ROBERT S. GARFINKEL,

Graduate School of Management
University of Rochester

GEORGE L. NEMHAUSER,

Department of Operations Research
Cornell University

Integer Programming

A WILEY-INTERSCIENCE PUBLICATION

JOHN WILEY & SONS

NEW YORK · LONDON · SYDNEY · TORONTO

Library of Congress Cataloging in Publication Data:

Garfinkel, Robert.
Integer programming.

(Series in decision and control)

Bibliography: p.
1. Integer programming. I. Nemhauser, George L., joint author. II. Title.

T57.7.G37 519.7'7 72-3881
ISBN 0-471-29195-1

Printed in the United States of America

10-9 8 7 6 5 4 3 2 1

Preface

Integer programming deals with the class of optimization problems in which some or all of the variables are required to be integer. This book is intended to be a comprehensive one on the theory, methodology, and applications of integer programming and can be used both as a text and as a reference.

Since the pioneering work of Ralph Gomory in the late 1950s, integer programming has been one of the most exciting and rapidly developing areas of operations research. Very many real-world applications of integer programming models have been developed, and algorithms have been devised for general and specific purpose models. These algorithms vary widely in spirit and mathematical sophistication. One of our objectives is to unify these approaches as far as possible and to classify them into a few generic categories so that similarities and differences may be easily seen.

The book is designed to be a text for a one- or two-semester course, depending on the sophistication of the students. Numerous exercises, ranging from simple numerical calculations to research problems, are provided. Every technique that is developed is illustrated by at least one numerical example. A comprehensive set of notes is included at the end of every chapter. These notes give an historical summary of the development of the material of each chapter. More importantly, they supply a (hopefully complete) set of references to integer programming, including many works that are not described in the body of the text.

Chapter 1 introduces the reader to some methods of formulating and solving integer programming problems. In addition, a number of applications are presented. Chapter 2 is an introduction to those linear programming methods used in integer programming. A knowledge of some basic linear algebra is needed, and this is actually the only mathematical requirement for reading the entire text. Chapter 3 deals with integer programming and graphs. The first four sections are basic and should be included in any course. Sections 3.5–3.7 are special topics of interest. Chapter 4 on enumeration is one of the fundamental chapters and should be studied in its entirety. Chapter 5 on cutting planes is also essential. The only topics that could reasonably be omitted are the rather complex details of the proof of termination in Section 5.12 and the methods described in Sections 5.14–5.16 and 5.18. The knapsack problem, discussed in Chapter 6 is a specific integer programming model. However, the development of Chapter 7 is, to some extent, dependent on Sections 6.1 and 6.5–6.7. Section 6.12 is of importance in its own right.

Chapter 7 contains some of the most recent approaches to integer programming problems. The first six sections should be included in any course. The set covering and partitioning problems discussed in Chapter 8 are special models with a wide range of applications and relatively efficient algorithms. Theoretical and computational aspects of covering and partitioning are currently an active area for research. Chapter 9 on approximate methods is quite straightforward but very important for anyone interested in solving large problems. We would recommend including this chapter in a course involving the computational aspects of integer programming. Chapter 10 contains a variety of special topics involving nonlinearities. Chapter 11 contains computational experience. It is the only chapter written in the form of the notes; that is, it contains references to authors within the body of the text. Anyone interested in solving real-world problems is strongly advised to read this chapter.

The book, in draft form, has been used as a text for courses in integer programming at Cornell and Rochester, as well as at other universities. Our students and colleagues have been very helpful in finding errors and making suggestions. In particular, we would like to thank Egon Balas, Gerry Bennington, Mark Eisner, Mendu Rao, Don Ratliff, and David Rubin for their comments. Ivan Rosenberg, at the University of Rochester, read a number of the chapters in detail and provided many corrections. We are especially grateful to Les Trotter, at Cornell University, who read the manuscript in its entirety, and whose suggestions and corrections were invaluable. Finally, we are grateful for the efficient typing service provided by Cornell and Rochester. In particular, special thanks are due Debbie Cagey, Kathy King, and Lorraine Ziegenfuss, all of whom did a marvelous job in typing the bulk of the manuscript.

ROBERT S. GARFINKEL
GEORGE L. NEMHAUSER

Rochester, New York
Ithaca, New York
February 1972

Contents

INTEGER PROGRAMMING

1 Introduction

1.1. BACKGROUND

Integer programming is a branch of *mathematical programming*. A general mathematical programming problem can be stated abstractly as

$$\max f(x), \qquad x \in S \subseteq R^n \tag{1}$$

where R^n is the set of all n-dimensional vectors of real numbers and f is a real-valued function defined on S. The set S is called the *constraint set* and f is called the *objective function*.

Every $x \in S$ is called a *feasible solution* to (1). If there is an $x^o \in S$ satisfying

$$\infty > f(x^o) \geq f(x) \qquad \text{for all } x \in S$$

then x^o is called an *optimal solution* to (1). The objective in a mathematical programming problem is to establish whether an optimal solution exists and then to find one, or perhaps all, optimal solutions.

Although there is no generally agreed-upon definition, an *integer programming problem* is a mathematical programming problen. in which

$$S \subseteq Z^n \subseteq R^n$$

where Z^n is the set of all n-dimensional integer vectors. A *mixed integer programming problem* is a mathematical programming problem in which at least one, but not all, of the components of $x \in S$ are required to be integers.

In an applied context, it is convenient to think of (1) as a model of decision making in which S represents the set of all permissible decisions and f assigns a utility or profit to each $x \in S$. Applications of this model abound in the real world and are relevant to various branches of engineering, business, and the physical and social sciences. Some of these applications are discussed in Sections 1.4 and 1.5 and briefly throughout the book.

Much of the work done in modeling and developing solution techniques for mathematical programming problems comes under the names *operations research* and *management science*. Stimulated by scientific applications to military planning problems during World War II, operations research and management science have now become disciplines in their own right.

Linear Programming Problems

Considerable success in analyzing (1) has been achieved for the *linear programming problem* (LP), where

$$f(x) = cx \tag{2}$$

and

$$S = \{x | Ax = b, x \geq 0\} \tag{3}$$

Here, A is an $m \times n$ matrix, b is an m-vector, c is an n-vector, and 0 is an n-vector of zeros. The set S is a *convex set*, since $x, y \in S$ imply $\alpha x + (1 - \alpha)y \in S$ for all $0 \leq \alpha \leq 1$ (see Exercise 1). A convex set defined by linear constraints is called a *polyhedron* or *polytope*.

The Use of Computers

It is very important to keep in mind that the aim of much of the research in mathematical programming is to develop a theory that leads to the construction of algorithms for use on modern, high-speed digital computers. An unsophisticated reader might be attracted to simple algorithms that can be used to solve small problems by hand. Instead, the following important aspects of algorithms should be considered:

1. Is finite termination guaranteed?
2. If so, is there an upper bound on the number of computations?
3. Has computational experience been promising with respect to the speed of the algorithm?
4. Are computer storage requirements reasonable?
5. If the algorithm does not terminate in a specified time, are feasible solutions generated?

Notation

In defining the linear programming problem, we referred to x as an n-vector, without specifying whether it was $n \times 1$ or $1 \times n$. However, from (3) it is clear that x is $n \times 1$, b is $m \times 1$, and 0 is $n \times 1$, since the various matrices must be conformable for matrix multiplication. From (2) it follows that c is $1 \times n$. In general, we will not use a transpose operation on a matrix, unless the possibility of confusion exists.

In comparing two n-dimensional vectors (n-vectors) a and b, we write $a \geq b$ to mean $a_j \geq b_j$, $j = 1, \ldots, n$. If $a \geq b$, and $a_j > b_j$ for some j, we write $a > b$. Similarly, for two sets A and B, $A \subseteq B$ means that A is a subset of B, while $A \subset B$ denotes A a proper subset of B. The number of elements (*cardinality*) of a finite set A is denoted by $|A|$.

1.2. DEFINING AN INTEGER LINEAR PROGRAMMING PROBLEM

Much of this book is devoted to analyzing the *integer linear programming* problem (ILP) in which $f(x)$ is given by (2) and

$$S = \{x \mid Ax = b, x \geq 0 \text{ integer}\} \tag{4}$$

In more standard form the ILP is written

$$\begin{aligned} &\max cx \\ &Ax = b \\ &x \geq 0 \text{ integer} \end{aligned} \tag{5}$$

In summation notation (5) is

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, \ldots, m \\ &x_j \geq 0 \text{ integer}, \qquad j = 1, \ldots, n \end{aligned} \tag{6}$$

All of the data in c, A, and b will always be assumed integer. This is actually equivalent to assuming the data rational, since multiplication of the objective function by any positive number, or any constraint by any nonzero number, does not change the problem (see Exercise 2).

The ILP can be written in any of a number of forms, and (5) and (6) have been arbitrarily chosen to be the ones generally used. Sometimes we will choose to minimize rather than maximize. This causes no difficulty, since for $x \in S$

$$-(\min - f(x)) = \max f(x)$$

Similarly, we could use the constraint form $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$, or $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$. Table 1 gives methods for converting from any of these constraint forms to any other.

Table 1

			To	
		$\le$	$\ge$	$=$
	$\le$		$-\sum_j a_{ij}x_j \ge -b_i$	$\sum_j a_{ij}x_j + s_i = b_i$ $s_i \ge 0$
From	$\ge$	$-\sum_j a_{ij}x_j \le -b_i$		$\sum_j a_{ij}x_j - t_i = b_i$ $t_i \ge 0$
	$=$	$\sum_j a_{ij}x_j \le b_i$ $-\sum_j a_{ij}x_j \le -b_i$	$\sum_j a_{ij}x_j \ge b_i$ $-\sum_j a_{ij}x_j \ge -b_i$	

The new variables s_i and t_i introduced in Table 1 are known as *slack* and *surplus* variables, respectively. Finally, if $x_j \ge 0$ is not required, an appropriate transformation is $x_j = x'_j - x''_j$, $x'_j, x''_j \ge 0$, so that x_j is unrestricted in sign.

The LP obtained by dropping the integrality constraints from the ILP (5) will be referred to as the *corresponding LP*. We also refer to the LP as a *relaxation* of the ILP. In general, the problem

$$P1: \quad \max f(x), \qquad x \in S_1$$

is said to be a relaxation of the problem

$$P2: \quad \max f(x), \qquad x \in S_2$$

if $S_1 \supseteq S_2$. Similarly, $P2$ is said to be a *restriction* of $P1$. The concepts of relaxation and restriction will be used often throughout the text. Note that if x^o is an optimal solution to $P1$ and x^* is an optimal solution to $P2$, then $f(x^o) \ge f(x^*)$. Furthermore, if $x^o \in S_2$, then x^o is an optimal solution to $P2$ (see Exercise 3).

An important special case of the ILP (6) is the *binary* ILP, where $x_j \ge 0$ and integer is replaced by $x_j = 0, 1$. Although the binary ILP could be written in the form (6) by adding the constraints $x_j \le 1$, $j = 1, \ldots, n$, it will generally be written

$$\begin{aligned} &\max cx \\ &Ax = b \\ &x \text{ binary} \end{aligned} \tag{7}$$

There is an important class of binary ILP's in which $a_{ij} = 0, 1$ for all i and j, and $b_i = 1$ for all i. Special methods have been developed for these so-called *covering* and *matching* problems, and they will be discussed in Chapters 3 and 8.

A generalization of the ILP is the *mixed integer linear program* (MILP), where only some of the variables are constrained to be integer. It is written

$$\begin{aligned} \max\ & c_1x + c_2v \\ & A_1x + A_2v = b \\ & x \geq 0 \text{ integer} \\ & v \geq 0 \end{aligned} \tag{8}$$

where A_1 is $m \times n_1$ and A_2 is $m \times n_2$. If $n_1 = 0$, (8) is an LP, and if $n_2 = 0$, (8) is an ILP.

1.3. METHODS FOR SOLVING THE ILP

In this section, a brief introduction is given to two fundamental approaches for solving ILP's, *enumeration* and *cutting planes*. Since most algorithms for solving ILP's solve subproblems that are LP's, a summary of the basic methodology of linear programming will be presented in Chapter 2. From the relaxation concept introduced in Section 1.2, it follows that if the corresponding LP has an optimal solution x^o which is an integer vector, then x^o is a feasible solution, and thus an optimal solution, to the ILP. There is a class of ILP's (see Chapter 3) for which the corresponding LP always has an integer optimal solution. In this class of problems, the relaxation of the ILP to the corresponding LP does not change the essence of the problem.

In general, the corresponding LP does not have an optimal solution that is integer. Such is the case in the following simple example for which two solution techniques are sketched.

Example

$$\begin{aligned} \max\ & 2x_1 + x_2 \\ & x_1 + x_2 + x_3 = 5 \\ & -x_1 + x_2 + x_4 = 0 \\ & 6x_1 + 2x_2 + x_5 = 21 \\ & x_1, \ldots, x_5 \geq 0 \text{ integer} \end{aligned}$$

Since x_3, x_4, and x_5 are slack variables, the problem can be written

$$\begin{aligned} \max\ & 2x_1 + x_2 \\ & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 0 \\ & 6x_1 + 2x_2 \leq 21 \\ & x_1, x_2 \geq 0 \text{ integer} \end{aligned} \tag{9}$$

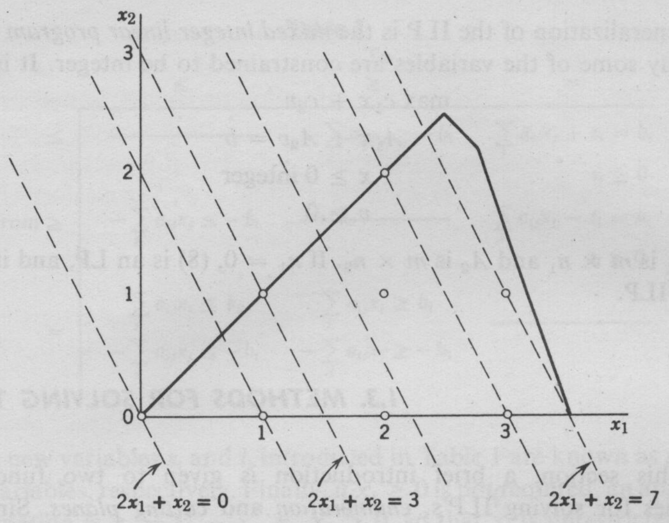

Figure 1

The feasible region of the corresponding LP is shown in Figure 1, along with a sequence of parallel lines representing constant values of the objective function. The circles represent the feasible solutions to (9). From Figure 1 it is clear that there are eight feasible solutions to (9) and that the optimal solution is $x_1 = 3$, $x_2 = 1$.

Enumeration

Without examining Figure 1, it is possible to get an upper bound on the number of feasible points. Note that the first constraint of (9), along with nonnegativity, implies $0 \leq x_1, x_2 \leq 5$. Also, the third constraint implies $0 \leq x_1 \leq 3$. Thus the constraints $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 5$, along with the integrality requirements, limit the number of feasible points to not more than 24. For such a small problem, these 24 points could be *totally enumerated* to find that 16 are infeasible, 8 are feasible, and $(x_1, x_2) = (3, 1)$ is optimal.

Using a little more imagination, we might have added the first and second constraints to derive the valid constraint $2x_2 \leq 5$. Along with integrality, this implies $x_2 \leq 2$ and reduces the upper bound on the number of feasible points from 24 to 12.

Note that the feasible point $x_1 = 3$, $x_2 = 0$ yields an objective value of $f(x) = 6$. Thus every optimal solution to (9) must satisfy $2x_1 + x_2 \geq 6$.