PSEUDO-BOOLEAN PROGRAMMING

FOR

BIVALENT OPTIMIZATION

PSEUDO-BOOLEAN PROGRAMMING FOR BIVALENT OPTIMIZATION

By

M. NATESAN, M. Tech.

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Engineering

McMaster University
March, 1973

MASTER OF ENGINEERING (1973)
(Mechanical Engineering)

McMaster University
Hamilton, Ontario

TITLE: PSEUDO-BOOLEAN PROGRAMMING FOR BIVALENT OPTIMIZATION

AUTHOR: M. NATESAN, M. Tech.

SUPERVISOR: Professor J. N. Siddall

NUMBER OF PAGES: vi,64

ABSTRACT

This thesis introduces an effective computational algorithm making use of Boolean algebra for solving bivalent optimization problems with linear and nonlinear constraints. This method is a combination of the algorithm suggested by Hammer and the branch and bound method. The whole system of constraints is replaced by a single Boolean resolvent function and the solutions of this resolvent are found by branch and bound method which are found to be the feasible solutions of the system of constraints. Some practical applications are also discussed.

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge the active cooperation of people who helped with this thesis work. In particular, I would like to thank Professor J. N. Siddall for suggesting the problem, and for his all round expert guidance and advice.

To Mr. Dave Bonham goes the credit for some invaluable suggestions towards program development.

And finally, a last word of thanks to Miss Veronica Komczynski, who typed the manuscript so well.

McMaster University Benefactor's Scholarship is gratefully acknowledged.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

It was Dantzig[1,2,3] who first recognized that a great variety of problems in operations research and related areas could be solved by means of mathematical programming with bivalent variables.

For a long time bivalent (zero-one) problems were solved by programs with integer variables by introducing additional constraints. Best-known among them are R.E. Gomory's algorithm[4,5] for solving linear programs with integer variables. The problems with zero-one variables were treated as a special case in the above algorithm.

The cutting plane approach has also been used by Beale[6] and Gomory to develop an algorithm for solving the mixed case when some but not all of the variables are required to be integers.

Another type of algorithm for integer and mixed integer linear programs developed by Land and Doig[7] also start with a non integer optimal solution and then finds the integer or mixed integer optimal solution subsequently.

However, special methods using the peculiarities of bivalent problems have also been studied. Working on these lines, Egon Balas[8] developed an algorithm for solving linear programs with variables constrained to take only one of the values, either zero or one. The algorithm starts by setting all the n variables equal to zero and consists of a systematic procedure of successively assigning to certain variables the value 1, in such a way after trying a part of all 2^n combinations, one obtains either an optimum solution, or evidence of the fact that no feasible solution exists. The only operations involved in this algorithm are additions and subtractions. So this algorithm is better known as the additive algorithm. The initial idea concerning the possibility of applying Boolean methods to economic problems came from Robert Fortet. He pointed out that the bivalent nature of Boolean algebra can be made use of in solving zero-one problems.

In 1963 Hammer, Rudeanu and Rosenberg(9,10) suggested a Boolean method for finding the minima of an integer valued function with bivalent (0,1) variables, the variables being possibly subject to certain constraints. Later on they extended the same method for real valued functions. This they called pseudo-Boolean programming. This pseudo-Boolean programming was then successfully applied for solving problems in operations research and economic problems. But the above method suggested by them involved manual inspection and a lot of hand computation. For problems with large number of variables this was quite time consuming and from the view point of making a computer program it was not efficient. Chapter IV explains some of the above concepts.

Later on Hammer(11) looking for an alternative method tried to replace the whole system of constraints by a single Boolean function which he called the resolvent. The system of constraints may include linear as well as nonlinear constraints. Those solutions which make up the resolvent zero were found to be the feasible solutions of the whole system of constraints. But the time consuming effort of solving the resolvent for its feasible solutions was not overcome.

Yoshida, Inagaki and Fukumura(12) suggested a branch and bound technique to minimize a pseudo-Boolean problem under a constraint equation expressed in the form of a Boolean function. The Boolean constraint function in n variables is systemattically reduced to a single variable by the technique of successive elimination.(13) At this point one can determine from the consistency of the constraint function whether it can have a feasible solution. Then, the feasible solutions are built up by adding the variables one by one. In this process the solutions which give an objective function value more than a prespecified limit are left behind. This speeds up the whole process of getting to an optimum solution without trying all the possible combinations.

At this point a brief introduction to branch and bound method on which the thesis work is developed is given.

Branch and Bound Method

Among the most general approaches to the solutions of constrained optimization problems is that of branching and bounding. This is an 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 an upper bound (in the case of minimization) is calculated within each subset. After each partitioning those subsets with a bound greater than the specified bound are excluded from all further partitioning. The total amount of computations is related to the number of distinct bounding problems created, and hence to the total number of nodes in the fully developed tree.

Some areas of application in mathematical programming which make use of branch and bound method to a large extent are integer programming, nonlinear programming, the travelling salesman problem, the quadratic assignment problem, etc. The branch and bound technique can of course be applied to a variety of problems in scheduling, decision processes, etc.

The name branch and bound arises from the two basic operations:

(a) Branching: which consists of dividing collections of sets of solutions into subsets.

(b) Bounding: which consists of establishing bounds on the values of the objective function over the subsets of solutions.

The branch and bound procedure involves recursive application of branching and bounding operations with provisions made for deleting subsets known not to contain an optimal solution.

Regarding the fields of application are considered, in operations research - travelling salesman problem, scheduling and transportation problems, in the field of science and engineering - graph theory, flows in network etc.(8) and also to a number of miscellaneous problems, some of which are discussed in detail in chapter VI. The field of application is slowly getting widened and in that connection this thesis work constitutes an introductory work for the larger problems of optimizing large problems, particularly structural problems, where the running time is prohibitive when the conventional optimization techniques are used. An interesting possibility is to discretize the variables into relatively few values and transform the problem into a zero-one programming problem. The first trial solution would then be rediscretized into a narrower region in the vicinity of the first solution. It is the anomaly of the current techniques that integer methods require more computer time than continuous variables methods, yet less information is required. The first step in this approach is to develop an efficient and rapid technique for zero-one programming which is the aim of this thesis work.

This is done by a good combination of the two methods, one to replace the whole system of constraints by a single Boolean function, called the resolvent and then to solve the resolvent for its feasible solutions by the branch and bound method. A computer program has been developed based on the above combination.

In chapter II some of the basic fundamentals of Boolean algebra which have been made use of in the development of the program are examined.

Chapter III deals with some of the pseudo-Boolean programming methods developed earlier for hand computation.

Chapter IV defines the resolvent $\phi(x_1,\ldots,x_n)$ of a system of linear and/or nonlinear inequalities in 0-1 variables, as being a Boolean function with the property that the set of solutions to the original problem coincides with the set of solutions of the Boolean equation

$$\phi(x_1,\ldots,x_n) = 0$$

A simple method of determining the function ϕ is given.

Chapter V shows how the resolvent is successively reduced to a single variable and how the branch and bound method is used to check the feasibility of the sequence of Boolean equations and to produce an optimal solution if there is any.

Chapter VI deals with some of the well known problems that can be solved using pseudo-Boolean programming.

In Appendix A some well known Boolean expressions are listed. Appendix B lists the computer program developed to solve the pseudo-Boolean problem.

CHAPTER II

BOOLEAN ALGEBRA

2.1 DEFINITION OF BOOLEAN ALGEBRA(13)

By a Boolean algebra we mean a set B_2 in which two elements 0 and 1 are distinguished and three operations disjunction (U), conjunction (·) and negation (¯) are defined.

2.2 NOTATIONS AND TERMINOLOGY

The disjunction (U) is defined by

x	U	y	=	z
0		0		0
0		1		1
1		0		1
1		1		1

(2.1)

The conjunction (·) is defined by

x	.	y	=	z
0		0		0
0		1		0
1		0		0
1		1		1

(2.2)

and the negation is defined by

x	$\bar{x}$
0	1
1	0

(2.3)

It is also very easy to note that, since a and b are variables belonging to the set {0,1} we can always write

$$a \cup b = a + b - ab$$

$$= \max(a,b) \qquad (2.4)$$

$$a \cdot b = \min(a,b) \qquad (2.5)$$

$$\bar{a} = 1 - a \qquad (2.6)$$

and that the operation of disjunction is commutative and associative.

These properties permit us to introduce the symbol

$$\bigcup_{\ell=1}^{k} x_\ell = x_1 \cup x_2 \cup \ldots\ldots \cup x_k$$

$$\prod_{\ell=1}^{k} x_\ell = x_1 \cdot x_2 \ldots\ldots x_k \qquad (2.7)$$

Some of the basic Boolean identities which will be frequently used in this work are*

$$a \cup a = a$$

$$a \cup b = b \cup a$$

$$a \cup ab = a$$

$$a \cup bc = (a \cup b)(a \cup c)$$

$$\overline{a \cup b} = \bar{a} \cdot \bar{b} \qquad (2.8)$$

$$a \cdot a = a$$

$$a \cdot b = b \cdot a$$

$$a \cdot (a \cup b) = a$$

$$a \cdot (b \cup c) = a \cdot b \cup a \cdot c$$

$$\overline{ab} = \bar{a} \cup \bar{b} \qquad (2.9)$$

*See Appendix A for more identities

Other important relations are $a \cup o = a$ and $a \cup 1 = 1$ for any $a \in \{0,1\}$.

2.3 BOOLEAN FUNCTIONS

A function $f(x_1,\ldots,x_n)$ whose variables and values belong to 0,1 is called a Boolean function.

The following observations regarding the Boolean function are worth noting.

Any Boolean function can be represented as a disjunction of elementary conjunctions. Such an expression is called the normal disjunctive form of the given function. By elementary conjunction we shall mean a product of the form

$$c = \prod_{j \in A} X_j \cdot \prod_{j \in B} \bar{X}_j$$

where $X_j \in \{0,1\}$ $j = \{1,\ldots,n\}$

while A and B are disjoint subsets of $\{1,\ldots,n\}$

This may be illustrated by the following example.

$$(x_2 \cup x_4) \cdot x_1 \cdot \bar{x}_3 \cup x_3 \cdot \bar{x}_4 \cdot x_5 \cup x_3 \cdot \bar{x}_4 \cup x_2 \cdot \bar{x}_5 \qquad (2.10)$$

Writing the above in disjunctive form

$$x_1 \cdot x_2 \cdot \bar{x}_3 \cup x_1 \cdot \bar{x}_3 \cdot x_4 \cup x_3 \cdot \bar{x}_4 \cdot x_5 \cup x_3 \cdot \bar{x}_4 \cup x_2 \cdot \bar{x}_5 \qquad (2.11)$$

(Normally (.) are omitted in between variables while writing a Boolean expression.)

Throughout this work all Boolean functions are written in the above form.

It is to be remarked that the disjunctive form of a given Boolean function is not unique. The equation (2.11) can also be written as

$$x_1x_2\bar{x}_3 \cup x_1\bar{x}_3x_4 \cup x_3\bar{x}_4 \cup x_2\bar{x}_5 \qquad (2.12)$$

By a Boolean equation (inequality) we mean an equation (inequality) of the form

$$f(x_1,\ldots,x_n) = g(x_1,\ldots x_n)$$

(respectively of the form

$$f(x_1,\ldots,x_n) \lesseqgtr g(x_1,\ldots,x_n)$$

where f and g are Boolean functions.

Two (systems of) Boolean equations (inequalities) are called equivalent if they have the same solutions.

The following remarks will be useful in understanding Boolean functions.(13)

1. A Boolean function $f = g$ is equivalent to the Boolean equation $f\bar{g} \cup g\bar{f} = 0$ and also $fg \cup \bar{f}\bar{g} = 1$, while a Boolean inequality $f \leq g$ is equivalent to the Boolean equation $f\bar{g} = 0$ and also $\bar{f} \cup g = 1$.

2. A system of Boolean equations of the form $h_j = 0$ $(j = 1,\ldots,m)$, is equivalent to the Boolean equation

$$\bigcup_{j=1}^{m} h_j = 0$$

while a system of Boolean equations of the form $k_j = 1$ $(j = 1,\ldots,m)$ is equivalent to the Boolean equation

$$\prod_{j=1}^{m} k_j = 1$$

3. Any system of Boolean equations and (or) inequalities is equivalent to a single Boolean equation of the form $h = 0$ (and also to an equation of the form $k = 1$).

2.4 PSEUDO-BOOLEAN FUNCTIONS

The pseudo-Boolean function is a real valued function with bivalent variables, for example

$$x_1x_2\bar{x}_3 \cup x_2x_4\bar{x}_5 \cup x_2x_3x_6 \cup x_5$$

is a Boolean function

whereas

$$6x_2x_3 - 5x_2\bar{x}_4x_5 + 4x_1x_6x_7$$

is a pseudo-Boolean function.

With regard to the properties of pseudo-Boolean functions, we notice that such a function is always linear in each of its variables.

i.e.,

$$f(x_1,\ldots,x_n) = x_i \cdot g(x_1,\ldots,x_{i-1}, x_{i+1},\ldots,x_n) + h(x_1,\ldots,x_{i-1}, x_{i+1}, x_n)$$

More generally we have the following result due to Gaspar,(14)

" Every pseudo-Boolean function may be written as a polynomial which is linear in each variable and which after the reduction of the similar terms is uniquely determined upto the order of the sums and products."

An equation (inequality) between two pseudo-Boolean functions is called a pseudo-Boolean equation (inequality).

A problem of minimizing or maximizing a pseudo-Boolean function whose variables are subject to a system of pseudo-Boolean inequalities is called a pseudo-Boolean program (or a 0,1 program).

CHAPTER III

A REVIEW

3.1 GENERAL

In this chapter some of the earlier methods suggested by Hammer and Rudeanu(13) for solving pseudo-Boolean equations and inequalities are reviewed. A procedure is described in which the solutions are either completely listed or grouped into families of solutions. Each family is characterized by the fact that for certain fixed indices $i_1,..,i_p$ the corresponding variables have fixed values. $x_{i_1} = k_{i_1},..,x_{i_p} = k_{i_p}$, while the other variables $x_{i_{p+1}},...,x_{i_n}$ remain arbitrary.

3.2 LINEAR PSEUDO-BOOLEAN EQUATIONS

Let us consider an equation

$$a_1z_1 + b_1\bar{z}_1 + \ . \ . \ . + a_nz_n + b_n\bar{z}_n = K \tag{3.1}$$

where a_i, b_i $(i = 1,...,n)$ and K are constants.

We may assume $a_i \neq b_i$.

For each i let us set

$$x_i \begin{cases} z_i & \text{if } a_i > b_i \\ \bar{z}_i & \text{if } a_i < b_i \end{cases} \tag{3.2}$$

Then the terms $a_iz_i + b_i\bar{z}_i$ may be transformed as follows

$$\begin{aligned} a_iz_i + b_i\bar{z}_i &= (a_i-b_i)x_i + b_i \quad \text{if } a_i > b_i \\ &= (b_i-a_i)x_i + a_i \quad \text{if } a_i < b_i \end{aligned} \tag{3.3}$$

Thus equation (3.1) is transformed into

$$C_1x_1 + C_2x_2 + \ldots C_nx_n = d \qquad (3.4)$$

where $C_1,..,C_n$, d are constants, $C_i > 0$ $(i = 1,..,n)$ and in reindexing the unknowns we can suppose that

$$C_1 \geq C_2 \geq \ldots \geq C_n > 0 \qquad (3.5)$$

Now we are interested in finding a procedure for solving a canonical form (3.4) under the assumption (3.5). But it would be unreasonable to try out all the 2^n possibilities. Hammer suggested that the systemmatic use of the following table(13) (3.1) would avoid most of the blind alleys.

Table (3.1) studies eight mutually exclusive cases concerning equation (3.4) and covering all situations. It is to be noted that unless equation (3.4) is inconsistent or it has a unique solution, we must continue in Table (3.1) to the new equations that resulted at the first step. This process is continued until all the possibilities are exhausted.

When applied to problems this procedure was found to give all the solutions of the equation (3.4). If T is the transformation from the equation (3.1) to equation (3.4) then the solutions of (3.1) are obtained by applying T^{-1} to the solutions of (3.4).

3.3 LINEAR PSUEDO-BOOLEAN INEQUALITIES

The most general form of inequality is either

$$a_1z_1 + b_1\bar{z}_1 + a_2z_2 + b_2\bar{z}_2 + \ldots + a_nz_n + b_n\bar{z}_n > h \qquad (3.5)$$

or

$$a_1z_1 + b_1\bar{z}_1 + a_2z_2 + b_2\bar{z}_2 + \ldots + a_nz_n + b_n\bar{z}_n \geq K \qquad (3.6)$$

TABLE 3.1

No.	Case	Conclusions
1.	$d < 0$	No solutions
2.	$d = 0$	The unique solution is $x_1 = x_2 = \ldots = x_n = 0$
3.	$d > 0$ and $C_1 \geq \ldots \geq C_p > d \geq C_{p+1} \ldots \geq C_n$	The solution, if any, satisfy $x_1 = \ldots = x_p = 0$ and $\sum_{j=p+1}^{n} C_j x_j = d$
4.	$d > 0$ and $C_1 = \ldots = C_p = d > C_{p+1} \geq \ldots \geq C_n$	a) for every $K=1,\ldots,p$: $x_K = 1$ $x_1 = \ldots = x_{K-1} = x_{K+1} = . = x_n = 0$ is a solution b) The other solutions, if any, satisfy $x_1 = \ldots = x_p = 0$ and $\sum_{j=p+1}^{n} C_j x_j = d$
5.	$d > 0$, $C_i < d$ $(i = 1,2,\ldots,n)$ and $\sum_{i=1}^{n} C_i < d$	No solutions
6.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$ and $\sum_{i=1}^{n} C_i = d$	The unique solution is $x_1 = \ldots = x_n = 1$
7.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$ $\sum_{i=1}^{n} C_i > d$ and $\sum_{j=2}^{n} C_i < d$	The solution, if any, satisfy $x_1 = 1$, $\sum_{j=2}^{n} C_j x_j = d - C_1$
8.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$ $\sum_{i=1}^{n} C_i > d$ and $\sum_{j=2}^{n} C_i \geq d$	The solution, if any, satisfy $x_1 = 1$ and $\sum_{j=2}^{n} C_j x_j = d - C_1$ $x_1 = 0$ and $\sum_{j=2}^{n} C_j x_j = d$

TABLE 3.2

No.	Case	Conclusions
1.	$d \leq 0$	The unique solution is $x_1 = x_2 = \ldots = x_n = 0$
2.	$d > 0$ $C_1 \geq \ldots \geq C_p \geq d > C_{p+1} \geq \ldots \geq C_n$	a) For every $k = 1,2,\ldots,p$ $x_K = 1$ $x_1 = \ldots = x_{K-1} = x_{K+1} = \ldots = x_n = 0$ is a basic solution b) The other basic solutions (if any) are characterized by the property, $x_1 = \ldots = x_p$ $= 0$ and $(x_{p+1},\ldots,x_n)$ is a basic solution of $\sum_{j=p+1}^{n} C_j x_j \geq d$
3.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$ and $\sum_{j=1}^{n} C_i < d$	No solutions
4.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$ and $\sum_{i=1}^{n} C_i > d$ and $\sum_{j=2}^{n} C_j < d$	The basic solutions(if any) are characterized by the property: $x_1 = 1$ and $(x_2,\ldots,x_n)$ is a basic solution of $\sum_{j=2}^{n} C_j x_j \geq d-C_1$
5.	$d > 0$, $C_i < d$ $(i = 1,\ldots,n)$	The basic solutions(if any) are characterize by the property,either

cont'd...

$\sum_{i=1}^{n} C_i > d$ and $\sum_{j=2}^{n} C_j \geq d$	$x_1 = 1$ and $(x_2,\ldots,x_n)$ is a basic solution of $\sum_{j=2}^{n} C_j x_j \geq d - C_1$ or $x_1 = 0$ and $(x_2,\ldots,x_n)$ is a basic solution of $\sum_{j=2}^{n} C_j x_j \geq d$
6. $d > 0$, $C_i < d$ $(i=1,\ldots.,n)$ and $\sum_{i=1}^{n} C_i = d$	The unique solution is $x_1 = x_2 = \ldots\ldots = x_n = 1$

TABLE (3.3a)

Equation

No.	Case	Conclusions	Fixed variables	Remaining equations
1	$d_i < 0$	No solutions		
2	$d_i = 0$	All the appearing variables fixed	$x'_{i_1} = \ldots = x'_{i_n} = 0$	
3	$d_i > 0$ and $C_{i_1} \geq \ldots \geq C_{i_p} > d_i \geq C_{i_{p+1}} \ldots \geq C_{i_n}$	Part of the appearing variables fixed	$x'_{i_1} = \ldots = x'_{i_p} = 0$	$\sum_{j=p+1}^{n} C_{ij} x'_{ij} = d_i$
4	$d_i > 0$ and $C_{i_1} = \ldots = C_{i_p} = d_i \geq C_{i_{p+1}} \ldots \geq C_{i_n}$	There are p+1 possibilities $a_1, \ldots, a_p$, b	a_K $x'_{i_K} = 1$, $x'_{i_j} (j \neq K) = 0$ $(K=1,\ldots,p)$ b: $x'_{i_1} = \ldots = x'_{i_p} = 0$	 $\sum_{j=p+1}^{n} C_{i_j} x'_{ij} = d_i$
5	$d_i > 0, C_{i_j} < d_i$ $(j=1,\ldots,n)$	No solutions		

cont'd...

No.	Case	Conclusions	Fixed variables	Remaining equation
6	$d_i > 0, C_{i_j} < d_i$ $(j=1,\ldots,n)$ and $\sum_{j=1}^{n} C_{i_j} = d_i$	All appearing variables fixed	$x'_{i_1} = \ldots x'_{i_n} = 1$	
7	$d_i > 0 \quad C_{i_j} < d_i$ $(j=1,\ldots,n)$ $\sum_{j=1}^{n} C_{i_j} > d_i$ and $\sum_{j=2}^{n} C_{i_j} < d_i$	One variable fixed	$x'_{i_1} = 1$	$\sum_{j=2}^{n} C_{i_j} x'_{i_j} = d_i - C_{i_1}$
8	$d_i > 0, C_{i_j} < d_i$ $(j=1,\ldots,n)$ $\sum_{j=1}^{n} C_{i_j} > d_i$ and $\sum_{j=2}^{n} C_{i_j} \geq d_i$	There are two possibilities	(a) $x'_{i_1} = 1$ (b) $x'_{i_1} = 0$	$\sum_{j=2}^{n} C_{i_j} x'_{i_j} = d_i - C_{i_1}$ $\sum_{j=2}^{n} C_{i_j} x'_{i_j} = d_i$

TABLE 3.3b

Inequalities

No.	Case	Conclusions	Fixed Variables	Remaining Inequality
1	$d_i \leq 0$	Redundant inequality		
2	$d_i > 0$ and $C_{i_1} \geq \cdots \geq C_{i_p} \geq d_i > C_{i_{p+1}} \geq \ldots \geq C_{i_n}$	There are p+1 possibilities $a_1, \ldots, a_p$, b	a_K: $x'_{i_1} = \ldots = x'_{i_{k-1}} = 0$ $x'_{i_k} = 1 (K=1,\ldots,p)$	
			b: $x'_{i_1} = \ldots = x'_{i_p} = 0$	$\sum_{j=p+1}^{n} C_{i_j} x'_{i_j} \geq d_i$
3	$d_i > 0, C_{i_j} < d_i$ $(j=1,\ldots,n)$ and $\sum_{j=1}^{n} C_{i_j} < d_i$	No solutions		
4	$d_i > 0$ $C_{i_j} < d_i$ $(j=1,\ldots,n)$ $\sum_{j=1}^{n} C_{i_j} = d_i$	All appearing variables are fixed	$x'_{i_1} = \ldots = x'_{i_n} = 1$	

cont'd...

No.	Case	Conclusions	Fixed Variables	Remaining Inequality
5	$d_i > 0$, $C_{i_j} < d_i$ $(j=1,\ldots,n)$ $\sum_{j=1}^{n} C_{i_j} > d_i$ and $\sum_{j=2}^{n} C_{i_j} < d_i$	One variable fixed	$x'_{i_1} = 1$	$\sum_{j=2}^{n} C_{i_j}\, x'_{i_j} \geq d_i - C_{i_1}$
6	$d_i > 0$ $C_{i_j} < d_i$ $(j=1,\ldots,n)$ $\sum_{j=1}^{n} C_{i_j} > d_i$ and $\sum_{j=2}^{n} C_{i_j} \geq d_i$	There are two possibilities	(a) $x'_{i_1} = 1$ (b) $x'_{i_1} = 0$	$\sum_{j=2}^{n} C_{i_j}\, x'_{i_j} \geq d_i - C_{i_1}$ $\sum_{j=2}^{n} C_{i_j}\, x'_{i_j} \geq d_i$

Where a_i, b_i, h and K are constants and we may assume that $a_i \neq b_i$ for all i (If we have the sign $<$ or $\leq$ instead of $>$ or $\geq$ respectively we multiply the whole inequality by -1). If the constants a_i, b_i and h are integers, then the strict inequality (3.5) may also be written in the form (3.6), if we take $K = h + 1$. Therefore we shall confine our attention to inequalities of the form (3.6). As a matter of fact the method reported in this section in Table (3.2) for solving inequality (3.6) will directly offer solutions of the equation (3.1) and strict inequality (3.5).

As in the case of pseudo-Boolean equations the pseudo-Boolean inequality may be written as

$$C_1x_1 + \ldots + C_nx_n \geq d \qquad (3.7)$$

where $C_1, C_2, \ldots, C_n, d$ are constants and

$$C_1 \geq C_2 \geq \ldots\ldots \geq C_n > 0$$

3.4 SYSTEMS OF LINEAR PSEUDO-BOOLEAN EQUATIONS AND/OR INEQUALITIES

The method just described in the previous two sections for solving a linear pseudo-Boolean equation or inequality can easily be adapted to the more general case of a system of linear equations and/or inequalities (with real coefficients).

The algorithm proposed by Hammer and Rudeanu(13) for solving linear systems comprises three stages.

Step 1 All inequalities of the type $g \leq 0$ are replaced by $-g \geq 0$. In case of integer coefficients strict inequalities of the form $f > 0$ can also be dealt with by replacing them by $f-1 \geq 0$.

Step 2 If $x_1, \ldots, x_n$ are the unknowns of the system, the relation $x_i = 1 - \bar{x}_i$ can be used to write the i th inequality for each i.

$$C_{i_1}x'_{i_1} + C_{i_2}x'_{i_2} + \ldots + C_{i_n}x'_{i_n} \geq d_i$$

where $x_{i_1},\ldots,x_{i_n}$ are the variables on which the i th inequality effectively depends on and x'_i is either x or $\bar{x}$ so that

$$C_{i_1} \geq C_{i_2} \geq \ldots \geq C_{i_n} > 0$$

All equations of the system are written in a similar way. In other words, we bring each inequality and equation to the canonical form with respect to variables occuring effectively in it, but without changing the notation.

Step 3 Each equation or inequality is considered separately, and each one is written in the canonical form with respect to the variables x' contained in it. Each equation or inequality is analyzed by use of Tables (3.3a,3.3b) and the results of this are combined for the whole system.

For instance when a certain inequality or equation of the system has no solutions, then the whole system is inconsistent. In the same way if an equation has a unique solution $x_{i_1} = \ell_1$, $x_{i_2} = \ell_2,\ldots,x_{i_n} = \ell_n$ this should satisfy the remaining equations and inequalities of the system.

It may be seen from the above two tables that there are cases in which some of the variables are fixed, or in which there are no solutions, or in which the considered equation or inequality is redundant. These cases are called determinate. There are other cases where practically no information is available and they must be split into two cases for discussion, these cases are called indeterminate. Finally there are cases when the discussion is to be split into p+1 cases with increased information and they are called 'partially determinate'. The classification is shown below.

TABLE 3.4

Preferential order	Equation (Table 3.3a)	Inequality (Table 3.3b)	Characterization
First	1, 2, 3	1, 3, 4	Determinate
	5, 6, 7	5	
Second	4	2	Partially determinate
Third	8	6	Indeterminate

Now step 3 continues as follows. If some equations and inequalities belong to determinate cases, all corresponding conclusions are drawn. Two situations may arise. If at least one equation or inequality has no solutions or if two distinct equations or inequalities lead to the conclusions of the form $x_i = 1$ and $x_i = 0$ respectively, then the system has no solutions. It is preferable to start solving the system in the order of determinate, partially determinate and indeterminate.

If none of the equations and inequalities are in a determinate case then we look for partially determinate cases and we follow the conclusions corresponding to one of these cases.

3.5 NONLINEAR PSEUDO-BOOLEAN EQUATIONS AND INEQUALITIES

3.5.1 Characteristic Function

In the preceeding two sections a method was described for the determination of all solutions of a system of linear pseudo-Boolean equations and/or inequalities. In this part a method to replace the whole system of constraints by characteristic equation which has the same solutions as the system of constraints will be discussed. The construction of a characteristic equation is based on the reduction of the general case to the linear one.

3.5.2 Characteristic Function for a Linear Case

Any system of linear pseudo-Boolean equations or inequalities has a characteristic equation in a Boolean form

$$\phi(x_1,\ldots,x_n) = 1 \tag{3.8}$$

which has the same solutions as the system of expressions.

The characteristic function is given by the following expression

$$\psi(x_1,\ldots,x_n) = \underset{\alpha_1,\ldots,\alpha_n}{U^p} \quad x_1^{\alpha_1}\ldots x_n^{\alpha_n} \tag{3.9}$$

where $\underset{\alpha_1,\ldots,\alpha_n}{U^p}$ means the disjunction is extended over all solutions $(\alpha_1,\ldots,\alpha_n)$ of the system of expressions.

The above can be derived from the well known Boolean expression

$$\psi(x_1,\ldots,x_n) = \underset{\alpha_1,\ldots,\alpha_n}{U} \psi(\alpha_1,\ldots,\alpha_n)x_1^{\alpha_1}\ldots x_1^{\alpha_n} \tag{3.10}$$

where $\underset{\alpha_1,\ldots,\alpha_n}{U}$ means that the disjunction is extended over all 2^n possible systems of values 0,1 of $\alpha_1,\ldots,\alpha_n$ and the notation x^α means

$$x^\alpha = \begin{cases} x & \text{if } \alpha = 1 \\ \bar{x} & \text{if } \alpha = 0 \end{cases} \tag{3.11}$$

In other words we have

$$\psi(x_1,\ldots,x_n) = \underset{\alpha_1,\ldots,\alpha_n}{U^1} x_1^{\alpha_1}\ldots x_n^{\alpha_n} \tag{3.12}$$

where $\underset{\alpha_1,\ldots,\alpha_n}{U^1}$ means that the disjunction is extended over only those values of the vector $(\alpha_1,\ldots,\alpha_n)$ for which $\psi(\alpha_1,\ldots,\alpha_n) = 1$.

3.5.3 Linear Equations

In the case of a linear equation the knowledge of all solutions, obtained as described in Table 3.1 permits the direct formulation of the characteristic equation. This is illustrated by the following example.

Table 3.5 gives the solution of a certain pseudo-Boolean equation as shown.

TABLE 3.5

x_1	x_2	x_3	x_4	x_5
1	0	0	0	1
0	1	1	-	0
0	0	1	0	-

The characteristic equation is formed as

$$x_1\bar{x}_2\bar{x}_3\bar{x}_4x_5 \cup \bar{x}_1x_2x_3\bar{x}_5 \cup \bar{x}_1\bar{x}_2x_3\bar{x}_4 = 1$$

3.5.4 Nonlinear Equations

Let us consider a nonlinear pseudo-Boolean equation with the unknowns $x_1,\dots,x_n$

$$a_1P_1 + \dots + a_mP_m = b \tag{3.13}$$

where each $P_i (i = 1,\dots,m)$ stands for a certain conjunction (i.e., a product of variables with or without negations). One can replace the product P_i by a single bivalent variable y_i and solve the resulting linear pseudo-Boolean equation

$$a_1y_1 + a_2y_2 + \dots + a_my_m = b \tag{3.14}$$

where $y_1, y_2,\dots,y_n$ are treated as independent variables. If $\psi(y_1,\dots,y_m)$ is the characteristic equation of (3.14), then the Boolean function

$$\phi(x_1,\ldots,x_n) = \psi(x_{11}\cdots\cdot x_{1k},\ldots,x_{m1}\cdots\cdot x_{mk}) \qquad (3.15)$$

will be the characteristic function of (3.13).

The whole process of substitution and resubstitution is best illustrated in the following example

$$-6x_1\bar{x}_2x_3 - 4x_2x_4 + 2x_2x_4\bar{x}_5 + 4\bar{x}_3\bar{x}_4 = -2 \qquad (3.16)$$

We let $x_1\bar{x}_2x_3 = y_1$

$x_2x_4 = y_2$

$x_2x_4\bar{x}_5 = y_3$

$\bar{x}_3\bar{x}_4 = y_4$

The resulting linear equation has the form

$$-6y_1 - 4y_2 + 2y_3 + 4y_4 = -2$$

This equation is solved as described in Table 3.1

y_1	y_2	y_3	y_4
0	1	1	0
1	0	0	1

Hence the characteristic function of (3.16) is

$$\psi(y_1,\ldots y_4) = \bar{y}_1y_2y_3\bar{y}_4 \cup y_1\bar{y}_2\bar{y}_3y_4$$

Substituting for y_1, y_2, y_3 and y_4 in terms of x_1, x_2, x_3, x_4, and x_5 gives

$$\phi(x_1,\ldots,x_5) = (\bar{x}_1 \cup x_2 \cup \bar{x}_3) \cdot x_2x_4 \cdot x_2x_4\bar{x}_5$$
$$(x_3 \cup x_4) \cup x_1\bar{x}_2x_3 (\bar{x}_2 \cup \bar{x}_4)$$
$$(\bar{x}_2 \cup \bar{x}_4 \cup x_5) \cdot \bar{x}_3 \cdot \bar{x}_4$$

This reduces to

$$\phi(x_1,\ldots,x_5) = x_2 x_4 \bar{x}_5$$

The characteristic equation $\phi = 1$ gives the solutions of (3.16)

$$x_2 = 1, \quad x_4 = 1, \quad x_5 = 0 \qquad x_1, x_3 \text{ arbitrary}$$

3.5.5 Linear and Nonlinear Inequalities

To find the characteristic equation and therefrom to solve the solutions of the linear and nonlinear inequalities is similar to that of the equations except that different tables should be used to find the family of solutions. A family F of solutions was defined as being a set of solutions characterized by the fact that certain variables have fixed values, while the other remain arbitrary.

$$F : \; x_1 = \ell_1, \ldots, x_m = \ell_m, \; x_{m+k} \text{ arbitrary} \quad \text{for } k = 1,\ldots,n-m$$

For a particular problem the solutions are as shown below

x_1	x_2	x_3	x_4	x_5
-	-	1	-	-
-	1	0	1	-
0	1	1	0	1
1	0	1	-	1

The characteristic equation for the above is

$$\phi = x_3 \cup x_2\bar{x}_3x_4 \cup \bar{x}_1x_2x_3\bar{x}_4x_5$$
$$\cup\; x_1\bar{x}_2x_3x_5$$

3.5.6 Characteristic Function for Systems

We take a system of pseudo-Boolean equations and inequalities

$$f_j(x_1,\ldots,x_n) = 0 \qquad j = 1,\ldots,m$$

$$f_h(x_1,\ldots,x_n) \geq 0 \qquad h = m+1,\ldots,q \tag{3.17}$$

and let

$$\begin{aligned} &\psi_1(x_1,\ldots,x_n) = 1 \\ &\vdots \\ &\psi_q(x_1,\ldots,x_n) = 1 \end{aligned} \tag{3.18}$$

be the corresponding characteristic equations determined as in previous sections. If ϕ is the characteristic function of the system (3.17) it is given by

$$\phi(x_1,\ldots,x_n) = \prod_{s=1}^{q} \psi_s(x_1,\ldots,x_n) \tag{3.19}$$

CHAPTER IV

RESOLVENT

4.1 GENERAL

The role of this chapter is to define the resolvent of a system of pseudo-Boolean inequalities and to explain its effectiveness as a tool to replace the whole system of constraints.

4.2 COVER OF A LINEAR INEQUALITY(11)

Let us consider a linear pseudo-Boolean inequality

$$a_1' x_1 + \ldots + a_n' x_n \le b' \tag{4.1}$$

where $x_j \in \{0,1\}$ $j = 1,\ldots,n$

b' and a_j' $(j = 1,\ldots,n)$ are given real numbers.

We can rewrite equation (4.1) as

$$a_1 x_1^{\alpha_1} + \ldots + a_n x_n^{\alpha_n} \le b \tag{4.2}$$

where

$$a_j = |a_j'| \qquad j = 1,\ldots,n$$

$$b = b' - \sum_{j=1}^{n} \min (a_j' , 0)$$

$$\alpha_j = \begin{cases} 1 \text{ if } a_j' \ge 0 \\ 0 \text{ if } a_j' < 0 \end{cases}$$

and where

$$x^\alpha = \begin{cases} x & \text{if } \alpha = 1 \\ \bar{x} & \text{if } \alpha = 0 \end{cases}$$

For the sake of simplicity we shall assume that the terms of (4.2) are reordered, so that

$$a_1 \geq \;\ldots.\; \geq a_n \geq 0 \qquad (4.3)$$

If we define a set $N = \{1,2,\ldots,n\}$

Then a set of indices

$$J = \{j_1,\ldots,j_m\} \subseteq N$$

will be called a cover of the inequality (4.2) if

$$\sum_{j \in J} a_j > b \qquad (4.4)$$

The equation (4.4) will be a basic cover if no proper subset of J is a cover.

Example: To get the minimal covers of

$$6x_1 + 5\bar{x}_2 + 4x_3 + 2\bar{x}_4 \leq 7$$

Rewriting the above, after substituting

$$y_1 = x_1 \;, \quad y_2 = \bar{x}_2 \;, \quad y_3 = x_3 \;, \quad y_4 = \bar{x}_4$$

$$6y_1 + 5y_2 + 4y_3 + 2y_4 \leq 7$$

The minimal covers are

$\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3\}$

4.3 RESOLVENT OF A LINEAR INEQUALITY

If J is a basic cover the product

$$I_J(x) = \prod_{j \in J} x_j^{\alpha_j} \qquad (4.5)$$

will be called the basic implicant of the inequality (4.2).

If Ω is the family of all basic covers of (4.2), then the Boolean function

$$w(x) = \bigcup_{J\varepsilon\Omega} \prod_{j\varepsilon J} x_j^{\alpha_j} \tag{4.6}$$

will be called the basic resolvent or simply the resolvent of the inequality (4.2).

In effect we have replaced the pseudo-Boolean inequality with a Boolean function. In the example described in Section 4.2, the resolvent would be

$$w = y_1y_2 \cup y_1y_3 \cup y_1y_4 \cup y_2y_3$$

or

$$w = x_1\bar{x}_2 \cup x_1x_3 \cup x_1\bar{x}_4 \cup x_2x_3$$

4.4 RESOLVENT OF A NONLINEAR INEQUALITY

The replacement of a nonlinear inequality with a resolvent is very similar to that of a linear case.

We can write

$$\sum_{j=1}^{m} a_j y_j^{\alpha_j} \leq b \tag{4.7}$$

where

$$y_j = \prod_{h\varepsilon T_j} x_h \cdot \prod_{k\varepsilon Z_j} \bar{x}_k \tag{4.8}$$

and

$$T_j \cup Z_j \subseteq N \qquad j = 1,\ldots,n \tag{4.9a}$$

$$T_j \cdot Z_j = 0 \qquad j = 1,\ldots,n \tag{4.9b}$$

and where

$$y^{\alpha} = y \quad \text{if } \alpha = 1$$
$$\quad = \bar{y} \quad \text{if } \alpha = 0$$

If the resolvent of (4.7) is denoted as w(y) and if we introduce into w(y) the expressions (4.8) of the y_j' s

we get $\quad \psi(x) = w(y(x)) \qquad (4.10)$

Example: A Linear Case

$$5x_1 - 6x_2 + 8\bar{x}_3 + 4x_4 \geq 4 \qquad (4.11)$$

Rewriting the above as a ≤ inequality type

$$-5x_1 + 6x_2 - 8\bar{x}_3 - 4x_4 \leq -4$$

$$5\bar{x}_1 + 6x_2 + 8x_3 + 4\bar{x}_4 \leq 13$$

$$8x_3 + 6x_2 + 5\bar{x}_1 + 4\bar{x}_4 \leq 13$$

The resolvent of the above inequality (4.11)

$$w(x) = x_3x_2 \cup x_3\bar{x}_1\bar{x}_4 \cup x_2\bar{x}_1\bar{x}_4$$

In case of a nonlinear inequality

$$5y_1 - 6y_2 + 8\bar{y}_3 + 4y_4 \geq 4$$

where $\quad y_1 = x_1x_3, \quad y_2 = x_2x_3, \quad y_3 = x_1\bar{x}_2\bar{x}_4$

$$y_4 = \bar{x}_4$$

$$w(y) = y_3y_2 \cup y_3\bar{y}_1\bar{y}_4 \cup y_2\bar{y}_1\bar{y}_4$$

Substituting for y_1, y_2, y_3 and y_4 in terms of x's

$$w(x) = x_1x_2x_3x_4 \cup x_1x_2\bar{x}_4 \cdot (\overline{x_1 \cdot x_3}) \cdot x_4$$

$$\cup\ x_2x_3\ (\overline{x_1 \cdot x_3}) \cdot x_4$$

Simplifying

$$w(x) = x_2 \cdot x_3 \cdot x_4$$

4.5 RESOLVENT OF A SYSTEM OF INEQUALITIES

Let us consider a system of linear or nonlinear inequalities

$$\phi_i\ (x_1,\dots,x_n) \le b_i \qquad i = 1,\dots,m \tag{4.12}$$

$$x_j \in \{0,1\} \quad j = 1,\dots,n$$

If $w_i(x_1,\dots,x_n)$ is the resolvent of the i th inequality and

$$w(x_1,\dots x_n) = w_1 \cup w_2 \cup \dots \cup w_m \tag{4.13}$$

then $w(x_1,\dots,x_n)$ is called the resolvent of the system (4.12).

Next we must prove that a solution vector $x \in B_2^n$ is a solution of the system (4.12) if and only if it is a solution of the Boolean equation

$$w(x_1,\dots,x_n) = 0$$

where w is the resolvent of (4.12)

The proof for the above is very simple and directly follows from equation (4.6),

$$w_i(x) = \bigcup_{J\varepsilon\Omega}\ \prod_{j\varepsilon J}\ x_j^{\alpha_j} \tag{4.6}$$

which was defined as the basic resolvent.

It is clear in equation (4.5), if all $x_j^{\alpha_j} = 1$, that this set of values will be violating the constraints. So it is obvious, if

$$x_j^{\alpha_j}\ , \quad j = (1,\dots J)$$

is to form a solution of the corresponding constraint, that at least one of the above $x_j^{\alpha_j}$ must take a value zero. So in that case the product

$$I_J(x) = \prod_{j\varepsilon J} x_j^{\alpha_j} = 0 \tag{4.5}$$

It should be noted that the above argument holds good only if J is formed out of minimal covers. If $I_J(x) = 0$

$$w(x) = \bigcup_{J\varepsilon\Omega} \prod_{j\varepsilon J} x_j^{\alpha_j} = 0$$

So those solutions which satisfy the constraints, alone will make $w(x) = 0$. Putting it in a different way, only those solutions $(x_1,\ldots,x_n)$ which make $w(x) = 0$ can be the feasible solutions of the system of constraints.

Remark: The system of constraint equations (4.13) is inconsistent if and only if

$$w(x_1,\ldots,x_n) = 1$$

CHAPTER V

PSEUDO-BOOLEAN PROGRAMMING

5.1 GENERAL

In the previous chapter it was explained how a system of constraints can be replaced by a single Boolean resolvent function and it was also proven that the solutions which satisfy the system of constraints, when substituted in the resolvent will make it zero. So we are mainly interested in finding the solutions of the resolvent and such solutions are known as feasible solutions of the system. Among these solutions one or more optimum solutions exist which will minimize or maximize the objective function as the case may be. In this chapter a method for obtaining the feasible solutions of the system of constraints and the optimum solution is discussed.

5.2 FORMULATION OF THE PROBLEM

This section formulates the general problem to be treated in this thesis. Let

$$B_2 = \{0,1\}, \quad B_2^n = B_2 \times B_2^{n-1}$$

and $\quad x_i \in B_2 \qquad i = 1,2,\ldots,n$

We also denote $X \equiv (x_1,\ldots,x_n)$

and further let the first r components of X be denoted as

$$^{r}X = (x_1,\ldots,x_r)$$

$$X \in B_2^n \quad \text{and} \quad {}^{r}X \in B_2^r$$

Next two Boolean functions $Y_i(x)$ and $F_i(x)$ are defined with arguments x.

$$y(x) = \sum_{i=1}^{M} a_i Y_i(x) \tag{5.1}$$

and

$$F(x) = \bigcup_{j=1}^{K} F_j(x) \tag{5.2}$$

Our ultimate aim is to find an X which makes equation (5.2) zero and minimizes the objective function (5.1).

It is required in the algorithm that a_i values in equation (5.1) may not be negative. Any negative value present must be replaced using the following transformation

$$a_i y_i(x) = (-a_i)\bar{Y}_i(x) + a_i \quad (a_i < 0) \tag{5.3}$$

When a maximizing problem is encountered it can be reduced to that of a minimizing one by taking note of the fact that the maximum value of y(x) is equal to the minimum value of -y(x). This algorithm treats minimization problems only.

5.3 SOME BASIC THEOREMS

Before going into the details of the algorithm it becomes necessary to state certain basic theorems which are made use of in proving the validity of the algorithm.

Theorem 1(2)

Any system of Boolean equations and (or) inequalities is equivalent to a single Boolean equation of the form h = 0 and also to an equation of the form K = 1.

A Boolean equation f = g is equivalent to the Boolean equation

$$f\bar{g} \cup \bar{f}g = 0$$

or (5.4)

$$fg \cup \bar{f}\bar{g} = 1$$

While a Boolean inequality $f \leq g$ is equivalent to the Boolean equation

$$f\bar{g} = 0$$

and also (5.5)

$$\bar{f} \cup g = 1$$

Also a system of Boolean equations of the form $h_j = 0$ $(j = 1,\ldots,m)$, is equivalent to the Boolean equation $\bigcup_{j=1}^{m} h_j = 0$, while a system of Boolean equations of the form $K_j = 1$ $(j = 1,\ldots,m)$ is equivalent to the Boolean equation

$$\prod_{j=1}^{m} K_j = 1 \qquad (5.6)$$

(Since $xy = 1$ if and only if $x = 1$ and $y = 1$)

Considering a very simple case of a Boolean equation in one unknown, we can write the above equation in the form

$$f(x) = 0 \qquad (5.7)$$

or equivalently

$$ax \cup b\bar{x} = 0 \qquad (5.8)$$

where

$$a = f(1), \quad b = f(0)^{(3)} \qquad (5.9)$$

Theorem 2

Equation (5.8) is consistent if and only if

$$a \, . \, b = 0 \qquad (5.10)$$

Proof of the above is very simple. If x be a solution of (5.8) then

$$ax = b\bar{x} = 0$$

and hence $a \leq \bar{x}$ and $\bar{x} \leq \bar{b}$ (By Appendix A).

Therefore $a \leq \bar{b}$ (By Appendix A) or else

$$ab = 0$$

5.4 METHOD OF SUCCESSIVE ELIMINATIONS(13)

Let us start off with a general Boolean equation in n unknowns.

$$f(x_1,\ldots,x_n) = 0 \qquad (5.11)$$

In order to get a recursive relationship we take

$$f(x_1,\ldots,x_n) = f_1(x_1,\ldots,x_n)$$

Then

$$f_1(x_1,\ldots,x_n) = 0 \qquad (5.12.1)$$

The above may be written in the form

$$f_1(x_1,\ldots,x_{n-1},1)x_n \cup f_1(x_1,\ldots,x_{n-1},0)\bar{x}_n = 0 \qquad (5.13.1)$$

If we set

$$f_1(x_1,\ldots,x_{n-1},1) \,.\, f_1(x_1,\ldots,x_{n-1},0)$$
$$= f_2(x_1,\ldots x_{n-1}) \qquad (5.14.1)$$

then the condition that equation (5.13.1) has a solution with respect to x_n becomes

$$f_2(x_1,\ldots,x_{n-1}) = 0 \qquad (5.12.2)$$

We assume that the above method of eliminations is carried out in the i th step and therefore must solve the equation

$$f_i(x_1,\ldots,x_{n-i+1}) = 0 \qquad (6.12.i)$$

We write the above in the form

$$f_i(x_1,\ldots,x_{n-i},1)x_{n-i+1} \cup f_i(x_1\ldots,x_{n-i},0)\bar{x}_{n-i+1} = 0 \qquad (5.13.i)$$

Then once again we set

$$f_i(x_1,\ldots,x_{n-i},1) \,.\, f_i(x_1,\ldots,x_{n-i},0)$$
$$= f_{i+1}(x_1,\ldots,x_{n-i}) \qquad (5.14.i+1)$$

and from the equation

$$f_{i+1}(x_1,\ldots,x_{n-i}) = 0 \qquad (5.12.i+1)$$

In the n th step we obtain the equation

$$f_n(x_1) = 0 \qquad (5.12.n)$$

which may be written in the form

$$f_n(1)\; x_1 \cup f_n(0)\; \bar{x}_1 = 0 \qquad (5.13.n)$$

and

$$f_n(1) \,.\, f_n(0) = f_{n+1} \in B_2 \qquad (5.14.n)$$

If $f_{n+1} \neq 0$, then (5.13.n) or equivalently (5.12.n) has no solution in view of theorem 2, section (5.3). Since equation (5.12.n) is just the consistency condition of equation (5.13.n-1), it follows that the latter, which coincides with (5.12.n-1), is also inconsistent. We define by induction that (5.12.1) has no solution.

If $f_{n+1} = 0$, then the equation (5.12.n) is consistent. By introducing its solutions into the equation (5.12.n-1)

$$f_{n-1}(x_1,x_2) = 0$$

The latter becomes an equation with single unknown x_2, and is consistent for the reason explained above. We obtain thus the solutions (x_1,x_2) of the equation (5.12.n-1). We introduce them in the equation (5.12.n-2) etc. In the last step we introduce the solutions $(x_1,\ldots,x_{n-1})$ of equation (5.12.2) into equation (5.12.1), obtaining thus an equation with a single unknown x_n. After solving this we have at hand the solutions $(x_1,\ldots,x_n)$ of equations (5.12.1).

Thus the method of successive eliminations consists of two stages. The first one, which includes the steps from the beginning to the finding of f_{n+1}, may also be considered as a way of deciding whether or not the given equation is consistent. If $f_{n+1} = 0$, then the second stage leads to the determination of all solutions of (3.12.1) and therefrom solutions of (5.11).

5.5 THE BASIC ALGORITHM

Two function series $\{F^r(r_X)\}^{(12)}$ and $\{y^{(r)}(r_X)\}$, $r = 1,\ldots,n$ are defined as shown below

$$F^{(n)}(x) = F(x) \tag{5.16}$$

$$F^{(r)}(r_X) = F^{(r+1)}(r_X,0) \, . \, F^{(r+1)}(r_X,1) \tag{5.17}$$

The above replacement becomes possible by the method of successive eliminations.

$$y^{(r)}(r_X) = \sum_{i=1}^{m} a_i y_i^{(r)}(r_X) \tag{5.18}$$

where

$$y_i^{(n)}(x) = y_i(x) \tag{5.19}$$

$$y_i^{(r)}(r_X) = y_i^{(r+1)}(r_X,0) \, . \, y_i^{(r+1)}(r_X,1) \tag{5.20}$$

Next we define a parameter 'c', which is an arbitrary upper limit on the objective function value. Only those feasible solutions which give an objective function value less than or equal to the parameter 'c' are searched for an optimum solution. Writing this mathematically, if we define a set series $\{\psi^{(r)}(c)\}$ corresponding to the function series defined above, we get

$$\psi^{(r)}(c) = \{r_X / F^{(r)}(r_X) = 0, y^{(r)}(r_X) \leq c\} \quad (r = 1,2,\ldots,n) \tag{5.21}$$

We still must prove that the vector r_X, consisting of the first r components of an arbitrary element $r{+}1_X$ in the set $\psi^{(r+1)}(c)$, belongs to the set $\psi^{(r)}(c)$. From equation (5.20), the expression

$$y_i^{(r)}(r_X) \leq y_i^{(r+1)}(r_X, x_{r+1})$$

holds for any $r_X \in B_2^r$ and $x_{r+1} \in B_2$. Since all $a_i > 0$, from the relation above and equation (5.18), the inequality

$$y^{(r)}(r_X) \leq y^{(r+1)}(r_X, x_{r+1})$$

holds for any r_X and x_{r+1}. It follows that if an $(r_X, x_{r+1}) \in B_2^{r+1}$ fulfilling

$$y^{(r+1)}(r_X, x_{r+1}) \leq c$$

exists, it also satisfies the inequality

$$y^{(r)}(r_X) \leq c$$

In addition by successive elimination, we have shown how to get the solution step by step. So regarding

$$F^{(r+1)}(r_X, x_{r+1}) = 0 \qquad (5.22)$$

as a Boolean equation with respect to x_{r+1}, the condition

$$F^{(r)}(r_X) = 0$$

is just the necessary and sufficient condition for the Boolean equation (5.22) to have a solution.

The number of feasible solutions that are obtained depends on the value of 'c' chosen. A theoretical upper limit would be the sum of all positive coefficients in the objective function. If this value of 'c' is chosen, all the feasible and optimum solutions will be found.

5.6 ALGORITHM

Step 1

Assume a constant 'c'

a) If $F^{(1)}(x_1)$ is identically equal to unity, no feasible solution exists and the algorithm terminates.

b) If there is an x_1 present such that $F^{(1)}(x_1) = 0$, construct a set $\psi^{(1)}(c)$. If $\psi^{(1)}(c)$ is empty increase the value of 'c' by Δc ($\Delta c > 0$) to obtain another $\psi^{(1)}(c)$ which is not empty. Proceed to next step.

Step r ($r = 2,\ldots,n-1$)

a) If $\psi^{(r-1)}(c) \neq 0$, obtain $\psi^{(r)}(c)$ from $\psi^{(r-1)}(c)$ and to to step ($r+1$)

b) If $\psi^{(r-1)}(c) = 0$ replace 'c' by $c + \Delta c$ and return to step 1 (b).

Step n

a) If $\psi^{(n)}(c) \neq 0$, obtain $\psi^{(n)}(c)$ and the algorithm terminates.

b) If $\psi^{(n)}(c) = 0$ replace c by $c + \Delta c$ and return to step 1 b.

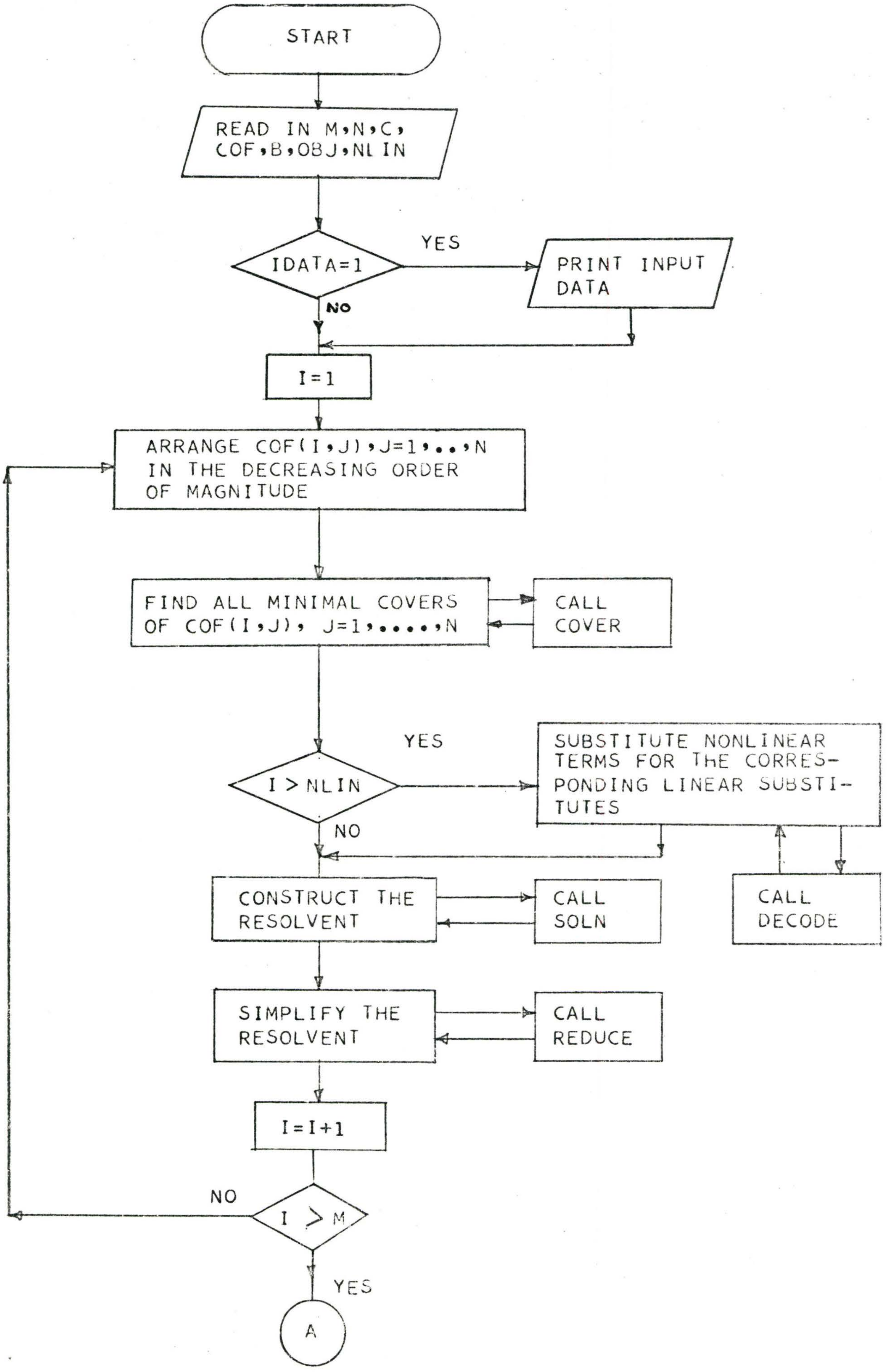
The algorithm terminates at (a) of step n, when a non-empty $\psi^{(n)}(c)$ is obtained. When no feasible solutions exist the algorithm terminates at (a) of step 1.

5.7 COMPUTER PROGRAM

5.7.1 Structure of the Program

A computer program has been developed, based essentially on the algorithm described in section (5.6) and is attached as Appendix B. This program solved problems with sixteen design variables and nineteen constraints efficiently. However the upper limit on size can not be specified. The basic difficulty is that it is not possible to predetermine the core memory requirements for any particular problem. Different problems with the same number of design variables and constraints may vary drastically in core memory requirements.

The computational time depends on the individual problem and also on the value of 'c', the upper limit. The time tends to increase with the



START
READ IN M,N,C, COF,B,OBJ,NLIN
IDATA=1
YES
NO
PRINT INPUT DATA
I=1
ARRANGE COF(I,J),J=1,..,N IN THE DECREASING ORDER OF MAGNITUDE
FIND ALL MINIMAL COVERS OF COF(I,J), J=1,....,N
CALL COVER
I > NLIN
YES
NO
SUBSTITUTE NONLINEAR TERMS FOR THE CORRESPONDING LINEAR SUBSTITUTES
CALL DECODE
CONSTRUCT THE RESOLVENT
CALL SOLN
SIMPLIFY THE RESOLVENT
CALL REDUCE
I=I+1
I > M
NO
YES
A

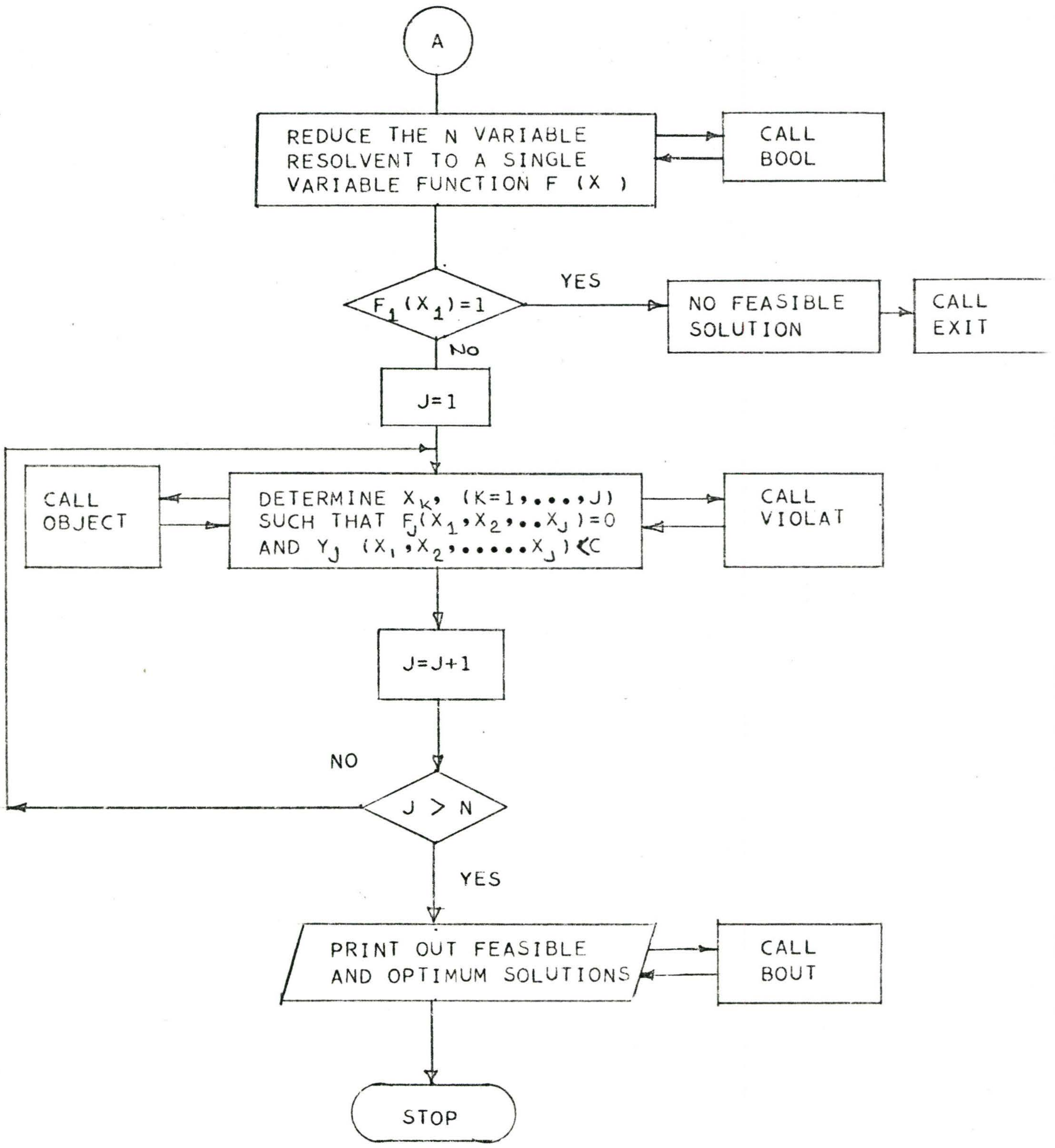

A
REDUCE THE N VARIABLE RESOLVENT TO A SINGLE VARIABLE FUNCTION F (X)
CALL BOOL
$F_1(X_1)=1$
YES
NO FEASIBLE SOLUTION
CALL EXIT
No
J=1
CALL OBJECT
DETERMINE X_K, (K=1,....,J) SUCH THAT $F_J(X_1,X_2,..X_J)=0$ AND $Y_J(X_1,X_2,.....X_J) \leqslant C$
CALL VIOLAT
J=J+1
NO
J > N
YES
PRINT OUT FEASIBLE AND OPTIMUM SOLUTIONS
CALL BOUT
STOP

Fig 5.1
General Arrangement of Subroutines.

value of 'c' since more and more feasible solutions must be completed and scanned for the optimum.

The computer program is capable of dealing with linear and nonlinear constraints and objective function. The nonlinear constraints may contain only product of variables as their terms. For nonlinear objective function subroutine UREAL should be written.

The accompanying flow diagram shows the general arrangement and the sequence in which the subroutines are called. A brief description of each subroutine is as follows:

Subroutine BABO calls the rest of the subroutines CANON, COVER, SOLN, LSTR, DECODE, STORE, REDUCE, ASEMBL, BOOL, CHOOSE, CHANGE, VIOLAT and OBJECT.

Subroutine CANON rearranges the coefficients of the variables in the constraints in the decreasing order of their magnitude, after changing the negative coefficients into positive and adding the corresponding quantity to the right hand side constant.

Subroutine COVER finds all the minimal covers of the constraints.

Subroutine SOLN identifies each member of the minimal cover with its corresponding variable and forms the resolvent.

Subroutine LSTR stores the terms of the resolvent, formed out of linear constraints.

If nonlinear constraints are present subroutine DECODE identifies the corresponding nonlinear terms for their linear substitutes and forms the resolvent. The terms of the resolvent are stored in subroutine STORE.

The total resolvent comprising of all the terms is simplified by the subroutine REDUCE using Boolean properties.

Subroutine ASEMBL uses the system routine SHIFT to assemble a number of single digit numbers into a single number. This is done essentially to save core memory space and whenever a particular digit is wanted out of this number, subroutines CHANGE and CHOOSE do the job.

Subroutine BOOL, using systematic elimination, forms various levels of the resolvent, from n variables to a single variable function.

Subroutine VIOLATE checks whether the Boolean resolvent function is satisfied at various levels, to determine whether a particular value of x can form a part of feasible solution.

The linear objective function is evaluated by the subroutine OBJECT.

The feasible and optimum solutions are printed out in a standard format by the subroutine BOUT.

5.7.2 The Limitations

If the time specified for the computation is insufficient, one has to start right from the beginning once again, since there are no iterations involved in finding the optimum solution and it is not possible to start from the stage where the computations were stopped.

This program contains two non-standard FORTRAN features. At one stage of program development, the complete resolvent formed with n variables, each term in the resolvent represented by a row and the various terms in the resolvent by the different rows in the matrix. If we denote the matrix of n variables as n th order, when this matrix was reduced step by step, by systematic elimination to a single variable, each step, depending on the number of variables it contains, had to be identified as the n-1 th order and so on down to the first order. Hence it became necessary to define a three dimensional array, in which the first number denotes the order, the second number denotes the rows in the matrix and the third a particular element in a row. This three dimensional array became too large for a problem with a comparatively large number of variables. So it was necessary to find an alternate way to reduce the core memory requirement. The CDC system routine SHIFT was used to pack the numbers previously represented by row, into a sixty bit word as a single number which reduces the core memory requirements by a factor n. Whenever a particular digit is needed for further computation, the same system routine SHIFT is used to pick up the required digit. This is available only on CDC 6400 computers and hence this program is machine dependent. Furthermore this facility restricts the number of variables that can be handled to thirty in the case of a problem with all linear constraints, since a sixty bit word can hold a maximum of thirty single digit numbers which can either be 0, 1 or 2, occupying two bits each. In the case of problems with nonlinear constraints, since the terms are read in octal format each number occupies three bits and a sixty bit word can hold a maximum of twenty numbers. Thus nonlinear problems are limited to a maximum of twenty variables.

The other non-standard feature is the use of octal format to read in the variables in the nonlinear constraints. This is necessary because any variable present in the nonlinear term is represented by 1, its negation by 2 and its absence by 0. For example the term $x_1\bar{x}_2x_3x_6$ (assuming there are only six variables) will be replaced by 121001. At a later stage we would like to pick up any particular digit for further computation. Unless the above number is read in octal format, under which each digit in a number is stored in three bits separately in a sixty bit word so that the picking up of any particular digit is nothing but extracting those three bits, it will not be possible to pick any desired digit.

CHAPTER VI

APPLICATIONS OF PSEUDO-BOOLEAN PROGRAMMING

6.1 GENERAL

In this chapter some practical applications of pseudo-Boolean programming methods, in various fields like operations research and science and engineering are presented. The computer program used to solve the problems is attached as Appendix B.

6.2 APPLICATIONS TO NETWORK PROBLEMS

6.2.1 The Travelling Salesman Problem(18,19)

During the last decade there has been a great deal of interest in problems that can be represented by networks. We will define a network as an array of nodes and branches. Each node is connected to at least one other node by at least one branch.

A well known example of the network problem is travel between cities, or the travelling salesman problem. We will let x_{ij} represent the branch from node i to j and specify $x_{ij} = 1$ if the branch from i to j is in the solution, and $x_{ij} = 0$ if the branch from i to j is not in the solution. A salesman is assigned n cities to visit. He is given the distances between all pairs of cities and instructed to visit each of the cities once, in one continuous trip and return to the starting city, using the route that is of minimum length. Since a complete cycle is involved, it does not make any difference which city is the starting city. The cities are numbered from 1 to n and let $x_{ij} = 1$ imply that salesman travels from city i directly to city j, and $x_{ij} = 0$ signify that the link from i to j is not in the tour.

In the matrix of distances, the distance from the city to itself (for every city) is set equal to an arbitrary large number. This is done to force each x_{ii} to be zero in the optimal solution. The problem is formulated as follows.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \, x_{ij}$$

where $C_{ij} \geq 0$ is the corresponding distances between cities i and j subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1,\ldots,n \quad \text{(departure)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for } j = 1,\ldots,n \quad \text{(arrival)}$$

Additional constraints are to be imposed to avoid subloops and to get a complete cycle as a solution.

$$x_{ij} + x_{ji} \leq 1 \qquad i = 1,\ldots n, \; j = 1,\ldots,n$$

prevents all subloops of order 2.

$$x_{it} + x_{tj} + x_{ji} \leq 2 \qquad i = 1,\ldots,n, \; j = 1,\ldots,n \quad t = 1,\ldots,n$$

prevents all subloops of order 3.

Subloops of higher order are prevented by sets of similar constraints. It is necessary to block out subloops of order n/2 or lower only, for higher order subloops can not exist if the lower order subloops have been prevented.

6.2.2 To Find the Longest Path

Many companies, involved in a long development project, such as designing and building missiles and space vehicles, use a technique called PERT. A PERT network is an array of steps in such a project. The first step is usually the receipt of the contract financing the project and the last step is the acceptance of the device. The steps in between consist of the design, construction and testing activity necessary in the project. There may be hundreds of nodes in the PERT network of a major project. The time to complete each activity (branch) is estimated from experience on similar projects. The length of the project is determined by the longest path from the start of the project to finish. The essential difference between PERT networks and the type discussed earlier is in the existence of precedence relations. There is a precedence of node i over node j if i can precede j but j cannot precede i. These arise naturally in development projects, for components should be designed before they can be built and should be built before they can be tested.

It should be noted that there are more efficient methods(20) of finding the longest route through a network, and one of these is normally used in PERT analysis rather than Boolean Programming. The demonstration of this programming approach to this problem is included here just as an application of this technique to network analysis.

6.3 ASSIGNMENT PROBLEM

In simple words this problem can be stated as follows.(20) Let us assume that we are given n requirements that must be satisifed and n methods of satisfying them, it being understood that each requirement must be satisfied by one of the methods and that one method cannot be used to satisfy more than one requirement. An n x n cost matrix is also given, each element C_{ij} being the cost of satisfying the j th requirement by the i th method. The assignment problem consists of finding that combination of methods and requirements that minimizes the total cost. It is specified that, if $x_{ij} = 1$, then the i th method is being used to satisfy the j th requirement, if $x_{ij} = 0$ the i th method is not being used to satisfy j th requirement. From the fact that each method is being associated with one, and only one requirement and that each requirement is associated with one and only one method the mathematical formulation is as shown

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \, x_{ij},$$

$$\text{Subject to} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n$$

$$x_{ij} = 0,\ 1.$$

6.4 QUADRATIC ASSIGNMENT PROBLEM

The quadratic assignment problem(17) differs from the ordinary assignment problem only in that a quadratic cost function is to be minimized rather than the linear one given above.

$$\text{Minimize} \quad \sum_{i,j} \sum_{p,q} C_{ijpq} \, x_{ij} \, x_{pq},$$

$$\text{Subject to} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n$$

$$x_{ij} = 0,\ 1 \qquad i,j = 1,\ldots,n$$

6.5 PLANT LOCATION

The problem of plant location has the following formulation.(21) Let $I = \{1,\dots,m\}$ be the set of places where the plants can be located, let $J = \{1,\dots,n\}$ be the set of consumers, let C_j be the annual rate of market requirements for location j, let a_i be the annual fixed cost of construction and of operation at plant i, let b_i be the manufacturing cost per unit plant at i, and let C_{ij} be the transportation cost per unit from i to j.

The problem consists in finding that subset of I, which assures a minimum for the total annual cost of construction, manufacturing and transportation.

Let us put $y_i = 1$ if a plant is to be located in i and $y_i = 0$, otherwise. Let x_{ij} denote the amount shipped from i to j and let $d_{ij} = bi + C_{ij}$. The problem is formulated as follows.

$$\text{Minimize} \quad \sum_{i=1}^{m} a_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}\, x_{ij}$$

$$\text{Subject to} \quad \text{If } y_i = 0,\ x_{ij} = 0 \qquad i = 1,\dots,m \quad j = 1,\dots,n$$

$$\sum_{i=1}^{m} x_{ij} = C_j \qquad (j = 1,\dots n)$$

$$x_{ij} \geq 0, \quad y_i \in \{0,1\} \quad (i = 1,\dots,m) \quad (j = 1,\dots,n)$$

It is necessary by suitable transformations to replace x_{ij}'s in the above expressions, in terms of (0-1) bivalent variables, so that the whole problem becomes a problem of pseudo-Boolean programming.

6.6 ELECTRONIC ASSEMBLY

Among so many other problems that can be solved by pseudo-Boolean programming a very interesting and a common problem(13) is formulated and solved below.

To construct an electronic device several ways are possible.

1. Any one of the three types T_1, T_2, T_3 of tubes may be used, but only one.

2. The box may be made of wood (W) or plastic material (M). But when using M dimensionality requirements impose the choice of T_2, and there is no place for the transformer F and a special power supply 'S' is needed.

3. T_1 needs F.

4. T_2 and T_3 need S (and not F)

The price of the above mentioned components are

Tube T_1	28 units
Tube T_2	30 units
Tube T_3	31 units
Transformer (F)	25 units
Special power supply(S)	23 units
Wooden Box (W)	9 units
Plastic material box (M)	6 units

The other necessary components of the device have the following costs.

27 units, if tube T_1 is used,

28 units, if tube T_2 is used,

and 25 units, if tube T_3 is used.

The assembly cost is 10 units for each set in all cases. Set is sold at 110 units if it is enclosed in a plastic material box and at 115 units in the other case.

It is to be determined which design is to be used in order to maximize the profit.

The problem is solved as follows. For each utilizable component X, we shall denote by x the Boolean variable

$$x = \begin{cases} 1 & \text{if } X \text{ is used} \\ 0 & \text{if } X \text{ is not used.} \end{cases}$$

The conditions become,

$$t_1 + t_2 + t_3 = 1 \tag{6.1}$$

$$w + m = 1 \tag{6.2}$$

$$\text{If } M=1,\ t_2 = s = 1 \tag{6.3}$$

$$\text{If } t_1 = 1,\ f = 1 \tag{6.4}$$

$$\text{If } t_2 = 1,\ s = 1 \tag{6.5}$$

$$\text{If } t_3 = 1,\ s = 1 \tag{6.6}$$

$$\text{and} \quad f + s = 1 \tag{6.7}$$

Under the above constraints, maximize

$$110w + 105m - (28t_1 + 30t_2 + 31t_3 + 25f + 235$$
$$+ 9w + 6m + 27t_1 + 28t_2 + 25t_3 + 10)$$

The constraints (6.3), (6.4), (6.5) and (6.6) are obviously equivalent to

$$m \leq t_2 s \tag{6.8}$$

$$t_1 \leq f \tag{6.9}$$

$$t_2 \leq s \tag{6.10}$$

$$t_3 \leq s \tag{6.11}$$

The above is a pseudo-Boolean problem with a linear objective function and system of mixed linear and nonlinear constraints.

The above problem was solved using the program in Appendix B and the obtained results are, $t_1 = 0$, $t_2 = 0$, $t_3 = 1$, $f = 0$, $s = 1$, $w = 1$, $m = 0$ which means we have to choose T_3 tube, the special power supply and the wooden box, assuring a profit of 12 units. This result was found to be the same as that of the solution obtained from hand computation(13) using Tables (3.2) and (3.3).

6.7 DESIGN OF A SYSTEM WITH RELIABILITY

The original problem(22) is to design a system with six controllers at six different stations, with sufficient redundant controllers at each station so as to maximize the reliability, for a maximum cost of \$65000. Four different alternate designs are available for each station. Since the number of design variables involved in solving the problem exceed the handling capacity of this developed program, the system is designed taking the first three stations and two design alteratives only.

Station	Design alternatives			
	R	C	R	C
1	.9983	2100	.9967	1800
2	.9992	3600	.9906	2900
3	.9846	1500	.9637	1400

The problem is formulated as follows:

Let x_i be the number of components that can be used at station i. If T is the upper limit on the availability of components

$$x_i = \sum_{K=1}^{T} K \cdot Z_{iK}$$

where $Z_{iK} = 0$ or 1

and $\sum_{K=1}^{T} Z_{iK} = 1$

The constraints have the form

$$\sum_{i=1}^{T} \sum_{j=1}^{m_i} C_{ij}\, y_{ij}{}^{*} \sum_{K=1}^{T} K \cdot Z_{ik} \leq C_s$$

where m_i is the number of design alternatives available, C_s is the total expenditure authorized, and C_{ij} is the cost of using the j th design alternative in the i th station.

Also

$$\sum_{K=1}^{T} Z_{iK} = 1 \qquad i = 1, 2, 3$$

$$\sum_{j=1}^{m_i} y_{ij} = 1 \qquad i = 1,2,3$$

Z_{iK} and y_{ij} are 0-1 variables.

The reliability is to be maximized

$$U = \prod_{i=1}^{3} [1-(1-\sum_{j=1}^{m_i} R_{ij}\, y_{ij})^{x_i}]$$

The solution of the above problem was obtained to give a maximum reliability of 0.99868 with two components of the first design alternatives at station one, two components of the first design alternatives at station two and two components of the second design alternative at station three.

CHAPTER VII

CONCLUSIONS

As mentioned in the introduction, for a long time bivalent problems have been solved as a part of integer programming. Growing applications of bivalent programming and the inherent difficulties in modifying other algorithms to solve bivalent problems necessitated the development of separate algorithms which deal with 0-1 problems exclusively. The algorithm suggested in this thesis, apart from the Balas' additive algorithm, is one of the very few that are directly applicable to bivalent optimization. An efficient computational program based on the particular combination of replacing the system of constraints by a Boolean function and then to solve the same branch and bound method for its feasible solutions, has been developed for the first time in this thesis.

The best feature of this computational program lies in the fact that the search for an optimum never stalls. The program finds the optimum solution (if there is any) always.

The efficiency of the algorithm can be improved by devising some method to deal with the equality constraints directly, instead of replacing each one of them by two inequalities as it is done in the present algorithm.

The wide range of applicability of Boolean techniques as explained in chapter VI makes obvious the necessity for further research in zero-one programming. The fields of integer programming, graph theory and other domains offer very attractive problems which are easy to translate into Boolean language. Further investigations are also necessary in developing improved techniques, perhaps to overcome the limitations explained in chapter V, and also it would be useful to have a pseudo-Boolean procedure for solving mixed continuous-bivalent programs.

REFERENCES

1. Dantzig, G.B., "Discreet Variable Extremum Problems", Operations Research, 5, 1957, n 2.

2. Dantzig, G.B., "On the Significance of Solving Linear Programming Problems with Integer Variables", Econometrica, 28, 1960 n.1.

3. Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, 1963.

4. Gomory, R.E., "Essentials of an Algorithm for Integer Solutions to Linear Programs", Bull. Amer. Math. Soc. 64, 275-278, 1958, No. 5.

5. Gomory, R.E., "An All Integer Programming Algorithm" in J.R. Muth and G.L. Thompson (ed), Industrial Scheduling, Chapter 13, Prentice-Hall, 1963.

6. Beale, E.M.L., "A Method of Solving Linear Programming Problems when some but not all the Variables must Take Integer Values", Statistical Techniques Research Group, Technical Report No. 19, Princeton University 1958.

7. Land, A.H., and Doig, A.G., "An Automatic Method for Solving Discreet Programming Problems", Econometrica, 28, 497-520, 1960.

8. Balas, E., "An Additive Algorithm for Solving Linear Programs with zero-one Variables", Operations Research, July-August 1965.

9. Ivanescu, P.L., Rosenberg, I., Rudeanu, S., "On the Determination of Minima of Pseudo-Boolean Functions". Studii Si Cercetari Mathematica, 14, 359-364, 1963, No. 3.

10. Ivanescu, P.L., Rosenberg, I., Rudeanu, S., "An Application of Discreet Linear Programming to the Minimization of Boolean Functions", Revue. Math. Pures et Appl., 8, 459-475, 1963, No. 3.

11. Granot, F., Hammer, P.L., On the Use of Boolean Functions in 0-1 Programming., Operations Research, Statistics and Economics Mimeograph Series. No. 70.

12. Yoshida, Y., Inagaki, Y., and Fukumura, T., "Algorithms of Pseudo-Boolean Programming Based on Branch and Bound Methods", Electronics and Communications in Japan, Vol. 50, October 1967, No. 10.

13. Hammer, P.L., and Rudeanu, S., Boolean Methods in Operations Research and Related areas, Springer-Verlag, Berlin, Heidelberg, New York, 1968.

14. Gaspar, T., "Programming the Algorithm for Minimization of Pseudo-Boolean Function for a MECIPT-1 Computer (in Rumanian)", Stud-Cere-Mat 19, 1135-1148, 1967.

15. Ivanescu, P.L., and Rudeanu, S., Pseudo-Boolean Methods for Bivalent Programming, Lecture Notes in Mathematics, Springer-Verlag, Berlin, Heidelberg, New York, Vol. 23, 1966.

16. Ivanescu, P.L., Pseudo-Boolean Programming and Applications, Lecture Notes in Mathematics, Springer-Verlag, Berlin, Heidelberg, New York, Vol. 9, 1965.

17. Lawler, E.L., Wood, D.E., "Branch and Bound Methods: A Survey", Operations Research, Vol. 14, 699-717, 1966.

18. Flood, M.M., "The Travelling Salesman Problem", Operations Research, 61-75, 1956.

19. Wagner, H.M., Principles of Operations Research with Applications to Managerial Decisions, Prentice-Hall, 1969.

20. Llewellyn, R.W., Linear Programming, Holt, Rinehart and Winston, New York, 1964.

21. Manne, A.S., "Plant Locations Under Economies of Scale; Decentralization and Computation", Management Science, 11, 213-235, 1964.

22. Siddall, J.N., Analytical Decision Making in Engineering Design, Prentice-Hall, 1972.

APPENDIX A

SOME PROPERTIES OF BOOLEAN ALGEBRA

Some Properties of Boolean Algebra

1. $x \cup y = y \cup x$
2. $x \,.\, y = y \,.\, x$
3. $(x \cup y) \cup z = x \cup (y \cup z)$
4. $(xy)z = x(yz)$
5. $x \cup x = x$
6. $x \,.\, x = x$
7. $x \cup xy = x$
8. $x(x \cup y) = x$
9. $x \cup yz = (x \cup y)(x \cup z)$
10. $x(y \cup z) = xy \cup xz$
11. $x \cup 1 = 1$
12. $x \,.\, 1 = x$
13. $x \cup 0 = x$
14. $x \,.\, 0 = 0$
15. $x \cup y = 0$ if and only if $x = y = 0$
16. $x \,.\, y = 1$ if and only if $x = y = 1$
17. $x \cup y = 1$ if and only if $x = 1$ or $y = 1$
18. $xy = 0$ if and only if $x = 0$ or $y = 0$
19. $x \cup \bar{x} = 1$
20. $x \,.\, \bar{x} = 0$
21. $\overline{x \cup y} = \bar{x} \,.\, \bar{y}$ the de Morgan laws
22. $\overline{x \,.\, y} = \bar{x} \cup \bar{y}$

The above properties are proved by direct verification for all possible values of x, y and z.

The following properties are also worth noting for every x, y, z $\varepsilon\{0,1\}$.

23. $x \leq y$ if and only if $x \cup y = y$

24. $x \leq y$ if and only if $xy = x$

25. $x \leq x \cup y$ and $y \leq x \cup y$

26. $x \,.\, y \leq x$ and $x \,.\, y \leq y$

27. If $x \leq z$ and $y \leq z$ then $x \cup y \leq z$

28. If $z \leq x$ and $z \leq y$ then $z \leq xy$

29. $x \leq y$ if and only if $\bar{x} \cup y = 1$

30. $x \leq y$ if and only if $x\bar{y} = 0$

31. $x = y$ if and only if $x\bar{y} \cup \bar{x}y = 0$

32. $x = y$ if and only if $(\bar{x} \cup y)(x \cup \bar{y}) = 1$

APPENDIX B

COMPUTER PROGRAMS AND USER'S MANUAL

HOW TO USE

In its simplest form, the calling program is written as follows:

a) DIMENSION Statement - check through the list of input, output and working variables. Include all subscripted variables, dimensioning as indicated.

b) Define logical variables LOGY,XY,XX,F.

c) Define input data in any manner desired.

d) Call subroutine BABO

e) Call subroutine BOUT to give printed output.

f) Add STOP and END.

If the optimization function, $U(x_1,x_2,\ldots,x_n)$ is nonlinear, it is defined in the user written subroutine UREAL.

```
SUBROUTINE BABO(A,AB,F,Y,X,XX,FF,YY,IA,C,M,N,COF,B,MAX,LOGY,CM,OBJ,IDATA,YC,
                IAV,XY,JC,XOB,DX,FM,SF,AY,D,NLIN,MTERM,NOBJ,ID,NFEAS,KA,NNON,IFN)
```

Purpose

To minimize $U=U(x_1,x_2,\dots,x_n)$

Subject to $\phi_j(x_1,x_2,\dots,x_n)\le b_j$

and all $x_i=0,\ 1$

The optimization function may have any form. The constraint functions may be nonlinear but must have the special form of a sum of terms, each of which contains only the simple product of any number of the variables. Variables must not appear in their complementary form, $\bar{x}_i$ in a linear constraint. Such forms may be removed by the transformation

$$\bar{x}_i = 1 - x_i$$

Method

The solution of the problem with linear constraints uses Boolean algebra to replace the set of constraints by a single resolvent function. The feasible solutions may then be conveniently found and scanned for the optimum solution.

If the constraints are nonlinear, the product of the variables in each term is replaced by a new set of variables so as to linearize the problem. This is illustrated in the following example:

$$-8x_4\, x_5\, \bar{x}_8 + 4\bar{x}_3\, \bar{x}_7\, x_8 - 3x_1\, x_2 - \bar{x}_3 - \bar{x}_4 - \bar{x}_5 \le -2$$

The transformation gives

$$-8y_1 + 4y_2 - 3y_3 - y_4 - y_5 - y_6 \le -2$$

REFERENCES

1. Hammer, P.L., "Boolean Methods in Operations Research", Springer-Verlag, Berlin, Heidelberg, New York, 1968.
2. Hammer, P.L., "A Boolean Approach to Bivalent Optimization", Centre de Recherches Matematiques, Universite de Montreal, June 1971.
3. Yoshida, Y., Inagaki, Y., and Fukumura, T., "Algorithms of Pseudo-Boolean Programming Based on Branch and Bound Methods", *Electronics and Communications in Japan*, Vol. 50, No. 10, October, 1967.

NOTE: This subroutine BABO is machine dependent in two respects. The first one is the use of CDC 6400 system routine SHIFT to pack and unpack a number of digits and the other one is the use of octal format to read in the variables of nonlinear constraints. For more information on these refer CDC 6400 FTN Reference Manual.

Input Variables

N	number of design or independent variables. Replace with the value of MTERM if there are nonlinear constraints and MTERM>N
M	number of constraints
NLIN	number of linear constraints
NNON	number of nonlinear constraints + 1
MAX	estimated number of terms in the resolvent = 100 for the first trial. A message will be printed out if MAX is too small
IAV	estimated number of feasible solutions = 50 for the first trial. A message will be printed out if IAV is too small

NOBJ	= 0 for linear objective function
	= 1 for nonlinear objective function
MTERM	maximum number of terms in any of the nonlinear constraints, = 1 for all linear constraints
C	upper limit on the linear optimization function. A suggested value for C would be half the sum of all positive coefficients in the optimization function
	= 0 for nonlinear objective function
IDATA	= 1 all input data printed out
	= 0 input data is not printed out
COF(I,J)	coefficient of the J th variable in the I th constraint for linear or linearized constraints, dimensioned with (M,N). See also Note (ii) below re dimensioning.
B(I)	right hand side constants of the constraints, dimensioned with (M)
OBJ(J)	coefficient of the J th variable in the optimization function, if it is linear dimensioned with (N+1), and with (1) if nonlinear
	(Note that if the optimization function is nonlinear, this array need not be defined, but Subroutine UREAL must be written)
	See also Note (ii) below re dimensioning.

OBJ(N+1)	value of the constant if it occurs in a linear optimization function = 0 if there is no constant (Note that this need not be defined for a nonlinear objective function) See also Note (ii) below re dimensioning
KA(I,J)	J th term in I th nonlinear constraint. Any variable present in the term is replaced by 1, its absence indicated by 0, and its negation by 2. Example: $x_1 \bar{x}_3 x_5 x_6$ (Assuming there are only 6 variables) will be replaced by 102011. This should be read in octal format dimensioned with (NNON, MTERM). (See the appendix for information on octal formats)

Note: i) In constraints which contain less than MTERM terms, the rest of the terms should be replaced by N zeroes each.

ii) If MTERM exceeds the value of N, the value of N should be replaced by MTERM in the input and dimension statements. The whole problem will be treated as a problem of MTERM variables.

Output Variables

AB(I,J)	array of feasible solutions. An element contains the value of the J th variable in the I th feasible solution, dimensioned with (MAX,N)
YY(I)	array of optimization function values corresponding to feasible solutions, dimensioned with (MAX)
NFEAS	total number of feasible solutions

Working Arrays

Variable	Dimension
IFN	(N,MAX)
SF	(N,N) if nonlinear constraints are present
	(N,1) if nonlinear constraints are absent
FM	(IAV,N)
YC	(M,N)
D	(M,N)
CM	(M,N)
IA	(MAX,N)
A	(MAX,N)
AY	(MTERM,N)
DX	(N)
F	(N)
Y	(N)
X	(N)
XX	(N)
FF	(N)
XY	(N)
XOB	(N)
LOGY	(MAX)
JC	(MAX)
ID	(MAX)

Logical Variables

Variables LOGY,XY,XX,F must be defined as logical variables in the calling program.

Programming Information

Subroutine BABO has full variable dimensioning except for several variables dimensioned with MAX and IAV. These depend on the number of terms in the resolvent which cannot be predicted in advance.

Subroutine BOUT may be used to print out the feasible solutions and the optimum solutions in standard form.

```
CALL BOUT(AB,YY,NFEAS,N,MAX)
```

The user may alternately write his own output logic. If the optimization function $U(x_1,x_2,\ldots,x_n)$ is nonlinear it must be defined in the user written subroutine UREAL. Subroutines called by BABO are CANON,COVER,SOLN,REDUCE, LSTR,STORE,BOOL,VIOLAT,OBJECT,DECODE,CHOOSE,ASEMBL and CHANGE.

SUBROUTINE UREAL(X,U)

Purpose

To calculate the value of the objective function at a point $U(x_1,x_2,.....,x_n)$ when the function is nonlinear,and where U=minimum at the optimum.

Method

The objective function may be defined by a simple arithmetic FORTRAN statement such as

```
U=6.*X(1)*(1.-X(2))-5.*(1.-X(3))*X(4)
```

Input Variables

X(I)	the current values of the independent variables

Output Variables

U	the value of the objective function corresponding to the input X(I) variables

How to Set Up Subroutine UREAL

The following cards must be punched by the user:

```
SUBROUTINE UREAL(X,U)
DIMENSION X(1)
U= arithmetic function
RETURN
END
```

A more complex analysis to define the value of U may require more complicated coding or additional subroutines.

Miscellaneous

If additional data is required to perform the analysis,the necessary READ ststements should be inserted in the MAIN program and the data transferred from MAIN to UREAL through labelled COMMON blocks.Where possible,the user should include conditional STOP's in his coding to prevent invalid results from being returned to the optimization procedure.

APPENDIX

Ow Input

Octal values are converted under O specifications.

Ow

w is an unassigned integer designating the total number of characters in the field. The input field may contain a maximum of 20 octal digits. Blanks are allowed and a plus or minus sign may precede the first octal digit. Blanks are interpreted as zeros and all blank field is interpreted as minus zero. A decimal point is not allowed.

The list item corresponding to the Ow specification should be an integer.

Example:

```
      READ (5,10) J,K
   10 FORMAT (O10,O2)
```

Input Card

$$\underbrace{3737373737}_{10}\underbrace{44}_{2}$$

Input Storage (octal representation):

```
      J  00000000003737373737
      K  00000000000000000044
      SUBROUTINE BABO (A,AB,F,Y,X,XX,FF,YY,IA,C,M,N,COF,B,MAX,LOGY,CM,OB
     1J,IDATA,YC,IAV,XY,JC,XOB,DX,FM,SF,AY,D,NLIN,MTERM,NOBJ,ID,NFEAS,KA
     2,NNON,IFN)
C
C
C     THIS SUBROUTINE USES BOOLEAN ALGEBRA TO REPLACE ALL THE CONSTRAINTS
C     BY A SINGLE RESOLVENT.
C     A SEARCH FOR THE OPTIMUM IS MADE BY BRANCH AND BOUND METHOD.
C
C
      DIMENSION YC(M,1), SF(N,1), AY(MTERM,1), DX(1), XOB(1), XY(1), JC(
     11), COF(M,1), B(1), OBJ(1), FF(1), A(MAX,1), AB(MAX,1), F(1), Y(1)
     2, X(1), XX(1), YY(1), CM(M,1), FM(IAV,1), IFN(N,1), D(M,1), LOGY(1
     3), ID(1)
      DIMENSION KA(NNON,1)
      DIMENSION IA(MAX,1)
      LOGICAL XY,XX,F,LOGY
      IF (IDATA.NE.1) GO TO 4
      WRITE (6,29)
      WRITE (6,30) N,M
      WRITE (6,31) NLIN,MAX,IAV,C
      WRITE (6,28) NOBJ,MTERM
      WRITE (6,32)
      DO 1 I=1,M
      WRITE (6,33) (COF(I,J),J=1,N)
1     CONTINUE
      WRITE (6,34)
      WRITE (6,35) (B(I),I=1,M)
      IF (NOBJ.EQ.1) GO TO 2
      WRITE (6,36) (OBJ(I),I=1,N)
      GO TO 3
2     CONTINUE
      WRITE (6,37)
3     CONTINUE
4     CONTINUE
      DO 5 KE=1,N
      DO 5 KW=1,MAX
      IFN(KE,KW)=00000000000000000000B
5     CONTINUE
C     SUBROUTINE CANON IS CALLED WHICH INTURN CALLS  SUBROUTINE SOLN TO
C     GET THE RESOLVENT OF THE SYSTEM OF CONSTRAINTS.
C
      CALL CANON (M,N,COF,B,CM,IA,D,YY,MAX,A,JC,IAV,YC,FM,AB,SF,AY,NLIN,
     1MTERM,KA,NNON,IFN)
C
C     SUBROUTINE BOOL IS CALLED TO SPLIT THE RESOLVENT INTO VARIOUS
C     LEVELS.
C
      CALL BOOL (N,IFN,KLM,A,AB,MAX,IAV,JC,ID,IA)
C
      DO 6 I=1,MAX
      YY(I)=0.
      DO 6 J=1,N
      AB(I,J)=0.
      A(I,J)=0.
      IA(I,J)=0
6     CONTINUE
7     CONTINUE
      K=0
      LCH=0
      J=1
C     X(1) IS GIVEN A VALUE OF 1 TO START WITH.
      X(J)=1.
      CALL VIOLAT (X,J,FF,N,KLM,F,LOGY,MAX,XX,XY,IFN,IA)
C     THE FIRST LEVEL OF THE RESOLVENT IS SATISFIED ONLY IF FF(J)=0.
      IF (FF(J).NE.0.) GO TO 9
      LCH=LCH+1
      IF (NOBJ.EQ.1) GO TO 8
      CALL OBJECT (X,J,Y,N,OBJ,XOB,DX)
C     ONLY THOSE VALUES OF X WHICH GIVE THE OBJECTIVE FUNCTION VALUE
C     LESS THAN OR EQUAL TO C ARE ACCEPTED AS PARTS OF FEASIBLE SOLUTION
      IF (Y(J).GT.C) GO TO 9
8     CONTINUE
      K=K+1
      A(K,J)=X(J)
9     X(J)=0.
      CALL VIOLAT (X,J,FF,N,KLM,F,LOGY,MAX,XX,XY,IFN,IA)
      IF (FF(J).NE.0.) GO TO 11
      LCH=LCH+1
      IF (NOBJ.EQ.1) GO TO 10
      CALL OBJECT (X,J,Y,N,OBJ,XOB,DX)
      IF (Y(J).GT.C) GO TO 12
10    CONTINUE
      K=K+1
      A(K,J)=X(J)
      GO TO 14
11    IF (K.EQ.0.AND.LCH.EQ.0) WRITE (6,38)
      IF (K.EQ.0.AND.LCH.EQ.0) CALL EXIT
C     IF NEITHER ZERO NOR ONE SUBSTITUTED FOR X(1) MAKES FIRST LEVEL
C     OF THE BOOLEAN CONSTRAINT FUNCTION ZERO THE WHOLE SYSTEM
C     IS INCONSISTANT AND NOFEASIBLE SOLUTION EXISTS.
      GO TO 14
12    IF (K.NE.0.AND.LCH.NE.0) GO TO 14
C     IF THE SOLUTION MATRIX IS EMPTY INCREASE THE UPPER LIMIT ON
C     OBJECTIVE FUNCTION VALUE.
13    C=C+0.1*C+5.0
      GO TO 7
14    CONTINUE
      L=1
      MN=J
      J=J+1
      JJ=0
15    DO 16 I=1,MN
      X(I)=A(L,I)
16    CONTINUE
      X(J)=1.
      CALL VIOLAT (X,J,FF,N,KLM,F,LOGY,MAX,XX,XY,IFN,IA)
      IF (FF(J).NE.0.) GO TO 20
      IF (NOBJ.EQ.1) GO TO 17
      CALL OBJECT (X,J,Y,N,OBJ,XOB,DX)
      IF (Y(J).GT.C) GO TO 20
      GO TO 18
17    CONTINUE
C     IF THE OBJECTIVE FUNCTION IS NONLINEAR  UREAL IS EVALUATED.
      IF (J.EQ.N) CALL UREAL (X,U)
      Y(J)=U
18    CONTINUE
```

```
      JJ=JJ+1
      IF (J.EQ.N) YY(JJ)=Y(J)
      DO 19 I=1,J
      AB(JJ,I)=X(I)
19    CONTINUE
C
20    X(J)=0.
      CALL VIOLAT (X,J,FF,N,KLM,F,LOGY,MAX,XX,XY,IFN,IA)
      IF (FF(J).NE.0.) GO TO 24
      IF (NOBJ.EQ.1) GO TO 21
      CALL OBJECT (X,J,Y,N,OBJ,XOB,DX)
      IF (Y(J).GT.C) GO TO 24
      GO TO 22
21    CONTINUE
      IF (J.EQ.N) CALL UREAL (X,U)
      Y(J)=U
22    CONTINUE
      JJ=JJ+1
      IF (J.EQ.N) YY(JJ)=Y(J)
      DO 23 I=1,J
      AB(JJ,I)=X(I)
23    CONTINUE
24    CONTINUE
      L=L+1
      IF (L.GT.K) GO TO 25
      GO TO 15
25    CONTINUE
      IF (JJ.EQ.0) GO TO 13
      K=JJ
      DO 26 MM=1,JJ
      DO 26 NN=1,J
      A(MM,NN)=AB(MM,NN)
26    CONTINUE
      IF (J.EQ.N) GO TO 27
      GO TO 14
27    CONTINUE
      NFEAS=JJ
      RETURN
C
C
28    FORMAT (/,5X,*THE VALUE OF NOBJ       ......................*,I5,//5
     1X,*THE VALUE OF MTERM      .....................*,I5)
29    FORMAT (//5X,*INPUT DATA FOR THE 0-1 PROBLEM*,/5X,*---------------
     1---------------*)
30    FORMAT (/,5X,*NUMBER OF DESIGN VARIABLES(N)...............*,I5,//,
     15X,*NUMBER OF CONSTRAINTS(M)     ................*,I5)
31    FORMAT (/,5X,*NUMBER OF LINEAR CONSTRAINTS(NLIN)..........*,I5,//,
     15X,*THE VALUE OF MAX          .....................*,I5,//,5X,*THE VA
     2LUE OF IAV         ......................*,I5,//,5X,*UPPER LIMIT ON O
     3BJECTIVE FUNCTION(C)........*,E9.2)
32    FORMAT (//,5X,*THE COEFFICIENT MATRIX FOR THE CONSTRAINT EQUATIONS
     1  ...COF(I,J)*)
33    FORMAT (//,5X,1P12E10.2)
34    FORMAT (//5X,* THE VALUES OF RIGHT HAND SIDE CONSTANTS...B(I)*)
35    FORMAT (//,5X,1P12E10.2)
36    FORMAT (//,5X,*THE OBJECTIVE FUNCTION COEFFICIENTS..OBJ(J)*,/,(//,
     15X,1P12E10.2))
37    FORMAT (//,5X,*THE OBJECTIVE FUNCTION COEFFICIENTS..*,//,5X,*NON-L
     1INEAR OBJECTIVE FUNCTION*)
38    FORMAT (////,10X,* NO FEASIBLE SOLUTION *)
      END
      SUBROUTINE VIOLAT (X,J,FF,N,KLM,F,Y,MAX,XX,XY,IFN,IA)
C
C     THIS SUBROUTINE CHECKS  WHETHER THE RESOLVENT IS SATISFIED FOR A
C     PARTICULAR SET OF VARIABLE VALUES.
C
      DIMENSION F(1), X(1), Y(1), FF(1), XX(1)
      DIMENSION XY(1), IFN(N,1)
      DIMENSION IA(MAX,1)
      LOGICAL Y,XY,XX,F
      K=KLM
      DO 1 KI=1,J
      IF (X(KI).EQ.1.) XX(KI)=.TRUE.
      IF (X(KI).EQ.0.) XX(KI)=.FALSE.
1     CONTINUE
C     INITIALISE ALL 'Y' FUNCTIONS AS TRUE SINCE WHEN THIS 'Y' GETS
C     INTO ANY INTERSECTION WITH ANY OTHER FUNCTION THE RESULTING
C     NATURE(EITHER TRUE OR FALSE) DEPENDS ONLY ON THE OTHER FUNCTION.
      DO 2 I=1,K
      Y(I)=.TRUE.
2     CONTINUE
C     INITIALISE ALL 'F'FUNCTIONS AS FALSE SINCE WHEN THIS 'F' GETS
C     INTO UNION WITH ANY OTHER FUNCTION THE REULTING NATURE
C     (EITHER TRUE OR FALSE)DEPENDS ON THE OTHER FUNCTION
      DO 3 I=1,J
      F(I)=.FALSE.
3     CONTINUE
      JJ=J
      DO 14 I=1,K
      CALL CHOOSE (IFN,JJ,N,MAX,I,IA)
      JKI=0
      DO 4 KJ=1,J
      IF (IA(I,KJ).NE.0) GO TO 5
4     CONTINUE
      GO TO 13
5     CONTINUE
      DO 9 IJ=1,J
      IF (IA(I,IJ).EQ.1) GO TO 7
C     -1 IN THE FUNCTION DENOTES A NEGATIONAND HENCE CHANGE.T. TO .F.
C     AND VICE VERSA.
      IF (IA(I,IJ).EQ.-1) GO TO 8
      IF (IA(I,IJ).EQ.0) GO TO 6
      Y(I)=.FALSE.
      GO TO 12
6     XY(IJ)=.TRUE.
      GO TO 9
7     XY(IJ)=XX(IJ)
      GO TO 9
8     XY(IJ)=.NOT.XX(IJ)
9     Y(I)=Y(I).AND.XY(IJ)
      IF (Y(I)) 10,11
10    CONTINUE
      F(J)=.TRUE.
      GO TO 15
11    CONTINUE
12    F(J)=F(J).OR.Y(I)
      GO TO 14
13    CONTINUE
      F(J)=.FALSE.
14    CONTINUE
```

```
15    CONTINUE
      IF (F(J)) 17,16
16    FF(J)=0.
      RETURN
17    FF(J)=1.
      RETURN
      END
      SUBROUTINE OBJECT (X,J,Y,N,OBJ,XOB,DX)
C
C     THIS SUBROUTINE DETERMINES THE LINEAR OBJECTIVE FUNCTION VALUE.
C
      DIMENSION DX(1), XOB(1), OBJ(1), X(1), Y(1)
      CONS=0.
      DO 1 I=1,N
      IF (OBJ(I).LT.0.) CONS=CONS+OBJ(I)
      XOB(I)=ABS(OBJ(I))
1     CONTINUE
      Y(J)=CONS
      DO 4 I=1,J
      IF (OBJ(I).LT.0.) GO TO 2
      DX(I)=X(I)
      GO TO 3
2     DX(I)=1.-X(I)
3     CONTINUE
      Y(J)=Y(J)+XOB(I)*DX(I)
4     CONTINUE
      Y(J)=Y(J)+OBJ(N+1)
      RETURN
      END
      SUBROUTINE BOOL (N,IFN,KLM,A,AA,MAX,IAV,C,D,IA)
C
C     THIS SUBROUTINE DETERMINES THE VARIOUS LEVELS OF THE RESOLVENT.
C     STARTING FROM THE 'N' TH VARIBLE  ELIMINATIONS ARE MADE
C     SUCCESSIVELY TO ONE VARIABLE.
C
      DIMENSION AA(MAX,1), A(MAX,1), C(1), D(1), IFN(N,1), IA(MAX,1)
      INTEGER C,D
      COMMON /BOSO/ L
      K=L
      KLM=K
      JJ=N
      CALL ASEMBL (A,IA,IFN,N,MAX,K,JJ)
      GO TO 2
1     CONTINUE
      CALL ASEMBL (A,IA,IFN,N,MAX,K,JJ)
2     CONTINUE
      IF (JJ.EQ.1) GO TO 38
      KK=0
      KL=0
      DO 5 I=1,K
      IF (A(I,JJ)) 3,5,4
3     KL=KL+1
      D(KL)=I
      GO TO 5
4     KK=KK+1
      C(KK)=I
5     CONTINUE
      IF (KK.EQ.K.OR.KL.EQ.K) GO TO 39
      IF (KK.EQ.0.AND.KL.EQ.0) GO TO 17
      IF (KK.EQ.0.OR.KL.EQ.0) GO TO 13
      I=0
      DO 9 L=1,KK
      JL=C(L)
      DO 8 LL=1,KL
      JLL=D(LL)
      I=I+1
      KU=JJ-1
      DO 6 J=1,KU
      AA(I,J)=A(JL,J)+A(JLL,J)
      IF (AA(I,J).EQ.0..AND.A(JL,J).NE.0.) GO TO 7
      IF (AA(I,J).EQ.2.) AA(I,J)=1.
      IF (AA(I,J).EQ.-2.) AA(I,J)=-1.
6     CONTINUE
      GO TO 8
7     CONTINUE
      IF (K.EQ.2) GO TO 10
      I=I-1
8     CONTINUE
9     CONTINUE
      GO TO 12
10    CONTINUE
      DO 11 J=1,JJ
      AA(I,J)=0.
11    CONTINUE
12    CONTINUE
      MN=I
      GO TO 23
13    IF (KK.EQ.0) GO TO 19
```

```
      MN=0
      DO 16 J=1,K
      DO 14 I=1,KK
      IF (J.EQ.C(I)) GO TO 16
14    CONTINUE
      MN=MN+1
      DO 15 IJ=1,JJ
      AA(MN,IJ)=A(J,IJ)
15    CONTINUE
16    CONTINUE
      GO TO 28
17    CONTINUE
      DO 18 I=1,K
      DO 18 J=1,JJ
18    AA(I,J)=A(I,J)
      GO TO 28
19    MN=0
      DO 22 J=1,K
      DO 20 I=1,KL
      IF (J.EQ.D(I)) GO TO 22
20    CONTINUE
      MN=MN+1
      DO 21 IJ=1,JJ
      AA(MN,IJ)=A(J,IJ)
21    CONTINUE
22    CONTINUE
      GO TO 28
23    CONTINUE
      DO 27 I=1,K
      DO 24 J=1,KL
      IF (I.EQ.D(J)) GO TO 27
24    CONTINUE
      DO 25 J=1,KK
      IF (I.EQ.C(J)) GO TO 27
25    CONTINUE
      MN=MN+1
      DO 26 J=1,JJ
      AA(MN,J)=A(I,J)
26    CONTINUE
27    CONTINUE
28    CONTINUE
      KU=JJ-1
      DO 32 LJ=1,MN
      KJR=LJ+1
      IF (KJR.GT.MN) GO TO 32
      DO 31 I=KJR,MN
      DO 29 J=1,KU
      IF (AA(LJ,J).EQ.AA(I,J)) GO TO 29
      GO TO 31
29    CONTINUE
      DO 30 J=1,JJ
      AA(I,J)=0.
30    CONTINUE
31    CONTINUE
32    CONTINUE
      KFJ=0
      DO 36 I=1,MN
      DO 33 J=1,JJ
      IF (AA(I,J).EQ.0.) GO TO 33
      GO TO 34
33    CONTINUE
      GO TO 36
34    CONTINUE
      KFJ=KFJ+1
      DO 35 J=1,JJ
      AA(KFJ,J)=AA(I,J)
35    CONTINUE
36    CONTINUE
      MN=KFJ
      DO 37 L=1,MN
      DO 37 J=1,JJ
      A(L,J)=AA(L,J)
37    CONTINUE
      JJ=JJ-1
      K=MN
      IF(K.GT.MAX)GOTO 41
      CALL REDUCE (A,MAX,JJ,K,AA)
      GO TO 1
38    CONTINUE
      RETURN
39    CONTINUE
      KU=JJ-1
40    CONTINUE
      IFN(KU,1)=0
      KU=KU-1
      IF (KU.EQ.0) RETURN
      GO TO 40
41    CONTINUE
      WRITE(6,42)
42    FORMAT(//,5X,*INCREASE THE VALUE OF MAX*)
      CALL EXIT
      END
      SUBROUTINE CHOOSE (IFN,JJ,N,MAX,I,IA)
      DIMENSION IFN(N,1), IA(MAX,1)
C
C     THIS SUBROUTINE  CHOOSES A PATICULAR DIGIT  OUT OF AN INTEGER
C     CONSTANT.
C
      KG=0
      DO 1 J=1,JJ
      IT=IFN(JJ,I)
      IT=SHIFT(IT,KG).AND.3B
      IF (IT.EQ.2) IT=-1
      L=JJ-J+1
      IA(I,L)=IT
      KG=KG-2
1     CONTINUE
      RETURN
      END
      SUBROUTINE CANON (M,N,A,B,CM,IA,D,CMAX,MAX,CO,JAN,IAV,C,FM,AB,SF,Y
     1,NLIN,MTERM,KA,NNON,IFN)
C
C     THIS SUBROUTINE REARRANGES THE  COEFFICIENTS OF THE VARIABLES IN
C     ORDER OF DECREASING MAGNITUDE.
C
      DIMENSION AB(MAX,1), SF(N,1), Y(MTERM,1), CM(M,1), B(1), IFN(N,1),
     1 A(M,1), D(M,1), CMAX(1), FM(IAV,1), CO(MAX,1), C(M,1), JAN(1)
      DIMENSION IA(MAX,1)
      DIMENSION KA(NNON,1)
      COMMON /BOSO/ L
      DO 1 I=1,M
      DO 1 J=1,N
      IF (A(I,J).LT.0.) B(I)=B(I)+ABS(A(I,J))
      C(I,J)=ABS(A(I,J))
      D(I,J)=C(I,J)
1     CONTINUE
      DO 7 L=1,N
      DO 3 I=1,M
      CMAX(I)=C(I,1)
      DO 2 J=1,N
      IF (C(I,J).GT.CMAX(I)) CMAX(I)=C(I,J)
2     CONTINUE
3     CONTINUE
      DO 5 I=1,M
      DO 4 J=1,N
      IF (C(I,J).NE.CMAX(I)) GO TO 4
      C(I,J)=-C(I,J)
      GO TO 5
4     CONTINUE
5     CONTINUE
      DO 6 I=1,M
      CM(I,L)=CMAX(I)
6     CONTINUE
7     CONTINUE
      NNN=0
      IB=0
      DO 8 KL=1,M
      CALL COVER (M,N,A,B,D,CM,MAX,IA,CO,JAN,IAV,C,FM,KL,AB,IB,SF,Y,NLIN
     1,MTERM,NNN,KA,NNON,IFN)
8     CONTINUE
      DO 9 I=1,MAX
      DO 9 J=1,N
      CO(I,J)=0.
9     CONTINUE
      MN=0
      DO 13 I=1,IB
      DO 10 J=1,N
      IF (AB(I,J).NE.0.) GO TO 11
10    CONTINUE
      GO TO 13
11    MN=MN+1
      DO 12 J=1,N
      CO(MN,J)=AB(I,J)
12    CONTINUE
13    CONTINUE
      L=MN
      CALL REDUCE (CO,MAX,N,L,AB)
      RETURN
      END
      SUBROUTINE COVER (M,N,A,B,D,CM,MAX,IA,CO,JAN,IAV,C,FM,KL,AB,IB,SF,
     1Y,NLIN,MTERM,NNN,KA,NNON,IFN)
C
C     THIS SUBROUTINE FINDS ALL THE MINIMAL COVERS OF THE CONSTRAINTS.
C
      DIMENSION SF(N,1), Y(MTERM,1), FM(IAV,1), CO(MAX,1), C(M,1), JAN(1
     1), CM(M,1), B(1), IFN(N,1), A(M,1), D(M,1), AB(MAX,1), IA(MAX,1)
      DIMENSION KA(NNON,1)
      IJ=1
      DO 1 LL=1,MAX
      DO 1 ML=1,N
      IA(LL,ML)=0
1     CONTINUE
      L=0
2     L=L+1
      IF (L.GT.N) GO TO 21
      DO 11 KMN=1,N
      KKK=KMN
      MO=1
      DO 3 IN=1,N
      JAN(IN)=0
3     CONTINUE
      NAM=1
      IF (L.GT.N) GO TO 12
      I=L
      SUM=0.
      IF (CM(KL,I).GT.B(KL)) GO TO 9
4     SUM=SUM+CM(KL,I)
      J=I+KKK
      KKK=1
      IF (J.GT.N) GO TO 10
5     TEST=SUM+CM(KL,J)
      IF (TEST.GT.B(KL)) GO TO 6
      I=J
      JAN(NAM)=I
      NAM=NAM+1
      IF (I.EQ.N) GO TO 11
      GO TO 4
6     CONTINUE
      IA(IJ,L)=1
      IA(IJ,J)=1
      MAT=L+1
      DO 8 K=MAT,I
      DO 7 MAN=1,NAM
      IF (K.EQ.JAN(MAN)) IA(IJ,K)=1
7     CONTINUE
8     CONTINUE
      IJ=IJ+1
      J=J+1
      IF (J.GT.N) GO TO 10
      GO TO 5
9     IA(IJ,I)=1
      IJ=IJ+1
      GO TO 2
10    I=I+1
      IF (I.GT.N) GO TO 12
11    CONTINUE
      GO TO 2
12    CONTINUE
```

```
      L=0
13    CONTINUE
      KKV=2
      L=L+1
      IF (L.EQ.N) GO TO 21
      I=L+1
14    CONTINUE
      SUM=0.
      NAM=0
      SUM=CM(KL,L)+CM(KL,I)
      IF (SUM.GT.B(KL)) GO TO 21
      NI=I+KKV
      IF (NI.GT.N) GO TO 13
      DO 15 IN=1,N
      JAN(IN)=0
15    CONTINUE
      J=NI
16    CONTINUE
      TEST=SUM+CM(KL,J)
      NAM=NAM+1
      JAN(NAM)=J
      IF (TEST.GT.B(KL)) GO TO 17
      SUM=TEST
      J=J+1
      IF (J.GT.N) GO TO 20
      GO TO 16
17    CONTINUE
      IA(IJ,L)=1
      IA(IJ,L+1)=1
      MAT=L+2
      DO 19 K=MAT,J
      DO 18 MAN=1,NAM
      IF (K.EQ.JAN(MAN)) IA(IJ,K)=1
18    CONTINUE
19    CONTINUE
      IJ=IJ+1
      JAN(NAM)=0
      J=J+1
      IF (J.GT.N) GO TO 20
      GO TO 16
20    CONTINUE
      KKV=KKV+1
      GO TO 14
21    CONTINUE
      NJ=2
      NO=NJ-1
22    CONTINUE
      DO 25 NN=1,NO
      DO 23 J=1,N
      IF (IA(NJ,J).EQ.IA(NN,J)) GO TO 23
      GO TO 25
23    CONTINUE
      DO 24 J=1,N
      IA(NJ,J)=0
24    CONTINUE
25    CONTINUE
      NO=NJ
      NJ=NJ+1
      IF (NJ.GT.IJ) GO TO 26
      GO TO 22
 26   CONTINUE
      CALL SOLN (M,N,IA,A,D,CM,CO,C,IAV,MAX,FM,KL,AB,IB,SF,Y,NLIN,MTERM,
     1NNN,KA,NNON,IFN)
      RETURN
      END
      SUBROUTINE SOLN (M,N,IA,A,D,CM,CO,C,IAV,MAX,FM,KL,AB,IB,SF,Y,NLIN,
     1MTERM,NNN,KA,NNON,IFN)
C
C     THIS SUBROUTINE CONSTRUCTS THE RESOLVENT OUT OF THE MINIMAL COVERS
C
      DIMENSION SF(N,1), Y(MTERM,1), AB(MAX,1), FM(IAV,1), CO(MAX,1), C(
     1M,1), IFN(N,1), A(M,1), D(M,1), CM(M,1), IA(MAX,1)
      DIMENSION KA(NNON,1)
      JA=1
      I=KL
      L=1
1     CONTINUE
      K=1
2     CONTINUE
      IF (D(I,L).EQ.CM(I,K)) GO TO 3
      K=K+1
      GO TO 2
3     IF (CM(I,K).EQ.A(I,L)) GO TO 4
      CM(I,K)=-CM(I,K)
      GO TO 6
4     CONTINUE
      CM(I,K)=-CM(I,K)
      DO 5 J=1,IAV
      FM(J,L)=FLOAT(IA(J,K))
5     CONTINUE
      GO TO 8
6     CONTINUE
      DO 7 J=1,IAV
      FM(J,L)=-FLOAT(IA(J,K))
7     CONTINUE
8     CONTINUE
      L=L+1
      IF (L.GT.N) GO TO 9
      GO TO 1
9     CONTINUE
      L=1
      DO 13 KJ=1,IAV
      DO 10 KM=1,N
      IF (FM(KJ,KM).EQ.0.) GO TO 10
      GO TO 11
10    CONTINUE
      GO TO 13
11    CONTINUE
      DO 12 J=1,N
      CO(L,J)=FM(KJ,J)
12    CONTINUE
      L=L+1
13    CONTINUE
      IF (I.GT.NLIN) GO TO 14
      CALL LSTR (CO,AB,IB,L,MAX,N)
      RETURN
14    CONTINUE
      CALL DECODE (CO,MAX,N,L,AB,IA,IB,FM,SF,Y,IAV,MTERM,NNN,KA,NNON,IFN
     1)
      RETURN
C
      END
      SUBROUTINE ASEMBL (A,IA,IFN,N,MAX,KY,JJ)
      DIMENSION A(MAX,1), IA(MAX,1), IFN(N,1)
C
C     THIS SUBROUTINE ASSEMBLES  A NUMBER OF SINGLE DIGIT  INTEGERS INTO
C     A SINGLE INTEGER CONSTANT.
C
      DO 2 I=1,KY
      K=0
      IT=77777777777777777777B
      KT=77777777777777777770B
      DO 1 J=1,JJ
      IF (A(I,J).EQ.1.) IA(I,J)=1B
      IF (A(I,J).EQ.0.) IA(I,J)=0B
      IF (A(I,J).EQ.-1.) IA(I,J)=2B
      MT=SHIFT(IT,K).OR.3B
      NT=SHIFT(KT,0).OR.IA(I,J)
      NT=NT.OR.4B
      IT=MT.AND.NT
      K=2
1     CONTINUE
      IFN(JJ,I)=IT
2     CONTINUE
      RETURN
      END
      SUBROUTINE REDUCE (CO,MAX,N,L,BA)
C
C     THIS SUBROUTINE SIMPLIFIES THE RESOLVENT.
C
      DIMENSION CO(MAX,1), BA(MAX,1)
      KJ=0
      DO 1 I=1,L
      DO 1 J=1,N
      BA(I,J)=CO(I,J)
1     CONTINUE
      K=L
      DO 9 I=1,K
      DO 2 J=1,N
      IF (BA(I,J).NE.0.) GO TO 3
2     CONTINUE
      GO TO 9
3     CONTINUE
      IF (I.EQ.K) GO TO 7
      JS=I+1
      DO 5 II=JS,K
      DO 4 J=1,N
      IF (BA(I,J).EQ.BA(II,J)) GO TO 4
      IF (BA(I,J).EQ.1..AND.BA(II,J).EQ.0.) GO TO 4
      IF (BA(I,J).EQ.-1..AND.BA(II,J).EQ.0.) GO TO 4
      GO TO 5
4     CONTINUE
      GO TO 9
5     CONTINUE
      KJ=KJ+1
      DO 6 J=1,N
      CO(KJ,J)=BA(I,J)
6     CONTINUE
      GO TO 9
7     CONTINUE
      KJ=KJ+1
      DO 8 J=1,N
      CO(KJ,J)=BA(I,J)
8     CONTINUE
9     CONTINUE
      L=KJ
      RETURN
      END
      SUBROUTINE DECODE (CO,MAX,N,L,AB,IA,IB,TN,SF,Y,IAV,MTERM,NNN,KA,NN
     1ON,IFN)
C
C     THIS SUBROUTINE RESUBSTITUTES THE NONLINEAR TERMS FOR THEIR
C     LINEAR SUBSTITUTES.
C
      DIMENSION CO(MAX,1), TN(IAV,1), SF(N,1), IFN(N,1), Y(MTERM,1), AB(
     1MAX,1), IA(MAX,1)
      DIMENSION KA(NNON,1)
      L=L-1
      NJ=N
      DO 1 J=1,MAX
      DO 1 K=1,N
      IA(J,K)=0
1     CONTINUE
      CALL CHANGE (N,MTERM,Y,NNN,KA,NNON)
      KI=0
      DO 27 I=1,L
      NJ=N
      KOUNT=0
      DO 2 JB=1,IAV
      DO 2 JC=1,N
      TN(JB,JC)=0.
2     CONTINUE
      KN=0
      DO 26 J=1,MTERM
      DO 3 JB=1,N
      DO 3 JC=1,N
      SF(JB,JC)=0.
3     CONTINUE
      IF (CO(I,J).EQ.-1.) GO TO 5
      IF (CO(I,J).EQ.0.) GO TO 26
      KOUNT=KOUNT+1
      DO 4 JL=1,N
      SF(J,JL)=Y(J,JL)
4     CONTINUE
      KH=1
      GO TO 7
5     CONTINUE
      KOUNT=KOUNT+1
      DO 6 JL=1,N
      SF(JL,JL)=-Y(J,JL)
6     CONTINUE
      KH=N
7     CONTINUE
      IF (KOUNT.EQ.1) GO TO 21
      IF (KH.EQ.N) GO TO 8
      JQ=J
      JR=J
      GO TO 9
8     JR=N
      JQ=1
9     CONTINUE
      DO 19 JH=JQ,JR
      DO 10 LA=1,N
      IF (SF(JH,LA).NE.0.) GO TO 11
10    CONTINUE
      GO TO 18
11    CONTINUE
```

```
      IF (KOUNT.EQ.2.OR.KOUNT.EQ.3) IS=1
      DO 17 JN=IS,NJ
      DO 12 LA=1,N
      IF (IA(JN,LA).NE.0) GO TO 13
12    CONTINUE
      GO TO 17
13    CONTINUE
      KN=KN+1
      DO 14 JK=1,N
      TN(KN,JK)=SF(JH,JK)+FLOAT(IA(JN,JK))
      IF (TN(KN,JK).EQ.-2.) TN(KN,JK)=-1.
      IF (TN(KN,JK).EQ.2.) TN(KN,JK)=1.
      IF (TN(KN,JK).EQ.0..AND.SF(JH,JK).NE.0.) GO TO 15
14    CONTINUE
      GO TO 17
15    CONTINUE
      DO 16 JK=1,N
      TN(KN,JK)=0.
16    CONTINUE
      IF (J.EQ.MTERM) GO TO 17
      KN=KN-1
17    CONTINUE
18    CONTINUE
      IF (KH.NE.N) GO TO 20
19    CONTINUE
20    CONTINUE
      IS=NJ+1
      NJ=KN
      GO TO 23
21    CONTINUE
      DO 22 JN=1,N
      DO 22 JK=1,N
      IA(JN,JK)=SF(JN,JK)
22    CONTINUE
      GO TO 26
23    CONTINUE
      IF (KOUNT.EQ.2) IS=1
      DO 25 MK=IS,KN
      DO 24 ML=1,N
      IA(MK,ML)=TN(MK,ML)
24    CONTINUE
25    CONTINUE
26    CONTINUE
      CALL STORE (IA,IS,IB,I,MAX,KN,AB,N,IFN)
27    CONTINUE
      L=IB+1
      RETURN
C
C
      END
      SUBROUTINE STORE (IA,IS,IB,I,MAX,NJ,AB,N,IFN)
C
C     THIS SUBROUTINE STORES THE TERMS OF THE RESOLVENT FORMED OUT OF
C     NONLINEAR CONSTRAINTS.
C
      DIMENSION IA(MAX,1), IFN(N,1), AB(MAX,1)
      DO 2 J=IS,NJ
      IB=IB+1
      IF (IB.GT.MAX) GO TO 3
      DO 1 K=1,N
      AB(IB,K)=IA(J,K)
1     CONTINUE
2     CONTINUE
      GO TO 4
3     CONTINUE
      WRITE (6,5)
      CALL EXIT
4     CONTINUE
      RETURN
C
C
5     FORMAT (1H1,5X,*NUMBER OF SOLUTIONS EXCEED DIMENSION*,//,5X,*INCRE
     1ASE THE VALUE OF  MAX *)
      END
      SUBROUTINE LSTR (CO,AB,IB,L,MAX,N)
C
C     THIS SUBROUTINE STORES THE TERMS OF THE RESOLVENT FORMED OUT OF
C     LINEAR CONSTRAINTS.
C
      DIMENSION CO(MAX,1), AB(MAX,1)
      L=L-1
      DO 2 J=1,L
      IB=IB+1
      IF (IB.GT.MAX) GO TO 3
      DO 1 K=1,N
      AB(IB,K)=CO(J,K)
1     CONTINUE
2     CONTINUE
      GO TO 4
3     CONTINUE
      WRITE (6,5)
      CALL EXIT
4     CONTINUE
      RETURN
C
C
5     FORMAT (1H1,5X,*NUMBER OF SOLUTIONS EXCEED DIMENSION*,//,5X,*INCRE
     1ASE THE VALUE OF  MAX *)
      END
      SUBROUTINE BOUT (AB,YY,NFEAS,N,MAX)
C
C     THIS SUBROUTINE PRINTS OUT THE FEASIBLE AND OPTIMUM SOLUTIONS.
C
      DIMENSION AB(MAX,1), YY(1)
      WRITE (6,4)
      IF (N.GT.15) K=15
      IF (N.LE.15) K=N
      WRITE (6,5) (I,I=1,K)
      DO 1 MM=1,NFEAS
      WRITE (6,6) YY(MM),(AB(MM,NN),NN=1,N)
1     CONTINUE
      YO=YY(1)
      DO 2 I=1,NFEAS
      IF (YY(I).GT.YO) GO TO 2
      YO=YY(I)
2     CONTINUE
      WRITE (6,7)
      IF(N.GT.15) K=15
      IF(N.LE.15)K=N
      WRITE (6,5) (I,I=1,K)
      DO 3 I=1,NFEAS
      IF (YY(I).NE.YO) GO TO 3
      WRITE (6,6) YY(I),(AB(I,NN),NN=1,N)
3     CONTINUE
      RETURN
C
4     FORMAT (1H1,//,10X,*FEASIBLE SOLUTIONS*,/10X,*------------------*)
5     FORMAT (//,8X,*U*,8X,15(*X(*,I2,*)*,3X))
6     FORMAT (//,2X,  E13.5,15(F5.0,3X ),/, 15X,15(F5.0,3X))
7     FORMAT (//,10X,*THE OPTIMUM SOLUTION(S)*,/10X,*-------------------
     1----*)
      END
      SUBROUTINE CHANGE (N,MTERM,Y,NNN,KA,NNON)
      DIMENSION KA(NNON,1)
      DIMENSION Y(MTERM,1)
C
C     THIS SUBROUTINE DETERMINES THE NATURE AND THE NUMBER OF VARIABLE
C     PRESENT IN A PATICULAR NONLINEAR TERM.
C
      NNN=NNN+1
      DO 2 I=1,MTERM
      K=0
      DO 1 J=1,N
      IT=KA(NNN,I)
      IT=SHIFT(IT,K).AND.7B
      IF (IT.EQ.2) IT=-1
      L=N-J+1
      Y(I,L)=FLOAT(IT)
      K=K-3
1     CONTINUE
      RETURN
2     CONTINUE
      END
```