ON PSEUDO-BOOLEAN PROGRAMMING AND GENERALIZATIONS: APPLYING MATHEMATICAL LOGIC TO 0-1 INTEGER PROGRAMMING

Leopold A. ZIEGLER

Research Memorandum No.114

July 1976

ON PSEUDO-BOOLEAN PROGRAMMING AND GENERALIZATIONS: APPLYING MATHEMATICAL LOGIC TO 0-1 INTEGER PROGRAMMING.

by

Leopold A.Ziegler

July 1976

PREFACE

Mathematical Programming deals with the optimization of a given function under constraints - usually in the form of inequalities - and provides not only a theoretical framework for a large class of optimization problems but also a basis for the development of useful techniques which can be used for formulation purposes, model construction and thederivation of solution procedures.

It is the aim of this paper to discuss the general background of integer programming as a part of mathematical programming and some aspects of applying mathematical logic to integer programming problems, particularly in 0-1 variables.

This paper is a revised and modified version of the author's MSc-Dissertation "On the Pseudo-Boolean Approach and 0-1 Integer Programming", University of Sussex, October 1975.

ACKNOWLEDGEMENTS

I would like to thank my supervisor at the University of Sussex, Professor Steven Vajda for his kind advice and numerous suggestions, and the interest he has taken in this work. I also thank Professor P.L.Hammer and Dr.H.P. Williams for the valuable information made available to me.

Research into the subject had been started while the author held a scholarship awarded by the Institute for Advanced Studies Vienna which is gratefully acknowledged.

I am also grateful to the British Council for the grant under which this work has been completed.

CONTENTS

CHAPTER 1

Introduction

§ 1.1 Preliminaries

Mathematical Programming which is concerned with a wide variety of constrained optimization problems plays an important and distinctive role among operational research approaches.

Both practical requirements and theoretical interests have been stimulating research into this subject and as a consequence a theoretical framework as well as efficient solution methods have been developed for numerous classes of mathematical programming problems. The case where some or all of the variables involved are required to be integer valued is dealt with in the context of integer programming.

In the first part of this paper the general back - ground of integer programming will be reviewed and a classification of different types of problems given. After a characterization of the main approaches some aspects of applying mathematical logic to integer programming problems will be considered. The Pseudo-Boolean Approach which makes use of results and concepts of Boolean algebra will be the subject of a more detailed discussion, in particular the concept of the resolvent. As the result of analysing the general assumptions of that approach an axiomatic

basis is obtained which allows the development of Pseudo-Boolean programming in a generalized way.

§ 1.2 Mathematical Programming

Mathematical programming deals with a particular class of constrained optimization problems according to the following definition:

minimize $f(x)$ for vectors $x \in \mathbb{R}^n$ satisfying the conditions $g_j(x) \geq b_j \quad j=1,2,\ldots,m$ where $f(x)$ and $g_j(x)$ are functions from $\mathbb{R}^n \to \mathbb{R}$, $b_j \in \mathbb{R}$.

$f(x)$ is called the objective function, $g_j(x) \geq b_j$ (inequality) constraints. Sometimes the additional requirement $x \geq 0$ i.e. the nonnegativity of x is explicitly made. The case of maximizing $f(x)$ can be reduced to the case mentioned by using the relation $\max f(x) = - \min(-f(x))$.

The set of vectors satisfying the constraints is called the feasible region, the vectors belonging to that set feasible solutions. If that set is nonempty the problem is called feasible, otherwise infeasible. A feasible solution which minimizes (maximizes) the objective function is called an optimal solution.

By imposing further conditions on the objective function and the constraints different classes of mathematical programming problems can be obtained,

e.g. convex programs if both the objective function and the feasible region are convex. Convexity of a set, say C, means that for $x,y \in C$ also $x+(1-\lambda)y \in C$ $0 \leq \lambda \leq 1$, whereas a function $f(x)$ is called convex on a convex set $D \neq \emptyset$, if it follows from $u,v \in D$ that $f(\mu u+(1-\mu)v) \leq \mu f(u)+(1-\mu)f(v)$, where $0 \leq \mu \leq 1$. Fundamental results and topics of the theory of mathematical programming are the subject of e.g. Vajda [28], [29] ; applications are dealt with in Vajda [30].

Since from the point of view of applying mathematical programming in practice the availability of efficient solution procedures is most important, research has alsobeen directed towards the development of algorithms, i.e. computational procedures which evaluate an optimal solution to the given problem or indicate its infeasibility.

The implementation of such algorithms requires additional considerations concerning the handling of the actual data, speed and accuracy of calculations, and a suitable representation of the results (see e.g. Beale [3], Land & Powell [21]).

§ 1.3 Linear Programming

If in the general definition of mathematical programming $f(x)$ and $g_j(x)$ $j= 1,2,\ldots,m$ are required to be linear in x we obtain the case of linear programming which can also be represented in the following form:

$$\min c'x \quad \text{subject to } Ax \geq b,\ x \geq 0$$

with $c, x \in \mathbb{R}^n$, A an (mxn) matrix, $b \in \mathbb{R}^m$.

An optimal solution to a linear program (LP) exists if the feasible region is nonempty and if the objective function has a lower bound for all feasible solutions. A feasible solution is called a basic feasible solution if m components of the solution vector are nonnegative, n-m components zero and the square matrix of coefficients corresponding to the m nonnegative components is non-singular.

The set of basic feasible solutions is a finite set because after introducing m additional variables - the slack variables - there are at most $\binom{m+n}{m}$ ways to select a non - singular (mxm) matrix among the m+n columns of [A,-I] with I the (mxm) identity matrix. If an optimal solution exists, then there exists also a basic optimal solution, we can therefore restrict our attention to the set of basic feasible solutions if we want to solve

a given linear programming problem.
An efficient way to do this is the well known Simplex algorithm (see e.g.Dantzig [6]) for instance in the form of the Revised Simplex algorithm which is most suitable for the use on electronic computers(see e.g. Orchard-Hays [23]). Without any doubt G.B. Dantzig's Simplex method has been a fundamental contribution towards the widespread use of mathematical programming techniques and has been stimulating further theoretical investigations. Many practical problems such as some production problems, network flow problems, optimal mix problems, transport problems, the diet problem etc. turn out to be suitable for modelling by linear programs. The applicability of linear programming to economic problems is no coincidence , the reason being that LP-models - at least under certain conditions- can successfully be interpreted in terms of economic concepts:
Assuming the total divisibility of all the variables involved and constant return to scale, the LP , say max $p'x$ subject to $Ax \leq b$ and $x \geq 0$, can represent the problem of finding the output level x^* which maximizes the profit of the firm taking into account the given restrictions on the inputs

(e.g. capacities like machine hours, etc.);
x is usually called the vector of the structural variables.

Beyond that rather obvious interpretation LP concepts like duality and decomposition make a much more subtle economic analysis possible.

For example let us consider the problem of profit maximization mentioned above. If- after introducing slack variables and deriving an optimal solution by means of the Simplex algorithm- a slack variable has a nonzero value in an optimal basic solution, this indicates that the corresponding resource has not been fully used, while slack variables at the level of zero imply the full use of the corresponding resource. If we look at their coefficients (-the reduced costs-) in the final representation of the objective function, obtained at the last stage when applying the Simplex algorithm to the problem, they can be interpreted as internal accounting or shadow prices of the corresponding resource. In the former case this price is zero since the resource is not fully used. For a capacity restraint this means that the value of increasing the capacity is zero. If we consider the reduced costs on non-basic structural variables in the final representation of the objective function, we can use

them for pricing decisions. Without going into further details it should be mentioned that linear programming allows the interpretation of economic concepts like decentralized planning, input - output analysis etc. as well.(see e.g. Hadley [13]).

On the other hand if, say, the variable x is to represent the amount of a certain good which can only be obtained in indivisible units, limitations of that approach become apparent and the LP solution will in general not be a satisfactory solution to the given problem.

Mathematically the additional condition imposed by requiring indivisible units can be represented by introducing integer valued variables.

This is of course not the only reason for considering integer valued variables: certain nonlinearities, logical conditions, or combinatorial problems suggest a similar formulation.

The significance of such problems has been pointed out in several instances (see e.g. Dantzig [5]) and they are dealt with in the context of integer programming.

CHAPTER 2

Integer Programming

§ 2.1 Introduction

In this chapter we shall highlight some problems which arise when on a given mathematical program - ming problem the additional requirement is imposed that some or all variables are to take on integer values only.

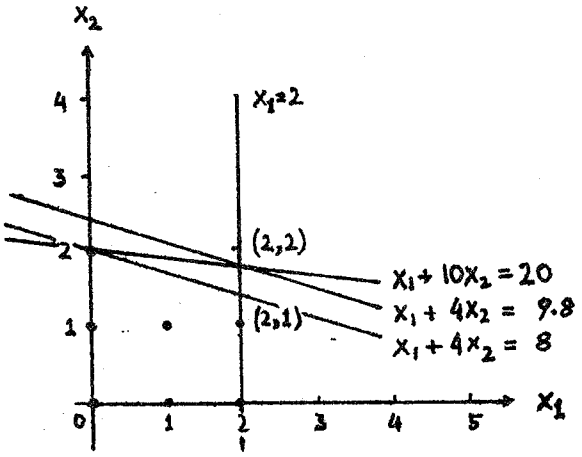
Usually the attention is restricted to the linear case i.e. to a LP-problem together with the integrality condition. The need for further theoretical investigations and the development of special integer programming algorithms can easily be demonstrated by the following example:

$$\begin{aligned} \text{maximize } & x_1 + 4x_2 \\ \text{subj.to } & x_1 + 10x_2 \leqslant 20 \\ & x_1 \qquad\quad \leqslant 2 \\ & x_1, x_2 \geqslant 0, \text{ integer valued.} \end{aligned}$$

An obvious approach to solve this problem is to ignore the integrality conditions first and to apply the simplex algorithm to the resulting ordinary LP-problem. If the optimal solution obtained in this way turns out to be integer valued the problem is already solved.

In our case we get as optimal solution to the LP-problem $x_1 = 2$, $x_2 = 1.8$ with 9.8 as the maximum

of the objective function. As this solution obviously violates the integrality condition one could expect one of the two points, obtained by rounding the noninteger LP solution, i.e.(2,2), (2,1) respectively, to be optimal solutions to the given problem. It can easily be checked that the first point is not even feasible while the second turns out to be feasible, though not optimal. Indeed, the optimal value of 8 is attained at (0,2) which in our case can be shown e.g. by a graphical method:



A sufficient condition for an LP with integer valued A and b to possess only integer valued basic solutions is the total unimodularity of the matrix A. An integer mxn matrix A is totally unimodular if the determinants of any of the square submatrices of A take the values 0,-1,+1 only. In such a case it is possible to obtain an optimal

integer valued solution by applying the Simplex algorithm directly to the problem: if there is an optimal solution, at least one basic optimal solution exists which under the stated assumptions will be integer valued.

An important class of problems which have a totally unimodular matrix A is that of the so called minimum cost capacitated flow problems, comprising among others the transportation, transshipment, assignment, maximum flow and shortest path problem.

Surveys on the subject are given in Garfinkel & Nemhauser [9] , Geoffrion & Marsten [10], Balinski & Spielberg [2] , and there are several books available, e.g. Garfinkel & Nemhauser [8] , Hu [20] , Zionts [33] .

§ 2.2 Definitions and solution procedures for integer programming problems

As already indicated we get an important class of mathematical programming problems if some or all of the variables involved are required to be integers. Usually the attention is restricted to the linear case only.

First we shall consider the following general definition, namely the problem of:

$\max c'x + d'y$ for $x \in \mathbb{Z}^k$, $x \geqslant 0$, $y \in \mathbb{R}^l$, $y \geqslant 0$

subj.to $Cx + Dy \leqslant b$ with C an (mxk) matrix, D (mxl), $c \in \mathbb{R}^k$, $d \in \mathbb{R}^l$, $b \in \mathbb{R}^m$, $\mathbb{Z}^k$ being the set of integer k-tuples;

which is called mixed integer linear programming problem. If $l = 0$, which means that only the integer valued variable vector x has to be considered, we get an integer linear programming problem (ILP) and it should be noted that the definition also covers the case of linear programming, whenever $k = 0$, such that the integrality condition is not in force.

If we replace $\mathbb{Z}^k$ by $\{0,1\}^k$ we obtain in an analogous way the cases of mixed 0-1 linear programming, and of 0-1 integer linear programming respectively.

Frequently a given integer programming problem can be modelled in several alternative formulations, therefore relationships between different models (e.g. the equivalence of formulations) and methods of transformations are of considerable interest. It has been shown earlier that the application of the simplex algorithm to an arbitrary ILP will in general not yield automatically an optimal solution which is also integer valued, and a sufficient condition has been mentioned.

To solve the problem of developing satisfactory

computational procedures several methods of solution have been proposed and the corresponding algorithms are commonly classified as:

- enumerative,
- cutting plane, and
- heuristic algorithms.

We shall first try to characterize the basic ideas underlying the enumerative approach. If we consider the set of feasible points of an LP, say, min $c'x$, subject to $Ax \geq b$, $x \geq 0$ i.e $\{x \mid Ax \geq b, x \geq 0\}$ assuming it to be bounded, then by imposing the integrality requirement the resulting set of feasible integer valued points will be a finite set consisting of the integer lattice points within the original feasible region, which can be enumerated. In the example above we get the set $\{(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0)\}$.

If we are given a 0-1 programming problem in n variables at most 2^n possible vectors have to be explored. Computationally efficient enumerative algorithms usually employ a tree-like search procedure and the calculation of lower and upper bounds on the values of the objective function.

Cutting plane algorithms are based on rather different concepts: the ILP is first treated as

an ordinary LP and the Simplex algorithm is applied. If the result is not integer valued it can be excluded from further considerations, and any other noninteger point could be neglected as well. A finitely convergent procedure of systematically excluding parts of the feasible region of the LP by introducing additional constraints ("cuts") has been pioneered by R.E.Gomory in 1958 (see e.g. Gomory [11]) and since then various methods of generating cuts have been proposed.

In our example the additional constraint $x_1+x_2 \leq 3$ would exclude the LP optimal solution (2,1.8) without making any of the integer feasible points infeasible. In fact, to maintain the applicability of the simplex algorithm and to ensure at the same time an integer valued result (if that exists at all), we could direct our attention to the smallest convex set containing the integer lattice points (the convex hull). This set will be contained in the feasible LP region, since the latter is itself convex.

It can be shown that solving the modified LP with the convex hull of integer feasible solutions to the original LP as feasible region - together with the original objective function - will yield an optimal solution which also satisfies the inte-

grality condition, i.e. the given ILP requirements altogether. The main problem lies in analytically deriving from the given set of constraints a suitable representation of the convex hull of integer feasible points.Attempts in this direction have been successful at least for a particular class of 0-1 programming problems -the so called monotone and regular 0-1 programs- for which a method of deriving a representation of the convex hull from the original constraints exists.(See Hammer,Johnson,Peled [16]).

An alternative approach to solving integer programming problems is provided by heuristic algorithms intended to derive at least approximate answers to a given problem and these can also be used for obtaining a good starting point for further tree-like search procedures as carried out by, say, enumerative algorithms.

An approach following quite different lines is the so called Pseudo-Boolean Approach - or simply Pseudo-Boolean programming - which can be characterized as making use of results and concepts of Boolean algebra and applying it to discrete optimization problems in general. It covers constrained and unconstrained optimization in 0-1 variables, with objective functions and/ or constraints not only

in linear but also in polynomial form, etc. The name "Pseudo-Boolean" is chosen, because this approach deals mainly with Pseudo-Boolean functions, i.e. real-valued functions in 0-1 variables. A monograph on this subject (Hammer & Rudeanu [19]) appeared in 1968 and more recent developments can be found e.g. in Granot & Hammer [12] and Hammer [15] .Although originally Pseudo-Boolean Programming -being mainly an algebraic approach- has been treated quite distinctly from other -more geometrically oriented- approaches, theoretical connections between these approaches on the one hand and Pseudo-Boolean programming on the other have been established (see e.g. Hammer et.al. [16]). The following chapters of this paper will continue with a more detailed discussion of that approach.

A final comment concerns the concept of duality in the context of integer programming upon which further investigations - quite common for LP problems - depend: recently a duality theory for 0-1 integer programming has been developed by Bell & Shapiro [4] .

§ 2.3 Mathematical Logic and Mathematical Programming.

The basic subject of mathematical logic is the study of formal systems, their interpretation and related problems. A formal system consists of a formal language, a set of axioms together with rules of inference. Theories are a special class of formal systems, usually consisting of a first-order language (variables, function and predicate symbols,quantifiers), a set of logical and nonlogical axioms and a particular set of rules. A model -in this context- is a structure for a theory for which the nonlogical axioms of the theory are valid. A structure is a set of elements (the individuals) with functions and predicates defined on it.

The relationship between mathematical logic on the one hand and mathematics on the other is very close, which is - among other reasons - due to the fact that the methodology applied is very similar and results found in one field of study are relevant to both. Let us take as an example the theory of Boolean algebras: basically a mathematical theory, it is most important to mathematical logic, since the structure of classical logic is essentially that of a Boolean algebra.

In the context of mathematical programming some other aspects of mathematical logic apart from Boolean algebra - which is widely used for integer, especially 0-1 programming,- are of particular interest, e.g. algorithm theory, decision procedures and problems.

The relationship between decision procedures and mathematical programming algorithms is the subject of Williams [31].

Applications of Boolean algebra are widespread (see e.g. Flegg [7]), the monograph by Hammer & Rudeanu [19] , mentioned earlier covers a variety of applications to operational research and related problems, the main subject of the study, however, is the application of Boolean algebra to discrete optimization problems. Earlier instances of using Boolean algebra for solving 0-1 programming problems are e.g. plant location (Smidt & Reis [27]),production scheduling (Akers & Friedman [1]), a selection problem (Root [24]), but it can be said that the main stream of research in this particular field has been influenced by P.L.Hammer's approach, namely that of Pseudo-Boolean programming.

Applications of mathematical programming to problems of mathematical logic provide another interesting aspect of the relationship between both fields. It

has been shown that mathematical programming algorithms can be used for solving optimization problems of threshold logic (switching theory) (see e.g.Muroga [22]), while Williams [32] describes a method of converting Boolean algebra statements into linear equations and inequalities in 0-1 variables which can be used to solve logical problems by means of integer programming algorithms.

It is the aim of this paper to emphasize the role methods and results of mathematical logic play in the context of mathematical programming. Therefore the attempt is made to review such applications and to pursue further investigations into the axiomatic background of 0-1 programming, in particular taking the Pseudo-Boolean approach into account.

CHAPTER 3

Pseudo-Boolean Programming

§ 3.1 Introduction

It is the objective of this chapter to outline the basic characteristics of the Pseudo-Boolean approach to 0-1 integer programming and we will follow the recent version contained in Hammer [14]. Earlier versions of that approach can be found e.g. in Hammer & Rudeanu [18], [19].

One section will be concerned with some concepts of Boolean algebra which will be needed later on, while in a further section an outline of the basic ideas of the approach and their relationship to concepts of Boolean algebra will be given.

§ 3.2 Elements of Boolean Algebra

A Boolean algebra is defined as an algebraic structure $\langle B, \wedge, \vee, -, 0, I \rangle$ which satisfies the following axioms:

B1.	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
B2.	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
B3.	$(x \vee y) \wedge y = y$	$(x \wedge y) \vee y = y$
B4.	$x \vee \bar{x} = I$	$x \wedge \bar{x} = 0$
B5.	$(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$	$(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z)$

In particular the set $\{0,1\}$ with $\wedge, \vee, -$ defined in the following way:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

	$-$
0	1
1	0

satisfies B1 - B5 and we shall denote it by $\langle\{0,1\},\wedge,\vee,-,0,1\rangle$ or shorter by B_2.

$\wedge$ is called conjunction (often denoted by . or simply left out entirely), $\vee$ disjunction, - negation.

If we consider the set $\{0,1\}$, i.e. the set consisting of the integers 0 and 1, we can define $\wedge, \vee, -$ by setting $x \wedge y = xy$, $x \vee y = x+y-xy$, $\bar{x} = 1-x$ with the usual arithmetical meaning of $.,+,-$.
Later on we shall restrict our attention to the particular Boolean algebra B_2 and its n-th cartesian product B_2^n .
Conjunctions of Boolean variables containing no disjunctions are called elementary conjunctions.
Conjunctions are often referred to as products.
Boolean polynomials can be defined recursively as follows:

P1. The constants 0,1 are Boolean polynomials.

P2. The Boolean variables $x_1, x_2, \ldots, x_m$ (literals) are Boolean polynomials.

P3. The conjunctions, disjunctions and negations of Boolean polynomials are Boolean polynomials.

We shall call a function in n variables from $B_2^n \to B_2$ a Boolean function.

If $f(x)$, $g(x)$ are Boolean functions, $f(x) = g(x)$ ($f(x) \leqq g(x)$) is called a Boolean equation (inequality). It is easy to see that a Boolean polynomial defines a Boolean function $B_2^n \to B_2$ if all 2^n possible vectors $\in B_2^n$ are substituted for the Boolean variables $x_1, x_2, \ldots, x_n$ and 0,1 for the constants 0,I.

For example: if the 2^2 possible values of $x = (x_1, x_2) \in B_2^2$ are substituted for x_1 and x_2 in $x_1x_2 \vee x_1\bar{x}_2$ we get the following table:

x_1	x_2	x_1x_2	$\vee$	$x_1\bar{x}_2$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	0

representing the function $f: B_2^2 \to B_2$ with:

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	0
1	0	1
1	1	1

But $f(x_1,x_2)$ can also be represented by x_1.

It is in general true that any Boolean function can be represented by at least one particular Boolean polynomial e.g. in the following way:

Let a particular function f: $B_2^n \to B_2$ be given. Considering the values $f(x_1,x_2,\ldots,x_n)$ at any of the 2^n points of B_2^n we form the following expression (*):

$f(0,0,\ldots,0)\bar{x}_1\bar{x}_2\ldots\bar{x}_n \vee f(0,0,\ldots,1)\bar{x}_1\bar{x}_2\ldots x_n \vee \ldots$

$\ldots \vee f(1,1,\ldots,1)x_1x_2\ldots x_n$ i.e. we are constructing a disjunction of elementary conjunctions of the variables $x_1,x_2,\ldots,x_n$; x_i unnegated, if 1 is substituted for it, negated, if 0. Each conjunction is preceded by the function value at this point, i.e. either 0 or 1, which is replaced by **O** and I in the final representation. According to our definition P1-P3 we obtain a Boolean polynomial. For example:

$\mathbf{O}\bar{x}_1\bar{x}_2 \vee \mathbf{O}\bar{x}_1x_2 \vee Ix_1\bar{x}_2 \vee Ix_1x_2$ is obtained from the Boolean function above.

We can formulate (*) in a more general way as:

$$\bigvee_{(\alpha_1,\alpha_2,\ldots,\alpha_n)\in B_2^n} f(\alpha_1,\alpha_2,\ldots,\alpha_n)x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n}$$

with $x^0 = \bar{x}$, $x^1 = x$ we have $x^\alpha = \begin{cases} 1 & \text{if } x = \alpha \\ 0 & \text{if } x \neq \alpha \end{cases}$

It can be checked easily that a Boolean polynomial derived in such a way represents the original Boolean function, because if we substitute 0,1 for the constants **O**,I and let take the variables $x_1,x_2,\ldots,x_n$ any of the 2^n points $(\alpha_1,\alpha_2,\ldots,\alpha_n) \in B_2^n$, exactly one elementary conjunction will yield the value 1, namely that one containing x_i unnegated if $\alpha_i=1$, negated, if $\alpha_i=0$. Thus $x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n} = 1$ will be preceded by $f(\alpha_1,\alpha_2,\ldots,\alpha_n)$ i.e. yielding exactly the function value equal to that

of the corresponding Boolean function at that point from which the polynomial under consideration has been derived.

For the sake of convenience we shall write elementary conjunctions preceded by I without I, those preceded by O will sometimes be left out of the presentation. Such a presentation will yield the so-called disjunctive normal form.

Two Boolean polynomials are called equivalent if they represent the same Boolean function.

For example: $x_1x_2 \vee x_1\bar{x}_2$ and x_1 are equivalent.

An elementary conjunction C is called an implicant of a Boolean function f if C=1 implies f(x) = 1, (e.g. in our case x_1)).

An implicant C of a Boolean function f is called a prime implicant of f if there is no other implicant of f such that all its literals appear among the literals of the latter implicant.

A Boolean function can also be represented by the disjunction of all its prime implicants, being the shortest normal form representation.

The so called consensus method provides an algorithm for the derivation of prime implicants, (see e.g. Quine [24]). It is well known that the notion of propositional variables and truth functions used in mathematical logic is closely related to the

concepts of Boolean variables, polynomials and functions (see e.g. Ristea [25]), and the consensus method for deriving prime implicants has been developed as a means of simplifying truth functions (see Quine [24]).

§ 3.3 Solving 0-1 integer programming problems by Pseudo-Boolean programming.

From among the variety of problems which have been dealt with by Pseudo-Boolean programming we shall choose the case of 0-1 linear integer programming as our main subject, but it can be shown that the nonlinearity of the objective function and/ or constraints can be reduced to the linear case (see e.g. Hammer & Rudeanu [19].

Let us therefore consider the problem:

$$\min c'x \quad \text{subject to } Ax \geq b \quad \text{with} \quad x \in \{0,1\}^n . \qquad (*)$$

When we are applying the Pseudo-Boolean approach to solving this problem the first step will consist of a transformation of the given problem into one of the following form:

$$\min c'x \quad \text{subject to } \Phi(x) = 0 \text{ with } x \in \{0,1\}^n. \qquad (**)$$

The function $\Phi : B_2^n \to B_2$ is called the resolvent of the set of feasible solutions to the given problem and it is defined to be the Boolean function with the property that, whenever $x \in B_2^n$ is a feasible

solution then $\Phi(x)=0$ and vice versa. It has been shown that (*) is equivalent to (**) (see e.g.Granot & Hammer [12]) and the transformation from (*) into (**) is essentially an algorithmic procedure for deriving a polynomial representation from the original set of constraints. After that the set of feasible solutions to the original problem is now represented as the set of solution to the Boolean equation $\Phi(x) = 0$.

The next stage is concerned with simplifying the actual Boolean polynomial which represents the resolvent and eventually deriving all its prime implicants. If the constant I appears among the prime implicants the problem is infeasible, as the resolvent - e.g. represented as a disjunction of all its prime implicants - will in such a case take on the value 1 for all x, such that $\Phi(x)=0$ cannot be satisfied at all.

There are limitations to that approach if the calculation of a large number of prime implicants is involved and practical algorithms will have to try to avoid this disadvantage. Quine [24] refers to a case where a formula in 9 variables has 1698 prime implicants, whereas the total number of possible assignments of values 0 and 1 to the variables of the formula is only $2^9 = 512$.

An algorithm proposed by Hammer [14] restricts therefore the attention to elementary conjunctions of a given length (i.e. the number of different literals in it), but requires a modification of the Boolean representation implying a relaxation of the given problem. The information given by the conjunctions can be interpreted in a similar way as in the previous case.

After the simplification procedure and further logical analysis the optimal value of the objective function can be derived if the problem is feasible and eventually the optimal solution(s). For practical purposes various algorithms have been proposed (see e.g. Hammer [14]).

Deriving the prime implicants of the resolvent has not only importance from the point of view of mathematical logic, indeed, it has been shown in a paper already mentioned (Hammer, Johnson & Peled [16]) that for special classes of Boolean functions $f(x)$ - monotone and regular functions - the knowledge of the prime implicants allows constructing the convex hull of the set of solutions to the equation $f(x) = 0$.

We shall now consider a particular method (proposed in Hammer [14]) for deriving a Boolean representation of the resolvent characterizing the set

of feasible solutions of a 0-1 integer linear programming problem, i.e. we shall consider $Ax \leqslant b$, $x \in B_2^n$, with A an (mxn) matrix, $b \in \mathbb{R}^m$. Each of the m inequalities corresponding to the m rows can be written as $ax \leqslant \beta$, $a \in \mathbb{R}^n$, $\beta \in \mathbb{R}$, thus omitting the row index.

First the following operations are performed:

If a component a_j is negative, we shall replace the corresponding x_j by $1 - \bar{x}_j$ (remembering that $x_j = 1 - \bar{x}_j$) and transfer a_j to the right hand side. This is done for every negative a_j and the notation is eventually changed in such a way that we get a transformed inequality of the form $a'y \leqslant \beta'$, say, with $a'_1 \geqslant a'_2 \geqslant a'_3 \geqslant \ldots \geqslant a'_n \geqslant 0$ with y_j unnegated throughout.

This particular representation of the original inequality allows the derivation of the so called minimal covers of that inequality in order to obtain a polynomial representation of the resolvent of the inequality. A subset C of the index set $J = 1,2,\ldots,n$ is called a cover of $a'y \leqslant \beta'$ if $\sum_{j \in C} a'_j \geqslant \beta'$. It is called a minimal cover of that inequality if it is a cover of it without properly containing any other cover of the inequality.

It has been shown (see e.g. Granot & Hammer [12]) that given the knowledge of all the minimal covers

$C_1, C_2, \ldots, C_s$ of an inequality $\bigvee_{k=1}^{s} \bigwedge_{j \in C_k} y_j = 0$ if and only if y is a solution to the inequality. This means that $\bigvee_k \bigwedge_j y_j$ can be considered as a polynomial representation of the resolvent of a single inequality, which can easily be reformulated in terms of the original variables x_j. This procedure is repeated for each inequality and eventually the resolvent of the system of constraints $\Phi(x) = \Phi_1(x) \vee \Phi_2(x) \vee \ldots \vee \Phi_m(x)$ is obtained.

This is the starting point for applying a simplification procedure (e.g. the consensus method) yielding the prime implicants of the resolvent. At this stage the so called logical analysis is completed by evaluating the information about the set of feasible solutions which is contained in the particular representation of the resolvent in terms of its prime implicants.

Example:

$$x_1 + 2x_2 - x_3 - 3x_4 \leq 0 .$$

Substituting $1-\bar{x}_3$ for x_3 and $1-\bar{x}_4$ for x_4 we get

$$x_1 + 2x_2 + \bar{x}_3 + 3\bar{x}_4 \leq 1 + 3 = 4.$$

Substituting y_1 for $\bar{x}_4$, y_2 for x_2, y_3 for $\bar{x}_3$, y_4 for x_1

$$3y_1 + 2y_2 + y_3 + y_4 \leq 4 \text{ is obtained.}$$

We get the following minimal covers $\{1,2\}$, $\{1,3,4\}$, therefore a polynomial representation of the resolvent

of the inequality is given by $y_1y_2 \vee y_1y_3y_4$, which in terms of the original variables is $x_2\bar{x}_4 \vee x_1\bar{x}_3\bar{x}_4$.

If we have to deal with polynomials constraints instead of linear ones the same method can be applied after introducing new variables for product terms of the original variables, thus obtaining a constraint which is linear in the newly introduced variables. If the polynomial representation of the resolvent of the so transformed inequality is derived the original variables can be resubstituted.(see e.g. Hammer [14]).

CHAPTER 4

Generalization of Pseudo-Boolean Programming

§ 4.1 Introduction

The final chapter of this paper shall be devoted to theoretical considerations concerning the mathematical and logical assumptions upon which -explicitly and implicitly- Pseudo-Boolean programming is based.

In earlier papers (e.g. Hammer & Rudeanu [18]) optimization of Pseudo-Boolean functions had been considered as the main subject of Pseudo-Boolean programming. Originally Pseudo-Boolean functions had been defined as functions $f: B_2^n \rightarrow \mathbb{Z}$, the set of integers, later on the definition has been extended to functions $f: B_2^n \rightarrow \mathbb{R}$, the set of real numbers.

In this context the question arises quite naturally, which properties of $\mathbb{R}$, or $\mathbb{Z}$, or any other structure, have to be assumed for the development of the Pseudo-Boolean approach. The applicability of elementary Boolean algebra suggests a relatively elementary axiomatic basis to be sufficient, the reason being that elementary Boolean expressions are closely related to formulae of the propositional calculus. As the result of further analysis it will be shown that the Pseudo-Boolean approach can be

generalized to mathematical programming problems involving functions $f: B_2^n \to \mathbb{A}$, where $\mathbb{A}$ is a linearly ordered abelian group. For this generalized approach we will be able to propose a system of axioms which can be formulated in a first-order language.

A similar generalization has been proposed ina paper by Hammer & Rosenberg [17]. It deals with the linear decomposition of Boolean functions defined on groups using a linearization procedure originally developed for Pseudo-Boolean functions.

§ 4.2 Linearly ordered abelian groups, $\mathbb{A}$-Boolean functions and polynomials.

In the following we shall consider the properties of linearly ordered abelian groups which are required for the further development of the proposed generalization.

Let H be an abelian group, we shall write the group operation as +, the neutral element as $\mathcal{O}$ and the inverse of any group element a as -a. If

1 all $a \in H$ are comparable with the neutral element $\mathcal{O}$, i.e. if either $a \geq \mathcal{O}$ or $-a \geq \mathcal{O}$ for all $a \in H$

2 $a \geq \mathcal{O}$ and $-a \geq \mathcal{O}$ implies $a = \mathcal{O}$

3 $a \geq \mathcal{O}$ and $b \geq \mathcal{O}$ implies $a + b \geq \mathcal{O}$ then H is called a linearly ordered group.

Whenever $a + (-b) \geqslant \mathcal{O}$ one can define $a \geqslant b$ and it can be shown that we get a linearly ordered set, i.e. the following axioms hold:

1 for all a, $a \geqslant a$

2 $a \geqslant b$ and $b \geqslant c$ implies $a \geqslant c$

3 $a \geqslant b$ and $b \geqslant a$ implies $a = b$

4 for all a, b $a \geqslant b$ or $b \geqslant a$.

Let $\mathbb{A}$ be a linearly ordered abelian group. We shall denote the operator which maps every element $a \in \mathbb{A}$ to itself: $\mathbb{I}$, i.e. $a.\mathbb{I} = a$, whilst $\mathbb{0}$ shall be used for the operator for which $a.\mathbb{0} = \mathcal{O}$ holds ($\mathcal{O}$ being the neutral element of $\mathbb{A}$) for every $a \in \mathbb{A}$. In each case we shall write the operator on the right hand side, the operation is written as $\cdot$.

Note: $(a + b).x = a.x + b.x \quad x \in \{\mathbb{0}, \mathbb{I}\}$;

proof: $x = \mathbb{I}$ implies $(a + b).\mathbb{I} = a + b = a.\mathbb{I} + b.\mathbb{I}$, if $x = \mathbb{0}$ we get $(a + b).\mathbb{0} = \mathcal{O} = a.\mathbb{0} + b.\mathbb{0}$

In analogy to the notation proposed in Hammer & Rosenberg [17] for functions from $B_2^n \rightarrow G$, with G an ordered (not necessarily linearly ordered) abelian group we shall call a function $f: B_2^n \rightarrow \mathbb{A}$ an $\mathbb{A}$-Boolean function. Since $\mathbb{A}$ fulfills automatically the properties of an ordered abelian group all properties of G-Boolean functions apply also to $\mathbb{A}$-Boolean functions.

We can impose easily on the set $\{\mathbb{0}, \mathbb{I}\}$ consisting of the two (trivial) operators on $\mathbb{A}$ the structure

of a Boolean algebra by defining $\wedge, \vee, -$ in the usual way and we obtain with $\langle \{0,I\}, \wedge, \vee, -, 0, I \rangle$ a Boolean algebra which is isomorphic to B_2.

An A-Boolean polynomial in n variables $x_i \in B_2$ is a finite sum consisting of product terms $a_k \cdot \prod_{i=1}^{n_k} x_i$ and possibly constant terms a_j with $a_j, a_k \in A$, $n_k \leq n$, $n_k = n$ for at least one k. If we interpret the elements 0,1 of B_2 as the operators $0, I$ on A, we can see that by substituting for $x_1, x_2, \ldots, x_n$ all 2^n possible values $\in B_2^n$ every A-Boolean polynomial defines an A-Boolean function $f: B_2^n \rightarrow A$.

On the other hand we can find for every A-Boolean function at least one polynomial representation, in particular the following one (exactly as before):

$$\sum a_{\alpha_1 \ldots \alpha_n} \cdot x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n},$$

where the summation ranges over all the 2^n possible values of $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in B_2^n$ and $a_{\alpha_1 \ldots \alpha_n}$ is uniquely determined by the relation $a_{\alpha_1 \ldots \alpha_n} = f(\alpha_1, \alpha_2, \ldots \alpha_n)$. We call it canonical (or normal) form and show that every A- Boolean polynomial can be converted into this form. Since on the one hand every A-Boolean polynomial corresponds to a unique A-Boolean function, but for every A-Boolean function there exists more than one polynomial representation the problem arises how to decide if two different polynomials

represent the same A-Boolean function or not.

To achieve this task we shall employ a special procedure capable of transforming a given A-Boolean polynomial into an equivalent one of the form:

$\sum_{(\alpha_1,\alpha_2,\ldots,\alpha_n)\in B_2^n} a_{\alpha_1\ldots\alpha_n} \cdot x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ according to the definition given earlier. Equivalence means that both polynomials represent the same function.

First we introduce a new operation $\oplus$ for the elements of B_2 :

$\oplus$	0	1
0	0	1
1	1	0

i.e.addition mod 2.

For this operation the following relations hold:

- $(x \oplus y)z = xz \oplus yz$, $x,y,z \in B_2$.

If $xy = \mathbb{0}$ then $a.(x \oplus y) = a.x + a.y$, $x,y \{\mathbb{0},\mathbb{I}\}$, $a \in A$.

Because: $xy = \mathbb{0}$, whenever one of the variables or both of them are equal to $\mathbb{0}$, leading to the two alternative cases:

1st case: without loss of generality let $x=\mathbb{0}$, thus $y=\mathbb{I}$ and $x + y = \mathbb{I}$. Therefore $a.(x + y) = a.\mathbb{I} = a$; $a.x + a.y$ becomes: $a.\mathbb{0} + a.\mathbb{I} = \mathcal{O} + a = a$.

2nd case: $x = y = \mathbb{0}$, hence $x + y = \mathbb{0}$ and $a.(x+y) = a.\mathbb{0} = \mathcal{O}$; $a.x + a.y = a.\mathbb{0} + a.\mathbb{0} = \mathcal{O}+\mathcal{O} = \mathcal{O}$.

In particular if we take $y = \bar{x}$ we get $a.(x \oplus \bar{x}) = a.x + a.\bar{x}$.

Let us now consider an arbitrary A-Boolean polynomial in n variables $x_1, x_2, \ldots x_n$;

we shall proceed in the following way:

- taking each term in turn and multiplying it by the product $\prod_j (x_j \oplus \bar{x}_j)$ of all x_j which do not appear yet among the variables of the term considered;
- applying the distributive laws $(x \oplus y)z = xz \oplus yz$ and (if $xy = 0$) $a.(x \oplus y) = a.x + a.y$
- we eventually arrive at the stage where there are left only terms of the form $a_k \cdot \prod_{i=1}^{n} x_i$ which contain all variables $x_1, x_2, \ldots x_n$ in the product;
- applying $(a.x + b.x) = (a + b).x$ we can add terms with equal product terms and after rearranging we obtain the desired result, namely a sum of the form $\sum_{(\alpha_1,\ldots,\alpha_n)\in B_2^n} a_{\alpha_1,\ldots,\alpha_n} x_1^{\alpha_1} \ldots x_n^{\alpha_n}$.

It can be shown easily (e.g. by induction) that by applying any of these operations we obtain equivalent polynomials only, i.e. polynomials representing the same function as the original polynomial.

Example:

$2+(-3)x_1+2\bar{x}_1x_2$; multiplying by $\prod_j (x_j \oplus \bar{x}_j)$ we get $2(x_1 \oplus \bar{x}_1)(x_2 \oplus \bar{x}_2)+(-3)x_1(x_2 \oplus \bar{x}_2)+2\bar{x}_1x_2$, the first distributive law yields $2(x_1x_2 \oplus \bar{x}_1x_2 \oplus x_1\bar{x}_2 \oplus \bar{x}_1\bar{x}_2) + (-3)(x_1x_2 \oplus x_1\bar{x}_2)+ 2\bar{x}_1x_2$; applying the second distributive law we get $2x_1x_2 + 2\bar{x}_1x_2 + 2x_1\bar{x}_2 + 2\bar{x}_1\bar{x}_2 + (-3)x_1x_2 + (-3)x_1\bar{x}_2 + 2\bar{x}_1x_2$; adding similar terms leads to $(-1)x_1x_2 + (-1)x_1\bar{x}_2 + 4\bar{x}_1x_2 + 2\bar{x}_1\bar{x}_2$

representing $f: B_2^2 \to Z$ according to

f	0	1
0	2	4
1	-1	-1

We have used the fact that Z fulfills the axioms of a linearly ordered abelian group and have interpreted the operation . as the ordinary multiplication. The equivalence of two A-Boolean polynomials can therefore be decided by means of their normal form representations which can be obtained in the described way.

§ 4.3 Constrained optimization of A-Boolean functions

After having made the necessary preparations we can now proceed to solving mathematical programming problems involving A-Boolean functions as objective function and also being used for the formulation of the constraints, i.e. we shall consider problems of the form:

$$\min f(x) \quad \text{subject to } g_j(x) \geq b_j \quad j=1,2,\ldots,m$$

$$x \in B_2^n,\ f(x),\ g_j(x) : B_2^n \to A,\ b_j \in A.$$

Since $f(x)$ and $g_j(x)$ can be represented by A-Boolean polynomials the method described earlier for deriving the normal form of a given polynomial can be applied -at least theoretically- and we shall first propose the following algorithmic solution procedure:

- Constant terms appearing in an inequality are

transferred to the right hand side.

- The given polynomial is transformed into normal form representation i.e. the product terms are arranged in the following way:

$$a_1x_1x_2\cdots x_n+a_2x_1x_2\cdots\bar{x}_n+\cdots+a_{2n}\bar{x}_1\bar{x}_2\cdots\bar{x}_n$$

- In each inequality the coefficients at the left hand side (i.e. the values it takes on at each point) are compared with the constants at the right hand side and the product terms fulfilling the inequality are marked.

- Eventually all the terms which are marked in each of the m inequalities are singled out, they correspond to the set of feasible solutions. If there are none, the problem is infeasible (assuming that all the terms with zero coefficients are listed as well).

- The product terms in the normal form representation of the objective function which appear among the product terms corresponding to feasible solutions are marked.

- The coefficients of the product terms of the objective function just marked are compared and the optimal value chosen, yielding the required optimum. Assigning 1 to each unnegated, 0 to each negated variable, within each product term with an optimal coefficient, we obtain the vector(s)

$\in \{0,1\}^n \subseteq B_2^n$ which constitute the optimal solution (s) to the given problem.

The validity of the procedure can be proved easily e.g. by induction. In fact, it involves the explicit enumeration of all the function values and will therefore be only of theoretical use. If however a large proportion of the product terms already contains almost all of the variables $x_1, x_2, .., x_n$ only a few steps may be needed to obtain the result.

Next we will show that the concept of the resolvent can be used analogously as in the ordinary 0-1 integer programming case.

To achieve this, the operations for deriving a polynomial representation of the resolvent, which have been discussed earlier, have to be slightly modified:

Instead of replacing x_j with negative coefficients a_j by $1-\bar{x}_j$, $a_j x_j$ has to be replaced by $a_j+(-a_j)\cdot \bar{x}_j$ which is permitted because $a_j x_j = a_j+(-a_j)\cdot \bar{x}_j$.

The further steps involve the transfer of constant elements to the right hand side and ranking the a_j which can be done assuming the structure of a linearly ordered abelian group for the set of coefficients.

After changing the notation exactly as before we can start deriving the minimal covers of each inequality. This involves comparisons of sums of

coefficients with the right hand side coefficients which again can be done under our assumptions. Altogether we have shown that constrained optimization problems involving A-Boolean functions can be formulated by means of A-Boolean polynomials which can be defined in terms of product terms consisting of conjunctions of variables $x_1, x_2, \ldots x_n$ each of them ranging over the set $\{\mathbb{0}, \mathbb{I}\}$ satisfying the axioms of a Boolean algebra. These conjunctions are preceded by coefficients a, which fulfill the axioms of a linearly ordered abelian group A together with the following operations:

$a.\mathbb{I} = a$ and $a.\mathbb{0} = \mathbb{0}$ for $a, \mathbb{0} \in A$.

We have also seen that basic characteristics of Pseudo-Boolean programming (e.g. the concept of the resolvent) remain valid in this more general context.

CONCLUSIONS

An exposition of the background of integer programming has been given, also a basic characterization of the Pseudo-Boolean approach particularly its application to solving 0-1 integer programming problems.

The notion of A-Boolean functions and polynomials is introduced as a result of generalizing Pseudo - Boolean functions and polynomials and is then used for formulating constrained optimization problems involving A-Boolean functions.

In this context a procedure for converting polynomials into normal form is of some importance.

It is hoped that this paper draws attention to the role mathematical logic is playing when applied to mathematical programming problems some aspects of which have been highlighted.

REFERENCES

Akers S.B. and J.Friedman [1], "A Non-Numerical Approach to Production Scheduling Problems", Operational Research 3 (1955)

Balinski M.L. and K.Spielberg [2], "Methods for Integer Programming: Algebraic, Combinatorial and Enumerative", 195-292, in Aronofsky J.(ed.), Progress in Operations Research Vol.3, John Wiley & Sons, 1969

Beale E.M.L. [3], Mathematical Programming in Practice, Pitman, 1971

Bell D.E. and J.F.Shapiro [4], "A Finitely Convergent Duality Theory for 0-1 Integer Programming", Research Memorandum RM-75-33, IIASA, 1975

Dantzig G.B. [5], "On the Significance of Solving Linear Programming Problems with Some Integer Variables", Econometrica 28(1960), 30-44

-"- [6], Linear Programming and Extensions, Princeton University Press, 1963

Flegg H.G. [7], Boolean Algebra and its Application, Blackie, 1964

Garfinkel R.S. and G.L.Nemhauser [8], Integer Programming, John Wiley & Sons, 1972

-"- -"- [9], "A Survey of Integer Programming Emphasizing Computation and Relations among Models", 77-155 in Hu T.C. and S.M.Robinson(eds.), Mathematical Programming, Academic Press, 1972

Geoffrion A.M. and R.E.Marsten [10], "Integer Programming Algorithms: A Framework and State-of-the-Art Survey", Management Science Vol.8, No.9(1972), 465-491

Gomory R.E. [11], "An Algorithm for Integer Solutions to Linear Programs", 269-302, in Graves R.L. and Ph.Wolfe (eds.), Recent Advances in Mathematical Programming, McGraw-Hill, 1963

Granot F. and P.L.Hammer [12], "On the Use of Boolean Functions in 0-1 Programming", Operations Research, Statistics and Economics Mimeograph Series No.70, Technion(Haiffa), 1970

Hadley G. [13], Linear Programming, Addison-Wesley, 1962

Hammer P.L. [14], "Boolean Procedures for Bivalent Programming", Research Report CORR 73-1, University of Waterloo (Ontario), 1973

-"- [15], "Boolean Elements in Combinatorial Optimization A Survey", Research Report CORR 74-22 University of Waterloo (Ontario), 1974

-"- ,E.L.Johnson and U.N.Peled [16], "Facets of Regular 0-1 Polytopes", Research Report CORR 73-19, University of Waterloo (Ontario), 1973

-"- and I.G.Rosenberg [17], "Linear Decomposition of a Positive Group-Boolean Function", Research Report CORR 73-25, University of Waterloo,(Ontario),1973

-"- and S.Rudeanu [18], "Pseudo-Boolean Methods for Bivalent Programming", Lecture Notes in Mathematics No.23, Springer 1966

-"- -"- [19], Boolean Methods in Operations Research and Related Areas, Springer 1968

Hu T.C. [20], "Integer Programming and Network Flows", Addison-Wesley, 1969

Land A. and S.Powell [21], Fortran Codes for Mathematical Programming, John Wiley & Sons, 1973

Muroga S. [22], Threshold Logic and its Applications, John Wiley & Sons, 1971

Orchard-Hays W. [23], Advanced Linear Programming Computing Techniques, McGraw-Hill, 1968

Quine W.V. [24], "On Cores and Prime Inplicants of Truth Functions", American Math.Monthly Vol.62(1955), 755-760

Ristea T. [25], "On Propositional, Truth and Boolean Functions", Notre Dame Journal of Formal Logic Vol.IX No.2 (1968), 160-166

Root J.G. [26], "An Application of Symbolic Logic to a Selection Problem", Operations Research 12 (1964), 519-526

Smidt R.M. and I.L.Reis [27], "Symbolic Logic and Plant Location", Journal of Industrial Engineering Vol.XIV No.1 (1963)

Vajda S. [28], Mathematical Programming, Addison-Wesley,1961

-"- [29], Theory of Linear and Non-Linear Programming, Longman, 1973

-"- [30], Planning by Mathematics (2nd ed.),Pitman,1973

Williams H.P. [31], "Decision Procedures in Formal Logic and Mathematical Programming", Research Report University of Sussex, 1973

-"- [32], "Logical Problems and Integer Programming", Research Report 75-2, University of Sussex, 1975

Zionts St. [33], Linear and Integer Programming, Prentice-Hall, 1974