL(LOOK)
      IF(JHERE.LT.J)GO TO 1
      IF(JHERE.GT.J)RETURN
      A=AA(LOOK)
      RETURN
    1 CONTINUE
      RETURN
      END
URMAN.ADDCON
```

```
J:ADDCON
      SUBROUTINE ADDCON
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      IF(SIZE1.GT.MXSIZE)GO TO 40
      I=NEWY-SIZE
      DO 10 L=1,SIZE
      INV(L,SIZE1)=0.0
   10 INV(SIZE1,L)=0.0
      ISTART=IROW(I)
      LAST=IROW(I+1)-1
      DO 30 LOOK=ISTART,LAST
      J=JCOL(LOOK)
      IF(INBASE(J).LE.0)GO TO 30
      K=INBASE(J)
      AIJ=AA(LOOK)
      DO 20 L=1,SIZE
   20 INV(SIZE1,L)=INV(SIZE1,L)-AIJ*INV(K,L)
   30 CONTINUE
      INV(SIZE1,SIZE1)=1.0
      XR(SIZE1)=SLACK(I)
      ISEFF(I)=SIZE1
      XBASIS(SIZE1)=I+N
      YBASIS(SIZE1)=1
      YR(SIZE1)=0.0
      SIZE=SIZE1
      SIZE1=SIZE1+1
      IF(SIZE.GT.ISBIG)ISBIG=SIZE
      NUMSLK=NUMSLK+1
      NEWY=SIZE
      GO TO 50
   40 ISTATE=4
   50 RETURN
      END
URMAN.CHACC
```

```
N:CHACC
C
C
      SUBROUTINE CHACC
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREP/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
 9000 FORMAT(1H0,'UNACCEPTABLE ERROR OF',F16.8,'FOUND IN B-SLACK-AX OF
     1CONSTRAINT',I6)
 9004 FORMAT(1H0,'UNACCEPTABLE RELATIVE ERROR OF',F16.8,'FOUND IN B-SLAC
     1K-AX OF CONSTRAINT',I6/1H,'THE ABSILUTE ERROR IS',F16.8,'AND B(I)
     2IS',F16.8)
 9008 FORMAT(1H0,'UNACCEPTABLE ERROR OF',F16.8,'FOUND IN YA-C OF BASIC
     1VARIABLE',I6)
 9012 FORMAT(1H0,'UNACCEPTABLE RELATIVE ERROR OF',F16.8,'FOUND IN YA-C O
     1F BASIC VARIABLE',I6/1H,'THE ABSOLUTE ERROR IS',F16.8,'AND C(J) I
     2S',F16.8)
      IF(NUMSLK.EQ.0)GO TO 10
C
      DO 5 K=1,SIZE
      IF(XBASIS(K).LE.N)GO TO 5
      I=XBASIS(K)-N
      SLACK(I)=XR(K)
    5 CONTINUE
C
   10 DO 20 J=1,N
      IF(INBASE(J).LE.0)GO TO 20
      YAC(J)=-C(J)
   20 CONTINUE
C
      TOL2=TOL(2)
      TOL6=TOL(6)
C
      DO 40 I=1,MNOW
      ISEFFI=ISEFF(I)
      YI=Y(I)
      BAXSL=B(I)-SLACK(I)
      ISTART=IROW(I)
      LAST=IROW(I+1)-1
C
      DO 30 LOOK=ISTART,LAST
      J=JCOL(LOOK)
      INJ=INBASE(J)
      IF(INJ.EQ.0)GO TO 30
      AIJ=AA(LOOK)
      BAXSL=BAXSL-X(J)*AIJ
      IF(INJ.GT.0.AND.ISEFFI.NE.0)YAC(J)=YAC(J)+YI*AIJ
   30 CONTINUE
C
      ERR=ABS(BAXSL)
      IF(ERR.GT.TOL2)GO TO 60
      ABSB=ABS(B(I))
      IF(ABSB.LT.1.0)ABSB=1.0
```

```
   40 CONTINUE
C
      TOL7=TOL(7)
      TOL4=TOL(4)
C
      DO 50 J=1,N
      IF(INBASE(J).LE.0)GO TO 50
      ERR=ABS(YAC(J))
      IF(ERR.GT.TOL4)GO TO 70
      ABSC=ABS(C(J))
      IF(ABSC.LT.1.0)ABSC=1.0
      IF(ERR/ABSC.GT.TOL7)GO TO 75
   50 CONTINUE
C
      GO TO 90
   60 WRITE(6,9000)ERR,I
      GO TO 80
   65 RELERR=ERR/ABSB
      WRITE(6,9004)RELERR,I,ERR,ABSB
      GO TO 80
   70 WRITE(6,9008)ERR,J
      GO TO 80
   75 RELERR=ERR/ABSC
      WRITE(6,9012)RELERR,J,ERR,ABSC
   80 ISTATE=7
   90 RETURN
      END
URMAN.CHBSIS
```

```
N:CHBSIS
  C
  C
        SUBROUTINE CHBSIS
        REAL INV
        INTEGER SIZE,SIZE1,XBASIS,YBASIS
        COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
       1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
       2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
       4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
       5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
       6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
       7   YAMINC
        ITR=ITR+1
        MOSNEG=0
        HOWNEG=0.0
        XOFNEG=0.0
        DRITEM=0.0
        TOL1=TOL(1)
        IF(INREV.EQ.1)GO TO 90
        IF(R.EQ.0.0)GO TO 40
  C
        DO 30 K=1,SIZE
        XR(K)=XR(K)-R*GR(K)*XKPOS
        XXX=XR(K)
        IF(ABS(XXX).LE.TOL1)XR(K)=0.0
        J=XBASIS(K)
        IF(J.LE.N)GO TO 10
        XXX=XR(K)
        I=J-N
        SI=S(I)
        IF(SI.EQ.0.0.AND.XXX.GT.0.0.OR.SI.EQ.-1.0)XXX=-XXX
        GO TO 20
    10  BOUNDJ=BOUND(J)
        IF(ABS(BOUNDJ-XXX).LE.TOL1)XR(K)=BOUNDJ
        XXX=XR(K)
        IF(XXX.LE.BOUNDJ.OR.BOUNDJ.EQ.-1.0)GO TO 20
        XXX=BOUNDJ-XXX
    20  IF(K.EQ.NEGINV)XOFNEG=XXX
        IF(XXX.GE.HOWNEG.OR.K.EQ.NEGINV)GO TO 30
        MOSNEG=K
        DRITEM=1.0
        IF(XR(K).GE.0.0)DRITEM=-1.0
        HOWNEG=XXX
    30  CONTINUE
  C
        IF(NEWY.NE.-1)GO TO 40
        IT=INBASE(NEWX)
        INBASE(NEWX)=-1
        IF(IT.EQ.-1)INBASE(NEWX)=0
        IXOUT=NEWX
        OBJ=OBJ-R*YAMINC
        GO TO 120
    40  IXOUT=XBASIS(NEWY)
        IF(IXOUT.GT.N)GO TO 50
        INBASE(IXOUT)=0
        IF(GR(NEWY)*XKPOS.LT.0.0.AND.NEWY.NE.NEGINV)INBASE(IXOUT)=-1
        IF(NEWY.EQ.NEGINV.AND.XR(NEWY).GT.0.0)INBASE(IXOUT)=-1
    50  IF(NEWX.GT.N)GO TO 60
        IHOLD=INBASE(NEWX)
        INBASE(NEWX)=NEWY
```

```
      IF(NEWX.GT.N)NUMSLK=NUMSLK+1
      IF(IXOUT.GT.N)NUMSLK=NUMSLK-1
      XR(NEWY)=R
      IF(NEWX.LE.N)GO TO 70
      I=NEWX-N
      IF(S(I).EQ.-1.0)XR(NEWY)=R
      GO TO 80
   70 IF(IHOLD.EQ.-1)XR(NEWY)=BOUND(NEWX)-R
   80 OBJ=OBJ-R*YAMINC
   90 RR=1.0/GR(NEWY)
C
      DO 110 L=1,SIZE
      IF(ABS(INV(NEWY,L)).LT.SMALL)GO TO 110
      RL=INV(NEWY,L)*RR
C
      DO 100 K=1,SIZE
      INV(K,L)=INV(K,L)-RL*GR(K)
  100 CONTINUE
C
      INV(NEWY,L)=RL
      IF(INREV.NE.1)YR(L)=YR(L)-RL*YAMINC*XKPOS
  110 CONTINUE
C
  120 IF(R.EQ.0.0.OR.XOFNEG.LT.0.0.AND.NEWY.NE.NEGINV)GO TO 130
      NEGINV=MOSNEG
      DRIVER=URITEM
  130 RETURN
      END
GURMAN.CHSLCK
```

```
N:CHSLCK
      SUBROUTINE CHSLCK
C
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      IF (R .NE. 0.0) NEGROW = 0
      HOWNEG = 0.0
      DO 10 J = 1,N
      YACJ = 0.0
      K = INBASE(J)
      IF(K.LE.0) YACJ = -C(J)
      YAC(J) = YACJ
      XJ=0.0
      IF(K.EQ.-1) XJ = BOUND(J)
      IF(K.GT.0) XJ = XR(K)
   10 X(J) = XJ
      TOL2 = TOL(2)
      DO 70 I = 1,MNOW
      L = ISEFF(I)
      Y(I) = 0.0
      IF(L.EQ.0) GO TO 30
      YI = YR(L)
      Y(I) = YI
      SLACK(I) = 0.0
      LAST = IROW(I+1) -1
      ISTART = IROW(I)
      DO 20 LOOK = ISTART,LAST
      J = JCOL(LOOK)
      IF(INBASE(J).GT.0) GO TO 20
      AIJ = AA(LOOK)
      YAC(J) = YAC(J) + YI * AIJ
   20 CONTINUE
      GO TO 70
   30 IF(INREV.NE.1) GO TO 50
      SLKI=B(I)
      DO 40 J=1,N
      IF(INBASE(J).EQ.0) GO TO 40
      SLKI=SLKI-A(I,J)*X(J)
   40 CONTINUE
      GO TO 60
   50 IF(R.EQ.0.) GO TO 70
      SLKI=SLACK(I)-R*G(I)*XKPOS
   60 IF(ABS(SLKI).LE.TOL2) SLKI=0.0
      SLACK(I)=SLKI
      IF(NEGINV.NE.0) GO TO 70
      SI=S(I)
      ABSLKI=ABS(SLKI)
      IF(SI.NE.0.0.AND.SI*SLKI.GE.HOWNEG.OR.SI.EQ.0.0.AND.
     *-ABSLKI.GE.HOWNEG) GO TO 70
      HOWNEG=-ABSLKI
      NEGROW=I
   70 CONTINUE
```

```
      IF(MARKI.NE.0) SLACK(MARKI)=XR(MARKK)
      RETURN
      END
URMAN.COPY
```

```
AN:COPY
      SUBROUTINE COPY
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      IF(MNOW.LE.MAXM) GO TO 5
      CALL IEXIT(6)
      GO TO 30
    5 ISTRIV=1
      IT=-1
      IRM=IROW(MNOW)
      DO 10 J=1,N
      AIJ=PIV(J)
      IF(AIJ.EQ.0.0) GO TO 10
      ISTRIV=0
      IT=IT+1
      LOOK=IRM+IT
      IF(LOOK.GT.MAXA) CALL IEXIT(6)
      IF(ISDONE.EQ.1) GO TO 30
      AA(LOOK)=AIJ
      JCOL(LOOK)=J
   10 CONTINUE
      IF(ISTRIV.EQ.0) GO TO 20
      LOOK=IRM
      IF(LOOK.GT.MAXA) CALL IEXIT(6)
      IF(ISDONE.EQ.1) GO TO 30
      AA(LOOK)=0
      JCOL(LOOK)=1
   20 NEXTM=MNOW+1
      IROW(NEXTM)=LOOK+1
      NEXTM=MNOW+2
      IROW(NEXTM)=LOOK+1
   30 RETURN
      END
GURMAN.FIRSTB
```

```
AN:FIRSTB
      SUBROUTINE FIRSTB
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     #  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7    YAMINC
      DO 10 J = 1,N
   10 INBASE(J) = 0
      DO 20 I = 2,MNOW
   20 ISEFF(I) = 0
      DRIVER = 0.0
      NEGINV = 0
      SS = S(1)
      BB = B(1)
      IF(SS .EQ. 1.0 .AND. BB .GE. 0.0 .OR. SS .EQ. -1.0 .AND. BB .LE.
     10.0 .OR. SS .EQ. 0.0 .AND. BB .EQ. 0.0) GO TO 30
      NEGINV = 1
      DRIVER = 1.0
      IF(BB.GT.0.0) DRIVER = -1.0
   30 SIZE = 1
      SIZE1 = 2
      NEWX = N + 1
      XBASIS(1) = N + 1
      INV(1,1) = 1.0
      XR(1) = BB
      OBJ = 0.0
      YR(1) = 0.0
      YBASIS(1) = 1
      ISEFF(1) = 1
      NUMSLK = 1
      MARKI = 1
      MARKK = 1
      ITR = ITR + 1
      INREV = 1
      RETURN
      END
GURMAN.1EXIT
```

```
N:IEXIT
      SUBROUTINE IEXIT(JK)
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(10I)
      COMMON/DAMN/MAXA
 9002 FORMAT (1H0,'INFEASIBLE.')
 9003 FORMAT (1H0,'UNBOUNDED.')
 9004 FORMAT (1H0,'THE MAXIMUM SIZE OF THE INVERSE HAS BEEN EXCEEDED.')
 9005 FORMAT (1H0,'THE MAXIMUM NUMBER OF ITERATIONS HAS BEEN REACHED.')
 9006 FORMAT (1H0,'EITHER THE AA VECTOR IS FULL WITH ',I6,' ELEMENTS, OR
     1THE NUMBER OF CONSTRAINTS IS ABOUT TO EXCEED ',I6,'.'/1X,'THE PROGR
     2AM IS TRYING TO COPY ROW ',I6,' WHICH IS AS FOLLOWS:.')
 9007 FORMAT (1H0,'STILL INACCURATE AFTER ',I5,' REINVERSIONS.')
 9008 FORMAT (1H0,'INTEGER PROGRAM OPTIMUM.')
 9009 FORMAT (1H0,'QUADRATIC PROGRAM OPTIMUM.')
 9011 FORMAT (1H0,'THE NEGINV ROW OF THE UPDATED A MATRIX IS AS FOLLOWS:
     1.')
 9012 FORMAT (8(1X,F14.7))
 9013 FORMAT (1H0,'THE VECTOR GR IS AS FOLLOWS.')
 9014 FORMAT (1H0,'THE VECTOR G IS AS FOLLOWS.')
      IF(JK.EQ.2.OR.JK.EQ.3) MOREPR = 1
      ISDONE = 1
      GO TO (1,2,3,4,5,6,7,8,9),JK
    1 GO TO 10
    2 WRITE (6,9002)
      WRITE (6,9011)
      WRITE (6,9012)(PIV(J),J=1,N)
      GO TO 10
    3 WRITE (6,9003)
      WRITE (6,9013)
      WRITE (6,9012)(GR(K),K=1,SIZE)
      WRITE (6,9014)
      WRITE (6,9012)(G(I),I=1,MNOW)
      GO TO 10
    4 WRITE (6,9004)
      GO TO 10
    5 WRITE (6,9005)
      GO TO 10
    6 WRITE (6,9006) MAXA,MAXM,MNOW
      WRITE (6,9012)(PIV(J),J=1,N)
      GO TO 10
    7 WRITE (6,9007) IR
      GO TO 10
    8 WRITE (6,9008)
      GO TO 10
    9 WRITE (6,9009)
   10 RETURN
      END
GURMAN.1SOPT
```

```
AN:ISOPT
      SUBROUTINE ISOPT
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJR,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      YAMINC=-TOL(3)
      NEWX=0
      DO 10 L=1,SIZE
      I=YBASIS(L)
      SI=S(I)
      IF(SI.EQ.0.0) GO TO 10
      YRL=YR(L)*SI
      IF(YRL.GE.YAMINC) GO TO 10
      YAMINC=YRL
      NEWX=I+N
   10 CONTINUE
      TOL4=TOL(4)
      DO 20 J=1,N
      INBJ=INBASE(J)
      IF(INBJ.GT.0.OR.BOUND(J).EQ.0.0) GO TO 20
      T=YAC(J)
      IF(ABS(T).LE.TOL4) T=0.0
      YAC(J)=T
      IF(INBJ.EQ.-1) T=-T
      IF(T.GE.YAMINC) GO TO 20
      YAMINC=T
      NEWX=J
   20 CONTINUE
      RETURN
      END
GURMAN.NEWVEC
```

```
N:NEWVEC
      SUBROUTINE NEWVEC
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      REAL INV
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      XKPOS=1.0
      IF(NEWX.GT.N) GO TO 40
      DO 10 K=1,SIZE
   10 GR(K)=0.0
      DO 30 L=1,SIZE
      I=YBASIS(L)
      AIJ=A(I,NEWX)
      IF(AIJ.EQ.0.0) GO TO 30
      DO 20 K=1,SIZE
   20 GR(K)=GR(K)+AIJ*INV(K,L)
   30 CONTINUE
      IF(INBASE(NEWX).EQ.-1) XKPOS=-1.0
      GO TO 60
   40 I=NEWX-N
      L=ISEFF(I)
      DO 50 K=1,SIZE
   50 GR(K)=INV(K,L)
      IF(S(1).EQ.-1.0)XKPOS=-1.0
   60 RETURN
      END
URMAN.REDUCE
```

```
N,REDUCE
      SUBROUTINE REDUCE
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      REAL INV
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      MARKI=0
      MARKK=0
      IF(NUMSLK.EQ.0) GO TO 80
      IT=SIZE
      DO 60 K=1,IT
   10 IF(SIZE.LE.1) GO TO 70
      J=XBASIS(K)
      IF(J.LE.N) GO TO  60
      I=J-N
      SI=S(I)
      IF(SI*XR(K).LT.0.0.OR.SI.EQ.0.0.AND.XR(K).NE.0.0) GO TO 60
      IF(K.EQ.SIZE) GO TO 30
      DO 20 L=1,SIZE
   20 INV(K,L)=INV(SIZE,L)
      J=XBASIS(SIZE)
      XBASIS(K)=J
      IF(J.LE.N) INBASE(J)=K
   30 SLACK(I)=XR(K)
      XR(K)= XR(SIZE)
      IF(NEGINV.EQ.SIZE) NEGINV=K
      L=ISEFF(I)
      ISEFF(I)=0
      IF(L.EQ.SIZE) GO TO 50
      DO 40 KK=1,SIZE
   40 INV(KK,L)=INV(KK,SIZE)
      YR(L)=YR(SIZE)
      YBASIS(L)=YBASIS(SIZE)
      I=YBASIS(SIZE)
      ISEFF(I)=L
   50 XBASIS(SIZE)=0
      SIZE=SIZE-1
      SIZE1=SIZE1-1
      NUMSLK=NUMSLK-1
      GO TO 10
   60 CONTINUE
   70 IF(SIZE.LT.2.AND.XBASIS(1).GT.N) MARKK=1
      IF(NEGINV.EQ.0.AND.MARKK.EQ.0) GO TO 80
      J=0
      IF(NEGINV.NE.0) J=XBASIS(NEGINV)
      IF(J.GT.N) MARKK=NEGINV
      IF(MARKK.EQ.0) GO TO 80
      MARKI=XBASIS(MARKK)-N
   80 RETURN
      END
JRMAN,REVERT
```

```
N;REVERT
      SUBROUTINE REVERT
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
 9000 FORMAT(1H ,20X,,REINVERTED AT ITERATION ,,I6)
      IR = IR + 1
      ITHOLD = ITR
      INREV = 1
      HOLD = SMALL
      SMALL = 0.0
      TOL8 = TOL(8)
      DO 20 K = 1,SIZE
   10 IF (XBASIS(K) .LE. N) GO TO 20
      I = XBASIS(K) - N
      L = ISEFF(I)
      IF (K .EQ. L) GO TO 20
      XBASIS(K) = XBASIS(L)
      XBASIS(L) = I + N
      J = XBASIS(K)
      IF (J .GT. N) GO TO 10
   20 CONTINUE
      DO 40 K = 1,SIZE
      IF (XBASIS(K) .LE. N) XBASIS(K) = -XBASIS(K)
      DO 30 L = 1,SIZE
   30 INV(K,L) = 0.0
   40 INV(K,K) = 1.0
      DO 50 J = 1,N
      IF (INBASE(J) .NE. -1) INBASE(J) = 0
   50 CONTINUE
      DO 90 K = 1,SIZE
      NEWX = -XBASIS(K)
      IF (NEWX .LT. 0 .OR. NEWX .GT. N) GO TO 90
   60 CALL NEWVEC
      NEWY = 0
      DO 80 KK = 1,SIZE
      IF (XBASIS(KK) .GT. 0) GO TO 80
      ABDIF = ABS(GR(KK))
      IF (ABDIF .LT. TOL8) GO TO 80
      IF (NEWY .NE. 0) GO TO 70
      BEST = ABS(1.0 - ABDIF)
      NEWY = KK
      GO TO 80
   70 ABDIF = ABS(1.0 - ABDIF)
      IF (ABDIF .GE. BEST) GO TO 80
      BEST = ABDIF
      NEWY = KK
   80 CONTINUE
      IF (NEWY .EQ. 0) GO TO 90
      IHOLD = -XBASIS(NEWY)
      IF (IHOLD .EQ. NEWX) IHOLD = 0
      CALL CHBSIS
      XBASIS(NEWY) = NEWX
      IF (IHOLD .EQ. 0) GO TO 90
      NEWX = IHOLD
```

```
      NUMSLK = 0
      DO 110 K = 1,SIZE
      J = XBASIS(K)
      IF (J .GT. 0) GO TO 100
      I = YBASIS(K)
      XBASIS(K) = N + I
      NUMSLK = NUMSLK + 1
      GO TO 110
  100 IF (J .LE. N) INBASE(J) = K
      IF (J .GT. N) NUMSLK = NUMSLK + 1
  110 CONTINUE
      SMALL = HOLD
      DO 120 K=1,SIZE
      XR(K) = 0.0
  120 YR(K) = 0.0
      DO 140 K = 1,SIZE
      I = YBASIS(K)
      J = XBASIS(K)
      TC = 0.0
      IF(J.LE.N) TC=C(J)
      TB=B(I)
      DO 130 JJ = 1,N
      IF (INBASE(JJ) .NE. -1) GO TO 130
      TB = TB- BOUND(JJ) * A(I,JJ)
  130 CONTINUE
      DO 140 L = 1,SIZE
      XR(L) = XR(L) + TB * INV(L,K)
      YR(L) = YR(L) + TC * INV(K,L)
      IF (ABS(YR(L)) .LE. SMALL) YR(L) = 0.0
      IF (ABS(XR(L)) .LE. SMALL) XR(L) = 0.0
  140 CONTINUE
      NEGINV = 0
      T = 0.0
      DO 180 K = 1,SIZE
      XRK = XR(K)
      J = XBASIS(K)
      IF (J .GT. N) GO TO 160
      IF (ISBND .EQ. 0) GO TO 150
      IF (BOUND(J) .EQ. -1.0) GO TO 150
      IF (XRK .GT. BOUND(J)) XRK = BOUND(J) - XR(K)
  150 IF (XRK .GE. T) GO TO 180
      GO TO 170
  160 I = J - N
      IF (S(I) .NE. 0.0 .AND. XRK * S(I) .GE. T .OR. S(I) .EQ. 0.0 .AND.
     1   ABS(XRK) * (-1.0) .GE. T) GO TO 180
  170 T = -1.0 * ABS(XRK)
      NEGINV = K
      DRIVER = 1.0
      IF (XR(K) .GT. 0.0) DRIVER = -1.0
  180 CONTINUE
      IF(NUMSLK .GE. 1) CALL REDUCE
      CALL CHSLCK
      CALL ISOPT
      ITR = ITHOLD
      INREV = 0
      OBJ = 0.0
      DO 190 J = 1,N
      IF (INBASE(J) .EQ. 0) GO TO 190
      OBJ = OBJ + X(J) * C(J)
  190 CONTINUE
      WRITE (6,9000) ITR
```

```
      RETURN
      END
RMAN.SEEKX
```

```
      SUBROUTINE SEEKX
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      NEWX = 0
      R = -BIG
      PIVMAX = 0.0
      JMAXP = 0
      BESPIV = 0.0
      TOL3 = TOL(3)
      TOL4 = TOL(4)
      TOL5 = TOL(5)
      DO 10 J = 1,N
   10 PIV(J) = 0.0
      DO 40 L = 1,SIZE
      I = YBASIS(L)
      SI = S(I)
      YI = YR(L) * SI
      IF (ABS(YI) .LT. TOL3) YI = 0.0
      RINVL = INV(NEGINV,L)
      ISTART = IROW(I)
      LAST = IROW(I+1) - 1
      DO 20 LOOK = ISTART,LAST
      J = JCOL(LOOK)
      IF (INBASE(J) .GE. 1 .OR. BOUND(J) .EQ. 0.0) GO TO 20
      AIJ = AA(LOOK)
      PIV(J) = PIV(J) + AIJ * RINVL
   20 CONTINUE
      IF (SI .EQ. 0.0) GO TO 40
      PIVOT=RINVL*SI*DRIVER
      IF (PIVOT .GE. -TOL5 .OR. PIVOT .GE. -0.5 .AND. NEWX .NE. 0 .AND.
     1YI .LT. 0.0) GO TO 40
      IF (PIVOT .GE. -0.5 .AND. YI .LT. 0.0) GO TO 30
      RATIO = YI / PIVOT
      IF (RATIO .LT. R .AND. NEWX .NE. 0) GO TO 40
      IF (RATIO .EQ. 0.0 .AND. PIVOT .GE. BESPIV) GO TO 40
      IF (RATIO .EQ. 0.0) BESPIV = PIVOT
      R = RATIO
      YAMINC = YI
      NEWX = N + I
      GO TO 40
   30 IF (PIVOT .GE. PIVMAX) GO TO 40
      YACP = YI
      JMAXP = N + I
      PIVMAX = PIVOT
   40 CONTINUE
      DO 60 J = 1,N
      INJ = INBASE(J)
      IF (INJ .GE. 1 .OR. BOUND(J) .EQ. 0.0) GO TO 60
      SJ = 1.0
      IF (INJ .EQ. -1) SJ = -1.0
      FUNC = YAC(J) * SJ
      IF (ABS(FUNC) .LT. TOL4) FUNC = 0.0
```

```
      PIVOT = PIV(J) * SJ * DRIVER
      IF (PIVOT .GE. -TOL5 .OR. PIVOT .GE. -0.5 .AND. NEWX .NE. 0
     1     .AND. FUNC .LT. 0.0) GO TO 60
      IF (PIVOT .GE. -0.5 .AND. FUNC .LT. 0.0) GO TO 50
      RATIO = FUNC / PIVOT
      IF (RATIO .LT. R .AND. NEWX .NE. 0) GO TO 60
      IF (RATIO .EQ. 0.0 .AND. PIVOT .GE. BESPIV) GO TO 60
      IF (RATIO .EQ. 0.0) BESPIV = PIVOT
      R = RATIO
      YAMINC = FUNC
      NEWX = J
      GO TO 60
   50 IF (PIVOT .GE. PIVMAX) GO TO 60
      PIVMAX = PIVOT
      YACP = FUNC
      JMAXP = J
   60 CONTINUE
      IF (NEWX .NE. 0) GO TO 70
      NEWX = JMAXP
      YAMINC = YACP
      IF (NEWX .NE. 0) R = YAMINC / PIVMAX
   70 RETURN
      END
URMAN.SEEKY
```

```
1;SEEKY
      SUBROUTINE SEEKY
      REAL INV
      INTEGER SIZE,SIZE1,XBASIS,YBASIS
      COMMON BOUND(200),C(200),INBASE(200),PIV(200),X(200),YAC(200),
     1 B(100),G(100),ISEFF(100),S(100),SLACK(100),Y(100),
     2 INV(100,100),XBASIS(100),XR(100),YBASIS(100),YR(100),GR(100),
     4  TOL(8),BIG,DRIVER,INREV,IR,IRMAX,ISBIG,ISBND,ISDONE,ISTATE,ITR,
     5  ITRMAX,M,MARKI,MARKK,MAXM,MAXN,MNOW,MORE,MOREPR,MXSIZE,N,
     6  NEGINV,NEGROW,NEWX,NEWY,NUMSLK,OBJ,R,SIZE,SIZE1,SMALL,XKPOS,
     7   YAMINC
      COMMON/AREF/ AA(1000),JCOL(1000),IROW(101)
      COMMON/DAMN/MAXA
      SI = 1.0
      BOUNDJ=-1.0
      R = BIG
      NEWY = 0
      TOL5 = TOL(5)
      IF (IBSND .EQ. 0 .OR. NEWX .GT. N) GO TO 10
      IF (BOUND(NEWX) .EQ. -1.0) GO TO 10
      R = BOUND(NEWX)
      NEWY = -1
   10 IF (NEGINV.EQ. 0) GO TO 30
      XRNEG = XR(NEGINV)
      J = XBASIS(NEGINV)
      IF (J .GT. N) GO TO 20
      BOUNDJ = BOUND(J)
      IF (BOUNDJ .GE. XRNEG .OR. BOUNDJ .EQ. -1.0) GO TO 20
      XRNEG = XRNEG - BOUNDJ
   20 RTRY = XRNEG / (XKPOS * GR(NEGINV))
      IF (RTRY .GT. R) GO TO 30
      R = RTRY
      IF (R .LE. SMALL) R = 0.0
      NEWY = NEGINV
      IF (R .EQ. 0.0) GO TO 140
   30 DO 90 K = 1,SIZE
      IF (K .EQ. NEGINV) GO TO 90
      GK = GR(K) * XKPOS
      IF (ABS(GK) .LE. TOL5) GO TO 90
      J = XBASIS(K)
      IF (J .GT. N) SI = S(J-N)
      IF (J .LE. N) BOUNDJ = BOUND(J)
      XX = XR(K)
      IF (GK .LE. 0.0) GO TO 70
      IF (XX .LT. 0.0) GO TO 90
      IF(J.LE.N.AND.BOUNDJ.EQ.-1.0.OR.J.LE.N.AND.XX.LE.BOUNDJ) GO TO 40
      IF (J .GT. N .AND. SI .EQ. 1.0) GO TO 40
      GO TO 90
   40 IF (XX.GE. GK * R) GO TO 90
   50 R = XX / GK
   60 IF (R .LE. SMALL) R = 0.0
      NEWY = K
      IF (R .EQ. 0.0) GO TO 140
      GO TO 90
   70 IF (J .GT. N) GO TO 80
      IF (BOUNDJ.EQ.-1.0.OR.XX.LT.0.0.OR.XX.GT.BOUNDJ) GO TO 90
      IF ((XX - GK * R) .LE. BOUNDJ) GO TO 90
      R = (BOUNDJ - XX) / (-1.0 * GK)
      GO TO 60
   80 IF (XX .GE. 0.0 .OR. S(J-N) .GE. 0.0) GO TO 90
      IF ((XX - GK * R) .LE. 0.0) GO TO 90
```

```
 90 CONTINUE
    DO 130 I = 1,MNOW
    IF (ISEFF(I) .EQ. 0) GO TO 100
    G(I) = 0.0
    GO TO 130
100 SLACKI = SLACK(I)
    SI = S(I)
    GI = 0.0
    ISTART = IROW(I)
    LAST = IROW(I+1) - 1
    DO 120 LOOK = ISTART,LAST
    J = JCOL(LOOK)
    INJ = INBASE(J)
    IF (INJ .LE. 0) GO TO 110
    GI = GI - AA(LOOK) * GR(INJ)
    GO TO 120
110 IF (J .EQ. NEWX) GI = GI + AA(LOOK)
120 CONTINUE
    G(I) = GI
    IF (ABS(GI) .LE. TOL5) GO TO 130
    IF (SI .EQ. 0.0 .AND. SLACKI .NE. 0.0) GO TO 130
    IF (SI * SLACKI .LT. 0.0) GO TO 130
    GI = GI * XKPOS
    T = SLACKI - GI * R
    IF(T.GE.0.0 .AND. SI.EQ.1.0 .OR. T.LE.0.0 .AND.SI.EQ.-1.0)GO TO130
    R = SLACKI / GI
    IF (R .LE. SMALL) R = 0.0
    NEWY = SIZE + I
    GR(SIZE1) = GI * XKPOS
    IF (R .EQ. 0.0) GO TO 140
130 CONTINUE
140 RETURN
    END
```

REFERENCES

(1) Moore, F.T., "Economies of Scale: Some Statistical Evidence", Quarterly Journal of Economics, 73 (May, 1959) 232 - 245.

(2) Haldi, J., "Economies of Scale in Economic Development", Memorandum No. E-7, Stanford Project for Quantitative Research in Economic Development, Stanford, California, 1960.

(3) Haldi,J and Whitcomb D., "Economies of Scale in Industrial Plants", The Journal of Political Economy, 75 (August, 1967) 373-385.

(4) Manne, A.S., "Capacity Expansion and Probabilistic Growth", Econometrica, 24 (Oc t., 1961) 637 - 649.

(5) Manne, A.S. Editor Investments for Capacity Expansion : Size, Location, and Time Phasing. Cabridge, Mass : The MIT Press, 1967.

(6) Effroymson M.A. and Ray, T.L., "A Branch-and-Bound Algorithm for Plant Location," Opns. Res., 14, 361 - 368 (1966)

(7) Soland, R.L., "Optimal Facility Location with Concave Costs", Opns Res- (May 1971).

(8) Tui, Huang, (1964), "Concave Programming under Linear Constraint Dokl. Akad. Nauk. SSSR 159 32-35. Translated (1964). Soviet Math Dokl. 4 1437-1440.

(9) Falk E.J. and Soland R.M., "An Algorithm for Separable Nonconvex Programming Problems." Mgt. Sc. Vol.15, No.9 May, 1969.

(10) Rech. P. and Barton L.G., "A Nonconvex Transportation Algorithm" Applications of Mathmetical Programming Techniques," E.M.L. Beale (ed.) American Elsevier Publishing Co., New York, 1970.

(11) Murty, K.G., "Solving the Fixed Charge Problem by Ranking the Extreme Points", Opns. Res. 16, 268-279 (1968)

(12) Cabot, A.V. and Francis, R.L. (1970), ""Solving Certain Non-convex Quadratic Minimization Problems by Ranking the Extreme Points," Opns. Res. 18 82-86.

(13) Taha, Hamdy, A. (1971). "Concave Minimization over a Convex Polyhedron", Technical Report No. 71-6, Department of Industrial Engineering, University of Arkansas, September.

(14) Falk, J.E. and Hoffman , K.R., "A Successive Underestimation Method for Concave Minimization Problems", Mathematics of Opns. Res. Vol. 1. No. 3, August 1976.

(15) Lawler, E.L., and Wood D.E., "Branch and Bound Methods: A. Survey", Opns, Res. 14, 699-719 (1966)

(16) Kleibohm, Von K. (1967), "Bermerkungen zum Problem der Nicht convexen Programmierung", Unternehmensforschung, Bank II, Heft 1, 49-60.

(17) Land, A.H. and Powell, S., "Fortran Codes for Mathematical Programming", John Wiley and Sons, 1973.