INTEGER PROGRAMMING

Volume 76

Harold Greenberg

INTEGER PROGRAMMING

HAROLD GREENBERG

Naval Postgraduate School
Monterey, California

ACADEMIC PRESS *New York · London*

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
Berkeley Square House, London W1X 6BA

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 73-137596

AMS (MOS) 1970 Subject Classification: 90C10

PRINTED IN THE UNITED STATES OF AMERICA

Contents

Chapter 3 All-Integer Methods

Chapter 4 Solving Integer Programs by Enumeration

Chapter 5 Continuous Solution Methods

Chapter 6 Number Theory Results

Chapter 7 Dynamic Programming Solutions

Chapter 8 Branch and Bound Procedures

Preface

This book is designed to serve as an introduction to the study of integer programming. The first two chapters contain applications of the subject and a review of linear programming.

The integer programming problem is then placed in a parametric context. From that perspective, an understanding of the integer programming process is acquired and the convergence proofs and algorithmic results are developed.

The approach is to express the variables of the integer program in terms of parameters and then to vary the parameters until the problem is solved. This flexible framework permits the handling of such previously difficult problems as those in which the variables have upper bounds and helps to establish the relationship between the all-integer and the continuous solution methods.

The book also offers an enumerative procedure for the solution of integer programs. The procedure is developed by means of a dynamic

programming format that makes it possible for many solutions to be generated at one time. The enumeration is then accelerated by a method that establishes the variable values leading to the solution of the problem.

The book concludes with a branch and bound scheme that eliminates the necessity for large computer storage capacity—a drawback in the early branch and bound methods.

I am very grateful to Professor Richard Bellman, who proposed the writing of the book, and to my wife, Eleanor, for her editorial assistance and typing of the manuscript.

1 INTRODUCTION TO INTEGER PROGRAMMING

1 PRESENTATION OF THE PROBLEM

A great many problems in mathematical programming are solved as linear programs. The usual linear programming problem is: Find $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\sum_{j=1}^{n} c_j x_j = z,$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m.$$

The solution to the problem usually results in fractional x_j values. The integer programming problem arises when the x_j variables are restricted to integer values.

To illustrate the range of integer programming, we present examples of practical problems that are amenable to solution within an integer programming format. We also describe mathematical programming problems that can be handled as integer linear programs. (Most of the examples presented were first collected by Dantzig [6].)

2 PILOT SCHEDULING

A major concern of many airlines is how to schedule their flight crews. A crew member is assigned to a particular flight. For example, Captain Bursting is pilot of Flight 92. It leaves New York City at 11.20 a.m. and arrives in Boston at 12.02 p.m. After arriving in Boston, Captain Bursting goes to the flight operations office to report on his trip and to learn of any changes in his forthcoming flight. He must allow 30 min for this period. Bursting then has 50 min for lunch. He reports to the flight operations office 40 min before the scheduled departure time of his next flight to receive the flight plan, weather reports, air traffic reports, etc. He is scheduled for a flight to leave Boston after 2.02 p.m. The flight could be any one of those shown in the accompanying tabulation.

Flight Number	Departure Time (p.m.)	Arrival Time (p.m.)	Arrival Station
121	2.30	3.25	New York City
171	2.35	3.59	Chicago
437	3.00	4.03	New York City
759	3.00	5.44	Miami
103	3.35	4.59	Chicago

Meanwhile, Captain Masters is pilot of Flight 230. It leaves Chicago at 8.55 a.m. and arrives in Boston at 11.48 a.m. He is also eligible to fly the same listed flights out of Boston. How should both Bursting and Masters be scheduled?

What we have begun to describe is a combinatorial problem of enormous size. The airline company has 100 flights into and out of Boston every day. How should the incoming and outgoing flights be linked as part of a pilot's continuous flight schedule? Before we can even answer the question, we must view the problem in a broader context. Boston connects directly to 25 other cities. A total of 75 cities are in the

overall network that comprises the flight schedule for the airline company. What is the complete flight schedule for all pilots when all the other cities are included?

Fortunately, the problem can be expressed as an integer program: Find $x_j = 0$ or 1 for $j = 1, 2, \ldots, n$ that minimize z when

$$\sum_{j=1}^{n} c_j x_j = z,$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \qquad i = 1, 2, \ldots, m,$$

where

i stands for a particular flight (number) between two cities,
m is the total number of flights,
j stands for a set of flights that describe a path,
n is the total number of possible paths,
$a_{ij} = 1$ means flight i appears in path j,
$a_{ij} = 0$ means flight i does not appear in path j,
c_j is the cost of path j,
$x_j = 1$ means path j is flown, and
$x_j = 0$ means path j is not flown.

The units on the right side of the inequalities cause all flights to be flown. The greater than or equal sign permits pilots to ride as passengers to other cities to meet their next flights. The costs c_j may include per diem rates for pilots, hotel costs, meals, premium pay, etc. If all $c_j = 1$, we minimize the number of pilots needed to fly the schedule. Otherwise, we minimize the total cost.

The flights that make up each path (i.e., the a_{ij} values) are found by enumerating all combinations of flights that can physically connect and that comply with the airline's flight regulations. Typical flight regulations for pilots are:

(1) The pilot must check in at flight operations 45 min before the scheduled departure of his first flight of the day.
(2) He must allow 30 min check-out time after his plane lands.
(3) The minimum stop between flights is 40 min.

(4) The maximum ground duty time is 5 hr.
(5) The minimum stop for food is 50 min.
(6) The maximum duty time is 12 hr.
(7) The maximum number of landings allowed in any 24-hr period is 4.
(8) Only paths that return to home base within 3 days are permitted.

The total number of combinations of flight paths has been found quite rapidly with the use of computer calculations. Even though a typical schedule often has several thousand paths possible, the problem can be solved as an integer program.

3 A QUADRATIC ASSIGNMENT PROBLEM

The dean at a large western university has heard of mathematical programming. He naturally thinks of using this powerful tool for scheduling classes at the university. He calls in several members of the Department of Operations Research and presents them with the problem. After several minutes of thought, the members suggest that the problem be solved as follows: Find $x_{ij} = 0$ or 1 that minimize z when

$$\sum_{j=1}^{n} \sum_{i=1}^{m-1} \sum_{k=i+1}^{m} c_{ik} x_{ij} x_{kj} = z,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, m,$$

where

z is the total scheduling cost,
$x_{ij} = 1$ means course i is assigned at time period j,
$x_{ij} = 0$ means course i is not assigned at time j,
c_{ik} is the cost of assigning courses i and k at the same time,
m is the number of courses, and
n is the number of time periods.

For a nontrivial solution, $m > n$. The costs c_{ik} occur when two courses are scheduled for the same time period and students wish to take both courses. If c_{ik} is taken as the number of students who desire to enroll in

both course i and course k, then z is the total number of students that cannot be accomodated.

The mathematical model is a variation of the quadratic assignment problem [17]; it was developed in the form given above by Carlson and Nemhauser [4], who found local minima. The same model was obtained by Freeman *et al.* [8] in a slightly different context.

We showed in [11] how to convert the problem to an integer program by making the change of variables

$$y_{ik} = \sum_{j=1}^{n} x_{ij} x_{kj}$$

to obtain the equivalent problem: Find $y_{ik} = 0$ or 1 that minimize z when

$$\sum_{i=1}^{m-1} \sum_{k=i+1}^{m} a_{ik} y_{ik} = z,$$

$$\sum_{i=1}^{m-1} \sum_{k=i+1}^{m} y_{ik} \geq \frac{nd(d-1)}{2} + dm_0,$$

$$y_{ik} + y_{ij} - y_{kj} \leq 1,$$

$$y_{ik} + y_{kj} - y_{ij} \leq 1,$$

$$y_{ij} + y_{kj} - y_{ik} \leq 1, \qquad \text{all} \quad i < k < j,$$

where $d = [m/n]$ and $m_0 \equiv m \bmod n$; $[x]$ being the greatest integer less than or equal to x.

When the optimal y_{ik} are found, the x_{ij} are obtained by inspection. For example, if $y_{ik} = 1$, then $x_{ij} = 1$ and $x_{kj} = 1$ for some arbitrary j value. Reference [11] shows how to find the solution to the integer program by enumeration. Any of the methods presented in this book can be used for the same purpose.

4 THE KNAPSACK PROBLEM

We can think of the knapsack problem in two senses. In the first, a given space is to be packed with items of differing value and volume. The problem is to select the most valuable packing. In the second sense,

a given item is to be divided into portions of differing value and our aim is to find the most valuable division of the item.

We can express the knapsack problem as: Find integers $x_j \geq 0$ that maximize z when

$$\sum_{j=1}^{n} \pi_j x_j = z,$$

$$\sum_{j=1}^{n} a_j x_j \leq L,$$

where the π_j are positive numbers, the a_j are positive integers, and L is an integer.

The most interesting application of the knapsack problem occurs in the solution of a large-size linear programming problem. In solving a one-dimensional cutting stock problem as a linear program, Gilmore and Gomory [*9*] use the knapsack solution to generate the pivot column in the simplex method. They employ the inverse matrix method of Chapter 2; the π_i are the simplex multipliers in the inverse matrix. The x_j in the knapsack solution correspond to the a_{ij} values of the original linear programming problem. If maximum z is small enough, then the current basic solution is optimal. Otherwise, the optimal x_j values are used to generate the pivot column and a new basic solution is found.

There are other applications for the knapsack problem: Hanssmann [*13*] for capital investment, Kolesar [*16*] for network reliability, Cord [*5*] for capital budgeting, and Kadane [*15*] for the Neyman–Pearson lemma of statistics.

The knapsack problem was formulated by Bellman and Dreyfus [*3*] as a dynamic programming model. Efficient algorithms for the solution are contained in Gilmore and Gomory [*10*], Shapiro and Wagner [*21*], and the author [*12*, this volume].

5 THE TRAVELING SALESMAN PROBLEM

The traveling salesman starts in one city, goes to n other cities once, and then returns to the first city. How can he determine the order in which to visit each city so that the total distance he covers is at a minimum?

The traveling salesman problem is formulated by Miller *et al.* [20] as an integer program: Find $x_{ij} = 0$ or 1 that minimize z when

$$\sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_j = z,$$

$$\sum_{i=0}^{n} x_{ij} = 1, \qquad j = 0, 1, 2, \ldots, n,$$

$$\sum_{j=0}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, n,$$

$$u_i - u_j + n x_{ij} \leq n - 1, \qquad 1 \leq i \neq j \leq n,$$

where

$x_{ij} = 1$ means that the salesman travels from city i to city j,
$x_{ij} = 0$ means that the salesman does not travel from city i to city j,
d_{ij} is the distance between cities i and j ($d_{ii} = \infty$), and
u_i are arbitrary real numbers.

The first $j \geq 0$ equalities insure that the salesman enters each city once from one of the n other cities. The next n equalities insure that the salesman leaves each city $i \geq 1$ once for one of the n other cities.

A circuit is a tour of $k \leq n + 1$ cities that starts and ends in one city. The last group of inequalities in the problem insures that every circuit contains city 0. Suppose there was a circuit that did not contain city 0 for some integer solution satisfying the equality constraints. We have $x_{ij} = 1$ around the circuit; thus

$$u_i - u_j \leq -1$$

for cities i and j in the circuit. By adding all such inequalities, we obtain $0 \leq -1$, a result which contradicts our supposition. It is clear, then, that if we have any circuit at all, it must be a possible salesman's tour.

What we must demonstrate now is the existence of u_i values that satisfy the inequalities. Take $u_i = t$ if city i is the tth city visited. If $x_{ij} = 0$, then $u_i - u_j + n x_{ij} \leq n - 1$ for all u_i and u_j values. If $x_{ij} = 1$, then $u_i - u_j + n x_{ij} = t - (t + 1) + n = n - 1$. In either case, the inequalities are satisfied. Any feasible solution of the constraints produces a possible tour; the minimization achieves the smallest length tour.

There are other problems in the traveling salesman category. For example, there is the case of sequencing jobs on a machine. We require that a number of jobs be performed successively. After job i is completed, the machine is set up for job j at a cost c_{ij}. The problem is to order the jobs in such a way that the total cost is at a minimum.

6 THE FIXED-CHARGE PROBLEM

Another problem which can be represented in integer programming terms is the fixed-charge problem. For example, a factory has a fixed charge $a > 0$ whenever $x > 0$ items are shipped. It incurs a cost C given by

$$C = \begin{cases} a + cx, & \text{if } x > 0, \\ 0, & \text{if } x = 0. \end{cases}$$

The objective is to minimize the total shipping cost.

The problem is generally formulated as: Find $x_j \geq 0$ that minimize z when

$$\sum_{j=1}^{n} (a_j y_j + c_j x_j) = z,$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m,$$

$$y_j = \begin{cases} 0, & \text{if } x_j = 0, \\ 1, & \text{if } x_j > 0, \end{cases}$$

where $a_j > 0$. Our method of solving the problem is to convert it to a mixed integer format by determining some upper bound d_j for each x_j, and replacing the last group of constraints by

$$x_j \leq d_j y_j, \qquad y_j = 0 \quad \text{or} \quad 1.$$

These inequalities occur because if $x_j > 0$, then y_j must be one, while if $x_j = 0$, then y_j must be zero since $y_j = 1$ produces a higher z value. If we restrict the x_j to be integers, the problem becomes an integer program.

The fixed-charge problem is discussed by Hirsch and Dantzig [*14*]. Other applications are by Balinski [*1*] for transportation problems, Balinski and Wolfe [*2*] and Manne [*18*] for plant location problems, and Weaver and Hess [*22*] in apportionment cases.

7 NONLINEAR APPROXIMATION

Nonlinear functions of a single variable may be approximated in a mixed integer format. Let the function be $f(x)$; the curve of $f(x)$ passes through the points $P_0, P_1, \ldots, P_k$, where the coordinates of points P_i are $(a_i, f(a_i)) = (a_i, b_i)$ with $a_i < a_{i+1}$. The linear interpolation between points P_i and P_{i+1} is used to approximate $f(x)$ for $a_i \leq x \leq a_{i+1}$. Thus we replace x and $f(x)$ by

$$\begin{aligned} x &= \lambda_0 a_0 + \lambda_1 a_1 + \cdots + \lambda_k a_k, \\ f(x) &= \lambda_0 b_0 + \lambda_1 b_1 + \cdots + \lambda_k b_k, \\ 1 &= \lambda_0 \quad + \lambda_1 \quad + \cdots + \lambda_k, \end{aligned}$$

where $\lambda_i \geq 0$. This approximation is the same one used in separable programming. For example, see Miller [*19*].

Since the interpolation is between points P_i and P_{i+1}, we need to impose the conditions that all $\lambda_j = 0$ except for one pair λ_i and λ_{i+1}. This is accomplished by defining integer variables $\delta_i = 0$ or 1. If $\delta_i = 1$, the interval between P_i and P_{i+1} is considered. If $\delta_i = 0$, some other region is considered. Hence, if we take the $k = 4$ case, we have

$$\begin{aligned} \lambda_0 &\leq \delta_0, \\ \lambda_1 &\leq \delta_0 + \delta_1, \\ \lambda_2 &\leq \quad \delta_1 + \delta_2, \\ \lambda_3 &\leq \qquad \delta_2 + \delta_3, \\ \lambda_4 &\leq \qquad\quad \delta_3, \end{aligned}$$

where

$$\delta_0 + \delta_1 + \delta_2 + \delta_3 = 1.$$

To illustrate the approximation in nonlinear programming, consider the problem: Find $x_j \geq 0$ that minimize z when

$$\sum_{j=1}^{n} f_j(x_j) = z,$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m.$$

We replace each $f_j(x_j)$ as above and obtain an approximate mixed integer problem in terms of continuous variables $\lambda_{rj} \geq 0$ and integer variables $\delta_{rj} \geq 0$. This and other methods for nonlinear approximation are given in Dantzig [7].

8 DICHOTOMIES

A *dichotomy* occurs in a mathematical programming problem when there is an "either–or" type of constraint. It can be expressed in integer programming terms. Assume we have the problem of coloring a map with four colors. Let the regions of the map be $r = 1, 2, \ldots, R$ and integer t_r one of four possible values; thus $t_r = 0, 1, 2, 3$. The values correspond to the four colors. If regions r and s have a common boundary, they are colored differently. Hence,

$$t_r - t_s \neq 0,$$

and *either*

$$t_r - t_s \geq 1$$

or

$$t_s - t_r \geq 1.$$

We introduce the integer variable $\delta_{rs} = 0, 1$. Thus,

$$t_r - t_s \geq 1 - 4\delta_{rs},$$
$$t_s - t_r \geq -3 + 4\delta_{rs}$$

replaces the dichotomy.

In general terms, suppose k of the conditions

$$\begin{aligned} G_1(x) &\geq 0, \\ G_2(x) &\geq 0, \\ &\vdots \\ G_p(x) &\geq 0 \end{aligned}$$

must hold simultaneously, where $x = (x_1, x_2, \ldots, x_n)$ is defined in some region S. We replace the alternatives by

$$\begin{aligned} G_1(x) - \delta_1 L_1 &\geq 0, \\ G_2(x) - \delta_2 L_2 &\geq 0, \\ &\vdots \\ G_p(x) - \delta_p L_p &\geq 0, \end{aligned}$$

where L_i is a lower bound for $G_i(x)$ for x in S and $\delta_i = 0$ or 1 satisfies

$$\delta_1 + \delta_2 + \cdots + \delta_p = p - k.$$

Thus for the four color problem where $k = 1$, $p = 2$, we have $L_r = -4$ and $L_s = -4$, which produces the result

$$\begin{aligned} t_r - t_s - 1 + 4\delta_{rs} &\geq 0, \\ t_s - t_r - 1 + 4\delta_{sr} &\geq 0 \end{aligned}$$

with

$$\delta_{rs} + \delta_{sr} = 1.$$

References

1. Balinski, M. L., Fixed-cost transportation problems, *Naval Res. Logist. Quart.* **8**, 41 (1961).
2. Balinski, M. L., and Wolfe, P., On Benders decomposition and the plant location problem, Mathematica Working Paper, **ARO-27**, Dec. 1963.
3. Bellman, R., and Dreyfus, S. E., "Applied Dynamic Programming." Princeton Univ. Press, New Brunswick, New Jersey, 1962.
4. Carlson, R. C., and Nemhauser, G. L., Scheduling to minimize interaction cost, *Operations Res.* **14** (1), 52 (1966).
5. Cord, J., A method for allocating funds to investment projects when returns are subject to uncertainty, *Management Sci.*, **10** (2), 335 (1964).
6. Dantzig, G. B., On the significance of solving linear programming problems with some integer variables, *Econometrica* **28** (1), 30 (1960).
7. Dantzig, G. B., "Linear Programming and Extensions." Princeton Univ. Press New Brunswick, New Jersey, 1963.
8. Freeman, R. J., Gogerty, D. C., Graves, G. W., and Brooks, R. B. S., A mathematical model of supply support for space operations, *Operations Res.*, **14** (1), 1 (1966).
9. Gilmore, P. C. and Gomory, R. E., A linear programming approach to the cutting-stock problem, *Operations Res.*, **19** (6), 849 (1961).
10. Gilmore, P. C., and Gomory, R. E., The theory and computation of knapsack functions, *Operations Res.* **14** (6), 1045 (1966).
11. Greenberg, H., A quadratic assignment problem without column constraints, *Naval Res. Logist. Quart.*, **16** (3), 417 (1969).
12. Greenberg, H., An algorithm for the computation of knapsack functions, *J. Math. Anal. Appl.* **26** (1), 159 (1969).
13. Hanssmann, F., Operations research in the national planning of underdeveloped countries, *Operations Res.* **9** (2), 230 (1961).
14. Hirsch, W. M., and Dantzig, G. B., The fixed charge problem, RAND Corp., Paper **P-648**, Dec. 1, 1954.
15. Kadane, J. B., Discrete search and the Neyman–Pearson lemma, *J. Math. Anal. Appl.* **22** (1), 156 (1968).
16. Kolesar, P. J., Assignment of optimal redundancy in systems subject to failure, Columbia Univ., Operations Res. Group, Technical Report, 1966.
17. Lawler, E. L., The quadratic assignment problem, *Management Sci.* **9**, 586 (1963).
18. Manne, A. S., Plant location under economics of scale-decentralization and computation, *Management Sci.* 11, 213 (1964).
19. Miller, C. E., The simplex method for local separable programming, *in* "Recent Advances in Mathematical Programming" (R. L. Graves and P. Wolfe, eds.), pp. 89–100. McGraw-Hill, New York, 1963.
20. Miller, C. E., Tucker, A. W., and Zemlin, R. A., Integer programming formulations and traveling salesman problems, *J. Assoc. Comput. Mach.* **7**, 326 (1960).
21. Shapiro, J. F., and Wagner, H. M., A finite renewal algorithm for the knapsack and turnpike models, *Operations Res.* **15**, 319 (1967).
22. Weaver, J. B., and Hess, S. W., A procedure for a non-partisan districting; development of computer techniques, *Yale Law J.* **73**, 288 (1963).

2 LINEAR PROGRAMMING

The theory and methods of linear programming have useful applications in integer programming. Developed largely by Dantzig [*1*], linear programming is an important technique in solving optimization problems when the variables are not integer restricted.

In this chapter we review the concepts of linear programming. These concepts are structured so that similar ideas in integer programming will be more readily grasped.

1 THE GENERAL LINEAR PROGRAM

Linear programming deals with the minimization of a linear function in which the variables are nonnegative and constrained by a system of linear equations. We can express the general linear programming problem as: Find $x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0$ that minimize z when

$$
\begin{aligned}
c_1x_1 + c_2x_2 + \cdots + c_nx_n &= z, \\
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m,
\end{aligned} \tag{1}
$$

and the a_{ij}, b_i, and c_j are given constants. The linear form z is called the *objective function*.

The general linear program is always defined in terms of minimization. When the objective in a given problem is to maximize $z = \Sigma c_j x_j$ we simply convert it to a minimization problem by minimizing $-z$. All maximization problems may thus be treated as minimization problems.

The general linear programming problem is written in terms of equality constraints. There may be, however, problems having inequality constraints. In these cases, we convert the inequalities to equalities.

A linear inequality constraint for the x_j is of the form

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \leq b.$$

To convert the inequality to an equation as in (1), we introduce a *slack variable* $x_{n+1} \geq 0$; thus

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + x_{n+1} = b,$$

and we consider that $c_{n+1} = 0$ in the objective function z.

If we have the inequality form

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \geq b,$$

we introduce a *surplus variable* $x_{n+1} \geq 0$; thus

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n - x_{n+1} = b$$

with $c_{n+1} = 0$. A different slack or surplus variable is of course used for each inequality.

2 RECOGNITION OF OPTIMALITY

In this section we present a linear programming problem in which the optimal solution is recognizable by inspection. This enables us to determine the format for solving *any* linear program.

We consider the problem: Find $x_1 \geq 0$, $x_2 \geq 0$, ..., $z_n \geq 0$ that minimize z when

$$
\begin{aligned}
-z + \bar{c}_{m+1}x_{m+1} + \cdots + \bar{c}_n x_n &= -\bar{z}_0, \\
x_1 + \bar{a}_{1,m+1}x_{m+1} + \cdots + \bar{a}_{1n}x_n &= \bar{b}_1, \\
x_2 + \bar{a}_{2,m+1}x_{m+1} + \cdots + \bar{a}_{2n}x_n &= \bar{b}_2, \\
&\vdots \\
x_m + \bar{a}_{m,m+1}x_{m+1} + \cdots + \bar{a}_{mn}x_n &= \bar{b}_m,
\end{aligned} \tag{2}
$$

with $\bar{a}_{ij}$, $\bar{c}_j$, $\bar{b}_i$, and $\bar{z}_0$ as constants.

The distinguished variables x_i for $i = 1, 2, \ldots, m$ as in Eq. (2) are called *basic variables*. Each basic variable has a coefficient of unity and occurs in one equation only. The variables for x_i for $i = m + 1, m + 2, \ldots, n$ are the *nonbasic variables*. Writing the objective function as in Eq. (2), we have $-z$ as a basic variable.

A system of linear equations, with each equation having one basic variable, again as in (2), is called a *canonical form*. The canonical form helps to define a basic solution to the equations of the linear program. Here the basic solution is $z = \bar{z}_0$, $x_1 = \bar{b}_1$, $x_2 = \bar{b}_2, \ldots, x_m = \bar{b}_m$, $x_{m+1} = 0$, $x_{m+2} = 0, \ldots, x_n = 0$. If, in addition, all $\bar{b}_i \geq 0$, then the x_i are nonnegative; we have a *basic feasible solution* and the linear program is in *feasible canonical form.*

A feasible canonical form provides a basic feasible solution. It also yields an optimal solution to the linear program of Eq. (2) when all $\bar{c}_j \geq 0$. We can show this by the following theorem:

Theorem 1

The basic feasible solution $x_1 = \bar{b}_1, x_2 = \bar{b}_2, \ldots, x_m = \bar{b}_m, x_{m+1} = 0$, $x_{m+2} = 0, \ldots, x_n = 0$ is a minimal solution with the objective value $z = \bar{z}_0$ if $\bar{c}_j \geq 0$ for $j = 1, 2, \ldots, n$.

Proof

If all $\bar{c}_j \geq 0$, then the smallest possible value of $\Sigma \bar{c}_j x_j$ is zero for any values of $x_j \geq 0$. The minimum possible value of $z - \bar{z}_0$ is thus zero and we can have only $\min z \geq \bar{z}_0$ when we consider the restrictions of

Eq. (2). We have produced a basic feasible solution, however, with $z = \bar{z}_0$; thus, min $z = \bar{z}_0$ and the solution is optimal. Note that any $x_j \geq 0$ for $j = m + 1, m + 2, \ldots, n$ can produce only the same or higher values of z. A solution with the property that $x_j > 0$ for $\bar{c}_j > 0$ is not minimal.

To obtain a solution to the problem in Eq. (1), we transform the equations to a form given by (2). We accomplish this with the simplex method, which we present in Section 3. To facilitate the transformation, write the objective function as

$$-z + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n = 0.$$

When we arrive at the optimal solution for (2), we see that the identical solution is optimal for (1).

Theorem 2

The minimal feasible solution $x_1 = \bar{b}_1$, $x_2 = \bar{b}_2$, $\ldots$, $x_m = \bar{b}_m$, $x_{m+1} = 0, x_{m+2} = 0, \ldots, x_n = 0, z = \bar{z}_0$ of Eq. (2) is also a minimal feasible solution of (1), where (2) is the given linear transformation of (1).

Proof

Any feasible solution of (2) also satisfies (1) and vice versa. If there exists a minimal solution of (1) with $z < \bar{z}_0$, the solution also satisfies (2). But this cannot be true because $\bar{z}_0$ is the minimum possible value of z for (2). Thus, we obtain a minimal feasible solution for (1) given by $x_1 = \bar{b}_1, x_2 = \bar{b}_2, \ldots, x_m = \bar{b}_m, x_{m+1} = 0, x_{m+2} = 0, \ldots, x_n = 0, z = \bar{z}_0$. When (2) is achieved we have the solution of linear program (1) by inspection.

In addition to the criterion of Theorem 1, if $\bar{c}_j = 0$ for a nonbasic variable, then x_j may be made into a basic variable and the value of z will remain unchanged. In this case we would have minimal solutions that are nonunique. Thus

Theorem 3

If x_j^t, $t = 1, 2, \ldots, k$ represent minimal feasible solutions to a linear program, then $x_j^* = \sum_{t=1}^{k} \lambda_t x_j^t$ is also a minimal solution for $j = 1, 2, \ldots, n$, where λ_t are any nonnegative constants satisfying $\sum_{t=1}^{k} \lambda_t = 1$.

Proof

Since the minimal solution is nonunique we must have objective value

$$z_0 = \sum_{j=1}^{n} c_j x_j^t \qquad \text{for} \quad t = 1, 2, \ldots, k.$$

The objective value given by

$$x_j^* = \sum_{t=1}^{k} \lambda_t x_j^t$$

is then

$$\begin{aligned} z &= \sum_{j=1}^{n} c_j \sum_{t=1}^{k} \lambda_t x_j^t \\ &= \sum_{t=1}^{k} \lambda_t \sum_{j=1}^{n} c_j x_j^t \\ &= \sum_{t=1}^{k} \lambda_t z_0 = z_0 . \end{aligned}$$

Thus the x_j^* produce the same objective value as the x_j^t. Furthermore, each x_j^t and λ_t are nonnegative; it follows that $x_j^* \geq 0$ for every j. Also, since

$$\sum_{j=1}^{n} a_{ij} x_j^t = b_i$$

we have

$$\sum_{t=1}^{k} \lambda_t \sum_{j=1}^{n} a_{ij} x_j^t = \sum_{t=1}^{k} \lambda_t b_i$$

and consequently

$$\sum_{j=1}^{n} a_{ij} x_j^* = b_i .$$

The x_j^* satisfy the constraints for any constants λ_t where $\lambda_t \geq 0$ and $\sum_{t=1}^{k} \lambda_t = 1$ and are a minimal feasible solution. In the nonunique case there are an infinite number of optimal solutions since any nonnegative λ_t summing to one produce optimal solutions. This selection of λ_t can be made in an infinite number of ways.

Observe that we have written Eq. (2) with the basic variables $x_1, x_2, \ldots, x_m$. We have selected these particular variables to illustrate the final form desired, but it should be understood that in a given problem other variables may be basic.

Thus far we have developed the canonical form and shown when we have an optimal solution. Theorem 1 provided the optimality criterion for a feasible canonical format; a minimal solution resulted when all $\bar{c}_j$ were nonnegative.

We now ask: When does a feasible canonical format not produce an optimal solution? Clearly, when one or more of the $\bar{c}_j$ are negative. In such a case, we can attempt to reduce the value of z by allowing nonbasic x_j to have a value. Thus x_j is made into a basic variable and a new canonical form is developed. If the new canonical form is still nonoptimal, we repeat the process; eventually, we will obtain the optimal solution.

In particular, suppose $\bar{c}_s$ is negative. Let x_s have a positive value while keeping all the other nonbasic variables at zero. To see the effect of increasing only x_s we write Eq. (2) in terms of the basic variables and x_s to obtain.

$$
\begin{aligned}
z &= \bar{z}_0 + \bar{c}_s x_s, \\
x_1 &= \bar{b}_1 - \bar{a}_{1s} x_s, \\
x_2 &= \bar{b}_2 - \bar{a}_{2s} x_s, \\
&\vdots \\
x_m &= \bar{b}_m - \bar{a}_{ms} x_s,
\end{aligned} \tag{3}
$$

where $\bar{b}_i \geq 0$ for $i = 1, 2, \ldots, m$. Since $\bar{c}_s$ is negative we readily see that z will be decreased in value if x_s is made positive.

In attempting to reduce the value of z, it is also advantageous to make x_s as large as possible. We should allow x_s to achieve some value

as long as the basic variables x_i for $i = 1, 2, \ldots, m$ remain nonnegative. If $\bar{a}_{is}$ is negative, then increasing x_s will also increase the basic variable x_i. If $\bar{a}_{is}$ is positive, x_s at this stage can only be increased to $\bar{b}_i/\bar{a}_{is}$. If x_s is given any larger value, x_i would be negative. Thus it is obvious that the greatest amount x_s can be increased is

$$x_s = \frac{\bar{b}_r}{\bar{a}_{rs}} = \min_{\bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}. \tag{4}$$

Further, if all $\bar{a}_{is} \leq 0$, then x_s can be made as large as possible, making z arbitrarily small. Thus

Theorem 4

Assume the canonical form, Eq. (2), in which $\bar{c}_s < 0$ and all $\bar{a}_{is} \leq 0$ for some index s. There then exists a class of programs with positive basic variables x_i for $i = 1, 2, \ldots, m$ where the nonbasic variable x_s may take on increasing nonnegative values and where, as a result, the set of z values has no lower bound.

If the conditions of Theorem 4 hold, the solution is unbounded and no finite minimal program exists. If at least one $\bar{a}_{is}$ is positive, however, the value of z will become

$$z = \bar{z}_0 + \bar{c}_s \frac{\bar{b}_r}{\bar{a}_{rs}} \leq \bar{z}_0, \tag{5}$$

and x_s will have a value given by Eq. (4). Equation (4) also acts as a choice rule for determining the index r.

The possible reduction of z in (5) indicates that x_s be made a basic variable and x_r be made nonbasic. We do this by dividing each term in the rth constraint of (2) by $\bar{a}_{rs}$ forming the equation

$$x_s + \frac{1}{\bar{a}_{rs}} x_r + \sum_{\substack{j=m+1 \\ j \neq s}}^{n} \frac{\bar{a}_{rj}}{\bar{a}_{rs}} x_j = \frac{\bar{b}_r}{\bar{a}_{rs}}. \tag{6}$$

We then eliminate x_s from the other equations of (2). Thus, to obtain the coefficient of x_j in the ith constraint, we multiply the coefficient of x_j in (6) by $\bar{a}_{is}$ and subtract the result from $\bar{a}_{ij}$. To obtain the coefficient of x_j in the z equation, we multiply the coefficient of x_j in (6) by $\bar{c}_s$ and subtract the result from $\bar{c}_j$. In this way we obtain the new feasible canonical form

$$
\begin{aligned}
&-z - \frac{\bar{c}_s}{\bar{a}_{rs}} x_r + \sum_{j=m+1}^{n} \left(\bar{c}_j - \frac{\bar{c}_s \bar{a}_{rj}}{\bar{a}_{rs}} \right) x_j = -\bar{z}_0 - \frac{\bar{c}_s \bar{b}_r}{\bar{a}_{rs}} \\
&x_i - \frac{\bar{a}_{is}}{\bar{a}_{rs}} x_r + \sum_{j=m+1}^{n} \left(\bar{a}_{ij} - \frac{\bar{a}_{is} \bar{a}_{rj}}{\bar{a}_{rs}} \right) x_j = \bar{b}_i - \frac{\bar{a}_{is} \bar{b}_r}{\bar{a}_{rs}}, \qquad i \neq r,
\end{aligned} \tag{7}
$$

with (6) as the additional equation. Equation (6) defines x_s as a basic variable. This procedure for producing the new canonical form is called *pivoting*; $\bar{a}_{rs}$ is called the *pivot element*. The elements $\bar{a}_{is}$ make up the *pivot column*.

The new feasible canonical form may have a smaller objective value than in (2) as seen by (5). The objective value does remain the same if $\bar{b}_r = 0$. Also, if $\bar{b}_r$ is zero, the basic solution has $x_r = 0$ in (2).

A basic solution is called *degenerate* if one or more of the basic variables has zero value. Thus, with the use of choice rule (4) and the development of (5), (6), and (7), we can state

Theorem 5

Given the feasible canonical form Eq. (2), in which $\bar{c}_s < 0$ and at least one $\bar{a}_{is}$ is positive, a new basic feasible solution can be constructed. If, in addition, the original canonical form is nondegenerate, the resultant objective value will be smaller in the new system.

In ordinary computational work when more than one $\bar{c}_j$ is negative, the practice is to select the index s where

$$
\begin{aligned}
\bar{c}_s &= \min \bar{c}_j \\
&< 0.
\end{aligned} \tag{8}
$$

We use choice rule (8) here because of its simplicity, but other rules may result in faster solutions to linear programs. Reference [4] gives the experimental results for various choice rules.

The altering of the canonical form Eq. (2), may not necessarily produce an optimal format in (6) and (7). In that event we know that at least one of the coefficients of the nonbasic variables in the objective function of (7) is negative. We repeat the procedure to obtain a new feasible canonical form, continuing until we have an optimal solution as expressed by Theorem 1 or until the solution is shown to be unbounded by Theorem 4. This method of solution constitutes the simplex algorithm.

3 THE SIMPLEX METHOD

The general linear program in Eq. (1) is presented without a basic feasible solution. But we have seen that this basic solution is essential; the corresponding canonical form enables us to recognize and to achieve optimality. If we arbitrarily select any of the m variables of (1) as basic variables, we may not produce a feasible solution because some of the variables may turn out to be negative. It is necessary that we have a method that insures the attainment of a basic feasible solution.

The simplex method starts with the general program (1). We introduce artificial variables into the general program to achieve a canonical form. The simplex algorithm is then used to achieve a feasible canonical form if one exists. When the feasible canonical form is obtained, the simplex algorithm is used to achieve the minimal canonical form if one exists.

Simplex Method

(A) Arrange the system of equations in (1) so that all b_i are nonnegative by multiplying, where necessary, any equation through by minus one.

(B) Introduce a set of variables that function initially as basic variables. These variables, not in the original equations, are called *artificial variables* and are later discarded. Thus we augment the system by the artificial variables $x_{n+1} \geq 0$, $x_{n+2} \geq 0$, $\ldots$, $x_{n+m} \geq 0$ so that we now have

$$
\begin{aligned}
c_1x_1 + c_2x_2 + \cdots + c_nx_n - z &= 0,\\
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + x_{n+1} &= b_1,\\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + x_{n+2} &= b_2,\\
&\vdots\\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + x_{n+m} &= b_m.
\end{aligned} \tag{9}
$$

(C) Define w as the sum of the artificial variables and write the *infeasibility form*

$$x_{n+1} + x_{n+2} + \cdots + x_{n+m} = w. \tag{10}$$

By summing the last m equations of (9) we can eliminate the artificial variables from (10) to obtain

$$d_1x_1 + d_2x_2 + \cdots + d_nx_n - w = -w_0, \tag{11}$$

where

$$
\begin{aligned}
d_j &= -(a_{1j} + a_{2j} + \cdots + a_{mj}), \qquad j = 1, 2, \ldots, n\\
-w_0 &= -(b_1 + b_2 + \cdots + b_m).
\end{aligned}
$$

The simplex algorithm is used in Phase I to minimize w to zero subject to (9) and $x_j \geq 0$ for $j = 1, 2, \ldots, n, n+1, \ldots, n+m$. Initially we treat $-w$, $-z$, $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ as basic variables. We eliminate the artificial variables from (10) to form (11) and obtain a feasible canonical form. At the conclusion of this phase, w can only be zero when all the artificial variables are zero, leaving usually m of the variables x_j for $j = 1, 2, \ldots, n$ as basic variables.

The Simplex Algorithm:

At the start of each iteration of the simplex algorithm the equations will be of the form

$$
\begin{aligned}
\bar{d}_1 x_1 + \bar{d}_2 x_2 + \cdots + \bar{d}_n x_n - w &= -\bar{w}_0, \\
\bar{c}_1 x_1 + \bar{c}_2 x_2 + \cdots + \bar{c}_n x_n - z &= -\bar{z}_0, \\
\bar{a}_{11} x_1 + \bar{a}_{12} x_2 + \cdots + \bar{a}_{1n} x_n + x_{s_1} &= \bar{b}_1, \\
\bar{a}_{21} x_1 + \bar{a}_{22} x_2 + \cdots + \bar{a}_{2n} x_n + x_{s_2} &= \bar{b}_2, \\
&\vdots \\
\bar{a}_{m1} x_1 + \bar{a}_{m2} x_2 + \cdots + \bar{a}_{mn} x_n + x_{s_m} &= \bar{b}_m,
\end{aligned} \tag{12}
$$

where $-w$, $-z$, x_{s_1}, x_{s_2}, ..., x_{s_m} are basic variables. Of course, if $s_i = j$, then x_j only appears in the ith equation of (12), with unit coefficient. Also we drop artificial variables from further consideration (i.e., do not carry them in the equations) when they become nonbasic.

1. Phase I: If all $\bar{d}_j \geq 0$ and
   (a) $\bar{w}_0 > 0$, end. No feasible solution exists.
   (b) $\bar{w}_0 = 0$, start Phase II by dropping all variables x_j with $\bar{d}_j > 0$. Drop the w equation.

   Otherwise, for some j we have $\bar{d}_j < 0$; choose x_s to become basic in the next iteration where $\bar{d}_s = \min \bar{d}_j < 0$.

   Phase II: If all $\bar{c}_j \geq 0$, end. A minimal solution is given by $x_{s_i} = \bar{b}_i$, $x_j = 0$, $z = \bar{z}_0$ for $j \neq s_i$ and $i = 1, 2, \ldots, m$. Otherwise, for some j we have $\bar{c}_j < 0$; choose x_s to become basic in the next iteration where $\bar{c}_s = \min \bar{c}_j < 0$.
2. If all $\bar{a}_{is} \leq 0$, end; the solution is unbounded. Otherwise, some $\bar{a}_{is} > 0$; the variable x_{s_r} is made nonbasic in the next iteration where index r is determined by

$$
\frac{\bar{b}_r}{\bar{a}_{rs}} = \min_{\bar{a}_{is} > 0} \frac{\bar{b}_i}{\bar{a}_{is}}.
$$

   In case of ties, choose r at random from those i which are tied, i.e., if T of the i are tied, make some selection scheme so that each has chance $1/T$ of being chosen.

3. Develop a new canonical form from (12), making x_s a basic variable. The new system of equations is obtained by using $\bar{a}_{rs}$ as the pivot element. Divide both sides of the rth equation by $\bar{a}_{rs}$ so that the resultant equivalent equation is

$$\sum_{j \neq s} \frac{\bar{a}_{rj}}{\bar{a}_{rs}} x_j + x_s = \frac{\bar{b}_r}{\bar{a}_{rs}}. \tag{13}$$

Equation (13) defines x_s as a basic variable. We eliminate x_s from the other equations of (12) to form

$$\begin{aligned}
&\sum_{j=1}^{n} \left(\bar{d}_j - \bar{d}_s \frac{\bar{a}_{rj}}{\bar{a}_{rs}} \right) x_j - w = -\bar{w}_0 - \bar{d}_s \frac{\bar{b}_r}{\bar{a}_{rs}}, \\
&\sum_{j=1}^{n} \left(\bar{c}_j - \bar{c}_s \frac{\bar{a}_{rj}}{\bar{a}_{rs}} \right) x_j - z = -\bar{z}_0 - \bar{c}_s \frac{\bar{b}_r}{\bar{a}_{rs}}, \\
&\sum_{j=1}^{n} \left(\bar{a}_{ij} - \bar{a}_{is} \frac{\bar{a}_{rj}}{\bar{a}_{rs}} \right) x_j + x_{s_i} = \bar{b}_i - \bar{a}_{is} \frac{\bar{b}_r}{\bar{a}_{rs}}, \qquad i \neq r, \\
&\sum_{j \neq s} \frac{\bar{a}_{rj}}{\bar{a}_{rs}} x_j + x_s = \frac{\bar{b}_r}{\bar{a}_{rs}}.
\end{aligned} \tag{14}$$

Obviously, the elimination of x_s in the w equation is necessary only during Phase I. When the constants have been properly relabeled, then Eq. (14) is exactly the form given in (12). We return to step 2 for the next iteration.

The use of the Phase I procedure will indicate when a system of equations has redundancies or inconsistencies, or when it is not solvable in nonnegative values. For example, the inconsistent inequalities

$$\begin{aligned}
x_1 - x_2 &\geq 3, \\
-x_1 + x_2 &\geq 4, \\
x_1 \geq 0, \qquad x_2 &\geq 0
\end{aligned}$$

produce the Phase I problem:

$$\begin{aligned}
-w \qquad\quad + x_3 + x_4 \qquad\quad &= -7, \\
x_1 - x_2 - x_3 \qquad + x_5 \qquad &= 3, \\
-x_1 + x_2 \qquad - x_4 \qquad + x_6 &= 4,
\end{aligned}$$

where x_3 and x_4 are surplus variables and x_5 and x_6 are artificial variables. Thus, all $\bar{d}_j \geq 0$ and $\bar{w}_0 > 0$; w cannot be reduced to zero and no feasible solution exists.

In another example showing the effectiveness of Phase 1, consider

$$\begin{aligned} -w - 6x_1 + 2x_2 + 3x_3 &= -6, \\ 3x_1 - x_2 - 2x_3 + x_4 &= 3, \\ 3x_1 - x_2 - x_3 + x_5 &= 3, \\ x_j \geq 0, \quad j &= 1, 2, \ldots, 5, \end{aligned}$$

where x_4 and x_5 are artificial variables. To make x_1 a basic variable, use the coefficient of x_1 in the third equation as pivot element to obtain the equations

$$\begin{aligned} -w + x_3 &= 0, \\ -x_3 + x_4 &= 0, \\ x_1 - \tfrac{1}{3}x_2 - \tfrac{1}{3}x_3 &= 1. \end{aligned}$$

Note that x_5 has been dropped; also that $\bar{w}_0 = 0$ and Phase I has ended. Since $\bar{d}_3 = 1 > 0$, we must take $x_3 = 0$ to make the equations consistent. Otherwise, x_3 may attain a positive value during Phase II. Observe also that the second equation defines the artificial x_4 as a basic variable but with zero value.

The example shows that Phase 1 may end with an artificial basic variable. At the end of the phase we have

$$\bar{d}_1 x_1 + \bar{d}_2 x_2 + \cdots + \bar{d}_{m+n} x_{m+n} = w, \tag{15}$$

where $\bar{d}_j \geq 0$. If any $\bar{d}_j > 0$, we take the corresponding $x_j = 0$. The only possible way for an x_j to have a positive value is when $\bar{d}_j = 0$. If any of the artificial x_j have a positive value, then w has a positive value from Eq. (10); this contradicts $w = 0$ from (15). Thus we have

Theorem 6

If an artificial variable is basic in Phase II, its value will never exceed zero.

Example for the Simplex Algorithm

Find $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ that minimize z when

$$\begin{aligned} 2x_1 - 3x_2 - x_3 + 2x_4 &= z, \\ -3x_1 + 2x_2 - x_3 + 3x_4 &= 2, \\ -\ x_1 + 2x_2 + x_3 + 2x_4 &= 3, \end{aligned}$$

Take the artificial variables $x_5 \geq 0$ and $x_6 \geq 0$ and start Phase I by writing

$$\begin{aligned} -w \quad + 4x_1 - 4x_2 \quad - 5x_4 \qquad\qquad &= -5, \\ -z + 2x_1 - 3x_2 - x_3 + 2x_4 \qquad\qquad &= 0, \\ -\ 3x_1 + 2x_2 - x_3 + 3x_4 + x_5 \qquad &= 2, \\ -\ x_1 + 2x_2 + x_3 + 2x_4 \qquad + x_6 &= 3. \end{aligned}$$

The initial basic feasible solution is $-w = -5$, $-z = 0$, $x_5 = 2$, $x_6 = 3$. The most negative coefficient in the w equation is $\bar{d}_4 = -5$; we make x_4 basic. Since $s = 4$ and $\min(\frac{2}{3}, \frac{3}{2}) = \frac{2}{3}$, we see that $r = 1$; x_5 is made nonbasic and 3 is the pivot element. We obtain

$$\begin{aligned} -w \quad - \ x_1 - \tfrac{2}{3}x_2 - \tfrac{5}{3}x_3 \qquad\qquad &= -\tfrac{5}{3}, \\ -z + 4x_1 - \tfrac{13}{3}x_2 - \tfrac{1}{3}x_3 \qquad\qquad &= -\tfrac{4}{3}, \\ -\ x_1 + \tfrac{2}{3}x_2 - \tfrac{1}{3}x_3 + x_4 \qquad &= \tfrac{2}{3}, \\ x_1 + \tfrac{2}{3}x_2 + \tfrac{5}{3}x_3 \qquad + x_6 &= \tfrac{5}{3} \end{aligned}$$

as the new canonical form. The artificial variable x_5 has been dropped. The basic solution is $-w = -\frac{5}{3}$, $-z = -\frac{4}{3}$, $x_4 = \frac{2}{3}$, $x_6 = \frac{5}{3}$. We have not reduced w to zero and so we repeat the algorithm. The most negative coefficient in the w equation is $\bar{d}_3 = -\frac{5}{3}$; we make x_3 basic. We see that $\frac{5}{3}$ in the last equation is the only positive coefficient of x_3 for the constraint equations. Thus, artificial x_6 is made nonbasic and dropped. Using $\frac{5}{3}$ as the pivot element, we obtain the new form

$$\begin{aligned} -w \qquad\qquad\qquad\qquad &= 0 \\ -z + \tfrac{21}{5}x_1 - \tfrac{21}{5}x_2 \qquad\qquad &= -1 \\ -\ \tfrac{4}{5}x_1 + \tfrac{4}{5}x_2 \qquad + x_4 &= 1 \\ \tfrac{3}{5}x_1 + \tfrac{2}{5}x_2 + x_3 \qquad &= 1. \end{aligned}$$

We have $w = 0$ and Phase I ends. We start Phase II by dropping the w equation. The basic solution is $-z = -1$, $x_4 = 1$, $x_3 = 1$. The most negative coefficient in the z equation is $\bar{c}_2 = -\frac{21}{5}$; we make x_2 basic and x_4 nonbasic. Using $\frac{4}{5}$ as the pivot element, we have

$$\begin{aligned} -z \qquad\qquad &+ \tfrac{21}{4}x_4 = \tfrac{17}{4}, \\ -x_1 + x_2 \qquad &+ \tfrac{5}{4}x_4 = \tfrac{5}{4}, \\ x_1 + x_3 - &\tfrac{1}{2}x_4 = \tfrac{1}{2}. \end{aligned}$$

We see that all $\bar{c}_j \geq 0$. Thus, the basic solution is minimal; $z = -\frac{17}{4}$, $x_2 = \frac{5}{4}$, and $x_3 = \frac{1}{2}$ is optimal for the problem. Note that $\bar{c}_1 = 0$; x_1 can be made basic to produce alternate minima.

4 TABLEAU FORM

In demonstrating the simplex method in Section 3 we used the equations of the problem. For ease of computation, we write the equations in tableau form using only the constants of the problem. At the start of the simplex method, the tableau equivalent to (9) with (11) we will call Tableau 1.

Basic Variables	x_1	x_2	x_3	$\cdots$	x_n	Equality Constants
$-w$	d_1	d_2	d_3	$\cdots$	d_n	$-w_0$
$-z$	c_1	c_2	c_3	$\cdots$	c_n	0
x_{n+1}	a_{11}	a_{12}	a_{13}	$\cdots$	a_{1n}	b_1
x_{n+2}	a_{21}	a_{22}	a_{23}	$\cdots$	a_{2n}	b_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
x_{n+m}	a_{m1}	a_{m2}	a_{m3}	$\cdots$	a_{mn}	b_m

Tableau 1

The tableau omits the x_{n+1}, x_{n+2}, $\ldots$, x_{n+m}, $-z$ and $-w$ columns. The values in each of these columns is zero or one, except when an artificial variable becomes nonbasic, in which case it is dropped.

At the start of any iteration, we have a tableau with constants taken from (12). The basic variables are listed at the left column of each tableau. The next tableau contains the constants from (14) resulting from the pivot operation. The difference between the tableaus in Phase I and Phase II is that the former contains the w row while the latter does not.

Example

To illustrate the use of the tableau form, we solve the example problem of Section 2 again. The initial and subsequent tableaus are written as Tableaus E1–E4. The minimal solution appears in Tableau E4.

	x_1	x_2	x_3	x_4	
$-w$	4	−4	0	−5	−5
$-z$	2	−3	−1	2	0
x_5	−3	2	−1	3	2
x_6	−1	2	1	2	3

Tableau E1

	x_1	x_2	x_3	x_4	
$-w$	−1	$-\frac{2}{3}$	$-\frac{5}{3}$	0	$-\frac{5}{3}$
$-z$	4	$-1\frac{3}{3}$	$-\frac{1}{3}$	0	$-\frac{4}{3}$
x_4	−1	$\frac{2}{3}$	$-\frac{1}{3}$	1	$\frac{2}{3}$
x_6	1	$\frac{2}{3}$	$\frac{5}{3}$	0	$\frac{5}{3}$

Tableau E2

	x_1	x_2	x_3	x_4	
$-z$	$2\frac{1}{5}$	$-2\frac{1}{5}$	0	0	−1
x_4	$-\frac{4}{5}$	$\frac{4}{5}$	0	1	1
x_3	$\frac{3}{5}$	$\frac{2}{5}$	1	0	1

Tableau E3

	x_1	x_2	x_3	x_4	
$-z$	0	0	0	$2\frac{1}{4}$	$1\frac{7}{4}$
x_2	−1	1	0	$\frac{5}{4}$	$\frac{5}{4}$
x_3	1	0	1	$-\frac{1}{2}$	$\frac{1}{2}$

Tableau E4

5 THE INVERSE MATRIX METHOD

In the simplex method it is necessary to calculate the coefficients from each equation in every iteration. We find the smallest $\bar{d}_j$ or $\bar{c}_j$ value from either the w or z objective function to select the pivot column and use

only the $\bar{a}_{ij}$ values from the column. The remaining constants are then changed in pivoting for possible use in a subsequent iteration.

It is possible, however, to calculate the coefficients in the equations only when they are needed if we use the *inverse matrix method*, sometimes known as the *revised simplex method.*

The inverse matrix method can best be described with matrix notation. We present first the method for Phase II of the simplex procedure. The constraints and objective function of (1) can be written as

$$\begin{pmatrix} 1 & c \\ 0 & A \end{pmatrix} \begin{pmatrix} -z \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}, \tag{16}$$

where c is an n-vector with components c_j, A is an $m \times n$ matrix with elements a_{ij}, b is an m-vector with components b_i and x is an n-vector with components x_j. Suppose we can achieve a canonical form similar to that in (2), where x_G represents the basic variables and is a vector with m of the components of x; x_H represents the nonbasic variables and is a vector made up of the remaining components of x. Thus, we may write (16) as

$$\begin{pmatrix} 1 & c_G & c_H \\ 0 & A_G & A_H \end{pmatrix} \begin{pmatrix} -z \\ x_G \\ x_H \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}, \tag{17}$$

where c_G and c_H are vectors with components that correspond to x_G and x_H respectively. Similarly, A_G and A_H are matrices with elements that correspond to x_G and x_H respectively. Since we can solve (17) for the basic variables $-z$ and x_G, the matrix

$$B = \begin{pmatrix} 1 & c_G \\ 0 & A_G \end{pmatrix}$$

has an inverse, shown as

$$B^{-1} = \begin{pmatrix} 1 & -c_G A_G^{-1} \\ 0 & A_G^{-1} \end{pmatrix}. \tag{18}$$

Premultiplying (17) by B^{-1}, we obtain the canonical form

$$\begin{pmatrix} 1 & 0 & c_H - c_G A_G^{-1} A_H \\ 0 & I & A_G^{-1} A_H \end{pmatrix} \begin{pmatrix} -z \\ x_G \\ x_H \end{pmatrix} = \begin{pmatrix} -c_G A_G^{-1} b \\ A_G^{-1} b \end{pmatrix}, \tag{19}$$

where I is an $m \times m$ identity matrix. The canonical form of (19) is the same as the canonical form achieved by the series of pivot operations in the simplex method.

After the development of (19), we see how to perform the simplex method with the inverse matrix B^{-1}. First we store the c, A, and b values of (16). We list the basic variables x_G and the inverse matrix B^{-1} in every iteration and calculate $\bar{c}_j$, the coefficient of nonbasic x_j in the current objective function. Stating B^{-1} from (18) as

$$B^{-1} = \begin{pmatrix} 1 & -\pi_1 & -\pi_2 & \cdots & -\pi_m \\ 0 & b_{11} & b_{12} & \cdots & b_{1m} \\ 0 & b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & b_{m1} & b_{m2} & \cdots & b_{mm} \end{pmatrix} \tag{20}$$

we see from (19) that $\bar{c}_j$ is given by

$$\bar{c}_j = c_j - \sum_{i=1}^{m} \pi_i a_{ij}.$$

The π_i are often called *simplex multipliers*.

When $\bar{c}_s = \min \bar{c}_j < 0$ is obtained to produce the pivot column, we calculate only the $\bar{a}_{is}$ values; no $\bar{a}_{ij}$ for $j \neq s$ values are needed. We have

$$\bar{a}_{is} = \sum_{k=1}^{m} b_{ik} a_{ks}.$$

Similarly,

$$\bar{z}_0 = \sum_{i=1}^{m} \pi_i b_i,$$

$$\bar{b}_i = \sum_{k=1}^{m} b_{ik} b_k.$$

We use the standard procedure to obtain the pivot element $\bar{a}_{rs}$.

The inverse matrix is available for calculations if the matrix A contains an identity matrix which is taken as B^{-1} initially. If A does not contain an identity matrix, then B^{-1} is at first arbitrarily taken as the identity matrix.

This is seen by introducing some new variable vector y to change (17) to

$$\begin{pmatrix} 1 & c_G & c_H \\ & & \\ 0 & A_G & A_H \end{pmatrix} \begin{pmatrix} -z \\ x_G \\ x_H \end{pmatrix} + Iy = \begin{pmatrix} 0 \\ \\ b \end{pmatrix}. \tag{21}$$

Premultiplying (21) by B^{-1}, we obtain the canonical form

$$\begin{pmatrix} 1 & 0 & c_H - c_G A_G^{-1} A_H \\ & & \\ 0 & I & A_G^{-1} A_H \end{pmatrix} \begin{pmatrix} -z \\ x_G \\ x_H \end{pmatrix} + B^{-1} y = \begin{pmatrix} -c_G A_G^{-1} b \\ \\ A_G^{-1} b \end{pmatrix}. \tag{22}$$

Note from (22) that B^{-1} is always the coefficient of y. Thus, we can start with the identity matrix in (21) and have the matrix B^{-1} available at every iteration. The actual pivoting from one iteration to the next is performed within B^{-1} to calculate a new inverse. The new inverse elements are

$$b'_{ij} = b_{ij} - \frac{\bar{a}_{is} b_{rj}}{\bar{a}_{rs}}, \qquad i \neq r$$

$$b'_{rj} = \frac{b_{rj}}{\bar{a}_{rs}}$$

$$\pi_j' = \pi_j + \frac{\bar{c}_s b_{rj}}{\bar{a}_{rs}}.$$

The variable vector y is not considered in the problem calculations.

We have described the inverse matrix method in terms of Phase II of the simplex algorithm; now we modify the analysis for use in Phase I of the algorithm. We consider the inverse matrix as

$$B^{-1} = \begin{pmatrix} 1 & 0 & -\sigma_1 & -\sigma_2 & \dots & -\sigma_m \\ 0 & 1 & -\pi_1 & -\pi_2 & \dots & -\pi_m \\ 0 & 0 & b_{11} & b_{12} & \dots & b_{1m} \\ 0 & 0 & b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & b_{m1} & b_{m2} & \dots & b_{mm} \end{pmatrix}, \tag{23}$$

where σ_i are the simplex multipliers relative to the w equation. Initially, B^{-1} is taken as the identity matrix. In any iteration of Phase I we calculate

$$\bar{d}_j = d_j - \sum_{i=1}^{m} \sigma_i a_{ij},$$

$$\bar{w}_0 = w_0 + \sum_{i=1}^{m} \sigma_i b_i,$$

and find min $\bar{d}_j$ to locate the pivot column. Pivoting then occurs in the inverse matrix (23). When Phase II is initiated, the first row and column of B^{-1} is discarded to form the B^{-1} of (20) and the procedure continues as previously explained.

Example

We list the values needed to solve the example problem in Section 3 as they are calculated. The problem is written in the form of Tableau E1.

x_1	x_2	x_3	x_4	
4	−4	0	−5	−5
2	−3	−1	2	0
−3	2	−1	3	2
−1	2	1	2	3

Tableau E1

The unit coefficients of $-w$, $-z$, x_5, and x_6 are omitted. Provision is made for listing the basic variables, the equality constant values, the inverse matrix B^{-1}, the $\bar{d}_j$ values, the $\bar{c}_j$ values, and the x_s column values.

ITERATION 1. Phase I.

	x_4	B^{-1}				
$-w$	−5	1	0	0	0	−5
$-z$	2	0	1	0	0	0
x_5	3	0	0	1	0	2
x_6	2	0	0	0	1	3

,

$\bar{d}_1 = 4, \bar{d}_2 = -4, \bar{d}_3 = 0, \bar{d}_4 = -5.$

ITERATION 2.

	x_3	B^{-1}			
$-w$	$-\frac{5}{3}$	1	0	$\frac{5}{3}$	0	$-\frac{5}{3}$
$-z$	$-\frac{1}{3}$	0	1	$-\frac{2}{3}$	0	$-\frac{4}{3}$
x_4	$-\frac{1}{3}$	0	0	$\frac{1}{3}$	0	$\frac{2}{3}$
x_6	$\frac{5}{3}$	0	0	$-\frac{2}{3}$	1	$\frac{5}{3}$

,

$\bar{d}_1 = -1, \bar{d}_2 = -\frac{2}{3}, \bar{d}_3 = -\frac{5}{3}, \bar{d}_4 = 0.$

ITERATION 3. Phase II.

	x_2	B^{-1}		
$-z$	$-\frac{21}{5}$	1	$-\frac{4}{5}$	$\frac{1}{5}$	-1
x_4	$\frac{4}{5}$	0	$\frac{1}{5}$	$\frac{1}{5}$	1
x_3	$\frac{2}{5}$	0	$-\frac{2}{5}$	$\frac{3}{5}$	1

,

$\bar{c}_1 = \frac{21}{5}, \bar{c}_2 = -\frac{21}{5}, \bar{c}_3 = 0, \bar{c}_4 = 0.$

ITERATION 4.

	B^{-1}		
$-z$	1	$\frac{1}{4}$	$\frac{5}{4}$	$\frac{17}{4}$
x_2	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{4}$
x_3	0	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

.

$\bar{c}_1 = 0, \bar{c}_2 = 0, \bar{c}_3 = 0, \bar{c}_4 = \frac{21}{4}$. The minimal solution is $z = -\frac{17}{4}$, $x_2 = \frac{5}{4}, x_3 = \frac{1}{2}, x_1 = 0, x_4 = 0.$

6 VARIABLES WITH UPPER BOUNDS

An important class of linear programs occurs when the variables have upper bound restraints. We wish to find the solution to (1) with the additional constraints $x_j \leq m_j$ for $j = 1, 2, \ldots, n$, where the m_j are given constant values.

It appears necessary at first to include these new restraints and their corresponding slack variables with the constraint equations of (1). Then the simplex method could be used on the considerably enlarged problem. By slightly generalizing the simplex procedure, however, we obviate the necessity to handle the larger problem. We see how the generalization is made as a result of

Theorem 7

Any values of x_j satisfying the constraint equations in Eq. (2) and $x_j \leq m_j$ for $j = 1, 2, \ldots, n$ are an optimal solution for min z if $\bar{c}_j \geq 0$ for variables at their lower bound zero and $\bar{c}_j \leq 0$ for variables at their upper bound m_j.

Proof

We have optimality by the conditions of the theorem since any inincrease in the variables at the zero level or any decrease in the variables at their upper bound can only increase z.

With the theorem providing the optimality conditions, we next improve a feasible solution that is not minimal. The method is to increase a nonbasic variable x_j at the value zero when $\bar{c}_j < 0$ or to decrease a nonbasic variable x_j at its upper bound when $\bar{c}_j > 0$. In either case there is an opportunity to decrease z; this is evident from (2). With some of the nonbasic variables at their upper bounds, the values of the objective function and basic variables are given by

$$\begin{aligned} z_0' &= \bar{z}_0 + \sum_{j \in U} \bar{c}_j m_j, \\ b_i' &= \bar{b}_i - \sum_{j \in U} \bar{a}_{ij} m_j; \end{aligned} \tag{24}$$

the objective value is z_0' and the ith basic variable has value b_i' with U as the index set of nonbasic variables at their upper bounds. Suppose now that $\bar{c}_s$ is negative and nonbasic x_s has zero value. The effect of letting $x_s = \theta$ is seen by changing (24) to the new values

$$\begin{aligned} z_0'' &= z_0' + \bar{c}_s \theta, \\ b_i'' &= b_i' - \bar{a}_{is} \theta. \end{aligned} \tag{25}$$

The objective value is z'' and is reduced from z_0' for positive θ. We also require that the basic variable values b_i'' be feasible, i.e., $0 \le b_i'' \le m_i$. If $\bar{a}_{is} < 0$, then x_s can be increased to $\theta = (m_i - b_i')/(-\bar{a}_{is})$. If $\bar{a}_{is} > 0$, then x_s can be increased to $\theta = b_i'/\bar{a}_{is}$. The greatest that x_s can be increased and still maintain feasibility is

$$\theta^* = \min \begin{cases} m_s, \\ \dfrac{m_i - b_i'}{-\bar{a}_{is}}, & \bar{a}_{is} < 0, \\ \dfrac{b_i'}{\bar{a}_{is}}, & \bar{a}_{is} > 0. \end{cases} \tag{26}$$

Suppose that $\bar{c}_s$ is positive and nonbasic x_s has value m_s. Let $x_s = \theta$ instead. The values obtained in (24) change to

$$\begin{aligned} z_0'' &= \bar{z}_0 + \sum_{\substack{j \in U \\ j \ne s}} \bar{c}_j m_j + \bar{c}_s \theta, \\ b_i'' &= \bar{b}_i - \sum_{\substack{j \in U \\ j \ne s}} \bar{a}_{ij} m_j - \bar{a}_{is} \theta. \end{aligned} \tag{27}$$

We rewrite (27) as

$$\begin{aligned} z_0'' &= \bar{z}_0 + \sum_{j \in U} \bar{c}_j m_j - \bar{c}_s(m_s - \theta) \\ &= z_0' - \bar{c}_s \gamma, \\ b_i'' &= \bar{b}_i - \sum_{j \in U} \bar{a}_{ij} m_j + \bar{a}_{is}(m_s - \theta) \\ &= b_i' + \bar{a}_{is} \gamma, \end{aligned} \tag{28}$$

where $\gamma = m_s - \theta$ is the amount of decrease of x_s. We use Eq. (28) to relate the old and new objective and basic values. The objective value is reduced for positive γ. We also require that $0 \le b_i'' \le m_i$ to maintain feasibility. If $\bar{a}_{is} < 0$, then x_s can be decreased by $\gamma = b_i'/(-\bar{a}_{is})$. If $\bar{a}_{is} > 0$, then x_s can be decreased by $\gamma = (m_i - b_i')/\bar{a}_{is}$. The greatest that x_s can be decreased by and still maintain feasibility is

$$\gamma^* = \min \begin{cases} m_s, \\ \dfrac{b_i'}{-\bar{a}_{is}}, & \bar{a}_{is} < 0, \\ \dfrac{m_i - b_i'}{\bar{a}_{is}}, & \bar{a}_{is} > 0. \end{cases} \tag{29}$$

The index s is obtained from $c_s' < 0$ where

$$(30) \quad c_s' = \min \begin{cases} \bar{c}_j, & \text{for nonbasic variables at their lower bound,} \\ -\bar{c}_j, & \text{for nonbasic variables at their upper bound.} \end{cases}$$

If no $c_s' < 0$, then of course the present solution is optimal. After selecting the nonbasic variable x_s by the choice rule of (30), we have

Case I.

Here $x_s = 0$ and $\bar{c}_s < 0$. In the next iteration, $x_s = \theta^*$, the new objective value is $z'' = z_0' + \bar{c}_s\theta^*$ and the variable values become $b_i'' = b_i' - \bar{a}_{is}\theta^*$ from (25); all other nonbasic variables retain their same values. In addition, if $\theta^* = m_s$ from (26), x_s assumes its upper bound value and no pivot operation is performed; the basic set remains unchanged. Otherwise, $\theta^* \neq m_s$ and x_s is made basic; choice rule (26) selects some basic variable x_r to become nonbasic from either

$$\theta^* = \frac{b_r'}{\bar{a}_{rs}} \qquad \text{or} \qquad \theta^* = \frac{m_r - b_r'}{-\bar{a}_{rs}}.$$

In the former instance, x_r assumes zero value; in the latter, x_r assumes its upper bound value m_r. The usual pivot operation is then performed. Note that the pivot element $\bar{a}_{rs}$ may be negative.

Case II.

Here $x_s = m_s$ and $\bar{c}_s > 0$. In the next iteration, $x_s = m_s - \gamma^*$, the new objective value is $z_0'' = z_0' - \bar{c}_s\gamma^*$ and the variable values become $b_i'' = b_i' + \bar{a}_{is}\gamma^*$ from (28); all other nonbasic variables retain their same values. In addition, if $\gamma^* = m_s$ from (29), x_s assumes zero value and no pivot operation is performed; the basic set remains unchanged. Otherwise, x_s is made basic and some basic variable x_r is made nonbasic. The variable x_r is selected from (29) with either

$$\gamma^* = \frac{b_r'}{-\bar{a}_{rs}} \qquad \text{or} \qquad \gamma^* = \frac{m_r - b_r'}{\bar{a}_{rs}}.$$

In the former instance, x_r assumes zero value; in the latter, x_r assumes its upper bound value. Pivoting then occurs.

The Phase I procedure is almost the same as the one just described. To select x_s, find $d_s' < 0$ where

$$d_s' = \min \begin{cases} \bar{d}_j, & \text{for } x_j \text{ at its lower bound} \\ -\bar{d}_j, & \text{for } x_j \text{ at its upper bound.} \end{cases}$$

We consider the artificial variables with infinite upper bounds. Similarly, the inverse matrix method remains substantially the same.

Example

Find nonnegative x_j with $x_1 \le 3$, $x_2 \le 2$, $x_3 \le 1$, $x_4 \le 3$, $x_5 \le 2$ that minimize z when

$$\begin{aligned} 4x_1 + 2x_2 + 3x_3 + x_4 - x_5 &= z, \\ 3x_1 + 4x_2 + 6x_3 + x_4 - x_5 &= 15, \\ -x_1 \qquad + 2x_3 + x_4 + x_5 &= 5. \end{aligned}$$

We write the Phase I problem

$$\begin{aligned} -w \quad - 2x_1 - 4x_2 - 8x_3 - 2x_4 \qquad\qquad\qquad &= -20, \\ -z + 4x_1 + 2x_2 + 3x_3 + x_4 - x_5 \qquad\qquad &= 0, \\ 3x_1 + 4x_2 + 6x_3 + x_4 - x_5 + x_6 \qquad &= 15, \\ -x_1 \qquad + 2x_3 + x_4 + x_5 \qquad + x_7 &= 5, \end{aligned}$$

with $-w$, $-z$, and artificials $x_6 \ge 0$ and $x_7 \ge 0$ as basic variables. The feasible solution is $w = 20$, $z = 0$, $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 15$, and $x_7 = 5$. Since $\bar{d}_3 = -8$ is negative, we increase x_3 to $\theta^* = \min(1, \frac{15}{6}, \frac{5}{2})$; the next feasible solution is $w = 12$, $z = 3$, $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$, $x_6 = 9$, and $x_7 = 3$. Since $\bar{d}_2 = -4$, we increase x_2 to $\theta^* = \min(2, \frac{9}{4})$; the next feasible solution is $w = 4, z = 7, x_1 = 0, x_2 = 2, x_3 = 1, x_4 = 1, x_5 = 0, x_6 = 1$, and $x_7 = 3$.

Since $\bar{d}_4 = -2$, we increase x_4 to $\theta^* = \min(3, \frac{1}{1}, \frac{3}{1})$; x_4 is made basic and x_6 is dropped. We pivot to obtain

$$\begin{aligned} -w \quad + 4x_1 + 4x_2 + 4x_3 \qquad -2x_5 \qquad &= \quad 10, \\ -z + \; x_1 - 2x_2 - 3x_3 \qquad\qquad &= -15, \\ 3x_1 + 4x_2 + 6x_3 + x_4 - \; x_5 \qquad &= \quad 15, \\ -4x_1 - 4x_2 - 4x_3 \qquad + 2x_5 + x_7 &= -10; \end{aligned}$$

the feasible solution is $w = 2$, $z = 8$, $x_1 = 0$, $x_2 = 2$, $x_3 = 1$, $x_4 = 1$, $x_5 = 0$, and $x_7 = 2$. Since $\bar{d}_2 = 4$ is positive, we decrease x_2 by $\gamma^* = \min(2, (3-1)/4, 2/4)$; x_2 is made basic and x_7 is dropped. We pivot to obtain

$$\begin{aligned} -z + 3x_1 \qquad - \; x_3 \qquad - \; x_5 &= -10, \\ -x_1 \qquad + 2x_3 + x_4 + \; x_5 &= \quad 5, \\ x_1 + x_2 + \; x_3 \qquad - \tfrac{1}{2}x_5 &= \quad \tfrac{5}{2}, \end{aligned}$$

where w is zero. Phase I ends. The feasible solution at the start of Phase II is $z = 9$, $x_1 = 0$, $x_2 = \frac{3}{2}$, $x_3 = 1$, $x_4 = 3$, and $x_5 = 0$. Since $\bar{c}_5 = -1$, we increase x_5 to $\theta^* = \min(2, \frac{3}{1}, \frac{1}{2}/\frac{1}{2})$; x_5 is made basic and x_2 becomes nonbasic at its upper bound value. We pivot to obtain

$$\begin{aligned} -z + \; x_1 - 2x_2 - 3x_3 \qquad &= -15, \\ x_1 + 2x_2 + 4x_3 + x_4 \qquad &= \quad 10, \\ -2x_1 - 2x_2 - 2x_3 \qquad + x_5 &= - \; 5, \end{aligned}$$

with the minimal solution $z = 8$, $x_1 = 0$, $x_2 = 2$, $x_3 = 1$, $x_4 = 2$, and $x_5 = 1$.

7 THE LEXICOGRAPHIC DUAL SIMPLEX METHOD

We will need to use a lexicographic dual simplex method in the study of integer programs. We present the method here after introducing the concept of duality.

Associated with every linear program is a second linear program

called the *dual program*. If we take the primal problem as: Minimize z when

$$\begin{aligned} cx &= z, \\ Ax &\geq b, \\ x &\geq 0. \end{aligned}$$

then the symmetric dual problem is: Maximize v when

$$\begin{aligned} ub &= v, \\ uA &\leq c, \\ u &\geq 0, \end{aligned}$$

where u is a row vector with m components u_i. These problems are symmetric in the sense that if an inequality format appears in the primal, then an inequality format appears in the dual problem.

The unsymmetric dual problem arises when the primal problem is: Minimize z when

$$\begin{aligned} cx &= z, \\ Ax &= b, \\ x &\geq 0. \end{aligned}$$

The dual problem is: Maximize v when

$$\begin{aligned} ub &= v, \\ uA &\leq c; \end{aligned}$$

the u_i are unrestricted in sign for the unsymmetric case.

There are other types of dual problems in which the primal constraints have both equalities and inequalities and some of the x_j are unrestricted. (Dantzig [*1*] and Hadley [*2*] give a fuller discussion of duality.)

The dual problems are of mathematical interest primarily. In computation the solution of one problem contains the solution to the other. For example, when the primal solution is found, the dual solution is given by $u_i = \pi_i$, the simplex multipliers of Eq. (20).

The simplex method of Section 3 operates with feasible solutions of the primal problem until a transformation is achieved with nonnegative constants in the objective function. A dual simplex method also operates on the primal problem, but it starts with the constants of the objective function, i.e., the c_j values, as nonnegative. The $\bar{c}_j$ values

are kept nonnegative in every iteration. When a feasible canonical form is reached, i.e., all $\bar{b}_i \geq 0$, we have the minimal solution. Because $\bar{c}_j \geq 0$ at each iteration produces a feasible solution to the dual problem, the method is known as the *dual simplex method.*

The dual simplex method is used with a lexicographic ordering to insure that the minimal solution is achieved in a finite number of steps. A vector R is defined as being lexicographically greater than zero, $R \succ 0$, if R has at least one non-zero component, the first of which is positive. (A fuller explanation of lexicographic ordering will be given in Chapter 3.)

The linear programming problem is written in the form: Find $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} z &= \sum_{j=1}^{n} c_j x_j, \\ x_{n+i} &= -b_i + \sum_{j=1}^{n} a_{ij} x_j \geq 0, \qquad i = 1, 2, \ldots, m. \end{aligned} \tag{31}$$

We assume that $c_j \geq 0$ for $j = 1, 2, \ldots, n$ and then write Eq. (31) as

$$\begin{aligned} z &= \sum_{j=1}^{n} c_j y_j, \\ x_j &= y_j \geq 0, \qquad\qquad j = 1, 2, \ldots, n, \\ x_{n+i} &= -b_i + \sum_{j=1}^{n} a_{ij} y_j \geq 0, \qquad i = 1, 2, \ldots, m. \end{aligned} \tag{32}$$

The y_j variables are used to relate the method to the integer programming algorithms presented later. See Simonnard [3] for the standard method. The starting basic solution is given by $x_{n+i} = -b_i$ for $i = 1, 2, \ldots, m$.

We can write (32) in the vector form

$$x = \beta + \sum_{j=1}^{n} \alpha_j y_j, \tag{33}$$

where x is a vector with components $z, x_1, x_2, \ldots, x_{n+m}$, β is a column

vector with components $0, 0, \ldots, 0, -b_1, -b_2, \ldots, -b_m$ and α_j is a column vector with components $c_j, 0, 0, \ldots, 1, \ldots, 0, a_{1j}, a_{2j}, \ldots, a_{mj}$. The one appears as the $(j+1)$ component of α_j.

Since all $c_j \geq 0$, then $\alpha_j \succ 0$. Suppose at some iteration we achieve a form like (33) where $\alpha_j \succ 0$. The nonbasic variables are given by the form $x_j = y_j$; the basic variables are then apparent. If all components of β are nonnegative, the minimal solution to (31) is given by all $y_j = 0$; i.e., $x = \beta$. Otherwise, one or more components of β are negative. Select a row from (33) with negative β component. Let the equation be

$$x_r = -b_0 + \sum_{j=1}^{n} a_j y_j, \tag{34}$$

where $-b_0$ is the negative component.

Since $y_j \geq 0$, at least one of the a_j is positive for the equation to have a solution. Define J^+ as the set of indices j where $a_j > 0$. Let s be in J^+; we solve (34) for y_s and obtain

$$y_s = \frac{b_0}{a_s} - \sum_{j \neq s} \frac{a_j}{a_s} y_j + \frac{1}{a_s} y_s',$$

where $y_s' \geq 0$ is taken for x_r. If y_s is eliminated from (33), then

$$x = \beta' + \sum_{j=1}^{n} \alpha_j' y_j, \tag{35}$$

where

$$\begin{aligned} \beta' &= \beta + \frac{b_0}{a_s} \alpha_s, \\ \alpha_j' &= \alpha_j - \frac{a_j}{a_s} \alpha_s, \qquad j \neq s \\ \alpha_s' &= \frac{1}{a_s} \alpha_s; \end{aligned} \tag{36}$$

also, y_s' has been replaced by new variable y_s.

We require that $\alpha_j' \succ 0$ in (35). Thus

$$\frac{1}{a_s}\alpha_s \prec \frac{1}{a_j}\alpha_j, \qquad j \neq s;$$

index s is chosen by

$$\frac{1}{a_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{a_j}\alpha_j,$$

where the minimum is defined in the lexicographic sense.

When (35) is developed, we designate the β' and α_j' values to be the current β and α_j values. Hence, (35) is of the same form as (33). If one or more components of β are negative again, we select a row with negative β component and repeat the process begun with (34). Eventually a form is attained where β has no negative components and $x = \beta$ is the minimal solution.

With $\alpha_j \succ 0$ at every iteration, β is lexicographically increasing as seen by (36). The minimal solution is reached in a finite number of steps since the same set of basic variables cannot be repeated.

Example

Find $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ that minimize z when

$$\begin{aligned} 2x_1 + 3x_2 + 3x_3 &= z, \\ 2x_1 - 3x_2 + x_3 &\geq 4, \\ x_1 + x_2 + 2x_3 &\geq 3. \end{aligned}$$

We introduce surplus variables $x_4 \geq 0$ and $x_5 \geq 0$ and put the problem in the format of (32) in the following tableau.

		1	2	3
z	0	2	3	3
x_1	0	1	0	0
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−4	2	−3	1
x_5	−3	1	1	2

ITERATION 1.

The basic variables are x_4 and x_5. The x_4 row is selected; $J^+ = (1, 3)$. $s = 1$ and we develop the tableau below using (36).

		1	2	3
z	4	1	6	2
x_1	2	$\frac{1}{2}$	$\frac{3}{2}$	$-\frac{1}{2}$
x_2	0	0	1	0
x_3	0	0	0	1
x_4	0	1	0	0
x_5	-1	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{3}{2}$

ITERATION 2.

The basic variables are x_1 and x_5. The x_5 row is selected; $J^+ = (1, 2, 3)$. $s = 3$ and we develop the following tableau.

		1	2	3
z	$\frac{16}{3}$	$\frac{1}{3}$	$\frac{8}{3}$	$\frac{4}{3}$
x_1	$\frac{5}{3}$	$\frac{2}{3}$	$\frac{7}{3}$	$-\frac{1}{3}$
x_2	0	0	1	0
x_3	$\frac{2}{3}$	$-\frac{1}{3}$	$-\frac{5}{3}$	$\frac{2}{3}$
x_4	0	1	0	0
x_5	0	0	0	1

ITERATION 3.

The minimal solution is reached; $z = \frac{16}{3}$, $x_1 = \frac{5}{3}$, $x_2 = 0$, and $x_3 = \frac{2}{3}$.

Problems

1. If any of the x_j in (1) are unrestricted in sign, show that each such x_j can be replaced by

$$x_j = x_j' - x_0$$

with $x_j' \geq 0$ and $x_0 \geq 0$. Thus, we need to introduce only one new variable to obtain the general linear programming format.

2. Find $x_j \geq 0$ that minimize z when

$$\begin{aligned} 3x_1 - x_2 + 12x_3 + x_4 &= z, \\ x_1 + x_2 + 3x_3 - 3x_4 &= 4, \\ 3x_1 - 3x_2 + 11x_3 + x_4 &= 2. \end{aligned}$$

3. Use the lexicographic dual simplex method to solve:

$$\begin{aligned} 2x_1 + 3x_2 &= z, \\ 3x_1 + 2x_2 &\geq 9, \\ 2x_1 + 5x_2 &\geq 8, \\ x_1 \geq 0, \qquad x_2 &\geq 0. \end{aligned}$$

4. Use the bounded variable technique to find nonnegative $x_1 \leq 1$, $x_2 \leq 1, x_3 \leq 1, x_4 \leq 1$ and min z for

$$\begin{aligned} 3x_2 + x_3 + 2x_4 &= z, \\ x_1 + x_2 + x_3 &= 2, \\ 5x_1 - 3x_2 + 3x_3 - 4x_4 &= 4. \end{aligned}$$

5. Find nonnegative x_j and min z in

$$\begin{aligned} 3x_1 + 4x_2 &= z, \\ 2x_1 + x_2 &\geq 1, \\ x_1 + 3x_2 &\geq 4. \end{aligned}$$

Use the inverse matrix method.

References

1. Dantzig, G. B., "Linear Programming and Extensions." Princeton Univ. Press, New Brunswick, New Jersey, 1963.
2. Hadley, G. "Linear Programming." Addison-Wesley, New York, 1962.
3. Simonnard, M. "Linear Programming" (W. S. Jewell, trans.). Prentice-Hall, Englewood Cliffs, New Jersey, 1966.
4. Wolfe, P., Experiments in linear programming, *in* "Recent Advances in Mathematical Programming" (R. L. Graves and P. Wolfe, eds.), pp. 177–200. Mc-Graw–Hill, New York, 1963.

3 ALL-INTEGER METHODS

We begin our study of integer linear programs with a presentation of all-integer methods. In an all-integer method the problem is stated with given integer coefficients; all calculations result in integer coefficients at each iteration.

We give the theory of all-integer methods using a parametric approach. Two main all-integer algorithms are developed: The first is similar to one by Gomory [3] but has different convergence properties; the second is used to solve bounded variable integer programs.

We then examine the interesting bounding form method developed by Glover [1]. The method may not produce rapid convergence in the solution of integer programs, and may not, therefore, seem practical; it is valuable, however, for the insights it offers into the integer programming process. We will use the method again in Chapter 6 to solve some of the classical problems of number theory.

These first methods are all based on the dual simplex algorithm. No feasible integer solutions are available until the optimal solution is achieved. We conclude the chapter with a simple primal algorithm, i.e., one that produces feasible integer solutions. We can offer no assurance that this algorithm will produce the optimal solutions, but modifications by Young [4] and Glover [2] insure convergence.

1 OPTIMALITY THEORY FOR INTEGER PROGRAMMING

We are interested in solving the integer programming problem: Find integer $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{n} c_j x_j = z, \\ &\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

and the a_{ij}, b_i, and c_j are given integer constants.

The method we use to find the optimal solution to Eq. (1) consists of making a series of changes of variables to achieve the transformation

$$x_j = d_j + \sum_{k=1}^{n} d_{jk} y_k, \qquad j = 1, 2, \ldots, n. \tag{2}$$

The integer constants d_j, d_{jk} are developed in an iterative procedure during the solution of the problem. The initial transformation is established by writing Eq. (1) in parametric form as

$$\begin{aligned} z &= \sum_{j=1}^{n} c_j y_j, \\ x_j &= y_j \geq 0, && j = 1, 2, \ldots, n, \\ x_{n+i} &= -b_i + \sum_{j=1}^{n} a_{ij} y_j \geq 0, && i = 1, 2, \ldots, m. \end{aligned} \tag{3}$$

The variables x_{n+i} for $i = 1, 2, \ldots, m$ are surplus variables.

Eliminating x_j from (1), using (2), we obtain the equivalent program: Find integer $y_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} z &= \bar{z}_0 + \sum_{j=1}^{n} \bar{c}_j y_j, \\ x_j &= d_j + \sum_{k=1}^{n} d_{jk} y_k \geq 0, && j = 1, 2, \ldots, n, \\ x_{n+i} &= -\bar{b}_i + \sum_{j=1}^{n} \bar{a}_{ij} y_j \geq 0, && i = 1, 2, \ldots, m. \end{aligned} \tag{4}$$

The constants $\bar{z}_0$, $\bar{c}_j$, $\bar{b}_i$, and $\bar{a}_{ij}$ are developed as a result of the transformation with

$$
\begin{aligned}
\bar{z}_0 &= \sum_{j=1}^{n} c_j d_j, \\
\bar{c}_j &= \sum_{k=1}^{n} c_k d_{kj}, && j = 1, 2, \ldots, n, \\
\bar{b}_i &= b_i - \sum_{j=1}^{n} a_{ij} d_j, && i = 1, 2, \ldots, m, \\
\bar{a}_{ij} &= \sum_{k=1}^{n} a_{ik} d_{kj}, && i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.
\end{aligned} \tag{5}
$$

We have

Theorem 1

If the constants are such that $\bar{c}_j \geq 0$, $d_j \geq 0$, and $\bar{b}_i \leq 0$ for all i and j, then the minimal solution to Eq. (4) is given by $z = \bar{z}_0$, $y_j = 0$ for $j = 1, 2, \ldots, n$.

Proof

If $d_j \geq 0$ and $\bar{b}_i \leq 0$, then $y_j = 0$ for $j = 1, 2, \ldots, n$ satisfy the constraints of (4) and produce $z = \bar{z}_0$. In addition, since $\bar{c}_j \geq 0$, a positive value for any y_j can only increase z.

We shall prove in Section 3 that Eq. (4) is an equivalent problem to (1), The minimal solution to (1) is then given by $z = \bar{z}_0$, $x_j = d_j$ for $j = 1, 2, \ldots, n$. We see from (5) that when $d_j \geq 0$ and $\bar{b}_i \leq 0$, then $x_j = d_j$ satisfies the constraints of (1) with objective value $\bar{z}_0$.

2 IMPROVING A NONOPTIMAL SOLUTION

We demonstrate here how to find the transformation, Eq. (2), that yields an equivalent problem, Eq. (4), and leads in a finite number of steps to the optimality conditions $\bar{c}_j \geq 0$, $d_j \geq 0$, and $\bar{b}_i \leq 0$.

Consider the case where all $c_j \geq 0$. We can write (4) as

$$x = \beta + \sum_{j=1}^{n} \alpha_j y_j, \tag{6}$$

where x is a column vector with components $z, x_1, x_2, \ldots, x_{n+m}$, β is a column vector with components $\bar{z}_0, d_1, d_2, \ldots, d_n, -\bar{b}_1, -\bar{b}_2, \ldots, -\bar{b}_m$, and α_j is a column vector with components $\bar{c}_j, d_{1j}, d_{2j}, \ldots, d_{nj}, \bar{a}_{1j}, \bar{a}_{2j}, \ldots, \bar{a}_{mj}$. Initially (3) is also in the form given by (6) with β components $0, 0, \ldots, 0, -b_1, -b_2, \ldots, -b_m$ and α_j components $c_j, 0, 0, \ldots, 0, 1, 0, \ldots, 0, a_{1j}, a_{2j}, \ldots, a_{mj}$. The one appears as the $(j+1)$ component of α_j.

To insure a finite algorithm we use *lexicographic ordering* in considering the α_j and β vectors. A vector R is defined as being lexicographically greater than zero, or *lexicopositive*, if R has at least one nonzero component, the first of which is positive. A vector R is less than vector S, $R \prec S$, in the lexicographic sense, if the vector S minus R is lexicopositive, $S - R \succ 0$. Define the symbols "$\prec$" and "$\succ$" to mean less than and greater than, respectively, in the lexicographic sense.

If (3) is written in the form of (6), we see that $\alpha_j \succ 0$ because all $c_j \geq 0$. Suppose at some iteration we have achieved a form like (6) where $\alpha_j \succ 0$ and one or more components of β are negative; the optimality conditions of Theorem 1 are not fulfilled. Select a row from (6) with negative β component. Let the inequality be

$$\sum_{j=1}^{n} a_j y_j \geq b_0, \tag{7}$$

where $-b_0$ is the negative component of β. For the inequality to have a solution, at least one of the a_j is positive. Taking λ a positive number, any value a_j/λ may be written as

$$\frac{a_j}{\lambda} = \left\{\frac{a_j}{\lambda}\right\} - \frac{r_j}{\lambda}, \qquad 0 \leq r_j < \lambda, \tag{8}$$

where $\{a\}$ denotes the smallest integer greater than or equal to a. Thus, after dividing by λ in (7) and using (8), we obtain

$$\sum_{j=1}^{n} p_j y_j \geq \frac{b_0}{\lambda} + \frac{1}{\lambda} \sum_{j=1}^{n} r_j y_j,$$

where $p_j = \{a_j/\lambda\}$. Note that

$$\frac{1}{\lambda}\sum_{j=1}^{n} r_j y_j \geq 0;$$

hence we have

$$\sum_{j=1}^{n} p_j y_j \geq \frac{b_0}{\lambda}. \tag{9}$$

Since the left side of (9) can have only an integer value, then

$$\sum_{j=1}^{n} p_j y_j \geq q, \tag{10}$$

where $q = \{b_0/\lambda\}$. It is desirable to make a change of variables for y_s, where $a_s > 0$; then if λ is chosen so that $\lambda \geq a_s$, from (10), we obtain

$$y_s = q - \sum_{j \neq s} p_j y_j + y_s', \tag{11}$$

where integer $y_s' \geq 0$ represents the surplus variable in (10). If y_s from (11) is substituted into (6), we have

$$x = \beta' + \sum_{j=1}^{n} \alpha_j' y_j. \tag{12}$$

The coefficients of (12) are

$$\begin{aligned} \alpha_j' &= \alpha_j - p_j \alpha_s, \qquad j \neq s \\ \alpha_s' &= \alpha_s, \\ \beta' &= \beta + q\alpha_s; \end{aligned} \tag{13}$$

also, y_s' has been replaced by new variable y_s in (12).

We require that the α_j' remain lexicopositive. Thus

$$\alpha_j - p_j \alpha_s \succ 0, \qquad j \neq s. \tag{14}$$

Condition (14) enables us to determine the index s and a value for λ, which produces the p_j values. If J^+ is defined as the set of indices j where $a_j > 0$ from (7), the index s is chosen by the rule

$$\alpha_s = l\text{-}\min_{j \in J^+} \alpha_j;$$

α_s is the lexicographically smallest of the α_j for $j \in J^+$. We define l-min to be the lexicographic minimum. Define integer value μ_j as the largest integer that maintains $\alpha_j - \mu_j \alpha_s$ lexicopositive for $j \in J^+$; also take $\mu_s = 1$. Condition (14) is fulfilled if $p_j \leq \mu_j$.

Now we can determine λ. If $\lambda = a_k/\mu_k$ for some index $k \in J^+$, then $p_k = \mu_k$. If $\lambda \geq a_k/\mu_k$, then p_k can only be reduced and condition (14) holds. The requirement for all j is that $\lambda \geq a_j/\mu_j$. In addition, to reduce the number of iterations, q is made as large as possible to affect the greatest change in β. Since β is bounded from above, as seen in Section 4, we make β approach the bound rapidly. For example, the first component of β is $\bar{z}_0$, which is bounded by the optimal value. By making q large we might produce an objective value that is close to optimum. Thus, to produce large q value, λ is made as small as possible; we take λ by the rule

$$\lambda = \max_{j \in J^+} \frac{a_j}{\mu_j}.$$

Note that λ may be fractional and that $\lambda \geq a_s$ so that p_s is unity in (10).

When (12) is developed, we designate the β' and α_j' values to be the current β and α_j values. Hence, (12) is of the same form as (6). If one or more components of β are negative again, we select a row with negative β component and repeat the process begun with inequality (7). Eventually a form is developed where β has no negative components. At this point the form (6) is like (4) and the conditions of Theorem 1 are fulfilled. The optimal solution to (1) is then produced by the first $(n + 1)$ components of β.

The solution of program (1) is obtained by using

Algorithm 1

1. Develop a tableau by listing the columns $\beta, \alpha_1, \alpha_2, \ldots, \alpha_n$. Before any iteration β has components $\bar{z}_0, d_1, d_2, \ldots, d_n, -\bar{b}_1, -\bar{b}_2, \ldots, -\bar{b}_m$; column α_j has components $\bar{c}_j, d_{1j}, d_{2j}, \ldots, d_{nj}, \bar{a}_{1j}, \bar{a}_{2j}, \ldots, \bar{a}_{mj}$. Initially $\bar{z}_0 = 0$, $d_j = 0$, $\bar{b}_i = b_i$; $\bar{c}_j = c_j \geq 0$, $d_{ii} = 1$, $d_{ij} = 0$, for $i \neq j$; $\bar{a}_{ij} = a_{ij}$. Go to 2.

2. If $d_j \geq 0$ for $j = 1, 2, \ldots, n$ and $\bar{b}_i \leq 0$ for $i = 1, 2, \ldots, m$, the

minimal solution is $z = \bar{z}_0$, $x_j = d_j$ for $j = 1, 2, \ldots, n$; stop. Otherwise, select the nonobjective row with the smallest β component. Suppose the row is $-b_0, a_1, a_2, \ldots, a_n$. Define J^+ as the set of indices j where $a_j > 0$. If all $a_j \leq 0$, the problem has no solution; stop. Otherwise, go to 3.

3. Determine index s from $\alpha_s = l\text{-}\min_{j \in J^+} \alpha_j$. Find the largest integer μ_j that maintains $\alpha_j - \mu_j \alpha_s \succ 0$ for $j \in J^+, j \neq s$. If α_j and α_s begin with an unequal number of zeroes, take $\mu_j = \infty$. Otherwise, suppose the first nonzero terms are e_j and e_s. If e_s does not divide e_j, take $\mu_j = [e_j/e_s]$, where $[a]$ denotes the greatest integer less than or equal to a. If e_s does divide e_j, then $\mu_j = e_j/e_s$ if $\alpha_j - (e_j/e_s)\alpha_s \succ 0$ and $\mu_j = e_j/e_s - 1$ otherwise. Also $\mu_s = 1$. Take $\lambda = \max_{j \in J^+} a_j/\mu_j$. Go to 4.

4. Calculate $q = \{b_0/\lambda\}$, $p_j = \{a_j/\lambda\}$ and new column values $\beta' = \beta + q\alpha_s$ and $\alpha_j' = \alpha_j - p_j\alpha_s$ for $j \neq s$. Designate β' and α_j' to be the current β and α_j. Return to 2.

Example

Find integers $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ that minimize z when

$$\begin{aligned} 2x_1 + 6x_2 + 3x_3 &= z, \\ x_1 + 3x_2 + x_3 &\geq 5, \\ 2x_1 + 5x_2 - 3x_3 &\geq 6, \\ 2x_1 + 3x_2 + 2x_3 &\geq 4. \end{aligned}$$

We follow the steps as they occur in Algorithm 1.

1. The problem is listed in the following tableau.

		1	2	3
z	0	2	6	3
x_1	0	1	0	0
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−5	1	3	1
x_5	−6	2	5	−3
x_6	−4	2	3	2

The integer surplus variables are x_4, x_5, and x_6.

ITERATION 1.

2. The x_5 row is selected. $J^+ = (1, 2)$.
3. $s = 1, \mu_1 = 1, \mu_2 = 2; \lambda = \max(\frac{2}{1}, \frac{5}{2}) = \frac{5}{2}$.
4. $q = 3$. $p_1 = 1, p_2 = 2, p_3 = -1$. We form the following tableau.

		1	2	3
z	6	2	2	5
x_1	3	1	−2	1
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−2	1	1	2
x_5	0	2	1	−1
x_6	2	2	−1	4

ITERATION 2.

2. The x_4 row is selected. $J^+ = (1, 2, 3)$.
3. $s = 2$. $\mu_1 = 1, \mu_2 = 1, \mu_3 = 2; \lambda = \max(\frac{1}{1}, \frac{1}{1}, \frac{2}{2}) = 1$.
4. $q = 2, p_1 = 1, p_2 = 1, p_3 = 2$. We form the tableau below.

		1	2	3
z	10	0	2	1
x_1	−1	3	−2	5
x_2	2	−1	1	−2
x_3	0	0	0	1
x_4	0	0	1	0
x_5	2	1	1	−3
x_6	0	3	−1	6

ITERATION 3.

2. The x_1 row is selected. $J^+ = (1, 3)$
3. $s = 1$. $\mu_1 = 1, \mu_3 = \infty; \lambda = 3$.

4. $q = 1, p_1 = 1, p_2 = 0, p_3 = 2$. We form the following tableau.

		1	2	3
z	10	0	2	1
x_1	2	3	−2	−1
x_2	1	−1	1	0
x_3	0	0	0	1
x_4	0	0	1	0
x_5	3	1	1	−5
x_6	3	3	−1	0

ITERATION 4.

2. The minimal solution is reached in the preceding tableau. It is $z = 10, x_1 = 2, x_2 = 1, x_3 = 0$.

In step 2 of Algorithm 1 the nonobjective row with smallest β component is chosen for inequality (7). This choice rule serves two purposes. First, it may make $q = \{b_0/\lambda\}$ large, thereby increasing β the most; secondly at the end of the iteration, the corresponding β component has value $-b_0 + qa_s > -b_0$, which may be positive and more nearly fulfill the optimality conditions. The choice rule leads to a finite algorithm. Other choice rules may be effective, such as the selection of

1. the row that makes the greatest lexicographic change in β,
2. the first nonobjective row with negative β component,
3. the rows in a cyclic manner,
4. the rows at random.

3 EQUIVALENT INTEGER PROGRAMS

We define two integer programs as equivalent if the minimal solution to one produces the minimal solution to the other and vice versa. We now show that the transformation (2), developed in Section 2, causes programs (1) and (4) to be equivalent.

Consider program (1) in matrix form, with relaxed integer restriction: Find $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} cx &= z, \\ Ax &\geq b, \end{aligned} \tag{15}$$

where x, c, and b are vectors with components x_j, c_j, and b_i, respectively; A is a matrix with elements a_{ij}. Write the transformation (2) as

$$x = d + Dy, \tag{16}$$

where d and y are vectors with components d_j and y_j respectively; D is a matrix with elements d_{ij}. Eliminate x from (15) using (16) and consider the program: Minimize z when

$$\begin{aligned} cDy &= z - cd, \\ Dy &\geq -d, \\ ADy &\geq b - Ad. \end{aligned} \tag{17}$$

We have

Theorem 2

If the inverse matrix D^{-1} exists and y^0 is a minimal solution to (17), then $x^0 = d + Dy^0$ is a minimal solution to (15).

Proof

Suppose $\bar{x}$ satisfies the constraints of (15) with objective value $\bar{z} = c\bar{x} < z^0 = cd + cDy^0$. Since D^{-1} exists, then $y = D^{-1}\bar{x} - D^{-1}d$ satisfies the constraints of (17) with objective value $\bar{z}$; thus, the assumption that y^0 is a minimal solution is contradicted. Hence, $x^0 = d + Dy^0$ is a minimal solution to (15). Similarly, the minimal solution to (15) produces the minimal solution to (17).

In addition, if d, D, and D^{-1} have only integer elements, then (15) and (17) are equivalent programs for integer $x_j \geq 0$ and integer y_j. Now we must show that our transformation (11) leads to the proper D matrix and that the equivalence of (15) and (17) then holds for nonnegative integer y_j.

Transformation (2), as contained in (4), is obtained by repeated use of transformations like (11). Transformation (11) may be written as

$$y^{(r)} = -Q_r + P_r y^{(r-1)}; \tag{18}$$

$y^{(r)}$ is the y vector after iteration r [i.e., the rth use of forms like (11)], Q_r is a vector with $n-1$ zero components and some component s_r with value q_r, and P_r is an $n \times n$ elementary matrix[1]

$$P_r = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & \ddots & & & & \\ p_{1r} & p_{2r} & \cdots & 1 & \cdots & & p_{nr} & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{bmatrix}, \tag{19}$$

where P_r differs from the identity matrix in row s_r. The values q_r, p_{jr}, and s_r are the q, p_j, and s values obtained from using (11) in iteration r. The transformation arises by writing (11) in the form

$$\begin{aligned} y_1^{(r)} &= y_1^{(r-1)}, \\ y_2^{(r)} &= y_2^{(r-1)}, \\ &\vdots \\ y_s^{(r)} &= -q + \sum_{j=1}^{n} p_j y_j^{(r-1)}, \\ &\vdots \\ y_n^{(r)} &= y_n^{(r-1)}, \end{aligned} \tag{20}$$

where the r subscripts on s, q, and the p_j have been neglected; $p_s = 1$.

The inverse to P_r is

$$P_r^{-1} = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & \ddots & & & & \\ -p_{1r} & -p_{2r} & \cdots & 1 & \cdots & & -p_{nr} & \\ & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{bmatrix}, \tag{21}$$

[1] An *elementary matrix* differs from the identity matrix in just one row or column.

and the inverse transformation to (18) exists and is

$$y^{(r-1)} = Q_r + P_r^{-1} y^{(r)}. \tag{22}$$

We use (22) recursively. Initially, $y^{(0)} = x$ and transformation (2) is achieved after some iteration t as

$$x = \sum_{r=1}^{t} T_{r-1} Q_r + T_t y^{(t)}, \tag{23}$$

where

$$T_0 = I,$$
$$T_t = P_1^{-1} P_2^{-1} \dots P_t^{-1}.$$

Since T_t is the product of elementary matrices of the form (21), the determinant of T_t is one; thus, T_t^{-1} exists. Also define d and D for (2), from (23), as

$$d = \sum_{r=1}^{t} T_{r-1} Q_r, \tag{24}$$
$$D = T_t.$$

Clearly, the conditions of Theorem 2 are fulfilled. Programs (15) and (17) are equivalent for D given by (24). Since the determinant of D is unity and D is composed of integer elements, D^{-1} is also composed of integer elements. Hence, (15) and (17) are equivalent problems for integer $x_j \geq 0$ and integer y_j when the transformation is given by (23). We proceed to show that (15) and (17) are also equivalent problems for integer $x_j \geq 0$ and integer $y_j \geq 0$.

In working out the development of (17) we are really considering a new problem for each iteration r. The successive use of (22) leads to the constraints of each new problem (17). Program r of (17) is defined as (17) with $y_j = y_j^{(r)}$. We prove inductively that any integer $y_j^{(r)}$ values that satisfy the constraints of problem r must be nonnegative.

We show that integer $y_j^{(r)} \geq 0$ for all r. Take $r = 1$; since $y_j^{(0)} = x_j$, we have $y_j^{(1)} = x_j$ for $j \neq s$ and

$$y_s^{(1)} = -q + \sum_{j=1}^{n} p_j x_j \tag{25}$$

from (20). Equation (25) is developed, as in the formation of (11), by

requiring that $x_j = y_j^{(0)} \geq 0$ be integers in (3) for the first iteration. Following (10), integers $x_j \geq 0$ must satisfy

$$\sum_{j=1}^{n} p_j x_j \geq q. \tag{26}$$

Program (15) with integer $x_j \geq 0$ and program one of (17) with integer $y_j^{(1)}$ are equivalent for transformation (20) with $r = 1$. Thus feasible integers $y_j^{(1)}$ to (17) will produce feasible integers $x_j \geq 0$ to (15). These feasible integers must satisfy (26). Hence, $y_s^{(1)} \geq 0$ from (25) and $y_j^{(1)} = x_j \geq 0, j \neq s$. Program (15) and program one of (17) are equivalent for integer $x_j \geq 0$ and integer $y_j^{(1)} \geq 0$.

Assume now that problems $r - 1$ and r of (17) are equivalent for integer $y_j^{(r-1)} \geq 0$ and integer $y_j^{(r)} \geq 0$ for $r = 1, 2, \ldots, t$ (the $r = 0$ problem of (17) is (15) for integer $x_j \geq 0$). We shall prove that integer $y_j^{(t+1)} \geq 0$. The equations in (20) hold for $r = t + 1$, where the s, q, and p_j may be different than in (25). The equations for $y_s^{(t+1)}$ in (20) result from the requirement that $y_j = y_j^{(t)} \geq 0$ be integers in (4). Thus, following (10), integers $y_j^{(t)} \geq 0$ must satisfy.

$$\sum_{j=1}^{n} p_j y_j^{(t)} \geq q. \tag{27}$$

Problem t of (17) with integer $y_j^{(t)} \geq 0$ and problem $t + 1$ with integer $y_j^{(t+1)}$ are equivalent for transformation (20) with $r = t$. Thus, feasible integers $y_j^{(t+1)}$ to problem $t + 1$ of (17) will produce feasible integers $y_j^{(t)} \geq 0$ to problem t of (17). These feasible integers must satisfy (27). Hence, $y_j^{(t+1)} \geq 0$ from (20). Problems t and $t + 1$ are equivalent for integer $y_j^{(t)} \geq 0$ and integer $y_j^{(t+1)} \geq 0$. Therefore, by induction, (15) and problem t of (17) are equivalent for integer $x_j \geq 0$ and integer $y_j^{(t)} \geq 0$ for any t value.

4 CONVERGENCE TO OPTIMALITY

The integer programming problem appears to be readily solvable by Algorithm 1. We must show that the algorithm produces the minimal solution in a finite number of iterations. The convergence of the algorithm is a result of the lexicographic ordering or the α_j columns of (6).

The α_j columns are lexicopositive in each step of the algorithm. Since β, from one iteration to the next, is given by

$$\beta' = \beta + q\alpha_s,$$

then $\beta' \succ \beta$; β is always increased lexicographically. Suppose the initial value of β obtained in (3) is β_0. After iteration t of the algorithm, the β value is β_t. A β sequence is produced with successively larger lexicographic values as $\beta_0 \prec \beta_1 \prec \beta_2 \prec \cdots$. Also, since the components of β are integers, the sequence changes by integer quantities.

Assume there exists a finite minimal solution x_0 with objective component z_0; i.e., $x_0 = (z_0, x_1^0, x_2^0, \ldots, x_n^0)$. Then x_0 must satisfy (6) with constants for β and α_j that result after any iteration. There must exist $y_j \geq 0$ that satisfy

$$\sum_{j=1}^{n} \alpha_j y_j = x_0 - \beta. \tag{28}$$

The convergence of the algorithm can now be proven. If the components of β are bounded from above by finite values, the algorithm requires a finite number of iterations. Consider the first component of β, the objective value $\bar{z}_0$. It can increase only for a finite number of iterations and then must remain at some fixed value $\bar{z}_0 \leq z_0$. If $\bar{z}_0 > z_0$ after some iteration, then write the objective row from (28) as

$$\sum_{j=1}^{n} \bar{c}_j y_j = z_0 - \bar{z}_0 < 0;$$

since $\bar{c}_j \geq 0$, no $y_j \geq 0$ values can possibly produce $z = z_0$. Thus, $\bar{z}_0$, the first component of β, is bounded from above by z_0.

An infinite number of iterations occur if some component of β becomes unbounded. Suppose the $(r+1)$ component, $1 \leq r \leq n$, takes an arbitrarily large value; then d_r, the value of variable x_r, becomes large. Let $d_r > x_r^0$ after some finite number of iterations. Write the corresponding row from (6)

$$\sum_{j=1}^{n} d_{rj} y_j = x_r^0 - d_r < 0;$$

if all $d_{rj} \geq 0$, then no $y_j \geq 0$ exist and d_r must be bounded by x_r^0. If some $d_{rj} < 0$, then the inequality

$$\sum_{j \in J^-} (-d_{rj}) y_j \geq d_r - x_r^0 \tag{29}$$

must hold. We define J^- as the set of indices j where $d_{rj} < 0$. The y_j are restrained by (29), which is of the form given by (7). We proceed as follows:

1. Obtain the index s by selecting α_s as the lexicographic smallest of the α_j for $j \in J^-$.
2. Find μ_j as the largest integer that keeps $\alpha_j - \mu_j \alpha_s \succ 0$ for $j \in J^-$; also, $\mu_s = 1$. Take $\lambda = \max_{j \in J^-} -d_{rj}/\mu_j$.
3. Calculate $q = \{(d_r - x_r^0)/\lambda\}$ and $p_j = \{-d_{rj}/\lambda\}$ for $j \in J^-$; $p_j = 0$ otherwise.

This leads to forms like (10) and (11); y_s is eliminated from (6) using (11) and (6) becomes

$$x = \beta' + q\alpha_s + \sum_{j=1}^{n} \alpha_j' y_j, \tag{30}$$

as in (12).

The convergence becomes apparent by induction. Suppose $d_1 > x_1^0$ after some iteration. The objective value from (30) is

$$\bar{z}_0' = \bar{z}_0 + q\bar{c}_s. \tag{31}$$

At least one d_{1j} must be negative and the α_j are lexicopositive, causing $\bar{c}_s$ to be positive. Since $\bar{z}_0' \leq z_0$, then $q \leq (z_0 - \bar{z}_0)/\bar{c}_s$ produces a bound for d_1 given by

$$d_1 \leq x_1^0 + \frac{\lambda(z_0 - \bar{z}_0)}{\bar{c}_s}.$$

Any larger value of d_1 can produce only a value of z larger than z_0, which is impossible.

Suppose, further, that $d_1, d_2, \ldots, d_{r-1}$ are bounded by $\delta_1, \delta_2, \ldots, \delta_{r-1}$, respectively, and that $d_r > x_r^0$ after some iteration. The objective value again is of the form of (31) In addition, the next $(r-1)$ components of β are

$$d_i' = d_i + qd_{is}, \qquad i = 1, 2, \ldots, r-1.$$

From the lexicopositive property of α_s and from the fact that $d_{rs} < 0$, one or more of the values $\bar{c}_s, d_{1s}, d_{2s}, \ldots, d_{r-1,s}$ is positive. Thus

$$d_r \leq \begin{cases} x_r^0 + \lambda \dfrac{z_0 - \bar{z}_0}{\bar{c}_s}, & \text{if} \quad \bar{c}_s > 0 \\ x_r^0 + \lambda \dfrac{\delta_i - d_i}{d_{is}}, & \text{if} \quad d_{is} > 0, \quad i = 1, 2, \ldots, r-1, \end{cases}$$

In any case d_r is bounded. By induction $d_1, d_2, \ldots, d_n$, the first n components of β, have finite bounds. Also, the next m components of β, representing the surplus variables, must be bounded. Since β is bounded and is lexicographically increasing by integer steps, the algorithm must take a finite number of iterations to achieve the optimal solution.

5 BOUNDED VARIABLE PROBLEMS

We are interested in solving the integer programming problem when the variables have upper bound restraints. We seek to find the solution to (1) with the additional constraints $x_j \leq m_j$ for $j = 1, 2, \ldots, n$, where the m_j are given integer values.

A possible solution to the bounded variable problem might be obtained by including the inequalities $-x_j \geq -m_j$ with the constraints of (1) and using Algorithm 1 to handle the considerably enlarged problem. We choose a different approach. We shall use Algorithm 1 and disregard the upper bounds on the variables until an iteration is reached, in developing (4), with $d_j > m_j$ for some index j. Naturally, if all $d_j \leq m_j$ for every iteration, Algorithm 1 solves the problem.

Suppose at some iteration we have achieved a form like (6), where $\alpha_j \succ 0$ and one or more of the d_j components of β have value $d_j > m_j$. Select one such row from (6). For that row we have

$$\begin{aligned} x_j &= d_j + \sum_{k=1}^{n} d_{jk} y_k \\ &\leq m_j. \end{aligned}$$

The y_k must then satisfy

$$-\sum_{k=1}^{n} d_{jk} y_k \geq d_j - m_j \tag{32}$$
$$> 0.$$

If we define a_k and b_0 by

$$a_k = -d_{jk}, \qquad k = 1, 2, \ldots, n,$$
$$b_0 = d_j - m_j,$$

then (32) is exactly of the form given by (7). Using (32) then as (7), the analysis follows as before; we develop transformation (11) and the new form of (6) given by (12). The lexicographic property of the α_j is maintained. The selection of index s and the λ calculation remain the same. Furthermore, the theory presented in Sections 3 and 4 holds; the bounded variable problem is simply solved by the use of (32). For upper bound problems we have

Algorithm 2

1. Same as step 1 of Algorithm 1 with the additional listing of $m_1, m_2, \ldots, m_n$.
2. (a) If $0 \leq d_j \leq m_j$ for $j = 1, 2, \ldots, n$ and $\bar{b}_i \leq 0$ for $i = 1, 2, \ldots, m$, the minimal solution is $z = \bar{z}_0$, $x_j = d_j$ for $j = 1, 2, \ldots, n$; stop. Otherwise, go to 2 (b).
   (b) If $d_j \leq m_j$ for $j = 1, 2, \ldots, n$, go to 2(c). Otherwise, select the row with d_j component of β that produces the largest value of $d_j - m_j > 0$. Take $b_0 = d_j - m_j$, $a_k = -d_{jk}$ for $k = 1, 2, \ldots, n$ as the row picked. Go to 2(d).
   (c) Select the nonobjective row with the smallest β component. Suppose the row is $-b_0, a_1, a_2, \ldots, a_n$. Go to 2(d).
   (d) Define J^+ as the set of indices j where $a_j > 0$. If all $a_j \leq 0$, the problem has no solution; stop. Otherwise, go to 3.
3. Same as step 3 of Algorithm 1.
4. Same as step 4 of Algorithm 1.

The choice rules of steps 2(b) and 2(c) may be changed just as in the previous algorithm. See the discussion immediately following Algorithm 1.

Example

Find nonnegative integers $x_1 \leq 1$, $x_2 \leq 1$, $x_3 \leq 1$ that minimize z when

$$3x_1 + 2x_2 + 5x_3 = z,$$
$$3x_1 + 5x_2 + 4x_3 \geq 3,$$
$$3x_1 + 2x_2 + 2x_3 \geq 3,$$
$$2x_1 + 2x_2 + 7x_3 \geq 4.$$

We follow the steps as they occur in the algorithm.

1. The problem is listed in the following tableau.

		1	2	3
z	0	3	2	5
x_1	0	1	0	0
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−3	3	5	4
x_5	−3	3	2	2
x_6	−4	2	2	7

ITERATION 1.

2. (c) The x_6 row is selected.
   (d) $J^+ = (1, 2, 3)$.
3. $s = 2$. $\mu_1 = 1, \mu_2 = 1, \mu_3 = 2$; $\lambda = \max(\frac{2}{1}, \frac{2}{1}, \frac{7}{2}) = \frac{7}{2}$.
4. $q = 2, p_1 = 1, p_2 = 1, p_3 = 2$. We form the tableau below.

		1	2	3
z	4	1	2	1
x_1	0	1	0	0
x_2	2	−1	1	−2
x_3	0	0	0	1
x_4	7	−2	5	−6
x_5	1	1	2	−2
x_6	0	0	2	3

ITERATION 2.

2. (a) $d_2 = 2, m_2 = 1$; $d_2 > m_2$.
   (b) $b_0 = 1, a_1 = 1, a_2 = -1, a_3 = 2$.
   (d) $J^+ = (1, 3)$.
3. $s = 3$. $\mu_1 = 1, \mu_3 = 1$; $\lambda = \max(\frac{1}{1}, \frac{2}{1}) = 2$.
4. $q = 1, p_1 = 1, p_2 = 0, p_3 = 1$. We form the following tableau.

		1	2	3
z	5	0	2	1
x_1	0	1	0	0
x_2	0	1	1	−2
x_3	1	−1	0	1
x_4	1	4	5	−6
x_5	−1	3	2	−2
x_6	3	−3	2	3

ITERATION 3.

2. (c) The x_5 row is selected.
   (d) $J^+ = (1, 2)$.
3. $s = 1$. $\mu_1 = 1, \mu_2 = \infty$; $\lambda = 3$.
4. $q = 1, p_1 = 1, p_2 = 1, p_3 = 0$. We form the tableau below.

		1	2	3
z	5	0	2	1
x_1	1	1	−1	0
x_2	1	1	0	−2
x_3	0	−1	1	1
x_4	5	4	1	−6
x_5	2	3	−1	−2
x_6	0	−3	5	3

ITERATION 4.

2. (a) The minimal solution is reached in the tableau of the previous iteration. It is $z = 5, x_1 = 1, x_2 = 1, x_3 = 0$.

6 NEGATIVE c_j VALUES

The methods discussed so far have required that $c_j \geq 0$ for $j = 1, 2, \ldots, n$. These methods can be modified, however, to include the case where *any* of the c_j are negative. We examine such a case here.

Let us define J^- as the set of indices j where $c_j < 0$. If the problem is to be of interest, the sum of x_j for $j \in J^-$ must have a finite upper bound. This upper bound is either apparent or readily obtained by solving the linear programming problem: Find $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that maximize z' when

$$\sum_{j \in J^-} x_j = z',$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m. \tag{33}$$

In either case, we can assume that

$$\sum_{j \in J^-} x_j \leq d_0,$$

where integer d_0 is initially given or obtained from (33) as the integer part of optimal z'. We express Eq. (1) in the form given by (6). The inequality with x_j replaced by y_j is

$$\sum_{j \in J^-} y_j \leq d_0; \tag{34}$$

we find index s from

$$\alpha_s = l\text{-}\min_{j \in J^-} \alpha_j.$$

We introduce a slack variable $y_s' \geq 0$ and solve for y_s in (34) to obtain

$$y_s = d_0 - \sum_{\substack{j \in J^- \\ j \neq s}} y_j - y_s'. \tag{35}$$

Using (35), we replace y_s in (6) to form (12) where

$$\alpha_j' = \alpha_j - \alpha_s,$$
$$\alpha_s' = -\alpha_s,$$
$$\beta' = \beta + d_0 \alpha_s.$$

Designate β' and the α_j' to be the current β and α_j. Observe that the α_j become lexicopositive. We begin Algorithm 1 in step 2.

Example

Find integers $x_j \geq 0$ that maximize z when

$$\begin{aligned} x_1 - 2x_2 + 3x_3 &= z, \\ -3x_1 + 3x_2 + x_3 &\leq 4, \\ 2x_1 + 2x_2 - x_3 &\leq 4, \\ 3x_1 - 2x_2 &\leq 1. \end{aligned}$$

Note that this is a maximization problem. We maximize $z' = x_1 + x_3$ subject to the constraints and obtain $\max z' = \frac{16}{3}$. We write the problem in Tableau E1; $s = 3$, $d_0 = 5$, and we form Tableau E2.

	1	2	3
z	0	−1	2	−3
x_1	0	1	0	0
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−4	3	−3	−1
x_5	−4	−2	−2	1
x_6	−1	−3	2	0

Tableau E1

	1	2	3
z	−15	2	2	3
x_1	0	1	0	0
x_2	0	0	1	0
x_3	5	−1	0	−1
x_4	−9	4	−3	1
x_5	1	−3	−2	−1
x_6	−1	−3	2	0

Tableau E2

Tableau E2 is the starting point for the algorithm.

7 THE USE OF BOUNDING FORMS

There is another dual simplex method in addition to the method described in Section 1, one that is formulated by Glover [2] and known as the *bound escalation method.* It is based on the observation that $y_s \geq q = \{b_0/a_s\}$ if an inequality of (4), namely $\sum_{j=1}^{n} a_j y_j \geq b_0 > 0$, has the property that $a_s > 0$ for some index s and $a_j \leq 0$ for $j \neq s$.

The inequality is then a *bounding form* because it provides a positive lower bound to variable y_s. We shall show how bounding forms are produced and how they lead to the solution of integer programs.

When a bounding form does not exist, it can always be developed through a single transformation. Let the inequality

$$\sum_{j=1}^{n} a_j y_j \geq b_0 \qquad (36)$$
$$> 0$$

have two or more positive coefficients a_j. Suppose $a_s > 0$; we desire a bounding form that maintains $a_s > 0$. If we write the inequality as

$$\sum_{j \in J^+} a_j y_j + \sum_{j \in J^-} a_j y_j \geq b_0, \qquad (37)$$

where J^+ is the index set for $a_j > 0$ and J^- is the index set for $a_j < 0$, then

$$\sum_{\substack{j \in J^+ \\ j \neq s}} v_j y_j + y_s = y_s' \geq 0, \qquad (38)$$

where $v_j = \{a_j/a_s\}$ for $j \in J^+$. Eliminating y_s from (37) using (38), we obtain

$$\sum_{j \in J^+} a_j' y_j + \sum_{j \in J^-} a_j y_j \geq b_0, \qquad (39)$$

where

$$a_j' = a_j - v_j a_s, \qquad j \in J^+, \quad j \neq s,$$
$$a_s' = a_s$$

and y_s' is replaced by new variable y_s. Since $a_j' \leq 0$ for $j \in J^+$, $j \neq s$, then (39) is a bounding form with $y_s \geq q = \{b_0/a_s\}$.

Take the integer programming problem as given by (6) where $\alpha_j > 0$ and one or more components of β are negative. Select a row with negative β component. Let the inequality be (36) and define index s from $\alpha_s = l\text{-}\min_{j \in J^+} \alpha_j$. If we use (38) to eliminate y_s from (6), we achieve (12) where

$$\begin{aligned} \alpha_j' &= \alpha_j - v_j \alpha_s, & j \in J^+, \quad j \neq s \\ \alpha_s' &= \alpha_s, & \\ \alpha_j' &= \alpha_j, & j \in J^-, \\ \beta' &= \beta. & \end{aligned} \qquad (40)$$

We now make an additional change of variable

$$y_s = q + y_s'$$

and replace β' by

$$\beta' = \beta + q\alpha_s. \tag{41}$$

When a sequence of lexicographically increasing β values are produced, the method solves the integer programming problem. The increasing β values occur if the α_j' remain lexicopositive in (40). To insure $\alpha_j' \succ 0$, we require that $v_j \leq \mu_j$, where μ_j is the largest integer that keeps $\alpha_j - \mu_j\alpha_s \succ 0$ for $j \in J^+$.

We have seen that if any $v_j > \mu_j$, then the bounding form may not lead to a finite algorithm, To maintain $\alpha_j' \succ 0$ we change (38) to

$$\sum_{\substack{j\in J^+ \\ j\neq s}} p_j y_j + y_s = y_s' \geq 0, \tag{42}$$

where $p_j = \min(v_j, \mu_j)$. We use (42) to eliminate y_s from (6) and achieve (12) with v_j replaced by p_j in (40). Hence, if any $v_j > \mu_j$, then (39) is not a bounding form and the β vector is not changed by (41). We repeat the process in an effort to obtain the bounding form; the β vector remains the same and we select the row that gave us (36). Eventually, we obtain a bounding form and a new β given by (41). Thus, we maintain $\alpha_j \succ 0$ and produce an increasing β.

We sum up the procedure in

Algorithm 3

Steps 1 and 2 are identical with those in Algorithm 1.

3. Determine index s from $\alpha_s = l\text{-}\min_{j\in J^+} \alpha_j$. Find the largest integer μ_j that maintains $\alpha_j - \mu_j\alpha_s \succ 0$ for $j \in J^+$; $\mu_s = 1$. Calculate $v_j = \{a_j/a_s\}$ for $j \in J^+$. Go to 4.

4. Take $p_j = \min(v_j, \mu_j)$ and calculate new column values $\alpha_j' = \alpha_j - p_j\alpha_s$ for $j \in J^+$, $j \neq s$. Designate the α_j' to be the current α_j. If all $v_j \leq \mu_j$, calculate $q = \{b_0/a_s\}$ and the new column $\beta' = \beta + q\alpha_s$; designate β' to be the current β and return to 2. Otherwise, some $v_j > \mu_j$.

The row selected in step 2 has new values $-b_0, a_1, a_2, \ldots, a_n$. Define J^+ as the set of indices j where $a_j > 0$ and return to 3.

We have presented an algorithm in which a bounding form is developed, but a search procedure may be included to determine whether or not a bounding form already exists. Glover [2] presents methods to force faster solution.

Example

Find integers $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ that minimize z when

$$\begin{aligned} 5x_1 + 7x_2 + 11x_3 &= z, \\ 4x_1 + 5x_2 + 5x_3 &\geq 6, \\ x_1 + x_2 + 3x_3 &\geq 7, \\ 5x_1 + 3x_2 + 2x_3 &\geq 5. \end{aligned}$$

We write the steps as they appear in the algorithm.

1. The problem is listed in the following tableau.

		1	2	3
z	0	5	7	11
x_1	0	1	0	0
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−6	4	5	5
x_5	−7	1	1	3
x_6	−5	5	3	2

The integer surplus variables are x_4, x_5, x_6.

ITERATION 1.

2. The x_5 row is selected. $J^+ = (1, 2, 3)$.
3. $s = 1$. $\mu_1 = 1, \mu_2 = 1, \mu_3 = 2$; $v_1 = 1, v_2 = 1, v_3 = 3$.
4. $p_1 = 1, p_2 = 1, p_3 = 2$; $v_3 > \mu_3$. We form the tableau below. For the x_5 row $J^+ = (1, 3)$.

		1	2	3
z	0	5	2	1
x_1	0	1	−1	−2
x_2	0	0	1	0
x_3	0	0	0	1
x_4	−6	4	1	−3
x_5	−7	1	0	1
x_6	−5	5	−2	−8

3. $s = 3$. $\mu_1 = 5, \mu_3 = 1$; $v_1 = 1, v_3 = 1$.
4. $p_1 = 1, p_3 = 1$; $q = 7$. We form the following tableau.

		1	2	3
z	7	4	2	1
x_1	−14	3	−1	−2
x_2	0	0	1	0
x_3	7	−1	0	1
x_4	−27	7	1	−3
x_5	0	0	0	1
x_6	−61	13	−2	−8

ITERATION 2.

2. The x_6 row is selected. $J^+ = (1)$.
3. $s = 1$. $\mu_1 = 1$; $v_1 = 1$.
4. $p_1 = 1$; $q = 5$. We form the following tableau.

		1	2	3
z	27	4	2	1
x_1	1	3	−1	−2
x_2	0	0	1	0
x_3	2	−1	0	1
x_4	8	7	1	−3
x_5	0	0	0	1
x_6	4	13	−2	−8

ITERATION 3.

The minimal solution is reached in the tableau in Iteration 2. It is $z = 27, x_1 = 1, x_2 = 0, x_3 = 2$.

8 A PRIMAL INTEGER METHOD

A primal integer programming method uses the primal simplex algorithm so that feasible values of the basic variables are present at each iteration. The advantages of the primal method are:

(1) a feasible solution obtained by any means may be used as a starting point for the algorithm, and

(2) a feasible solution is available at any time in the calculations.

The simple all-integer primal algorithm which we present has been known for a considerable time and may be attributed to Gomory. Assuming that a feasible canonical form is available initially, the problem is: Find integer x_j for $j = 1, 2, \ldots, n, n+1, \ldots, n+m$ that minimize z when

$$-z + \sum_{j=1}^{n} c_j x_j = 0,$$

$$x_{n+i} + \sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m, \tag{43}$$

$$x_j \geq 0, \qquad j = 1, 2, \ldots, n, n+1, \ldots, n+m,$$

and the a_{ij}, b_i, and c_j are given integer constants. In addition, $b_i \geq 0$ so that a feasible solution $x_{n+i} = b_i$ exists.

To find the optimal solution, we make the transformation

$$x_j = d_j - \sum_{k=1}^{n} d_{jk} y_k, \qquad j = 1, 2, \ldots, n. \tag{44}$$

We develop the integer constants d_j, d_{jk} in an iterative process during the

solution of the problem. The initial transformation is established by writing (43) as

$$
\begin{aligned}
-z + \sum_{j=1}^{n} c_j y_j &= 0, \\
x_j - y_j &= 0, \qquad j = 1, 2, \ldots, n, \\
x_{n+i} + \sum_{j=1}^{n} a_{ij} y_j &= b_i, \qquad i = 1, 2, \ldots, m.
\end{aligned}
$$

Eliminating x_j from (43), using (44), we obtain the equivalent problem: Find integer $y_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$
\begin{aligned}
-z + \sum_{j=1}^{n} \bar{c}_j y_j &= -\bar{z}_0, \\
x_j + \sum_{k=1}^{n} d_{jk} y_k &= d_j, \qquad j = 1, 2, \ldots, n, \\
x_{n+i} + \sum_{j=1}^{n} \bar{a}_{ij} y_j &= \bar{b}_i, \qquad i = 1, 2, \ldots, m.
\end{aligned} \tag{45}
$$

The constants, $\bar{z}_0$, $\bar{c}_j$, $\bar{a}_{ij}$, and $\bar{b}_i$ are developed from the transformation. If the constants are such that all $\bar{c}_j \geq 0$, $\bar{b}_i \geq 0$, and $d_j \geq 0$, then the minimal solution to (43) is given by $z = \bar{z}_0$, $x_j = d_j$ for $j = 1, 2, \ldots, n$ and $x_{n+i} = \bar{b}_i$ for $i = 1, 2, \ldots, m$.

Equation (45) can be written as

$$
x + \sum_{j=1}^{n} \alpha_j y_j = \beta, \tag{46}
$$

where x is a column vector with components $-z$, x_1, x_2, $\ldots$, x_{n+m}, α_j is a column vector with components $\bar{c}_j, d_{1j}, d_{2j}, \ldots, d_{nj}, \bar{a}_{1j}, \bar{a}_{2j}, \ldots, \bar{a}_{mj}$, and β is a column vector with components $-\bar{z}_0$, d_1, d_2, $\ldots$, d_n, $\bar{b}_1$, $\bar{b}_2$, $\ldots$, $\bar{b}_m$. Initially, $\bar{c}_j = c_j$, $d_{ij} = 0$ for $i \neq j$, $d_{jj} = -1$, $\bar{z}_0 = 0$, $d_j = 0$, $\bar{b}_i = b_i$.

After some iteration, suppose equations like those in (45) are obtained where all $\bar{b}_i \geq 0$, $d_j \geq 0$, and some $\bar{c}_j$ are negative; we develop the transformation that converts (46) to an optimal format. We determine index s from $\bar{c}_s = \min \bar{c}_j < 0$; if all $\bar{a}_{is} \leq 0$ and $d_{js} \leq 0$, the solution

is unbounded. Otherwise, find $\theta = \min(\bar{b}_i/\bar{a}_{is}, d_j/d_{js})$, where the minimum is over indices i and j so that $\bar{a}_{is} > 0$ and $d_{js} > 0$. Select some equation of (45) with $[\bar{b}_i/\bar{a}_{is}] \leq \theta$ or $[d_j/d_{js}] \leq \theta$, where $\bar{a}_{is} > 0$ and $d_{js} > 0$.

Let the selected equality be

$$x_r + \sum_{j=1}^{n} a_j y_j = b_0 . \tag{47}$$

Dividing through by $a_s > 0$ and noting that $a_j/a_s = [a_j/a_s] + r_j$ with $0 \leq r_j < 1$, (47) appears as

$$y_s + \sum_{j \neq s} p_j y_j = \frac{b_0}{a_s} - \sum_{j \neq s} r_j y_j - \frac{1}{a_s} x_r , \tag{48}$$

where $p_j = [a_j/a_s]$. From (48) the inequality

$$y_s + \sum_{j \neq s} p_j y_j \leq \frac{b_0}{a_s} \tag{49}$$

results. Since the left side of (49) can take on only integer values, a more restricted relation holds, i.e.,

$$y_s + \sum_{j \neq s} p_j y_j \leq q, \tag{50}$$

where $q = [b_0/a_s]$. Inequality (50) then leads to

$$y_s = q - \sum_{j \neq s} p_j y_j - y_s', \tag{51}$$

where integer y_s' represents the slack variable in (50). If y_s from (51) is substituted for the y_s in (46), the latter becomes

$$x + \sum_{j=1}^{n} \alpha_j' y_j = \beta', \tag{52}$$

where

$$\begin{aligned} \alpha_j' &= \alpha_j - p_j \alpha_s , \qquad j \neq s, \\ \alpha_s' &= -\alpha_s , \\ \beta' &= \beta - q\alpha_s , \end{aligned}$$

and y_s' is redefined as y_s. Designate α_j' and β' as the current α_j and β;

(52) is again of the form given by (46). The same process can be repeated until an iteration is reached in which the optimality conditions hold. We are now ready to state a primal simplex algorithm for the solution to (43).

Algorithm 4

1. Develop a tableau by listing the columns $\alpha_1, \alpha_2, \ldots, \alpha_n, \beta$. Before any iteration α_j has components $\bar{c}_j, d_{1j}, d_{2j}, \ldots, d_{nj}, \bar{a}_{1j}, \bar{a}_{2j}, \ldots, \bar{a}_{mj}$; column β has components $-\bar{z}_0, d_1, d_2, \ldots, d_n, \bar{b}_1, \bar{b}_2, \ldots, \bar{b}_m$. Initially, $\bar{c}_j = c_j$, $d_{jj} = -1$, $d_{ij} = 0$ for $i \neq j$, $\bar{a}_{ij} = a_{ij}$; $\bar{z}_0 = 0$, $d_j = 0$, $\bar{b}_i = b_i$. Go to 2.

2. If $\bar{c}_j \geq 0$ for $j = 1, 2, \ldots, n$, the optimal solution is $z = \bar{z}_0, x_j = d_j$, for $j = 1, 2, \ldots, n$ and $x_{n+i} = \bar{b}_i$, for $i = 1, 2, \ldots, m$; stop.
Otherwise, select index s by $\bar{c}_s = \min \bar{c}_j < 0$. Go to 3.

3. If all $d_{js} \leq 0$ and $\bar{a}_{is} \leq 0$, the solution is unbounded; stop. Otherwise, calculate $\theta = \min(\bar{b}_i/\bar{a}_{is}, d_j/d_{js})$ for $\bar{a}_{is} > 0$ and $d_{js} > 0$. Select a row with the property that $[d_j/d_{js}] \leq \theta$ or $[\bar{b}_i/\bar{a}_{is}] \leq \theta$, where $d_{js} > 0$ and $\bar{a}_{is} > 0$. Suppose the row is $a_1, a_2, \ldots, a_n, b_0$. Calculate $p_j = [a_j/a_s]$ for $j = 1, 2, \ldots, n$ and $q = [b_0/a_s]$. Calculate new column j values, $j \neq s$, by multiplying the values of column s by p_j and subtracting the result from column j. Calculate a new constant value column by multiplying the values of column s by q and subtracting the result from the constant value column. Finally, change the sign of each element of column s. Go to 2.

Example

Find integers $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ that minimize z when

$$\begin{aligned} -2x_1 - 3x_2 + x_3 &= z, \\ 4x_1 - x_2 - 3x_3 &\leq 5, \\ -2x_1 + 2x_2 + 3x_3 &\leq 7. \end{aligned}$$

We introduce slack variables x_4 and x_5 to achieve the feasible solution $x_4 = 5$, $x_5 = 7$. We write the steps as they occur in the algorithm.

1. The problem is listed in the tableau below.

	1	2	3	
$-z$	−2	−3	1	0
x_1	−1	0	0	0
x_2	0	−1	0	0
x_3	0	0	−1	0
x_4	4	−1	−3	5
x_5	−2	2	3	7

ITERATION 1.

2. $s = 2$.

3. The x_5 row is selected; $a_s = 2$, $p_1 = -1$, $p_2 = 1$, $p_3 = 1$, $q = 3$. We form the following tableau.

	1	2	3	
$-z$	−5	3	4	9
x_1	−1	0	0	0
x_2	−1	1	1	3
x_3	0	0	−1	0
x_4	3	1	−2	8
x_5	0	−2	1	1

ITERATION 2.

2. $s = 1$.

3. The x_4 row is selected; $a_s = 3$, $p_1 = 1$, $p_2 = 0$, $p_3 = -1$, $q = 2$. We form the tableau below.

	1	2	3	
$-z$	5	3	−1	19
x_1	1	0	−1	2
x_2	1	1	0	5
x_3	0	0	−1	0
x_4	−3	1	1	2
x_5	0	−2	1	1

ITERATION 3.

2. $s = 3$.
3. The x_5 row is selected; $a_s = 1$, $p_1 = 0$, $p_2 = -2$, $p_3 = 1$, $q = 1$. We form the following tableau.

	1	2	3	
$-z$	5	1	1	20
x_1	1	−2	1	3
x_2	1	1	0	5
x_3	0	−2	1	1
x_4	−3	3	−1	1
x_5	0	0	−1	0

ITERATION 4.

The minimal solution is reached in the tableau in Iteration 3. It is $z = -20, x_1 = 3, x_2 = 5, x_3 = 1, x_4 = 1, x_5 = 0$.

Algorithm 4 is not very useful in computational work because it generally requires many iterations even in very small problems. Since it is possible that $q = [b_0/a_s]$ may be zero for each iteration, the algorithm may not produce an optimal solution in a finite number of iterations. Young and Glover present means for preventing this occurrence, but the resulting algorithms converge very slowly.

Problems

1. Show that the m equalities

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m,$$

and $m + 1$ inequalities

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m,$$

$$\sum_{j=1}^{n} h_j x_j \geq h$$

are equivalent where

$$h_j = -\sum_{i=1}^{m} a_{ij},$$

$$h = -\sum_{i=1}^{m} b_i.$$

Thus equality constraint integer programming problems can be converted to inequality constraint problems.

2. Minimize z in the following problem by means of the all-integer algorithm:

$$\begin{aligned} 3x_1 + 2x_2 + 6x_3 &= z, \\ 2x_1 - 3x_2 + 4x_3 &\geq 5, \\ x_1 + 4x_2 - 5x_3 &\geq 8, \\ 3x_1 - 2x_2 - x_3 &\geq 4, \\ x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 &\geq 0. \end{aligned}$$

3. Find min z and integers $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ for

$$\begin{aligned} 2x_1 + 3x_2 + 2x_3 &= z, \\ 3x_1 - 2x_2 - x_3 &\geq 4, \\ -2x_1 - 2x_2 + x_3 &\geq 5, \\ -x_1 + 3x_2 - x_3 &\geq 3. \end{aligned}$$

4. Solve the negative c_j value problem

$$\begin{aligned} 2x_1 - x_2 + x_3 &= z, \\ x_1 + 2x_2 - x_3 &\geq 6, \\ -x_1 + 2x_2 - x_3 &\geq 2, \\ x_1 - x_2 + 2x_3 &\geq 4. \end{aligned}$$

5. Solve by the bounding form method

$$\begin{aligned} 3x_1 + 2x_2 + 4x_3 &= z, \\ x_1 - 2x_2 + 5x_3 &\geq 3, \\ 6x_1 + x_2 + 3x_3 &\geq 4, \\ 4x_1 + 7x_2 - 2x_3 &\geq 2. \end{aligned}$$

6. Minimize z when $x_j = 0$ or 1 for $j = 1, 2, 3$

$$\begin{aligned} x_1 + 2x_2 + x_3 &= z, \\ 2x_1 + 2x_2 + x_3 &\geq 3, \\ 2x_1 - 2x_2 + x_3 &\geq 1, \\ x_1 + 2x_2 - 2x_3 &\geq 1. \end{aligned}$$

7. Solve by the primal integer method: min z for integers $x_1 \geq 0$ and $x_2 \geq 0$ when

$$\begin{aligned} -x_1 - x_2 &= z, \\ 3x_1 + 2x_2 &\leq 7. \end{aligned}$$

References

1. Glover, F., A bound escalation method for the solution of integer programs, *Centre D'Etudes Rech. Oper.* **6** (3), 131 (1964).
2. Glover, F., A new foundation for a simplified primal integer programming algorithm, *Operations Res.* **16** (4), 727 (1968).
3. Gomory, R. E., An all-integer programming algorithm, *in* "Industrial Scheduling" (J. F. Muth and G. L. Thompson, eds.), pp. 193–206. Prentice-Hall, Englewood Cliffs, New Jersey, 1963.
4. Young, R. D., A simplified primal (all-integer) integer programming algorithm *Operations Res.*, **16** (4), 750 (1968).

4 SOLVING INTEGER PROGRAMS BY ENUMERATION

An enumeration method for solving integer programs often has advantages over other methods. Obviously, since the variables take on only discrete values, they can be listed easily, although they may be numerous. For the procedure to be manageable, the enumeration should be ordered in such a way that solutions are obtained with a minimal amount of calculation.

We enumerate all possible values of the objective function for an integer program and demonstrate how to use the enumeration in finding the solution to the program. We also provide rules to accelerate the enumerative process. The enumeration is given by a dynamic programming formulation of the problem. (See Bellman [3] for a general presentation of dynamic programming.) In conclusion we give an enumerative solution to the knapsack problem.

1 A DIRECT ENUMERATION METHOD

We shall find the solution to the integer programming problem: Find integer values of $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{n} c_j x_j = z, \\ &\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m. \end{aligned} \tag{1}$$

To solve (1) directly by enumeration, we begin by finding all the values of z from

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{2}$$

that are produced by nonnegative integer values of x_j where the c_j are positive numbers. Equation (2) is the objective function of the integer program (1). We find all feasible values of z as a monotonic increasing sequence. For a feasible value of z, say z_0, we also find the x_j values that produce z_0.

In the process of developing the monotonic sequence of the z, we obtain the solution to (1) when the smallest z value in the sequence has corresponding x_j values that satisfy the constraints. Since the x_j values produce the smallest objective function value consistent with the constraints, the solution must be optimal. Thus the enumeration of (2) is performed in order of increasing values and stopped when the constraints are satisfied.

The method for generating feasible values of z from (2) is contained in

Theorem 1

If z_0 is a feasible value for $z = \sum_{j=1}^{n} c_j x_j$ that is produced by integer values x_j^0 for $j = 1, 2, \ldots, n$, then other feasible values of z are produced by $z_0 + c_j$ for $j = 1, 2, \ldots, n$.

Proof

Since $z_0 = \sum_{j=1}^{n} c_j x_j^0$, we can form another feasible z value by increasing any x_j^0 value, say x_k^0, by one. The result is yet another feasible z value given by $z = \sum_{j=1}^{n} c_j x_j^0 + c_k = z_0 + c_k$. Other feasible z values then can be formed by $z_0 + c_k$ for $k = 1, 2, \ldots, n$.

In Theorem 1 we see how an enumeration of z values can be performed when a feasible z already exists: Given a list of feasible z, additional feasible z are found by using a z value, such as z_0, as generator and forming $z_0 + c_j$ for $j = 1, 2, \ldots, n$. We then add the new z values

to the list and select another z value as generator. If we select the smallest unused z, we insure that every z value will act as a generator. The only problem is to find the first generator. Since each feasible z value has corresponding x_j values, the first generator is $z = 0$ with $x_j = 0$.

We give the formal algorithm for generating all feasible z values and then prove that no z values are omitted using the algorithm.

Algorithm 1

1. List the values of the problem as

1	2	3	...	n
c_1	c_2	c_3	...	c_n
$x_j = 0$				

Go to 2.

2. Given the list, find $c_r = \min c_j$ for all unmarked columns in all sections. Take $c = c_r$ and go to 3.

3. Add a new section of columns to the list as follows:
   (a) Calculate $c_j' = c + c_j$ for the indices j of the unmarked columns in the section containing column r. The values of c_j used are from the first section.
   (b) Mark the r column.
   (c) Add columns headed by the prescribed indices in a new section with values c_j'. Consider the c_j' as new c_j values.
   (d) Underneath the section added, write the x_j values from the section containing the newly marked r column. Increase x_r by one for the new section. Go to 2.

This concludes the algorithm.

There are no feasible z values omitted by the algorithm as seen by

Theorem 2

Every feasible value of z for $z = \sum_{j=1}^{n} c_j x_j$ with integer $x_j \geq 0$ is listed by Algorithm 1.

Proof

The values of z generated by $z = 0$ are c_j for $j = 1, 2, \ldots, n$ and are listed in the algorithm. Suppose z^* is a feasible value that is not listed; we have $z^* = \sum_{j=1}^{n} c_j x_j^*$. If $z^* > 0$ is feasible, then some x_j^*, say x_k^*, is positive. Another feasible z exists for $x_k = x_k^* - 1$, $x_j = x_j^*$, $j \neq k$, and has value $z' = \sum_{j=1}^{n} c_j x_j^* - c_k$ and we have $z' = z^* - c_k$. Thus, z^* would be generated by z using Theorem 1. Furthermore, z' is not listed by the algorithm; if it were, it would generate the value z^* which would be listed. If $z' > 0$, then it can be generated by some smaller value of z which is not on the list. Each value of $z > 0$ not on the list can be generated by a smaller value not on the list. We can find smaller and smaller feasible z values that are not produced by the algorithm and which will generate z values that lead to z^*. Since we can always find a smaller feasible z as generator and because there are only a finite number of z values smaller than z^*, we see that $z = 0$ must be a generator and thus must generate all values of z that lead to z^*. Moreover, $z = 0$ cannot generate values of z on the list that lead to z^*. All possible feasible values of z generated by $z = 0$, however, are on the list; these are the c_j. Thus we have a contradiction and all feasible values of z are produced by Algorithm 1.

We have just shown that Theorem 1 leads to Algorithm 1 for the enumeration of z and corresponding x_j values that satisfy $z = \sum_{j=1}^{n} c_j x_j$ with integer $x_j \geq 0$ and $c_j > 0$. Theorem 2 proves that every feasible z value is enumerated by the algorithm.

Example

Find feasible z that satisfy

$$z = 3x_1 + 4x_2 + 7x_3$$

for $x_1, x_2, x_3 \geq 0$ and integer. We list

(3)

1	2	3
3*	4*	7
$x_1, x_2, x_3 = 0$		

Note that min c_j is the 3 of column 1. We mark the 3 and form

(4)

1	2	3
6*	7	10
$x_1 = 1$		

The minimum unmarked element is the 4 in column 2 of (3). We mark the 4 and form

(5)

2	3
8	11
$x_2 = 1$	

We do not have a column headed by 1 in (5) because it would duplicate the column headed by 2 in (4). Both would require that $x_1 = 1$, $x_2 = 1$. This storage reduction is handled by step 3(a) of the algorithm. The 6 from (4) generates the next values

1	2	3
9	10	13
$x_1 = 2$		

As the method continues, the list in subsequent steps becomes

3	2 3	2 3	1 2 3	3	2 3	3	2 3
14	11 14	12 15	12 13 16	17	14 17	18	15 18
$x_3 = 1$	$x_1 = 1$ $x_2 = 1$	$x_2 = 2$	$x_1 = 3$	$x_1 = 1$ $x_3 = 1$	$x_1 = 2$ $x_2 = 1$	$x_2 = 1$ $x_3 = 1$	$x_1 = 1$ $x_2 = 1$

The list goes on indefinitely since z may take on unbounded values.

Feasible values of z are listed in order as

z	0	3	4	6	7	7	8	9	10	10	11	11	12	12	13	13	14	14	14	15	15	15
x_1	0	1	0	2	0	1	0	3	1	2	0	1	0	4	2	0	0	1	2	0	1	5
x_2	0	0	1	0	0	1	2	0	0	1	1	2	3	0	0	2	0	1	2	2	3	0
x_3	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	2	1	0	1	0	0

2 SOLUTION TO THE INTEGER PROGRAM

We are now ready to solve Eq. (1) by considering the problem: Find $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$c_1 x_1 + c_2 x_2 + \cdots + c_n x_n = z, \tag{6}$$
$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = \alpha,$$

where the α_j are column vectors with elements a_{ij} and α is a feasible variable column vector. Take α_0 as a column vector with elements b_i.

As in Section 1, we enumerate values for z in (6) as a monotonic increasing function. This time, however, the corresponding x_j values produce values for α in the constraints. We achieve a solution to (1) for the smallest z value and corresponding x_j values that produce a vector α with the property that $\alpha \geq \alpha_0$ (each component of α is greater than or equal to the corresponding component of α_0).

We can remove the $c_j > 0$ restriction in solving (6). If Algorithm 1 is used where $c_r = \min\ c_j \leq 0$, the enumeration will continue by increasing only x_r without limit. In a particular problem, however, x_r may be bounded either implicity through the constraints of (1) or by having an upper bound initially. In either case we will assume that

$$x_j \leq m_j, \qquad j = 1, 2, \ldots, n; \tag{7}$$

m_j are bounds on the variables (any m_j value may be infinite). For those

indices j where $c_j \leq 0$, we must first attempt to find a finite m_j for variable x_j. If a variable is not initially bounded, then a bound, if it exists, may be obtained by solving the linear program: Maximize x_k subject to the constraints of (1). The variable x_k is simply one of the variables from (1) for which it is desirable to achieve a bound. If $x_k{}^0$ is the resultant value of max x_k, then we have $x_k \leq [x_k{}^0]$ where $[x_k{}^0]$ is the integer part of $x_k{}^0$.

While it is most useful to bound each variable, it may not be practical because of the need to solve a linear programming problem for each variable. An alternate strategy is to solve the single linear program: Maximize $\sum_{j \in J} x_j$ subject to the constraints of (1), J being the set of integers j where $c_j \leq 0$. We take the resultant bound as integer d_0 and have $\sum_{j \in J} x_j \leq d_0$.

For the solution of (1) with the additional constraints $x_j \leq m_j$ for $j = 1, 2, \ldots, n$, we present Algorithm 2. If $c_j > 0$, then m_j may be infinite; if $c_j \leq 0$, then m_j must be finite for the method to solve every problem. In any case, m_j may also be prescribed by the problem. (We see, for example, that $m_j = 1$ restricts the variable x_j to being zero or one, which constitutes an important class of integer programs.) For each $c_j < 0$ we make the change $x_j' = m_j - x_j$. We can then assume all $c_j \geq 0$. Thus we have

Algorithm 2

1. Define solution vector $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$, where x_j^* for $j = 1, 2, \ldots, n$ is a feasible integer solution to the problem with objective value z^*. If no feasible solution is apparent take $z^* = \infty$. Go to 2.

2. List the values of problem (1) as

1	2	3	...	n
c_1	c_2	c_3	...	c_n
α_1	α_2	α_3	...	α_n

;

α_0 and the m_j are also listed. Component i of α_j is a_{ij}, $j \neq 0$. Component i of α_0 is b_i. Go to 3.

3. In the newly listed section, find index r from $c_r = \min c_j$ for indices j with $\alpha_j \geq \alpha_0$. Mark each such column. If no such column exists or $c_r \geq z^*$, go to 4. Otherwise take $z^* = c_r$ and form $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$ where the x_j^* are the nonzero x_j values found below the new section. The other x_j^* values equal zero. Increase x_r^* by one. Go to 4.

4. Given the list, the solution $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$ is the minimal integer solution if there are no unmarked columns with $c_j < z^*$. Stop. Otherwise, find index r from $c_r = \min c_j$ for all unmarked columns in all sections. Take $c = c_r$ anc $\alpha = \alpha_r$. Go to 5.

5. Add a new section of columns to the list, if possible, as follows:
   (a) Calculate $c_j' = c + c_j$ and $\alpha_j' = \alpha + \alpha_j$ for the indices j of the unmarked columns in the section containing column r. The c_j, α_j values are taken from the list in step 2. Mark the r column.
   (b) Add columns with values c_j' and α_j' headed by the prescribed indices j if $c_j < z^*$. Include a column headed by r only if $x_r + 1 < m_r$ for the x_r value found below the section containing the newly marked r column. Designate the c_j' and α_j' values as c_j and α_j values respectively.
   (c) Underneath the section added write the x_j values from the section containing the newly marked r column. The non-appearance of a variable means it has zero value. Increase x_r by one for the new section. Go to 3. If no new section is added, return to 4.

While enumerating z values for $z = \Sigma c_j x_j$ in the algorithm, we are simultaneously enumerating α values for $\alpha = \Sigma \alpha_j x_j$. The α enumeration is performed with α_r as generator. We form new values of α from $\alpha_r + \alpha_j$ for $j = 1, 2, \ldots, n$. As in the z enumeration, the method succeeds in enumerating all feasible α values. The order of enumeration of the z values directs the α enumeration and permits all feasible α to be produced.

The solution vector $S(z, x_1, x_2, \ldots, x_n)$ is used in the algorithm to reduce the amount of computation. If z and corresponding x_j values are listed and feasible, then there is no need to list the same or larger z

values. In addition, if we have a bound for the variables given by $\sum_{j \in J} x_j \leq d_0$, we modify Algorithm 2 by the variable change $x_j' = d_0 - x_j$ for $j \in J$. We can then assume all $c_j \geq 0$. List the values of the problem in step 2 as

1	2	3	...	n
c_1	c_2	c_3	...	c_n
α_1	α_2	α_3	...	α_n
a_1	a_2	a_3	...	a_n

Consider the bound inequality as $\sum_{j=1}^{n} a_j x_j \leq d_0$. Thus we have $a_j = 1$ if $j \in J$ and $a_j = 0$ otherwise.

In step 5(a) of the algorithm, calculate $c_j' = c_r + c_j$, $\alpha_j' = \alpha_r + \alpha_j$, $a_j' = a_r + a_j$ for the indices j of the unmarked columns in the section containing column r. The c_r, α_r, and a_r values are from the section containing column r. The c_j, α_j, and a_j values are taken from the list in step 2. Mark the r column.

In step 5(b), add columns with values c_j', α_j', and a_j' headed by the prescribed indices j. Include the j column only if $a_j' \leq d_0$. Also, include the r column only if $x_r + 1 < m_r$ for the x_r value found below the section containing the newly marked r column. Designate the c_j', α_j', and a_j' values as c_j, α_j, and a_j values, respectively.

Algorithm 2 succeeds in solving the integer program with a finite number of steps when a finite minimal solution exists. We enumerate all objective values $z \leq z_0$ where z_0 is the optimal one. We also find all x_j values that produce each z. Since there are only a finite number of integer values less than or equal to z_0, the minimal solution must be listed in a finite number of steps.

We do not claim any efficiency for the direct enumeration method, although it works well when the optimal x_j values are small. Because the two operations performed are additions and comparisons, the method is particularly amenable to computer calculation and can readily be coded to produce extremely rapid solutions to some integer programs. In Section 3 we show how to accelerate the enumeration.

Example

Consider the problem: Find integers $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$ that minimize z when

$$\begin{aligned} 3x_1 + 4x_2 + 7x_3 &= z, \\ 3x_1 + 2x_2 + 3x_3 &\geq 8, \\ 4x_1 - x_2 - 2x_3 &\geq 6, \\ -x_1 + 3x_2 + 4x_3 &\geq 2. \end{aligned}$$

We list the steps as they occur in the algorithm.

1. $z^* = \infty$.
2. The problem is listed in Tableau E1; $\alpha_0 = (8, 6, 2)$, all $m_j = \infty$.

1*	2*	3*	1*	2*	3	2	3	1	2	3*	2	2
3	4	7	6	7	10	8	11	9	10	13	11	12
3	2	3	6	5	6	4	5	9	8	9	7	6
4	−1	−2	8	3	2	−2	−3	12	7	6	2	−3
−1	3	4	−2	2	3	6	7	−3	1	2	5	9
			$x_1 = 1$			$x_2 = 1$		$x_1 = 2$			$x_1 = 1$ $x_2 = 1$	$x_2 = 2$
E1			E2			E3		E4			E5	E6

Tableaus

4. $r = 1$ in Tableau E1, $c_r = 3$.
5. We form Tableau E2. Mark column 1 of Tableau E1 (the marking is shown by the *).
4. $r = 2$ in Tableau E1, $c_r = 4$.
5. We form Tableau E3. Mark column 2 of Tableau E1.
4. $r = 1$ in Tableau E2, $c_r = 6$.
5. We form Tableau E4. Mark column 1 of Tableau E2.
3. A feasible solution is apparent in Tableau E4. $r = 3$, $\alpha_r \geq \alpha_0$, where $\alpha_r = (9, 6, 2)$. $z^* = 13$, $x_1^* = 2$, $x_2^* = 0$, $x_3^* = 1$. Mark column 3.
4. $r = 1$ in Tableau E1, $c_r = 7$.
5. No new tableau is formed. Mark column 3 of Tableau E1.
4. $r = 2$ in Tableau E2, $c_r = 7$.

5. We form Tableau E5. Mark column 2 of Tableau E2.
4. $r = 2$ in Tableau E3, $c_r = 8$.
5. We form Tableau E6. Mark column 2 of Tableau E3.

The method continues easily. No new tableaus are formed. The feasible solution $z = 13, x_1 = 2, x_2 = 0, x_3 = 1$ is minimal.

3 AN ACCELERATED ENUMERATION

As we indicated in Section 2, it is sometimes desirable to accelerate the enumeration. This can be done once the variables become evaluated. We present several rules for making the variables known.

In (1) it may occur that a positive value for one of the x_j variables may not allow a feasible solution to be found; alternatively, some of the x_j must have positive values in order for the solution to be found. Similar conditions may exist when Algorithm 2 is applied in the search for the solution to (1).

Consider (1) as listed in step 2 of Algorithm 2. We form an index set K with initial members $j = 1, 2, \ldots, n$. We then form

$$U_i = \sum_{k \in K} a_{ik} y_k, \qquad i = 1, 2, \ldots, m; \tag{8}$$

y_k for $k \in K$ is a nonnegative integer that represents the subsequent increase in x_k. Thus $y_k \leq m_k'$ where $m_k' = m_k - x_k'$. Initially all $x_k' = 0$.

We require that $U_i \geq b_i$. Hence, we may be able to determine from (8) whether no y_k can produce feasibility or whether some y_k must have a value to produce feasibility. Take

$$U_i^* = \sum_{k \in K_i^+} a_{ik} m_k', \qquad i = 1, 2, \ldots, m, \tag{9}$$

Where K_i^+ is the set of indices in K with $a_{ik} > 0$ and U_i^* is the maximum value of U_i. If K_i^+ has no members, then $U_i^* = 0$. The following rules apply where initially $c = 0$ and $d_i = 0$ for $i = 1, 2, \ldots, m$. (These rules are extensions of the ones developed by Le Garff and Malgrange [6] and Balas [1] in the $m_j = 1$ case. For a concise presentation of the $m_j = 1$ case, see Beale [2].)

A1. If any $U_i^* < b_i - d_i$, then no y_k values can produce feasibility. No feasible solution to the problem can be found. Stop.

A2. If $U_i^* + a_{ij} < b_i - d_i$ for $j \in K$, then $y_j = 0$ since $y_j \geq 1$ produces infeasibility. Mark column j and remove index j from K. Write a new form for the U_i in (8) and calculate the U_i^* in (9). Return to rule A1.

A3. (a) If j is the only index in K_i^+ and $d_i < b_i$, then $a_{ij} y_j \geq b_i - d_i$ requires that $y_j \geq \theta$ where $\theta = \{(b_i - d_i)/a_{ij}\}$.[1]

(b) If $U_i^* - a_{ij} m_j' < b_i - d_i$ for $j \in K$, then $y_j \geq \theta$, where θ is the smallest integer with the property that $U_i^* - a_{ij} m_j' + a_{ij}\theta \geq b_i - d_i$. Any $y_j < \theta$ produces infeasibility.

(c) When θ is found in (a) or (b), increase x_j' by θ, decrease m_j' by θ, replace c by $c + c_j\theta$ and d_i by $d_i + a_{ij}\theta$. If $m_j' = 0$, remove index j from K. Write a new form for the U_i and calculate U_i^*. Return to rule A1.

If, as a result of adhering to these rules, all $x_j' = 0$, continue in the algorithm with step 3. If, on the other hand, some x_j' is positive, problem (1) can be solved only with $x_j \geq x_j'$ for all j. We then take value d_i as component i of α. If $\alpha \geq \alpha_0$, the problem is solved with $z = c$ and $x_j = x_j'$ for all j. Otherwise, add a new section of columns to the list. Each column is headed by an index $k \in K$. The column values are given by $c_k' = c + c_k$ and $\alpha_k' = \alpha + \alpha_k$. The c_k and α_k values are taken from the list in step 2. Underneath the section write the $x_j = x_j'$ values. Mark all unmarked columns on the list from step 2 and go to 3.

We are able, further, to evaluate the variables within Algorithm 2. In the enumeration to the solution of (6) we have a feasible solution to (1) whenever $\alpha = \alpha_r$ is achieved with the property that $\alpha_r \geq \alpha_0$. Each column k developed in Algorithm 2 is a listing of z and α that results when the x_j values appearing below the column section, with x_k increased by unity, are substituted in (6). It may be that these values of x_j do not allow a feasible α_r to be produced or that some of the x_j must be increased to obtain values of $\alpha_r \geq \alpha_0$. We can therefore modify Algorithm 2 to ascertain whether a current stage of the enumeration will lead to a feasible α or whether some of the x_j require specific values in order to produce a feasible α.

When index r is found from $c_r = \min c_j$ in step 4 (with $c = c_r$),

[1] This rule is useful if $m_j = \infty$. If m_j is finite, then rule (b) holds as well.

we define a set of columns K as follows:

(a) Form set K' where column $k \in K'$ if k is an index of one of the unmarked columns in the section containing column r.
(b) Remove k from K' if $c_k \geq z^*$ (the c_k value in the section).
(c) Remove r from K' if $x_r + 1 = m_r$ for the x_r value found below the section.
(d) Remove k from K' if $c + c_k \geq z^*$ where the c_k value is taken from the list in step 2.
(e) The remaining set K' is taken as set K.

Define the column values for α_r and $d_1, d_2, \ldots, d_m$. The possible values of z and component V_i of α that can be enumerated at this stage are given by

$$
\begin{aligned}
z &= c + \sum_{k \in K} c_k y_k, \\
V_i &= d_i + U_i, \qquad i = 1, 2, \ldots, m,
\end{aligned}
\tag{10}
$$

where U_i is given by (8) for the present set K; y_k is a nonnegative integer that represents the subsequent increase in x_k. Thus, $y_k \leq m_k'$ where $m_k' = m_k - x_k'$. The x_k' are the x_k values that appear below the section containing column r with $x_r' = x_r + 1$.

In terms of Eq. (1), we require that $V_i \geq b_i$. This means that we may be able to determine from (10) whether no y_k can produce feasibility or whether some y_k must have a value to produce feasibility. Consider

$$V_i^* = d_i + U_i^*, \qquad i = 1, 2, \ldots, m,$$

where U_i^* is given by (9) with K_i^+ as the set of indices in current K with $a_{ik} > 0$. Then V_i^* is the maximum value that V_i can attain when we perform the enumeration starting with the section containing column r. The following rules then apply.

B1. If any $V_i^* < b_i$, then no y_k values can produce feasibility. Mark column r and continue the algorithm in step 4.

B2. If $V_i^* + a_{ij} < b_i$ for $j \in K$, then $y_j = 0$ since $y_j \geq 1$ produces infeasibility. Remove index j from K and write new forms for the U_i. Calculate the V_i^* and return to rule B1.

B3. (a) If j is the only element in K_i^+ and $d_i < b_i$, then $d_i + a_{ij}y_j \geq b_i$ requires that $y_j \geq \theta$ where $\theta = \{(b_i - d_i)/a_{ij}\}$.

(b) If $V_i^* - a_{ij}m_j' < b_i$ for $j \in K$, then $y_j \geq \theta$ where θ is the smallest integer with the property that $V_i^* - a_{ij}m_j' + a_{ij}\theta \geq b_i$. Any $y_j < \theta$ produces infeasibility.

(c) When θ is found in (a) or (b) and if $c + c_j\theta \geq z^*$, then mark column r and continue the algorithm in step 4. Otherwise, increase x_j' by θ, decrease m_j' by θ, replace c by $c + c_j\theta$ and d_i by $d_i + a_{ij}\theta$. If $m_j' = 0$, remove j from K. If $c + c_k \geq z^*$, remove k from K.

Write new forms for the U_i; calculate V_i^* and return to rule B1.

When the application of the rules is completed, we return to the algorithm in a modified step 5. The modifications consist of the following:

5. (a) Mark column r. Take value d_i as component i of α. If $\alpha \geq \alpha_0$ take $z^* = c$ and form $S(z^*, x_1', x_2', \ldots, x_n')$; go to 4. Otherwise, go to 5 (b).

(b) If set K has no elements, go to 4. Otherwise, add a new section of columns to the list. Each column is headed by an index $k \in K$. The column values are given by $c_k' = c + c_k$ and $\alpha_k' = \alpha + \alpha_k$. The c_k, α_k values are taken from the list in step 2. Underneath the section write the $x_j = x_j'$ values. Go to 3.

Example

Find nonnegative integers $x_j \leq 1$ that minimize z when

$$\begin{aligned} 5x_1 + 8x_2 + 10x_3 + 2x_4 + x_5 &= z, \\ x_1 - 3x_2 + 5x_3 + x_4 - 3x_5 - 6x_6 &\geq 2, \\ -2x_1 + 6x_2 - x_3 - 3x_4 + 2x_5 + 2x_6 &\geq 2, \\ -x_1 - x_2 + 2x_3 - x_4 + x_6 &\geq 1. \end{aligned}$$

We list the problem values in Tableau E1 with $\alpha_0 = (2, 2, 1)$ and all $m_j = 1$. We obtain $U_1^* = 7$, $U_2^* = 10$, $U_3^* = 3$ with $K = (1, 2, \ldots, 6)$. For rule A2, $U_1^* + a_{16} = 1 < 2$; we mark column 6 and remove the 6 from K. Hence, $U_1^* = 7$, $U_2^* = 8$, $U_3^* = 2$. For rule A3, $y_3 \geq \theta = 1$;

$x_3' = 1$, $c = 10$, $d_1 = 5$, $d_2 = -1$, $d_3 = 2$. We remove 3 from K; no further rules apply and we form Tableau E2. All unmarked columns in tableau E1 are marked.

1	2	3	4	5	6*
5	8	10	3	1	0
1	−3	5	1	−3	−6
−2	6	−1	−3	2	2
−1	−1	2	−1	0	1

Tableau E1

1	2*	4	5
15	18	13	11
6	2	6	2
−3	5	−4	1
1	1	1	2
$x_3 = 1$			

Tableau E2

A feasible solution is apparent in column 2 of Tableau E2. We have $z^* = 18$, $x_2^* = 1$, $x_3^* = 1$, $x_1^* = 0$, $x_4^* = 0$, $x_5^* = 0$, $x_6^* = 0$; mark column 2. In step 4 of Algorithm 2, $r = 5$ in Tableau E2 with $c_r = 11$. We obtain $K = (1, 4)$ and $V_1^* = 4$, $V_2^* = 1 < 2$, $V_3^* = 2$ for rule B1. We mark column 5 and continue the algorithm. No other column can pass rule B1. The current feasible solution is minimal.

4 A DYNAMIC PROGRAMMING METHOD

The enumeration in Section 2 may be interpreted as a dynamic programming procedure. In this section we proceed to formalize the method in terms of the dynamic programming equations.

We define the function

$$F(\alpha) = \min\left(\sum_{j=1}^{n} c_j x_j \,\middle|\, \sum_{j=1}^{n} \alpha_j x_j = \alpha, \quad \text{integer } x_j \geq 0 \right), \tag{11}$$
$$F(0) = 0,$$

where α is a feasible variable vector. Problem (6) is equivalent to the functional relationship given by (11). Note the the zero argument of $F(0)$ is a vector with zero elements. We are able to convert (11) to a dynamic programming recursion in

Theorem 3

The functional relationship (11) leads to the recursion

$$F(\alpha) = \min_j (c_j + F(\alpha - \alpha_j)),$$
$$F(0) = 0, \tag{12}$$

where α is feasible. (The recursion given in (12) was first formulated by Gilmore and Gomory [4] for knapsack functions.)

Proof

Consider (11) with $\alpha \neq 0$; then some x_j is positive. Suppose we have x_k positive; then

$$F(\alpha) = \min\left(c_k + \sum_{j \neq k} c_j x_j + c_k(x_k - 1) \middle| \sum_{j \neq k} \alpha_j x_j + \alpha_k(x_k - 1) = \alpha - \alpha_k, \right.$$
$$\left. \text{integer } x_j \geq 0\right).$$

Thus

$$F(\alpha) = c_k + \min\left(\sum_{j=1}^{n} c_j x_j' \middle| \sum_{j=1}^{n} \alpha_j x_j' = \alpha - \alpha_k', \quad \text{integer } x_j' \geq 0\right), \tag{13}$$

where the change of variables $x_j' = x_j, j \neq k$, $x_k' = x_k - 1$ is made; note that the min term is $F(\alpha - \alpha_k)$ from definition (11). We have

$$F(\alpha) = c_k + F(\alpha - \alpha_k). \tag{14}$$

Equation (14) holds in the case where $x_k > 0$; hence, we have

$$F(\alpha) \leq c_j + F(\alpha - \alpha_j), \qquad j = 1, 2, \ldots, n, \tag{15}$$

in all cases. Inequality (15) is another way of writing (12) provided that the equality holds for at least one j, as in (14). Since at least one x_j is positive for $\alpha \neq 0$, we have achieved (12) and proven the theorem.

The recursion in (12) may be solved as a direct enumeration because $F(\alpha_r) = c_r$, where $c_r = \min c_j$. Thus we produce one solution value for $F(\alpha)$. Replacing α by $\alpha - \alpha_r$ in (12), we form

$$F(\alpha - \alpha_r) = \min_j (c_j + F(\alpha - \alpha_j - \alpha_r)),$$

which we substitute for the $F(\alpha - \alpha_r)$ term on the right side of (12). We then obtain

$$(16) \qquad F(\alpha) = \min(\min_j(c_j' + F(\alpha - \alpha_j')), \min_{j \neq r}(c_j + F(\alpha - \alpha_j))),$$

where $c_j' = c_j + c_r$ and $\alpha_j' = \alpha_j + \alpha_r$ for $j = 1, 2, \ldots, n$. Designate c_j' and α_j' as c_j and α_j values. Note that (16) is of the same form as (12) except that the original $c_r + F(\alpha - \alpha_r)$ term is missing and other terms are included. Since we have the same form as (12), we obtain the solution value $F(\alpha_r) = c_r$ by finding a new $c_r = \min c_j$. In this way we find $F(\alpha)$ for all feasible α. By keeping track of the indices j that produce each $F(\alpha_r)$, we obtain the corresponding x_j values.

The method of solution just outlined is the same as given in Section 2. Algorithm 2 conveniently lists the values of $F(\alpha)$ at any stage of the enumeration.

5 KNAPSACK FUNCTIONS

The knapsack problem and its applications were introduced in Chapter 1. Following the author's [5] approach, we shall present a dynamic programming solution to the problem. We define the knapsack problem as: Find integers $x_j \geq 0$ that maximize z when

$$\sum_{j=1}^{n} c_j x_j = z,$$

$$\sum_{j=1}^{n} a_j x_j \leq L,$$

where each c_j is a positive number, each a_j is a positive integer, and L is a positive integer.

We express the one-dimensional knapsack function as

$$(17) \qquad F(x) = \max\left(\sum_{j=1}^{n} c_j x_j \middle| \sum_{j=1}^{n} a_j x_j = x, \quad \text{integer } x_j \geq 0 \right)$$

for feasible values of x. (See Gilmore and Gomory [4] for higher dimensional knapsack functions.) The knapsack problem is solved when we find the maximum of $F(x)$ for $x \leq L$.

In a manner like that used in Section 4, the knapsack function can be written as the dynamic programming recursion

$$F(x) = \max_{j:a_j \leq x} (c_j + F(x - a_j)), \tag{18}$$
$$F(0) = 0.$$

The maximum in (18) is for indices j with the property that $a_j \leq x$. This occurs because the x_j values are zero when the corresponding a_j values are greater than x in (17).

We can obtain an immediate solution to (18) by finding $a_r = \min a_j$ for all j. If $a_k = a_r$ for $k \neq r$, then index r is chosen where $c_r \geq c_k$. We take $x = a_r$ and produce $F(a_r) = c_r$. We then replace x by $x - a_r$ in (18) and have

$$F(x - a_r) = \max_{j:a_r + a_j \leq x} (c_j + F(x - a_r - a_j)),$$

which we substitute for the $F(x - a_r)$ term on the right side of (18). Hence,

$$F(x) = \max\Big(\max_{j \neq r:a_j \leq x} (c_j + F(x - a_j)), \quad \max_{j:a_j' \leq x} (c_j' + F(x - a_j'))\Big), \tag{19}$$

where $c_j' = c_r + c_j$ and $a_j' = a_r + c_j$ for all j. If all c_j' and a_j' are defined as additional c_j and a_j values, then (19) is of the form given by (18) and another immediate solution can be obtained by seeking a new $a_r = \min a_j$; again $F(a_r) = c_r$. We continue this way until $F(L)$ is found or until no other $F(x)$ for $x \leq L$ can be produced. At the same time, we remember the $j = r$ that produces the solution so that we can determine the optimal x_j at the end of the procedure. We also consider only distinct a_j values. If $a_j = a_k$ and $c_j \geq c_k$ for $j \neq k$, then take $x_k = 0$.

The method for solving the knapsack problem is contained in

Algorithm 3

1. List the values of the problem as follows:

1	2	3	...	n
c_1	c_2	c_3	...	c_n
a_1	a_2	a_3	...	a_n

;

L is also listed. Go to 2.

2. Given the list, find index r from $a_r = \min a_j \leq L$ for all unmarked columns in all sections. If $a_r = L$ or if no a_r exists, go to 4. Otherwise, take $c = c_r$, $a = a_r$ and go to 3.

3. Add a new section of columns to the list, if possible. Calculate $c_j' = c + c_j$ and $a_j' = a + a_j$ for the indices j of the unmarked columns in the section containing column r. Mark the r column. The c_j, a_j values are taken from the list in step 1. Add a column with values c_j' and a_j' if:

(a) $a_j' \leq L$.
(b) a_j' is not on the list.
(c) a_j' is on the list and has corresponding c_j value that is smaller than c_j'.

Underneath the section added, write the x_j values from the section containing the newly marked r column. Increase x_r by one for the new section. Designate the c_j', a_j' as c_j, a_j values and go to 2.

4. The problem is solved with solution $c_s = \max c_j$ for $a_s \leq L$ for columns in all sections. The values of the variables are found below the section where c_s appears; increase x_s by one.

Algorithm 3 is a single pass algorithm in the sense that the x_j values are always available. If desirable, the x_j values need not be maintained but may be calculated in a back-tracking procedure. We would then define the function $I(a_r) = r$ in step 2. When index s is found in step 4, we calculate the x_j values as follows:

(a) Initially all $x_j = 0$.

(b) Take $a = a_s$ in step 4.
(c) Increase x_s by one.
(d) Calculate $m = a - a_s$, where a_s is the value from the list in step 1. If $m = 0$, the current x_j values are optimal. Otherwise, take $s = I(m)$ and return to (c).

Example

Find integers $x_j \geq 0$ that maximize z when

$$3x_1 + 4x_2 + 3x_3 = z,$$
$$9x_1 + 7x_2 + 6x_3 \leq 15,$$

We list the values in Tableau E1 with $L = 15$. We have $r = 3$, $c = 3$, $a = 6$ in Tableau E1. We mark the 3 column and form Tableau E2. We have $r = 2$, $c = 4$, $a = 7$ in Tableau E1. We mark the 2 column and form Tableau E3. No other columns are added. The maximum solution is in column 2 of Tableau E3. It is $z = 8$ with $x_2 = 2$, $x_1 = 0$, $x_3 = 0$.

1	2*	3*
3	4	3
9	7	6

E1

1	2	3
6	7	6
15	13	12
$x_3 = 1$		

E2

2
8
14
$x_2 = 1$

E3

Tableaus

Problems

1. The map coloring problem was described in Chapter 1. Show how a feasible solution may be found by enumerating values of α_{rs} for

$$t_r - t_s = \alpha_{rs}.$$

When the problem is tabulated as in Algorithm 1, what is a good column selection criterion that leads to all $\alpha_{rs} \neq 0$.

2. The traveling salesman problem was discussed in Chapter 1. Show how to enumerate solutions to the objective function until the constraints are satisfied.
3. Solve the example on page 90 by the accelerated enumeration of Section 3.
4. Find integers $x_j \geq 0$ and min z for

$$\begin{aligned} 2x_1 + 5x_2 + 4x_3 &= z, \\ 3x_1 + 2x_2 + 3x_3 &\geq 8, \\ x_1 + 3x_2 + 5x_3 &\geq 11. \end{aligned}$$

5. Minimize z when $0 \leq x_1 \leq 1$, $0 \leq x_2 \leq 2$, $0 \leq x_3 \leq 3$, $0 \leq x_4 \leq 1$ in

$$\begin{aligned} 2x_1 + x_2 + 3x_3 + 5x_4 &= z, \\ -7x_1 - x_2 + x_3 + 2x_4 &\geq 3, \\ 2x_1 + x_2 + x_3 + 2x_4 &\geq 6. \end{aligned}$$

References

1. Balas, E., An additive algorithm for solving linear programs with zero–one variables, *Operations Res.* **13** (4), 517 (1965).
2. Beale, E. M. L., "Mathematical Programming in Practice." Pitman, London, 1968.
3. Bellman, R., "Dynamic Programming." Princeton Univ. Press, New Brunswick, New Jersey, 1957.
4. Gilmore, P. C., and Gomory, R. E., The theory and computation of knapsack functions, *Operations Res.* **14** (6), 1045 (1966).
5. Greenberg, H., An algorithm for the computation of knapsack functions, *J. Math. Anal. Appl.*, 26 (1), 159 (1969).
6. Le Garff, A., and Malgrange, Y., Résolution des programmes linéaires a valeurs entières par une méthode Booleienne "compacte", *Proc. 3rd Inter. Conf. Operational Res.*, Dunod, Paris, 1964, pp. 695–701.

5 CONTINUOUS SOLUTION METHODS

We turn our attention to a study of continuous solution methods for solving integer programs. The simplex algorithm is an efficient method for solving a linear program for continuous variables and can be used to advantage for the integer problem.

First, we present the theory for continuous solution methods in which the integer restriction is temporarily relaxed and the problem solved as a linear program. If the continuous solution is also integer, then the integer problem is solved. If the solution values are fractional, we obtain the optimal integer values from the continuous solution. The method is analogous to the all-integer dual simplex method of Chapter 3. The resulting algorithm is comparable to one by Gomory [*2*], but has different convergence properties.

We then show how to combine the continuous solution and all-integer methods; this process tends to improve the convergence. The approach taken in Sections 4 and 5 was indicated by Gomory [*3*]. The chapter concludes with a new algorithm for problems having variables with upper bounds, and a method to solve the mixed integer problem.

1 A CONTINUOUS SOLUTION METHOD

The integer programming problem may be written as: Find integer $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{n} c_j x_j = z, \\ &\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

and the a_{ij}, b_i, and c_j are given integer constants. Suppose the integer constraint on the x_j is relaxed, i.e., the x_j may take on fractional values. The resulting continuous problem may be solved by the simplex method which converts the equations of (1) to an optimality format

$$\begin{aligned} z &= \bar{z}_0 + \sum_{j=1}^{t} \bar{c}_j x_j, \\ x_{t+i} &= \bar{b}_i + \sum_{j=1}^{t} \bar{a}_{ij} x_j, \qquad i = 1, 2, \ldots, m, \end{aligned} \tag{2}$$

where the first $t = n - m$ and the last m variables have been arbitrarily selected as the nonbasic and basic variables, respectively. Since (2) is an optimal canonical form, $\bar{c}_j \geq 0$ and $\bar{b}_i \geq 0$; the optimal continuous solution to program (1) is $x_{t+i} = \bar{b}_i$ for the basic variables and $x_j = 0$ for the nonbasic variables. In addition, if the continuous solution has all $\bar{b}_i$ as integer, program (1) is solved. When any of the $\bar{b}_i$ are fractional, the equations of (2) are used to obtain the integer solution. Program (1) is equivalent to: Find integer $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when the objective function and constraints are given by (2).

We shall follow closely the approach of Chapter 3. The method to find the optimal integer solution from (2) consists of making a series of changes of variables to achieve the transformation

$$x_j = d_j + \sum_{k=1}^{t} d_{jk} y_k, \qquad j = 1, 2, \ldots, t. \tag{3}$$

The integer constants d_j, d_{jk} are developed in an iterative procedure during the solution of the problem. The initial transformation is established by writing (2) in parametric form as

$$
\begin{aligned}
& z = \bar{z}_0 + \sum_{j=1}^{t} \bar{c}_j y_j, \\
(4) \qquad & x_j = y_j \geq 0, \qquad j = 1, 2, \ldots, t, \\
& x_{t+i} = \bar{b}_i + \sum_{j=1}^{t} \bar{a}_{ij} y_j, \qquad i = 1, 2, \ldots, m.
\end{aligned}
$$

Eliminating x_j from (2) using (3), we obtain the equivalent problem: Find integer $y_j \geq 0$ for $j = 1, 2, \ldots, t$ that minimize z when

$$
\begin{aligned}
& z = z_0' + \sum_{j=1}^{t} c_j' y_j, \\
(5) \qquad & x_j = d_j + \sum_{k=1}^{t} d_{jk} y_k, \qquad j = 1, 2, \ldots, t, \\
& x_{t+i} = b_i' + \sum_{j=1}^{t} a_{ij}' y_j, \qquad i = 1, 2, \ldots, m.
\end{aligned}
$$

The constants z_0', c_j', b_i', and a_{ij}' are developed as a result of the transformation with

$$
\begin{aligned}
& z_0' = \bar{z}_0 + \sum_{j=1}^{t} \bar{c}_j d_j, \\
& c_j' = \sum_{k=1}^{t} \bar{c}_k d_{kj}, \qquad j = 1, 2, \ldots, t, \\
& b_i' = \bar{b}_i + \sum_{j=1}^{t} \bar{a}_{ij} d_j, \qquad i = 1, 2, \ldots, m, \\
& a_{ij}' = \sum_{k=1}^{t} \bar{a}_{ik} d_{kj}, \qquad \text{all } i \text{ and } j.
\end{aligned}
$$

If the constants are such that $c_j' \geq 0$, d_j and b_i' are nonnegative integers, then the minimal solution to (5) is given by $z = z_0'$, $y_j = 0$ for $j = 1, 2, \ldots, t$. In addition, the minimal solution to (1) and (2) is given by $x_j = d_j$ for $j = 1, 2, \ldots, t$ and $x_{t+i} = b_i'$ for $i = 1, 2, \ldots, m$.

In Section 2 we develop transformation (3). The proof of optimality and the equivalence of programs (2) and (5) then follow by adapting the methods and proofs of Chapter 3. We leave the details to the reader.

2 IMPROVING A NONOPTIMAL SOLUTION

We demonstrate here how to find the transformation (3) that yields an equivalent problem, Eq. (5), and leads in a finite number of steps to the optimality conditions $c_j' \geq 0$, integer $d_j \geq 0$, and integer $b_i' \geq 0$.

We can write (5) as

$$x = \beta + \sum_{j=1}^{t} \alpha_j y_j, \tag{6}$$

where x is a column vector with components $z, x_1, x_2, \ldots, x_n$. β is a column vector with components $z_0', d_1, d_2, \ldots, a_t', b_1', b_2', \ldots, b_m'$ and α_j is a column vector with components $c_j', d_{1j}, d_{2j}, \ldots, d_{tj}, a_{1j}', a_{2j}', \ldots, a_{mj}'$. Initially (4) is also in the form given by (6) with β components $\bar{z}_0, 0, 0, \ldots, 0, \bar{b}_1, \bar{b}_2, \ldots, \bar{b}_m$ and α_j components $\bar{c}_j, 0, 0, \ldots, 0, 1, 0, \ldots, 0, \bar{a}_{1j}, \bar{a}_{2j}, \ldots, \bar{a}_{mj}$. The one appears as the $(j+1)$ component of α_j.

We insure a finite algorithm by maintaining α_j lexicopositive. If (4) is written in the form of (6), we see that $\alpha_j \succ 0$ because all $\bar{c}_j \geq 0$. Suppose at some iteration we have achieved a form like (6) with $\alpha_j \succ 0$, all nonobjective function components of β nonnegative and at least one component of β fractional. Select a row with fractional β component. Suppose the corresponding equation is given by

$$x_0 = b_0 + \sum_{j=1}^{t} a_j y_j. \tag{7}$$

Define $q = \{b_0\} - b_0 > 0$ and $p_j = a_j - [a_j] \geq 0$. Thus (7) may be written as

$$x_0 - \sum_{j=1}^{t} [a_j] y_j = b_0 + \sum_{j=1}^{t} p_j y_j. \tag{8}$$

Defining integer y as the left side of (8), we have $y \geq \{b_0\}$ or $y - y' = \{b_0\}$ for integer $y' \geq 0$. From (8) we obtain

$$y' = -q + \sum_{j=1}^{t} p_j y_j \geq 0. \tag{9}$$

Equation (9) is a new restriction that the y_j must satisfy to produce an integer value for x_0. Note that b_0 may be negative so that the objective function may be used to generate a new restriction. To produce a finite algorithm, we will use the topmost fractional component of β to select the row for (7). Thus x_0 in (7) represents z or some x_j.

Next find index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j,$$

where J^+ is the set of indices j where $p_j > 0$. The constraint (9) may then be written as

$$y_s' = -q' + \sum_{j \neq s} p_j y_j + p_s y_s. \tag{10}$$

If y_s is eliminated from (6) using (10), we obtain

$$x = \beta' + \sum_{j=1}^{t} \alpha_j' y_j, \tag{11}$$

where

$$\beta' = \beta + \frac{q}{p_s}\alpha_s,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s, \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s,$$

and y_s' has been redefined by new variable y_s. If in addition β' and α_j' are redefined as β and α_j, then (11) is again of the form of (6) with $\alpha_j \succ 0$.

The dual lexicographic method is then used in conjunction with (11) to find a new continuous solution. In effect, we employ a continuous simplex method in an attempt to achieve an integer solution. As in the lexicographic dual method of Chapter 2, we determine the equation of

(6) [after redefining (11) as (6)] with most negative β component, excluding the objective function. If all such components are nonnegative, the continuous solution is given by $x = \beta$. Otherwise, let the equation determined by a negative component be

$$x_r = -b_0 + \sum_{j=1}^{t} a_j y_j, \tag{12}$$

where $b_0 > 0$. Define J^+ as the set of indices j where $a_j > 0$. Define index s from

$$\frac{1}{a_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{a_j}\alpha_j$$

and write (12) as

$$\begin{aligned} x_r &= -b_0 + \sum_{j \neq s} a_j y_j + a_s y_s \\ &= y_s' \\ &\geq 0, \end{aligned} \tag{13}$$

where y_s' is a nonnegative integer defined equal to x_r. Eliminate y_s from (6) using (13) with y_s' and obtain the form of (11) where

$$\beta' = \beta + \frac{b_0}{a_s}\alpha_s,$$

$$\alpha_j' = \alpha_j - \frac{a_j}{a_s}\alpha_s, \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{a_s}\alpha_s,$$

and y_s' has been redefined as y_s. Note that the constant term in the x_r row becomes zero and the equation becomes $x_r = y_s$. Thus x_r is considered nonbasic. This enables us to make a lexicographic argument to show that optimality for the continuous solution is reached in a finite number of steps.

When the form of (11) is achieved, it is redefined as (6). If any nonobjective components of β are negative, we repeat the procedure that led to the selection of (12) and the new form of (6). Otherwise, we have

achieved a new continuous solution. If the continuous solution is fractional, then (9) is developed and the method repeated until the optimal continuous solution is all-integer. The entire procedure is contained in

Algorithm 1

1. Solve (1) by the simplex algorithm of Chapter 2 to obtain a form equivalent to (2). If the basic continuous solution has all integer values, the integer programming problem is solved; stop. Otherwise, go to 2.

2. Develop a tableau by listing the columns $x \mid \beta, \alpha_1, \alpha_2, \ldots, \alpha_t$. Column x is a listing of the variables. The first element of x is z; the next t are the nonbasic variables in order and the next m are the basic variables in order of equation appearance. Assuming the basic and nonbasic variables are indexed as in (4), the β and α_j values are assigned accordingly. Initially, then, $z_0' = \bar{z}_0$, $c_j' = \bar{c}_j$, $d_j = 0$, $d_{jj} = 1$, $d_{ij} = 0$ for $i \neq j$, $b_i' = \bar{b}_i$ and $a_{ij}' = \bar{a}_{ij}$ where the unprimed values are from (2). Use the corresponding values when the problem format is different; the basic and nonbasic variables are indexed according to the problem. Go to 3.

3. Select the topmost row with fractional β component. The row values are $b_0, a_1, a_2, \ldots, a_t$. Calculate $q = \{b_0\} - b_0$ and $p_j = a_j - [a_j]$, for $j = 1, 2, \ldots, t$. Go to 4 (a).

4. (a) Define J^+ as the index set where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j .$$

Calculate new column values

$$\beta' = \beta + \frac{q}{p_s}\alpha_s ,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s , \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s .$$

Designate β', α_j' as the current β, α_j, and go to 4 (b).

(b) If no component of β beyond the first is negative, the optimal continuous solution is achieved. If the optimal solution is fractional, go to 3. If the solution is all-integer, the problem is solved with values $x = \beta$. Stop. Otherwise, select the nonobjective row with most negative β component. The row values are $-b_0, a_1, a_2, \ldots, a_t$. Take $q = b_0, p_j = a_j, j = 1, 2, \ldots, t$ and go to 4(a).

Example

Find integers $x_j \geq 0$ that minimize z when

$$3x_1 - x_2 + 12x_3 + x_4 = z,$$
$$x_1 + x_2 + 3x_3 - 3x_4 = 4,$$
$$3x_1 - 3x_2 + 11x_3 + x_4 = 2.$$

	1	2
z	$\frac{16}{3}$	$\frac{5}{3}$	$\frac{10}{3}$
x_3	0	1	0
x_4	0	0	1
x_1	$\frac{7}{3}$	$-\frac{10}{3}$	$\frac{4}{3}$
x_2	$\frac{5}{3}$	$\frac{1}{3}$	$\frac{5}{3}$

Tableau E1

	1	2
z	7	$\frac{5}{2}$	$\frac{5}{2}$
x_3	1	$\frac{3}{2}$	$-\frac{1}{2}$
x_4	0	0	1
x_1	-1	-5	3
x_2	2	$\frac{1}{2}$	$\frac{3}{2}$

Tableau E2

	1	2
z	$\frac{47}{6}$	$\frac{20}{3}$	$\frac{5}{6}$
x_3	$\frac{5}{6}$	$\frac{2}{3}$	$-\frac{1}{6}$
x_4	$\frac{1}{3}$	$\frac{5}{3}$	$\frac{1}{3}$
x_1	0	0	1
x_2	$\frac{5}{2}$	3	$\frac{1}{2}$

Tableau E3

	1	2
z	8	6	1
x_3	$\frac{4}{5}$	$\frac{4}{5}$	$-\frac{1}{5}$
x_4	$\frac{2}{5}$	$\frac{7}{5}$	$\frac{2}{5}$
x_1	$\frac{1}{5}$	$-\frac{4}{5}$	$\frac{6}{5}$
x_2	$\frac{13}{5}$	$\frac{13}{5}$	$\frac{3}{5}$

Tableau E4

	1	2
z	$\frac{33}{4}$	5	$\frac{5}{4}$
x_3	$\frac{3}{4}$	1	$-\frac{1}{4}$
x_4	$\frac{1}{2}$	1	$\frac{1}{2}$
x_1	$\frac{1}{2}$	-2	$\frac{3}{2}$
x_2	$\frac{11}{4}$	2	$\frac{3}{4}$

Tableau E5

	1	2
z	12	5	5
x_3	0	1	-1
x_4	2	1	2
x_1	5	-2	6
x_2	5	2	3

Tableau E6

1. Using the simplex algorithm, we transform the equations to

$$z = \tfrac{16}{3} + \tfrac{5}{3}x_3 + \tfrac{10}{3}x_4$$
$$x_1 = \tfrac{7}{3} - \tfrac{10}{3}x_3 + \tfrac{4}{3}x_4$$
$$x_2 = \tfrac{5}{3} + \tfrac{1}{3}x_3 + \tfrac{5}{3}x_4.$$

2. Tableau E1 is formed.
3. The z row is selected. $q = \frac{2}{3}, p_1 = \frac{2}{3}, p_2 = \frac{1}{3}$.
4. (a) $J^+ = (1, 2); s = 1$. We form Tableau E2.
   (b) Reminimize. The x_1 row is selected. $q = 1, p_1 = -5, p_2 = 3$.
   (a) $J^+ = (2); s = 2$. We form Tableau E3.
3. The z row is selected. $q = \frac{1}{6}, p_1 = \frac{2}{3}, p_2 = \frac{5}{6}$.
4. (a) $J^+ = (1, 2); s = 2$. We form Tableau E4.
3. The x_3 row is selected. $q = \frac{1}{5}, p_1 = \frac{4}{5}, p_2 = \frac{4}{5}$.
4. (a) $J^+ = (1, 2); s = 2$. We form Tableau E5.
3. The z row is selected. $q = \frac{3}{4}, p_1 = 0, p_2 = \frac{1}{4}$.
4. (a) $J^+ = (2); s = 2$. We form Tableau E6.
   (b) The minimal solution is $z = 12, x_3 = 0, x_4 = 2, x_1 = 5, x_2 = 5$.

3 CONVERGENCE IN THE ALGORITHM

Since $\alpha_j \succ 0$, every utilization of step 4(a) of Algorithm 1 causes β to increase lexicographically. In an argument similar to that in Chapter 3, we show that the components of β have finite upper bounds. The problem of producing a finite algorithm arises because of the fact that although β is lexicographically increasing, the components of β may change only by minute amounts over an infinite number of iterations. To prevent these infinitessimal changes and obtain a finite algorithm, we select the topmost row with fractional β component in step 3 of the algorithm. We shall prove that the algorithm takes a finite number of iterations with this row selection strategy.

There are two types of convergent processes occurring during the use of the algorithm. One of these occurs during the reoptimization phase. Each minimization cycle takes only a finite number of iterations because the number of basic variables is fixed during the cycle. Since β is lexicographically increasing from iteration to iteration, the same set of basic variables may not recur and since there exist a finite number of sets of basic variables, the optimal continuous solution must be arrived at in a finite number of iterations.

The second convergent process occurs at the end of each minimization phase. At the end of some cycle the continuous solution must eventually be all-integer. The number of minimization cycles must remain finite. To achieve a finite number of cycles, we first show that the components of β have finite upper bounds when a finite optimal integer solution exists.

Assume there exists a finite optimal solution x^0 with objective component z^0, i.e., $x^0 = (z^0, x_1{}^0, x_2{}^0, \ldots, x_n{}^0)$. Thus x^0 must satisfy (6); there must exist $y_j \geq 0$ that satisfy

$$\sum_{j=1}^{t} \alpha_j y_j = x^0 - \beta. \tag{14}$$

Consider the first component of β, the objective value z_0'. Although monotonically increasing, $z_0' \leq z^0$ holds for all minimization cycles. This is readily apparent because if $z_0' > z^0$ after some cycle, the objective row from (14) is

$$\begin{aligned} \sum_{j=1}^{t} c_j' y_j &= z^0 - z_0' \\ &< 0; \end{aligned} \tag{15}$$

since $c_j' \geq 0$, then, as seen from (15), no $y_j \geq 0$ values can possibly produce $z = z^0$. Thus z_0', the first component of β, is bounded from above by z^0.

The other components of β remain bounded. Suppose, on the contrary, that the $(r+1)$ component, $1 \leq r \leq n$, takes on arbitrarily large values. Take the $(r+1)$ component as d_r with $d_r > x_r{}^0$ (for $r > t$, take $d_r = b_r'$ and $d_{rj} = a_{ij}'$). The corresponding equation from (14) is

$$\begin{aligned} \sum_{j=1}^{t} d_{rj} y_j &= x_r{}^0 - d_r \\ &< 0. \end{aligned} \tag{16}$$

If all $d_{rj} \geq 0$, then no $y_j \geq 0$ exist that satisfy (16) and d_r would be bounded by $x_r{}^0$. Thus if (16) exists, some $d_{rj} < 0$ and

$$\sum_{j \in J^-} d_{rj} y_j \leq x_r{}^0 - d_r \tag{17}$$

must hold; J^- is the set of indices j where $d_{rj} < 0$. We perform the following steps.

1. Obtain index s from

$$\frac{1}{-d_{rs}}\alpha_s = l\text{-}\min_{j\in J^-}\frac{1}{-d_{rj}}\alpha_j .$$

2. Write (17) in the equality form

$$\sum_{j\in J^-} d_{rj}y_j + y_s' = x_r^0 - d_r , \tag{18}$$

where $y_s' \geq 0$. Use (18) to eliminate y_s from (6) and obtain the form of (11) where

$$\beta' = \beta + \frac{x_r^0 - d_r}{d_{rs}}\alpha_s . \tag{19}$$

The boundedness of β follows easily by induction. Suppose $d_1 > x_1^0$. The objective value from (19) becomes

$$z_0'' = z_0' + \frac{x_1^0 - d_1}{d_{1s}}c_s' ;$$

since $\alpha_s \succ 0$ and $d_{1s} < 0$, then $c_s' > 0$. Furthermore, $z_0'' \leq z^0$, which results in

$$d_1 \leq x_1^0 + \frac{z^0 - z_0'}{c_s'}(-d_{1s}).$$

An upper bound has been achieved for d_1, the second component of β.

Suppose now that $d_1, d_2, \ldots, d_{r-1}$ are bounded by $\delta_1, \delta_2, \ldots, \delta_{r-1}$, respectively, and $d_r > x_r^0$. Since $\alpha_s \succ 0$ and $d_{rs} < 0$, one or more of the values $c_s', d_{1s}, d_{2s}, \ldots, d_{r-1,s}$ is positive. We achieve an upper bound for d_r

$$d_r \leq \begin{cases} x_r^0 + \dfrac{z^0 - z_0'}{c_s'}(-d_{rs}), & \text{if } c_s' > 0, \\ x_r^0 + \dfrac{\delta_i - d_i}{d_{is}}(-d_{rs}), & \text{if } d_{is} > 0, \quad i = 1, 2, \ldots, r-1. \end{cases} \tag{20}$$

Thus by induction all components of β are bounded. Note from (20),

however, that the bounds can be quite large since c_s' and the d_{is} may be small in a given problem. Large bounds will reduce the efficiency of the algorithm. The means for rectifying this deficiency will be discussed in Sections 4 and 5.

We need to show that the components of β may not change by minute amounts over an infinite number of cycles while remaining below their upper bounds. If the process takes an infinite number of cycles, then an infinite sequence, $\beta_1 \prec \beta_2 \prec \beta_3 \prec \ldots$, results where β_k is the continuous solution at the end of minimization cycle k. Suppose that the first r components, $1 \leq r \leq n+1$, of β remain fixed at integer values starting at some cycle k_0. This supposition will be proven true for at least $r = 1$.

In the $r = 1$ case, let ${z_0}^k$ be the objective value at the end of minimization cycle k. Let n_0 be the smallest integer with the property that ${z_0}^k \leq n_0$ for all k. Some bound like n_0 exists since ${z_0}^k$ is bounded by z^0. We now show that ${z_0}^k = n_0$ for all cycles $k \geq k_0$.

From the definition of n_0 we must have ${z_0}^i = n_0 - q$ with $0 \leq q < 1$ after some cycle i. If $q = 0$, then ${z_0}^k = n_0$ for $k \geq i$. If $q > 0$, then we use the objective row in step 3 of the algorithm to produce a new objective value $z_0' = {z_0}^i + q{c_s}^i/p_s$, where $p_s = {c_s}^i - [{c_s}^i] > 0$. Since ${c_s}^i > 0$, then $p_s \leq {c_s}^i$ and $z_0' \geq {z_0}^i + q = n_0$. Thus ${z_0}^k = n_0$ for $k \geq i + 1 = k_0$ and the first component of β remains fixed at an integer value after some cycle k_0.

If we assume that the first r components of β remain fixed at integer values after some cycle, we see that the algorithm takes an infinite number of cycles if component $r+1$ increases over an infinite number of cycles. Take b_0 as component $r+1$ of β and $a_1, a_2, \ldots, a_t$ as the corresponding row values. Since b_0 is bounded, then for it to increase over an infinite number of cycles there exists an integer n_r such that for all cycles $k \geq i \geq k_0$ we have $n_r - 1 < {b_0}^k < n_r$, where n_r is the smallest integer with the property that ${b_0}^k < n_r$. Row r is the eligible row, after cycle i, in step 3 of the algorithm. The new value $b_0' = {b_0}^i + qa_s/p_s$ is produced where $q = n_r - {b_0}^i$ and $p_s = a_s - [a_s] > 0$. The first r components of β must remain fixed in the next minimization cycle. Thus a_s is the first nonzero component of $\alpha_s \succ 0$ and $a_s > 0$. Since $a_s > 0$, then $p_s \leq a_s$ and $b_0' \geq {b_0}^i + q = n_r$, which contradicts the condition

that $b_0{}^k$ be smaller than n_r and the assumption that b_0 increases over an infinite number of cycles.

We have demonstrated that it is impossible for component $r+1$ of β to increase for an infinite number of cycles; similarly, every component of β attains a fixed value which it maintains from some cycle onwards after a finite number of cycles. The fixed value is also an integer. The choice of the topmost fractional component of β is thus sufficient to achieve the finiteness.

4 REDUCING THE CONTINUOUS SOLUTION TO AN ALL-INTEGER FORMAT

The continuous solution in the integer programming problem is usually found rapidly with the simplex method. The integer solution, however, may be more difficult to obtain because the components of β may have large upper bounds as shown in (20). In an effort to produce the integer solution rapidly, we present a method that changes the continuous solution (2) to an all-integer format. When the constants are integers, Algorithm 1 of Chapter 3 may be used to find the minimal solution. Alternatively, the simplex method may be used to reminimize.

We begin by investigating the constants of (2). Equation (1) may be written in matrix form as

$$Ax_H + Bx_G = b, \tag{21}$$

where $x_H = (x_1, x_2, \ldots, x_t)$ and $x_G = (x_{t+1}, x_{t+2}, \ldots, x_n)$ are vectors representing the nonbasic and basic variables, respectively; the inverse matrix B^{-1} must exist. Thus, multiplying (21) through on the left by B^{-1}, we obtain the matrix form of the constraints of (2) as

$$x_G = B^{-1}b - B^{-1}Ax_H . \tag{22}$$

Since B^{-1} exists it may be written as

$$B^{-1} = \frac{\tilde{B}}{\det B},$$

where $\tilde{B}$ is the adjoint matrix of B and $\det B$ is the determinant of B. Since the elements of B are assumed to be integers, the elements of $\tilde{B}$

are integers. Furthermore, all elements of A and b are assumed to be integers; thus; every constant appearing in (2) or (22) is of the form $i/\det B$, where numerator i is an integer.

We are interested in D, the absolute value of $\det B$. If $D = 1$, an integer solution is produced in (2) and the integer program is solved. If D is very large, the constants of (2) may be quite small with resulting large upper bounds in (20). The use of constraint (10) and the reminimization may result in still larger $D = D'$ at later cycles. Ultimately the process results in a D' value that equals one (i.e., if D' does not divide equally into the respective numerators for the variable values). Thus, Algorithm 1 may take many iterations to provide the integer solution if D' becomes large.

The D value develops during the simplex algorithm as the product of the pivot elements. The pivot step in the first iteration is represented by left-multiplying the matrix (a_{ij}) of (1) by the elementary matrix

$$B_1 = \begin{pmatrix} 1 & & & k_1 & & \\ & 1 & & k_2 & & \\ & & \ddots & \vdots & & \\ & & & k_r & & \\ & & & \vdots & \ddots & \\ & & & k_m & & 1 \end{pmatrix}, \tag{23}$$

where

$$k_i = -\frac{a_{is}}{a_{rs}}, \qquad i \neq r,$$

$$k_r = \frac{1}{a_{rs}}.$$

Pivoting in iteration i consists of left-multiplying $B_{i-1}B_{i-2}\cdots B_1(a_{ij})$ by B_i, where B_i is an elementary matrix of the same form as (23) with $k_i = -\bar{a}_{is}/\bar{a}_{rs}$, $i \neq r$ and $k_r = 1/\bar{a}_{rs}$ (with usually different r and s values for each iteration). Thus each row of B^{-1} is given as a row in the matrix formed by the product $B_N B_{N-1}\cdots B_1$ of elementary matrices,[1]

[1] The inverse matrix in this form is called the *product form* of the inverse and as such is useful in computational work.

where N is the number of iterations to achieve the optimal continuous solution (2). Matrix B^{-1} itself does not usually result from the product since the order of the basic variables will most likely be different from that shown for x_G in (22). Since $\det B_i = 1/\bar{a}_{rs}$, where $\bar{a}_{rs}$ is the pivot element (initially $\bar{a}_{rs} = a_{rs}$), then the absolute value of $\det B^{-1}$ is one divided by the product of the pivot elements. Also, $BB^{-1} = I$; hence, $\det B \det B^{-1} = 1$ and D is proven to be the product of the pivot elements.

By calculating D we have a measure of how close we are to achieving an integer solution. The effect of using constraint (10) is the same as adding an equivalent constraint with integer coefficients to (1). Thus p_s becomes the pivot element in developing (11). If D is the determinant value at step 3 of Algorithm 1, then $D' = p_s D$ is the determinant value at the end of step 4(a). Similarly, a_s is the pivot element when (13) is used during the minimization cycle. Thus, the value D may be determined during the use of Algorithm 1. When (2) is achieved we desire to reduce D to unity.

The repeated use of (10) without the minimization cycle enables us to reduce D to unity. This reduction is possible because the pivot element p_s is always less than one. In fact, p_s must be of the form i/D, where integer $i < D$; thus $D' = i$. Whenever fractions appear in (6), constraint (10) is developed with subsequent reduction in determinant D. Eventually all integer coefficients will result (the variable values, however, may turn out to be negative integers).

The procedure for converting the continuous solution (2) to an all-integer format is summed up in Algorithm 2. The optimal integer solution is then found by the all-integer algorithm of Chapter 3.

Algorithm 2

1. Same as step 1 of Algorithm 1 except that D is calculated at each simplex iteration as the product of the pivot elements.
2. Same as step 2 of Algorithm 1.
3. Select the topmost row with fractional β component, if possible. If all components of β are integers, select a row with fractional α_j

component. The row values are $b_0, a_1, a_2, \ldots, a_t$. Calculate $q = \{b_0\} - b_0$ and $p_j = a_j - [a_j]$ for $j = 1, 2, \ldots, t$. Go to 4.

4. Define J^+ as the set of indices where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j.$$

Calculate $D' = p_s D$ and new column values

$$\beta' = \beta + \frac{q}{p_s}\alpha_s,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s, \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s.$$

Designate D', β', α_j' as the current D, β, α_j. If all components of β are nonnegative integers, the problem is solved with values $x = \beta$. Stop. If $D = 1$ or all tabular values are integers (set $D = 1$), go to 5. Otherwise, go to 3.

5. Use Algorithm 1 of Chapter 3 to complete the solution.

Example

We shall solve the example problem on page 110. Tableaus E1 and E2 are identical with those of the first algorithm. We start with Tableau E2 with $D = 2$.

	1	2
z	7	$\frac{5}{2}$ $\frac{5}{2}$
x_3	1	$\frac{3}{2}$ $-\frac{1}{2}$
x_4	0	0 1
x_1	-1	-5 3
x_2	2	$\frac{1}{2}$ $\frac{3}{2}$

Tableau E2

	1	2
z	7	0 5
x_3	1	2 -1
x_4	0	-1 2
x_1	-1	-8 6
x_2	2	-1 3

Tableau E3

	1	2
z	12	5 5
x_3	0	1 -1
x_4	2	1 2
x_1	5	-2 6
x_2	5	2 3

Tableau E4

3. The z row is selected. $q = 0, p_1 = \frac{1}{2}, p_2 = \frac{1}{2}$.
4. $J^+ = (1, 2)$; $s = 2$. We form Tableau E3. $D = 1$; an all-integer format is obtained.
5. The all-integer algorithm: the x_1 row is selected and $J^+ = (2)$ and $s = 2$. $\lambda = 6$, $q = 1$, $p_1 = -1$, $p_2 = 1$. Form Tableau E4. The minimal solution is reached in Tableau E4; $z = 12$, $x_3 = 0$, $x_4 = 2$, $x_1 = 5$, $x_2 = 5$.

The same procedure for converting the continuous solution (2) to an all-integer format is given by Algorithm 3. The lexicographic dual simplex method is then used to obtain a new continuous solution. The method alternates between the reduction of D phase and the minimization phase until the optimal integer solution is reached.

Algorithm 3

Steps 1, 2, 3, and 4 are the same as those in Algorithm 2.

5. Select the nonobjective row with most negative β component. The row values are $-b_0, a_1, a_2, \ldots, a_t$. Take $q = b_0, p_j = a_j$ for $j = 1, 2, \ldots, t$. Go to 6.
6. Define J^+ as the index set where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j .$$

Calculate $D' = p_s D$ and new column values

$$\beta' = \beta + \frac{q}{p_s}\alpha_s ,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s , \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s .$$

Designate D', β', α_j' as the current D, β, α_j. If all components of β beyond the first are nonnegative, the optimal continuous solution is achieved. Otherwise, go to 5. If the optimal solution is all-integer,

the problem is solved with values $x = \beta$. Stop. Otherwise, the solution is fractional. Go to 3.

Example

Find integers $x_j \geq 0$ that minimize z when

$$2x_1 + 3x_2 = z,$$
$$3x_1 + 2x \geq 9,$$
$$2x_1 + 5x_2 \geq 8,$$

	1	2
z	$\frac{76}{11}$	$\frac{4}{11}$	$\frac{5}{11}$
x_3	0	1	0
x_4	0	0	1
x_2	$\frac{6}{11}$	$-\frac{2}{11}$	$\frac{3}{11}$
x_1	$\frac{29}{11}$	$\frac{5}{11}$	$-\frac{2}{11}$

Tableau E1

	1	2
z	7	0	1
x_3	0	1	0
x_4	$\frac{1}{5}$	$-\frac{4}{5}$	$\frac{11}{5}$
x_2	$\frac{3}{5}$	$-\frac{2}{5}$	$\frac{3}{5}$
x_1	$\frac{13}{5}$	$\frac{3}{5}$	$-\frac{2}{5}$

Tableau E2

	1	2
z	7	0	1
x_3	4	5	-1
x_4	-3	-4	3
x_2	-1	-2	1
x_1	5	3	-1

Tableau E3

	1	2
z	8	$\frac{4}{3}$	$\frac{1}{3}$
x_3	3	$\frac{11}{3}$	$-\frac{1}{3}$
x_4	0	0	1
x_2	0	$-\frac{2}{3}$	$\frac{1}{3}$
x_1	4	$\frac{5}{3}$	$-\frac{1}{3}$

Tableau E4

1. Using the simplex algorithm, we transform the equations to

$$z = \tfrac{76}{11} + \tfrac{4}{11}x_3 + \tfrac{5}{11}x_4,$$
$$x_2 = \tfrac{6}{11} - \tfrac{2}{11}x_3 + \tfrac{3}{11}x_4,$$
$$x_1 = \tfrac{29}{11} + \tfrac{5}{11}x_3 - \tfrac{2}{11}x_4;$$

2. x_3 and x_4 are the surplus variables. Tableau E1 is formed. $D = 11$.
3. The z row is selected. $q = \frac{1}{11}, p_1 = \frac{4}{11}, p_2 = \frac{5}{11}$.
4. $J^+ = (1, 2); s = 2$. Tableau E2 is formed. $D = 5$.

3. The x_4 row is selected. $q = \frac{4}{5}, p_1 = \frac{1}{5}, p_2 = \frac{1}{5}$.
4. $J^+ = (1, 2); s = 1$. Tableau E3 is formed. $D = 1$.
5. The x_4 row is selected. $q = 3, p_1 = -4, p_2 = 3$.
6. $J^+ = (2); s = 2$. Tableau E4 is formed. $D = 3$. The optimal integer solution is apparent in Tableau E4. $z = 8, x_3 = 3, x_4 = 0, x_2 = 0, x_1 = 4$.

Martin [4] has a method somewhat similar to Algorithm 3. He achieves $D = 1$ by using a constraint different from (10) and then re-minimizes. In Algorithm 3, however, the lexicographic increase of β is maintained and the form of (10) permits b_0' to be at least the next highest integer value after b_0, a property that produces convergence.

5 BOUNDING THE DETERMINANT VALUE

Let the integer program be written in the form of (6) with $\alpha_j \succ 0$ and the components of β having finite upper bounds. A bound γ may be selected for D so that for every iteration $D \leq \gamma$. If β is now forced to be lexicographically increasing by any method, then from the analysis made in Section 3, the optimal integer solution will be produced in a finite number of steps.

We present one of several possible strategies for maintaining the bound on the determinant value. If a pivot element a_s should cause $D' = a_s D > \gamma$, we employ either the determinant reduction procedure of Algorithm 3 or the all-integer algorithm. We show this in

Algorithm 4

1. Assume that the integer program is in the nonoptimal format of (6) with $\alpha_j \succ 0$ and available D value. Develop a tableau by listing the columns $x \mid \beta, \alpha_1, \alpha_2, \ldots, \alpha_t$. Assign the upper bound value γ. Go to 2.

2. If all components of β and α_j are integers, set $D = 1$. If $D \leq \gamma$ go to 3. Otherwise, go to 4.

3. If no component of β beyond the first is negative, the optimal continuous solution is achieved; if the optimal solution is fractional, go to 4. If the solution is all integer, the problem is solved with values $x = \beta$. Stop. Otherwise, select the nonobjective row with most negative β component. The row values are $-b_0, a_1, a_2, \ldots, a_t$. Take $q = b_0$, $p_j = a_j$ for $j = 1, 2, \ldots, t$. Define J^+ as the set of indices j where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j .$$

If $D = 1$, and $p_s > \gamma$, go to 6. If $D > 1$ and $p_s D > \gamma$, go to 4. Otherwise, $D' = p_s D$. Designate D' as the current D and go to 5.

4. Select the topmost row with fractional β component, if possible. If all components of β are integers, select a row with fractional α_j component. The row values are $b_0, a_1, a_2, \ldots, a_t$. Calculate $q = \{b_0\} - b_0$ and $p_j = a_j - [a_j]$ for $j = 1, 2, \ldots, t$. Define J^+ as the set of indices j where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j .$$

Calculate $D' = p_s D$; designate D' as the current D and go to 5.

5. Calculate new column values

$$\beta' = \beta + \frac{q}{p_s}\alpha_s ,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s , \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s .$$

Designate β', α_j' as the current β, α_j. If $D = 1$, go to 3. Otherwise, go to 2.

6. Select the nonobjective row with most negative β component. The row selected is $-b_0, a_1, a_2, \ldots, a_n$. Define J^+ as the set of indices j where $a_j > 0$. Determine index s from

$$\alpha_s = l\text{-}\min_{j \in J^+} \alpha_j .$$

Find μ_j as the largest integer that maintains $\alpha_j - \mu_j \alpha_s \succ 0$ for $j \in J^+$, $j \neq s$; $\mu_s = 1$. Take $\lambda = \max_{j \in J^+} a_j/\mu_j$. Calculate $p_j = \{a_j/\lambda\}$ and $q = \{b_0/\lambda\}$. Go to 5.

Example

We shall solve the example problem for Algorithm 1. Take $\gamma = 5$. Tableaus E1 and E2 are identical with those of the first algorithm. We start with Tableau E2; $D = 2$.

3. The x_1 row is selected. $q = 1$, $p_1 = -5$, $p_2 = 3$; $J^+ = (2)$, and $s = 2$. $p_2 D = 6 > \gamma$.
4. The z row is selected. $q = 0$, $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{2}$; $J^+ = (1, 2)$, and $s = 2$. $D = 1$.
5. We form Tableau E3.
3. The x_1 row is selected. $q = 1$, $p_1 = -8$, $p_2 = 6$; $J^+ = (2)$, and $s = 2$. $p_2 = 6 > \gamma$.
6. $J^+ = (2)$; $s = 2$. $\lambda = 6, q = 1, p_1 = -1, p_2 = 1$.
5. We form Tableau E4.
3. The minimal solution is reached in Tableau E4. $z = 12$, $x_3 = 0$, $x_4 = 2$, $x_1 = 5$, $x_2 = 5$.

	1	2
z	7	$\frac{5}{2}$ $\frac{5}{2}$
x_3	1	$\frac{3}{2}$ $-\frac{1}{2}$
x_4	0	0 1
x_1	-1	-5 3
x_2	2	$\frac{1}{2}$ $\frac{3}{2}$

Tableau E2

	1	2
z	7	0 5
x_3	1	2 -1
x_4	0	-1 2
x_1	-1	-8 6
x_2	2	-1 3

Tableau E3

	1	2
z	12	5 5
x_3	0	1 -1
x_4	2	1 2
x_1	5	-2 6
x_2	5	2 3

Tableau E4

6 BOUNDED VARIABLE PROBLEMS

We can use the approach developed in Chapter 3 for the bounded variable problem. We seek to find the solution to (1) with the additional constraints $x_j \leq m_j$ for $j = 1, 2, \ldots, n$, where the m_j are given integer values.

The integer constraint on the x_j is relaxed and (1) is solved as a linear program by the bounded variable technique of Chapter 2. The canonical form of (2) is then achieved with feasible x_j values. The optimality conditions are: $\bar{c}_j \geq 0$ for nonbasic variables at their lower bound zero and $\bar{c}_j \leq 0$ for nonbasic variables at their upper bound m_j.

If the feasible solution is fractional, we again find the optimal integer solution from (2) through transformation (3). The initial transformation is given in (4). If (4) is then written in vector format as in (6), we may not have all α_j lexicopositive. If $\bar{c}_j < 0$, then we make the change of variables

$$y_j = m_j - y_j',$$

where integer $y_j' \geq 0$. Thus we obtain the form of (11) with

$$\beta' = \beta + \sum_{j \in J^-} m_j \alpha_j,$$

$$\alpha_j' = -\alpha_j, \qquad \text{for} \quad j \in J^-,$$

$$\alpha_j' = \alpha_j, \qquad \text{otherwise};$$

J^- is the set of indices j where $\bar{c}_j < 0$. If β' and α_j' are redefined as β and α_j, then we have the form of (6) with $\alpha_j \succ 0$.

Nonobjective components of β are infeasible in the continuous minimization phase whenever they are negative or whenever they exceed their upper bound values. When a component of β is negative, the lexicographic dual simplex method is used as in Algorithm 1. When a component of β exceeds its upper bound value, we can modify the dual simplex method. Suppose at some iteration we have achieved a form like (6) where $\alpha_j \succ 0$ and one or more of the components of β exceed their upper bound values. Select one such row from (6). For that row we must have

$$x_r = b_0 + \sum_{j=1}^{t} a_j y_j \leq m_r,$$

where $b_0 > m_r$. The y_j then satisfy

$$y' = m_r - b_0 - \sum_{j=1}^{t} a_j y_j \geq 0 \tag{24}$$

for integer y'. If we define

$$\begin{aligned} q &= b_0 - m_r \\ &> 0, \\ p_j &= -a_j, \qquad j = 1, 2, \ldots, t, \end{aligned}$$

then (24) is exactly of the form given by (9).[2] Using (24) as (9), the analysis follows as before; the selection of index s remains the same and we develop transformation (10) and the new form of (6) given by (11). The lexicographic property of the α_j is maintained. For upper bound problems, then, we have

Algorithm 5

1. List $m_1, m_2, \ldots, m_n$ and solve (1) by a bounded variable simplex algorithm to obtain a form equivalent to (2). If the feasible continuous solution has all-integer values, the integer programming problem is solved; stop. Otherwise, go to 2.

2. Same as step 2 of Algorithm 1.

3. Define J^- as the set of indices j where $\bar{c}_j < 0$. If J^- has no members, go to 4(a). Otherwise, calculate new column values as

$$\begin{aligned} \beta' &= \beta + \sum_{j \in J^-} m_j \alpha_j, \\ \alpha_j' &= -\alpha_j, && \text{for } j \in J^-, \\ \alpha_j' &= \alpha_j, && \text{otherwise.} \end{aligned}$$

Designate β', α_j' as the current β, α_j and go to 4(a).

4. (a) If any component of β exceeds the corresponding upper bound value, go to 5(a). Otherwise, go to 4(b).
   (b) If any nonobjective component of β is negative, go to 5(b). Otherwise, go to 4(c).

[2] If (24) is handled the same way as (7), a constraint stronger than (24) may be found by defining $q = \{b_0 - m_r\} - b_0 + m_r$, $p = -a_j - [-a_j]$ for (9).

(c) The optimal continuous solution is achieved. If the solution is also all-integer, the problem is solved with values given by $x = \beta$. Stop. Otherwise, the solution is fractional; go to 5(c).

5. (a) Select the row with β component that exceeds the upper bound by the greatest amount. The corresponding variable and row values are $x_r, b_0, a_1, a_2, \ldots, a_t$. Take $q = b_0 - m_r$, $p_j = -a_j$, and go to 6.

(b) Select the nonobjective row with most negative β component. The row values are $-b_0, a_1, a_2, \ldots, a_t$. Take $q = b_0$, $p_j = a_j$ and go to 6.

(c) Select the topmost row with fractional β component. The row values are $b_0, a_1, a_2, \ldots, a_t$. Take $q = \{b_0\} - b_0$, $p_j = a_j - [a_j]$ and go to 6.

6. Define J^+ as the set of indices j where $p_j > 0$. Determine index s from

$$\frac{1}{p_s}\alpha_s = l\text{-}\min_{j \in J^+} \frac{1}{p_j}\alpha_j .$$

Calculate new column values

$$\beta' = \beta + \frac{q}{p_s}\alpha_s ,$$

$$\alpha_j' = \alpha_j - \frac{p_j}{p_s}\alpha_s , \qquad j \neq s,$$

$$\alpha_s' = \frac{1}{p_s}\alpha_s .$$

Designate β', α_j' as the current β, α_j and go to 4.

Example

Find nonnegative integers $x_1 \leq 1$, $x_2 \leq 1$, $x_3 \leq 1$, $x_4 \leq 1$ that minimize z when

$$\begin{aligned} 3x_2 + x_3 + 2x_4 &= z, \\ x_1 + x_2 + x_3 &= 2, \\ 5x_1 - 3x_2 + 3x_3 - 4x_4 &= 4. \end{aligned}$$

1. Using the bounded variable technique of Chapter 2, we transform the equations to

$$z = \tfrac{8}{3} - \tfrac{1}{3}x_1 + \tfrac{2}{3}x_4,$$
$$x_3 = \tfrac{5}{3} - \tfrac{4}{3}x_1 + \tfrac{2}{3}x_4,$$
$$x_2 = \tfrac{1}{3} + \tfrac{1}{3}x_1 - \tfrac{2}{3}x_4;$$

the minimal continuous solution is $z = \frac{7}{3}, x_1 = 1, x_2 = \frac{2}{3}, x_3 = \frac{1}{3}, x_4 = 0$.

2. Tableau E1 is formed.

3. $J^- = (1)$. We form Tableau E2.

5. (c) The z row is selected. $q = \frac{2}{3}, p_1 = \frac{1}{3}, p_2 = \frac{2}{3}$.

	1	2
z	$\frac{8}{3}$	$-\frac{1}{3}$	$\frac{2}{3}$
x_1	0	1	0
x_4	0	0	1
x_3	$\frac{5}{3}$	$-\frac{4}{3}$	$\frac{2}{3}$
x_2	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$

Tableau E1

	1	2
z	$\frac{7}{3}$	$\frac{1}{3}$	$\frac{2}{3}$
x_1	1	-1	0
x_4	0	0	1
x_3	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{2}{3}$
x_2	$\frac{2}{3}$	$-\frac{1}{3}$	$-\frac{2}{3}$

Tableau E2

	1	2
z	3	1	0
x_1	-1	-3	2
x_4	0	0	1
x_3	3	4	-2
x_2	0	-1	0

Tableau E3

	1	2
z	3	1	0
x_1	1	1	1
x_4	1	2	$\frac{1}{2}$
x_3	1	0	-1
x_2	0	-1	0

Tableau E4

6. $J^+ = (1, 2); s = 1$. We form Tableau E3.

5. (a) The x_3 row is selected. $q = 2, p_1 = -4, p_2 = 2$.

6. $J^+ = (2); s = 2$. We form Tableau E4.

4. (b) The minimal solution is achieved. $z = 3, x_1 = 1, x_4 = 1, x_3 = 1, x_2 = 0$.

7 THE MIXED INTEGER PROBLEM

The mixed integer problem is expressed as in (1) with the difference that not all of the x_j variables are restricted to be integers. We find the continuous solution and follow the procedure developed in Section 1. Hence, we assume that at any iteration we have the form (6). We follow Gomory's [*1*] method.

If a component of β is fractional while the corresponding variable is an integer variable, then we have not achieved the minimal solution. Select such a row. Suppose the corresponding equation is given by

$$x_0 = b_0 + \sum_{j=1}^{t} a_j y_j; \tag{25}$$

b_0 is fractional-valued and x_0 is one of the integer restricted variables. We then write (25) in the form

$$x_0 = b_0 + \sum_{j \in J^+} a_j y_j + \sum_{j \in J^-} a_j y_j;$$

define J^+ as the set of indices j where $a_j > 0$ and J^- as the set of indices j where $a_j < 0$.

If

$$\sum_{j \in J^+} a_j y_j + \sum_{j \in J^-} a_j y_j \leq 0,$$

then $x_0 \leq b_0$; also, $x_0 \leq \{b_0\} - 1$. Hence, we obtain

$$\sum_{j \in J^+} a_j y_j + \sum_{j \in J^-} a_j y_j \leq q - 1,$$

where $q = \{b_0\} - b_0 > 0$. Therefore,

$$\sum_{j \in J^-} a_j y_j \leq q - 1$$

and we can write

$$-\sum_{j \in J^-} \frac{q}{1-q} a_j y_j \geq q.$$

We then certainly have

$$\sum_{j \in J^+} a_j y_j - \sum_{j \in J^-} \frac{q}{1-q} a_j y_j \geq q. \tag{26}$$

If, on the other hand

$$\sum_{j\in J^+} a_j y_j + \sum_{j\in J^-} a_j y_j \geq 0,$$

then $x_0 \geq b_0$; also, $x_0 \geq \{b_0\}$. Hence, we obtain

$$\sum_{j\in J^+} a_j y_j + \sum_{j\in J^-} a_j y_j \geq q.$$

Therefore,

$$\sum_{j\in J^+} a_j y_j \geq q.$$

We then certainly have

$$\sum_{j\in J^+} a_j y_j - \sum_{j\in J^-} \frac{q}{1-q} a_j y_j \geq q,$$

which is the same as (26). Thus, (26) holds in either case.

In the mixed integer problem, we replace Eq. (9) by

$$y' = -q + \sum_{j\in J^+} a_j y_j - \sum_{j\in J^-} \frac{q}{1-q} a_j y_j \geq 0. \tag{27}$$

The analysis then is the same as after (9), where we define

$$p_j = \begin{cases} a_j, & \text{for } j \in J^+, \\ -\dfrac{q}{1-q} a_j, & \text{for } j \in J^-. \end{cases}$$

We develop the new form of (6) given by (11).

We can also produce a stronger constraint than (27) if some of the y_j variables are integer. The integers y_j are apparent when we have $x_j = y_j$ as one of the equations of (6) for integer x_j. Otherwise, y_j is fractional. We write (25) as

$$x_0 = b_0 + \sum_{j\in I} a_j y_j + \sum_{j\in F^+} a_j y_j + \sum_{j\in F^-} a_j y_j, \tag{28}$$

where we define:

I as the set of indices j where y_j is integer,
F^+ as the set of indices j where y_j is fractional and $a_j > 0$, and
F^- as the set of indices j where y_j is fractional and $a_j < 0$.

For $j \in I$, let

$$a_j = [a_j] + f_j, \qquad \text{if} \quad f_j \leq q, \tag{29}$$

or else

$$a_j = \{a_j\} - g_j, \qquad \text{if} \quad g_j < q. \tag{30}$$

Define I^+ as the set of indices $j \in I$ when (29) holds and I^- as the set of indices $j \in I$ when (30) holds. Thus, we can write (28) as

$$y = b_0 + \sum_{j \in I^+} f_j y_j + \sum_{j \in F^+} a_j y_j - \sum_{j \in I^-} g_j y_j + \sum_{j \in F^-} a_j y_j,$$

where integer y is

$$y = x_0 - \sum_{j \in I^+} [a_j] y_j - \sum_{j \in I^-} \{a_j\} y_j.$$

Following the previous analysis, we obtain

$$\begin{aligned} y' &= -q + \sum_{j=1}^{t} p_j y_j \\ &\geq 0 \end{aligned}$$

to replace (9) where

$$p_j = \begin{cases} f_j, & \text{for} \quad j \in I^+ \\ a_j, & \text{for} \quad j \in F^+ \\ \dfrac{q}{1-q} g_j, & \text{for} \quad j \in I^- \\ -\dfrac{q}{1-q} a_j, & \text{for} \quad j \in F^-. \end{cases}$$

The new constraint is stronger than (27) in the sense that β' may be lexicographically larger with the possibly smaller p_s value.

The convergence in the mixed integer case follows exactly as in the integer case if z is constrained to be integer and the row selected for (25) is for the topmost eligible component of β.

Example

Consider the example of page 110 with only z, x_1, and x_3 as integers.

2. Tableau E1 is formed.
3. The z row is selected. y_1 is integer. $q = \frac{2}{3}, p_1 = \frac{2}{3}, p_2 = \frac{10}{3}$.
4. (a) $J^+ = (1, 2); s = 2$. We form Tableau E2.
3. The x_1 row is selected. y_1 is integer. $q = \frac{2}{5}, p_1 = \frac{2}{5}, p_2 = \frac{2}{5}$.
4. (a) $J^+ = (2); s = 2$. We form Tableau E3.
   (b) The minimal solution is $z = 7, x_3 = 0, x_4 = \frac{1}{2}, x_1 = 3, x_2 = \frac{5}{2}$.

	1	2
z	$\frac{16}{3}$	$\frac{5}{3}$	$\frac{10}{3}$
x_3	0	1	0
x_4	0	0	1
x_1	$\frac{7}{3}$	$-\frac{10}{3}$	$\frac{4}{3}$
x_2	$\frac{5}{3}$	$\frac{1}{3}$	$\frac{5}{3}$

Tableau E1

	1	2
z	6	1	1
x_3	0	1	0
x_4	$\frac{1}{5}$	$-\frac{1}{5}$	$\frac{3}{10}$
x_1	$\frac{13}{5}$	$-\frac{18}{5}$	$\frac{2}{5}$
x_2	2	0	$\frac{1}{2}$

Tableau E2

	1	2
z	7	0	$\frac{5}{2}$
x_3	0	1	0
x_4	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{3}{4}$
x_1	3	-4	1
x_2	$\frac{5}{2}$	$-\frac{1}{2}$	$\frac{5}{4}$

Tableau E3

Problems

1. Find integers $x_j \geq 0$ that minimize z for

$$\begin{aligned} 16x_1 + 15x_2 + 12x_3 &= z, \\ 8x_1 + 5x_2 + 2x_3 - x_4 &= 4, \\ 2x_1 + 3x_2 + 3x_3 - x_4 &= 3. \end{aligned}$$

2. Find min z in the following integer programming problem:

$$\begin{aligned} -3x_1 + x_2 - 5x_3 + x_4 &= z, \\ x_1 - x_2 + 3x_3 + x_4 &= 8, \\ x_1 + 2x_2 - 2x_3 - 3x_4 &= 4. \end{aligned}$$

3. Minimize z when

$$\begin{aligned} z &= 3 + \tfrac{4}{3}x_1 + \tfrac{16}{3}x_3, \\ x_2 &= \tfrac{1}{2} - \tfrac{1}{6}x_1 - \tfrac{1}{6}x_3, \\ x_4 &= \tfrac{5}{2} - \tfrac{1}{2}x_1 - \tfrac{1}{2}x_3. \end{aligned}$$

4. By the method in this chapter, minimize z when $x_j = 0$ or 1 for $j = 1, 2, 3$:

$$\begin{aligned} x_1 + 2x_2 + x_3 &= z, \\ 2x_1 + 2x_2 + x_3 &\geq 3, \\ 2x_1 - 2x_2 + x_3 &\geq 1, \\ x_1 + 2x_2 - 2x_3 &\geq 1. \end{aligned}$$

5. Find integers $x_j \geq 0$ and min z for

$$\begin{aligned} 2x_1 + 3x_2 - x_3 + x_4 &= z, \\ x_1 - 2x_2 + 2x_3 - x_4 &= 4, \\ 2x_1 + x_2 - x_3 + x_4 &= 7. \end{aligned}$$

References

1. Gomory, R. E., An algorithm for the mixed integer problem, RAND Corp., Paper P-**1885**, February 22, 1960.
2. Gomory, R. E., An algorithm for integer solutions to linear programs, *in* "Recent Advances in Mathematical Programming" (R. L. Graves and P. Wolfe, eds.), pp. 269–302. McGraw-Hill, New York, 1963.
3. Gomory, R. E., An all-integer integer programming algorithm, *in* "Industrial Scheduling" (J. F. Muth and G. L. Thompson, eds.), pp. 193–206. Prentice-Hall, Englewood Cliffs, New Jersey, 1963.
4. Martin, G. T., An accelerated Euclidean algorithm for integer linear programming, *in* "Recent Advances in Mathematical Programming" (R. L. Graves and P. Wolfe, eds.), pp. 311–317. McGraw-Hill, New York, 1963.

6 NUMBER THEORY RESULTS

It is instructive to regard various concepts of number theory in the context of integer programming. Since some of the number theory results are needed in the next chapter, it is appropriate to offer the applicable material at this point. We shall also present algorithms using integer programming methods for some of the classical problems of number theory.

1 THE EUCLIDEAN ALGORITHM

The Euclidean algorithm provides a method for computing the greatest common divisor (gcd) of a set of positive integers $a_1, a_2, \ldots, a_n$. We can assume positive integers since the greatest common divisor of a set of integers is unchanged if any a_i of the set is replaced by $-a_i$. The problem can also be solved as an integer program: Find integers $x_1, x_2, \ldots, x_n$ that minimize

$$z = \sum_{j=1}^{n} a_j x_j, \tag{1}$$

where $z \geq 1$.

Proof that the optimal value of z is the greatest common divisor of the a_j is contained in the following four theorems. (See Griffin [1]).

Theorem 1

The minimum positive integer in the set of integers

$$L = \sum_{j=1}^{n} a_j x_j$$

is the greatest common divisor of the set, where the a_j are given integers and the x_j are all possible integers.

Proof

There is a finite number of integers between zero and any positive integer. The set L contains at least one positive integer; hence, the set has a minimum positive integer. Denote the minimum positive integer by

$$d = \sum_{j=1}^{n} a_j x_j'. \tag{2}$$

By Euclid's theorem, for any integer s and positive d, there exist integers p and q such that

$$s = pd + q, \qquad 0 \leq q < d. \tag{3}$$

For any s in L we also have

$$s = \sum_{j=1}^{n} a_j x_j''. \tag{4}$$

From (2), (3), and (4) we obtain

$$\sum_{j=1}^{n} a_j x_j'' = p \sum_{j=1}^{n} a_j x_j' + q$$

or

$$\sum_{j=1}^{n} a_j (x_j'' - p x_j') = q.$$

Since $x_j'' - p_j x_j'$ is an integer for each j, then q is an integer in L. Also, since $q < d$ and d is the minimum positive integer in L, q must equal zero. From (3) we see that d divides s and thus divides every member of L;[1] d is a common divisor of L. No integer greater than d is a common divisor of L since d is in L and it would have to be a divisor of d. Therefore, d is the greatest common divisor of L.

Theorem 2

The greatest common divisor of the a_j exists and is the minimum positive integer in the set L.

Proof

Each integer a_j is in L. From Theorem 1 the minimum positive integer d in the set is a common divisor of all the a_j. Therefore, there exist integers x_j' where

$$d = \sum_{j=1}^{n} a_j x_j';$$

consider d' any common divisor of the a_j. We have $a_j = a_j' d'$, where the a_j' are integers. Hence,

$$d = d' \sum_{j=1}^{n} a_j' x_j';$$

any common divisor of the a_j divides d. The greatest common divisor of the a_j then exists and is d.

Theorem 3

If d is the greatest common divisor of the a_j, then d is the greatest common divisor of the set of integers L.

[1] An integer b divides an integer a if there exists an integer c such that $a = bc$.

Proof

If $d = \gcd(a_1, a_2, \ldots, a_n)$, then $a_j = a_j' d$, where the a_j' are integers. Also, d is a common divisor of set L since

$$L = d \sum_{j=1}^{n} a_j' x_j .$$

Furthermore, any common divisor of the set divides the a_j because the a_j are in L. Any common divisor of the a_j is also a divisor of d (as seen in the proof of Theorem 2); hence, a common divisor of the set L must divide d. Thus d is the greatest common divisor of the set.

Theorem 4

The greatest common divisor of the set L is unique.

Proof

Suppose d_1 and d_2 are greatest common divisors of L. Since d_1 and d_2 are in L, d_1 must divide d_2 and d_2 must divide d_1. Consequently, $d_1 \leq d_2$ and $d_2 \leq d_1$. Therefore, $d_1 = d_2$.

From Theorems 1–4, it is clear that the greatest common divisor of $a_1, a_2, \ldots, a_n$ may be obtained by solving the integer program (1). The optimal value of z is then the greatest common divisor.

The application of Algorithm 3 of Chapter 3, properly modified for variables unrestricted in sign, provides a means for solving (1) and determining $\gcd(a_1, a_2, \ldots, a_n)$. Write problem (1) as

$$\begin{aligned} z &= \sum_{j=1}^{n} a_j y_j \geq 1, \\ x_j &= y_j, \qquad j = 1, 2, \ldots, n. \end{aligned} \tag{5}$$

We make a series of transformations of the y_j in (5) until an integer solution is apparent. At the start of any iteration we have

$$x = \sum_{j=1}^{n} \alpha_j y_j , \tag{6}$$

where x is a vector with components $z, x_1, x_2, \ldots, x_n$ and α_j is a column vector. Initially, the first component of α_j is a_j, the $(j+1)$ component is unity and all other components are zero.

Suppose the first component of α_j is taken as a_j. At any iteration define J^+ as the set of indices j where $a_j > 0$. If J^+ contains more than one member, find index s from

$$a_s = \min_{j \in J^+} a_j$$

and calculate $v_j = [a_j/a_s]$ for all $j \in J^+$. Make a change of variables for y_s given by

$$\sum_{\substack{j \in J^+ \\ j \neq s}} v_j y_j + y_s = y_s', \tag{7}$$

where y_s' is an integer. Eliminating y_s from (6) using (7), we obtain

$$x = \sum_{j \neq s} \alpha_j' y_j + \alpha_s y_s' \tag{8}$$

where

$$\begin{aligned} \alpha_j' &= \alpha_j - \alpha_s v_j, \qquad j \in J^+, \quad j \neq s \\ \alpha_j' &= \alpha_j, \qquad\qquad\quad j \notin J^+. \end{aligned} \tag{9}$$

Designating α_j' and y_s' as the current α_j and y_s, respectively, we obtain the form (6). The first component of α_j is again called a_j. We repeat the process. It is clear that in a finite number of iterations a form must develop with only one positive a_j value. The equivalent optimization problem becomes: Find integer y_s that minimizes z when

$$\begin{aligned} z &= a_s y_s \\ &\geq 1, \end{aligned} \tag{10}$$

where s is the only index in J^+. The other equations of (6) are not restricting and can be temporarily disregarded. The obvious solution of (10) is $y_s = 1$ with optimal $z = a_s$. (Again, the a_s value need not be one of the original a_j values; it represents the result of (9) over possibly many iterations.) The optimal z value obtained is the greatest common divisor

of the original a_j values. The complete solution is then

$$x = \alpha_s + \sum_{j \neq s} \alpha_j y_j, \tag{11}$$

where the y_j, for $j \neq s$, act as parameters.

We use (11) to find all possible x_j values that produce the gcd(a_1, a_2, ..., a_n). If the greatest common divisor is desired and the x_j values not needed, the method is performed with only the first components of the α_j.

Example

Find the greatest common divisor of 15, 36, and 81 together with the x_j values. We list successively

15	36	81	15	6	6	3	6	0	3	0	0
1	0	0	1	−2	−5	5	−2	−3	5	−12	−3
0	1	0	0	1	0	−2	1	−1	−2	5	−1
0	0	1	0	0	1	0	0	1	0	0	1

and obtain $z = 3$ as the greatest common divisor with $x_1 = 5 - 12y_2 - 3y_3$, $x_2 = -2 + 5y_2 - y_3$, and $x_3 = y_3$.

Theorem 5

The set of integers L consists of all and only multiples of d, the greatest common divisor of $a_1, a_2, \ldots, a_n$.

Proof

Every integer of the set L has been shown to be a multiple of d. Since there exist integer x_j^* values with

$$d = \sum_{j=1}^{n} a_j x_j^*,$$

then

$$kd = \sum_{j=1}^{n} a_j (k x_j^*)$$

is obviously in L. Thus, all, and only, multiples of d are in L.

Theorem 6

If d_1 is the greatest common divisor of $a_1, a_2, \ldots, a_n$, then the greatest common divisor of $a_1, a_2, \ldots, a_n, a_{n+1}$ is the greatest common divisor of d_1 and a_{n+1}.

Proof

There exist integers x_j so that

$$d_1 = \sum_{j=1}^{n} a_j x_j .$$

If $d_2 = \gcd(d_1, a_{n+1})$, then there exist integers y_1, y_2 so that

$$d_2 = d_1 y_1 + a_{n+1} y_2 .$$

We have

$$d_2 = \sum^{n} a_j (y_1 x_j) + a_{n+1} y_2 .$$

If $d_3 = \gcd(a_1, a_2, \ldots, a_n, a_{n+1})$, we see that d_3 divides d_2. Furthermore, since d_1 divides a_j, for $j = 1, 2, \ldots, n$, and since d_2 divides d_1 and a_{n+1}, we see that d_2 is a common divisor of a_j, for $j = 1, 2, \ldots, n, n+1$; hence, d_2 divides d_3 from the proof of Theorem 2. Therefore $d_2 = d_3$.

2 LINEAR DIOPHANTINE EQUATIONS

A *Diophantine equation* is a rational integral equation in which the constants are integers and the solutions, or values of the variables that satisfy the equations, are integers.

Theorem 7

The linear Diophantine equation

$$\sum_{j=1}^{n} a_j x_j = b \tag{12}$$

has a solution if and only if the greatest common divisor of $a_1, a_2, \ldots, a_n$ divides b.

Proof

In the previous theorems, we showed that $d = \gcd(a_1, a_2, \ldots, a_n)$ divides $L = \sum_{j=1}^{n} a_j x_j$ for all integral values x_j. If (12) has a solution, then d divides b. Alternatively, if d divides b, let $b = b_0 d$. Since d is in L there exist integers x_j' so that

$$b = b_0 \sum_{j=1}^{n} a_j x_j'.$$

Thus, $x_j = b_0 x_j'$ solves Eq. (12).

Integers a_1 and a_2 are *relatively prime* if the greatest common divisor of a_1 and a_2 is unity. Thus, a necessary and sufficient condition that a_1 and a_2 are relatively prime is that there is a solution of the linear Diophantine equation

$$a_1 x_1 + a_2 x_2 = 1.$$

Theorem 8

If m divides ab, and m and a are relatively prime, then m divides b.

Proof

Since m and a are relatively prime,

$$mx_1 + ax_2 = 1$$

has a solution x_1', x_2'. Then

$$b(mx_1' + ax_2') = b$$

and

$$mbx_1' + abx_2' = b.$$

Since m divides ab, then $ab = mr$ for some integer r and

$$m(bx_1' + rx_2') = b.$$

Hence, m divides b.

The system of linear Diophantine equations

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m \tag{13}$$

may be solved by modifying Algorithm 3 of Chapter 3 for variables unrestricted in sign. Write (13) as

$$\sum_{j=1}^{n} a_{ij} y_j = b_i, \qquad i = 1, 2, \ldots, m, \tag{14}$$

$$y_j = x_j, \qquad j = 1, 2, \ldots, n.$$

We make a series of transformations of the y_j in (14) until an integer solution is apparent. At the beginning of any iteration we have

$$\sum_{j=1}^{n} \alpha_j y_j = x. \tag{15}$$

At the first iteration x is a vector with components $b_1, b_2, \ldots, b_m$, $x_1, x_2, \ldots, x_n$, and α_j is a column vector. Also, initially, component i, $1 \leq i \leq m$ of α_j is a_{ij}; component $m+j$ is unity and all other components are zero.

Consider the first equation of (15) as

$$\sum_{j=1}^{n} a_j y_j = b_0. \tag{16}$$

Define J as the set of indices j where $a_j \neq 0$. If J contains more than one member, find index s from

$$|a_s| = \min_{j \in J} |a_j|,$$

where $|a|$ means the absolute value of a. For each $j \in J, j \neq s$, determine an integer v_j value that produces the minimum of $|a_j - v_j a_s|$. Note that v_j may be positive or negative. Make a change of variables for y_s given by

$$\sum_{\substack{j \in J \\ j \neq s}} v_j y_j + y_s = y_s', \tag{17}$$

where y_s' is an integer. Eliminating y_s from (16) using (17), we obtain

$$\sum_{j \neq s} \alpha_j' y_j + \alpha_s y_s' = x, \tag{18}$$

where

$$\alpha_j' = \alpha_j - \alpha_s v_j, \qquad j \in J, \quad j \neq s,$$
$$\alpha_j' = \alpha_j, \qquad j \notin J.$$

Designating α_j' and y_s' as the current α_j and y_s respectively, we obtain the form (15). We repeat the process. It is clear that in a finite number of iterations a form must develop with only one nonzero term in (16). The equation then reads

$$a_s y_s = b_0. \tag{19}$$

To have a solution, a_s must divide b_0. If this is the case, we eliminate y_s from (15) using (19) and obtain

$$\beta + \sum_{j \neq s} \alpha_j y_j = x, \tag{20}$$

where $\beta = \alpha_s b_0 / a_s$. If β is subtracted from both sides of (20), we again have the form (15) with x replaced by $x - \beta$. Note that in this new form of (15) the first row has all zero values on both sides. The s column has been eliminated. Next the first row is discarded. The new first row is then changed like (16) until only one nonzero element is left. We obtain (19) and (20) once more. Another column and row are eliminated and the process is continued until only the last n rows of (15) remain; at this point the solution is

$$\beta' + \sum \alpha_j y_j = x, \tag{21}$$

where β' is the result of m applications of (19) and (20) and the sum in (21) is over the columns remaining. The y_j variables in (21) represent parameters.

Example

Find integers x_j that satisfy

$$4x_1 - 4x_2 + 3x_3 = 3,$$
$$5x_1 - 7x_2 + 6x_3 = 4.$$

We list successively with $x = (3, 4, x_1, x_2, x_3)$

4	−4	3	1	−1	3	1	0	0
5	−7	6	−1	−1	6	−1	−2	9
1	0	0	1	0	0	1	1	−3
0	1	0	0	1	0	0	1	0
0	0	1	−1	1	1	−1	0	4

We obtain $\beta = (3, -3, 3, 0, -3)$; x is replaced by $(0, 7, x_1 - 3, x_2, x_3 + 3)$. We discard the first row and first column and list successively

−2	9	−2	1	0	1
1	−3	1	1	3	1
1	0	1	4	9	4
0	4	0	4	8	4

We obtain $\beta = (7, 7, 28, 28)$. The solution is

$$x_1 = 10 + 3y,$$
$$x_2 = 28 + 9y,$$
$$x_3 = 25 + 8y.$$

3 LINEAR CONGRUENCES

For any nonzero integer m an integer a can be expressed uniquely as

$$a = qm + r,$$

where $0 \leq r < |m|$. All integers can be separated into m classes according to the remainder r developed after division by m. Two integers, a and b, are *congruent modulo m* if and only if division of each by m results in the same remainder r. Thus

(22)
$$a = q_1 m + r,$$
$$b = q_2 m + r,$$

where $0 \leq r < |m|$; a and b are considered to be in the same *residue class*. The integers congruent to a given integer for the modulus m make up a residue class modulo m.

The notation

$$a \equiv b \mod m$$

means that a is congruent to b modulo m or that a is congruent to b for the modulus m.

We have

Theorem 9

Two integers are congruent modulo m if and only if their difference is divisible by nonzero m.

Proof

Suppose a is congruent to b modulo m. From (22)

$$a - b = m(q_1 - q_2)$$

and m divides $a - b$. Conversely, if $a - b = mk$, than $a \equiv b \bmod m$, since otherwise

$$a = q_1 m + r_1,$$
$$b = q_2 m + r_2,$$

with $0 \leq r_1, r_2 < |m|$. We have

$$a - b = m(q_1 - q_2) + r_1 - r_2;$$

hence, m divides $r_1 - r_2$, which can occur only if $r_1 = r_2$. Thus we have proved that a and b are in the same residue class.

The following properties of congruences can be stated:

1. If $a \equiv b \bmod m$ and $c \equiv d \bmod m$, then $a \pm c \equiv b \pm d \bmod m$ and $ac \equiv bd \bmod m$.
2. If d divides m and $a \equiv b \bmod m$, then $a \equiv b \bmod d$.

3. If $a \equiv b \bmod m_1$ and $a \equiv b \bmod m_2$, then $a \equiv b \bmod m_3$ where m_3 is the least common multiple of m_1 and m_2.[2]

4. If $ac \equiv bc \bmod m$ and $\gcd(c, m) = 1$, then $a \equiv b \bmod m$.

5. If $ac \equiv bc \bmod m$ and $\gcd(c, m) = d$, then $a \equiv b \bmod m_0$, where $m = m_0 d$.

We need to solve the linear congruence

$$ax \equiv b \mod m.$$

We have first

Theorem 10

When a is prime to m, the congruence

$$ax \equiv 1 \mod m \tag{23}$$

has one and only one solution modulo m. This solution is also prime to m.

Proof

The congruence (23) is equivalent to the equation

$$ax + my = 1, \tag{24}$$

which has an integer solution, from Theorem 7, when a is prime to m. If both x_1 and x_2 satisfy (23), then

$$ax_1 \equiv ax_2 \mod m$$

and

$$x_1 \equiv x_2 \mod m.$$

Furthermore, if x_1 satisfies (23), then the equation

$$x_1 u + mv = 1$$

[2] If $m = q_1 a = q_2 b$, then m is a *common multiple* of a and b. If, in addition, every common multiple of a and b is a multiple of m, then m is the *least common multiple* of a and b.

has a solution for u and v, namely, $u = a$ and $v = y$ from (24). Thus x_1 and m are relatively prime.

Theorem 11

When a is prime to m, the congruence

$$ax \equiv b \mod m \tag{25}$$

has one and only one solution modulo m.

Proof

Since a is prime to m, from Theorem 10, the congruence

$$ax \equiv 1 \mod m$$

has a unique solution

$$x \equiv x_1 \mod m.$$

Clearly, $x_1 b$ satisfies (25). As proved in Theorem 10, there can be but one solution modulo m; the solution, however, may not be prime to m.

Theorem 12

If d is the greatest common divisor of a and m, then

$$ax \equiv b \mod m \tag{26}$$

has a solution if and only if d divides b. There are d solutions modulo m when d divides b.

Proof

Since $d = \gcd(a, m)$, then $a = a_0 d$ and $m = m_0 d$. If the congruence has a solution, there exist integers x' and y' so that

$$ax' + my' = b.$$

Thus

$$d(a_0 x' + m_0 y') = b$$

and d divides b. Conversely, if d divides b, let $b = cd$ and reduce the congruence to

$$a_0 x \equiv c \mod m_0. \tag{27}$$

Furthermore, there exist integers x^* and y^* so that

$$ax^* + my^* = d;$$

hence,

$$a_0 x^* + m_0 y^* = 1.$$

Thus a_0 and m_0 are relatively prime and (27) has one solution x_1 modulo m_0. Consider the class of integers of the form $x_1 + km_0$. All of the integers in the class also solve (26). Some of the integers are congruent to each other modulo m and would represent the same solution to (26). For these integers there would be values k_1 and k_2 with

$$x_1 + k_1 m_0 \equiv x_1 + k_2 m_0 \mod m.$$

Hence,

$$m_0(k_1 - k_2) \equiv 0 \mod m$$

and

$$k_1 \equiv k_2 \mod d.$$

Therefore, when k is 0, 1, ..., $d - 1$ the integers $x_1 + km_0$ are exactly d solutions of (26) that are incongruent for the modulus m.

Example

Solve $9x \equiv 15 \mod 24$.

Since $\gcd(9, 24) = 3$ and 3 divides 15, the congruence is reduced to $3x \equiv 5 \mod 8$, which has one solution $x \equiv 7 \mod 8$. Hence, the solutions of the original congruence are $x \equiv 7, 15, 23 \mod 24$. All other solutions are in one of these 3 residue classes modulo 24.

We are now interested in a method for solving

$$ax \equiv b \mod m.$$

The equivalent problem is: Find integers $x = x_1$ and x_2 that satisfy

$$ax_1 + mx_2 = b.$$

We solve this equation by the method of Section 2. We list

a	m
1	0

and reduce the first row to a form having zero and nonzero elements. The second row then contains the solution for x.

Example

Solve $3x \equiv 5 \mod 8$.
We list

3	8
1	0

and form successively

3	8	3	−1	0	−1
1	0	1	−3	−8	−3

The first row produces

$$-y_2 = 5$$

and the solution is then apparent in the second row as

$$x = 15 - 8k.$$

Thus $x \equiv 15 \mod 8$; $x \equiv 7 \mod 8$ solves the congruence.

4 THE SOLUTION OF A SYSTEM OF LINEAR CONGRUENCES

The system of linear congruences

$$\sum_{j=1}^{n} a_{ij} x_j \equiv b_i \mod m_i, \qquad i = 1, 2, \ldots, t$$

may be solved by writing the equivalent system of linear Diophantine equations

$$\sum_{j=1}^{n} a_{ij} x_j + m_i y_i = b_i, \qquad i = 1, 2, \ldots, t.$$

The theorems and method of solution presented in Section *2* may then be applied to the system of equations.

Example

The Chinese remainder theorem states that if $m_1, m_2, \ldots, m_t$ are relatively prime in pairs, then the system

$$\begin{aligned} x &\equiv a_1 \mod m_1, \\ x &\equiv a_2 \mod m_2, \\ &\vdots \\ x &\equiv a_t \mod m_t \end{aligned}$$

has a unique solution modulo m, where $m = m_1 m_2 \cdots m_t$. Solve $x \equiv 3 \bmod 14$, $x \equiv 4 \bmod 9$, $x \equiv 6 \bmod 11$.

We write the equivalent equations

$$\begin{aligned} x + 14y_1 &= 3, \\ x + 9y_2 &= 4, \\ x + 11y_3 &= 6. \end{aligned}$$

Using the method in Section 2, we write the tableau

3	1	14	0	0
4	1	0	9	0
6	1	0	0	11
x	1	0	0	0

;

We do not need to keep track of the y_i variables. Thus we obtain successively

3	1	0	0	0
4	1	−14	9	0
6	1	−14	0	11
x	1	−14	0	0

1	−14	9	0	4	9	0	4	1	0	0	1	0
3	−14	0	11	−14	0	11	−14	28	11	−126	28	11
$x-3$	−14	0	0	−14	0	0	−14	28	0	−126	28	0

−25	−126	11	−5	11	−5	1	0	1
$x-31$	−126	0	−126	0	−126	−252	−1386	−252

Thus $x = 6331 - 1386k$ or $x \equiv 787 \bmod 1386$.

Problems

1. Find integers x_j that minimize

$$z = 42x_1 + 63x_2 + 78x_3 + 102x_4$$

for $z \geq 1$.

2. Find the greatest common divisor of 117, 108, 54, and 135.
3. Solve the simultaneous Diophantine equations

$$\begin{aligned} 7x_1 - 3x_2 + 3x_3 - 2x_4 &= 12, \\ 3x_1 + 2x_2 + 3x_3 - 7x_4 &= 4, \\ 2x_1 - x_2 + 5x_3 + x_4 &= 9. \end{aligned}$$

4. Find integers that satisfy

$$33x_1 - 7x_2 + 5x_3 - 4x_4 = 54.$$

5. Solve the system of congruences

$$\begin{aligned} x &\equiv 6 \mod 7 \\ x &\equiv 1 \mod 5 \\ x &\equiv 3 \mod 4 \\ x &\equiv 9 \mod 11. \end{aligned}$$

Reference

1. Griffin, H., "Elementary Theory of Numbers." McGraw-Hill, New York, 1954.

7 DYNAMIC PROGRAMMING SOLUTIONS

We solve the integer programming problem by means of a dynamic programming enumeration. The method can be used to solve the important class of problems in which the variables have upper bound values. This includes the problems where the variables are restrained to zero or one.

In Chapter 5, we relaxed the integer constraint on the variables and used the simplex method to find the continuous solution. If the continuous solution was fractional, we used additional constraints and minimization cycles to produce the optimal integer solution. In this chapter, we again relax the integer constraint on the variables and use the simplex method to find the continuous solution. We go on to develop linear congruences that the nonbasic variables must satisfy. We then obtain the integer solution by means of a dynamic programming enumeration analogous to the one presented in Chapter 4. This approach was developed in [2]. The remainder of the chapter contains rules for accelerating the enumeration.

1 A DYNAMIC PROGRAMMING SOLUTION

Once again, consider the integer programming problem: Find integers $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{n} c_j x_j = z, \\ &\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

where the a_{ij}, b_i, and c_j are given integer constants. Equation (1) may be solved as a linear program by relaxing the integer constraints to obtain the optimal canonical format

$$\begin{aligned} &-z + \sum_{j=1}^{t} \bar{c}_j x_j = -\bar{z}_0, \\ &x_{t+i} + \sum_{j=1}^{t} \bar{a}_{ij} x_j = \bar{b}_i, \qquad i = 1, 2, \ldots, m, \end{aligned} \tag{2}$$

where the first $t = n - m$ and the last m variables have been arbitrarily selected as the nonbasic and basic variables, respectively. Since (2) is an optimal format, $\bar{c}_j \geq 0$ and $\bar{b}_i \geq 0$; the optimal continuous solution to (1) is $x_{t+i} = \bar{b}_i$ for the basic variables and $x_j = 0$ for the nonbasic variables. If the continuous solution has all $\bar{b}_i$ as integer, the integer program is solved. If any of the $\bar{b}_i$ are fractional, however, the integer variables x_j must still satisfy (2), which are then used for obtaining the integer solution.

Select some equation of (2) with $\bar{b}_i$ as fractional. Suppose it is

$$x_r + \sum_{j=1}^{t} a_j x_j = b_0. \tag{3}$$

Define $f_0 > 0$ from $b_0 = [b_0] + f_0$; Eq. (3) may be written as

$$\sum_{j=1}^{t} a_j x_j - f_0 = [b_0] - x_r. \tag{4}$$

The right side of (4) is an integer; thus

$$\sum_{j=1}^{t} a_j x_j \equiv f_0 \mod 1. \tag{5}$$

In (5) we have developed a congruence, which means that $\sum_{j=1}^{t} a_j x_j - f_0$ is an integer. Furthermore, define the fractional parts $f_j \geq 0$ from $a_j = [a_j] + f_j$; Eq. (5) may be written as

$$\sum_{j=1}^{t} f_j x_j \equiv f_0 \mod 1. \tag{6}$$

Congruence (6) arises because $[a_j]x_j \equiv 0 \mod 1$ for integer x_j. We have now developed a congruence that the nonbasic integer variables must satisfy to produce an integer value for basic variable x_r. If b_0 is integer and some a_j are fractional, then (6) still holds with $f_0 = 0$. If b_0 and the a_j are all integers, then the trivial congruence $0 \equiv 0 \mod 1$ results. Thus we may write a congruence similar to (6) for every equation of (2). We need not develop a congruence from the objective equation of (2) since it is apparent that if all solutions to the congruences like (6) produce integers x_j, then z must have an integer value from (1).

Let us write (2) in the form

$$\begin{aligned} -z + \sum_{j=1}^{t} \bar{c}_j x_j &= -\bar{z}_0, \\ x + \sum_{j=1}^{t} \alpha_j x_j &= \alpha_0, \end{aligned} \tag{7}$$

where x is a vector with components x_{t+i}, each α_j for $j \neq 0$ is a column vector with components $\bar{a}_{ij}$, and α_0 is a column vector with components $\bar{b}_i$. The congruences developed from (7), as in (6), may be written as

$$\sum_{j=1}^{t} \beta_j x_j \equiv \beta_0 \mod 1 \tag{8}$$

where the β_j are the columns of fractional parts of the α_j. In addition, since $x_{t+i} \geq 0$, the constraints of (7) may appear as

$$\sum_{j=1}^{t} \alpha_j x_j \leq \alpha_0.$$

It is clear that the equivalent problem to (1) is: Find integers $x_j \geq 0$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{t} \bar{c}_j x_j = z - \bar{z}_0, \\ &\sum_{j=1}^{t} \alpha_j x_j \leq \alpha_0, \\ &\sum_{j=1}^{t} \beta_j x_j \equiv \beta_0 \mod 1. \end{aligned} \tag{9}$$

When the optimal solution x_j^0 of (9) is determined, then the basic variables x of (7) are given by

$$x^0 = \alpha_0 - \sum_{j=1}^{t} \alpha_j x_j^0.$$

We can solve (9) by writing the function

$$F(\alpha, \beta) = \min\left(\sum_{j=1}^{t} \bar{c}_j x_j \,\middle|\, \sum_{j=1}^{t} \alpha_j x_j = \alpha, \sum_{j=1}^{t} \beta_j x_j \equiv \beta \mod 1, \text{ integer } x_j \geq 0 \right),$$

where α and β are variable vectors and the inequalities of (8) are changed to equalities.[1] Since some $x_j > 0$, we can use the same argument as in Chapter 4 to develop the dynamic programming recursion

$$F(\alpha, \beta) = \min_j(\bar{c}_j + F(\alpha - \alpha_j, \beta - \beta_j)). \tag{10}$$

Note that the second argument of F is taken modulo 1.

The recursion in (10) may be solved as a simple enumeration. Observe that $F(\alpha_r, \beta_r) = \bar{c}_r$, where $\bar{c}_r = \min \bar{c}_j$. We then form $F(\alpha - \alpha_r, \beta - \beta_r)$ by replacing α and β by $\alpha - \alpha_r$ and $\beta - \beta_r$ respectively in (10) as

$$F(\alpha - \alpha_r, \beta - \beta_r) = \min_j(\bar{c}_j + F(\alpha - \alpha_j - \alpha_r, \beta - \beta_j - \beta_r)). \tag{11}$$

[1] Gomory [1] initiated this type of round-off procedure by considering $F(\beta) = \min(\sum_{j=1}^{t} \bar{c}_j x_j \mid \sum_{j=1}^{t} \beta_j x_j \equiv \beta \bmod 1, \text{integer } x_j \geq 0)$; (9) is solved by his method if x^0 emerges as nonnegative.

We substitute (11) for the $F(\alpha - \alpha_r, \beta - \beta_r)$ term on the right side of (10). Thus we obtain

$$(12) \quad F(\alpha, \beta) = \min(\min_j(c_j' + F(\alpha - \alpha_j', \beta - \beta_j')), \min_{j \neq r}(\bar{c}_j + F(\alpha - \alpha_j, \beta - \beta_j)))$$

where $c_j' = \bar{c}_r + \bar{c}_j$, $\alpha_j' = \alpha_j + \alpha_r$, and $\beta_j' \equiv \beta_j + \beta_r \bmod 1$. Designate c_j' α_j', and β_j' as the current $\bar{c}_j$, α_j, and β_j values; observe that (12) is of the same form as (10), except that the $\bar{c}_r + F(\alpha - \alpha_r, \beta - \beta_r)$ term is missing and other terms are included. Thus from (12) we can produce the solution value $F(\alpha_r, \beta_r) = \bar{c}_r$ by finding a new $\bar{c}_r = \min \bar{c}_j$. Continuing in this way, we find $F(\alpha, \beta)$ for all feasible α and β. We can also obtain the corresponding x_j values by keeping track of the indices j that produce each $F(\alpha, \beta)$.

The procedure just outlined is an enumeration of all possible values of $\sum_{j=1}^{t} \bar{c}_j x_j$ in order of increasing value. To solve the integer program given by (9), we stop the procedure when $F(\alpha, \beta_0)$ is calculated for some $\alpha \leq \alpha_0$ (i.e., when each component of α is less than or equal to the corresponding component of α_0).

The method for the solution to (10) may fail in case any $\bar{c}_j$ are zero; the same r value may be picked an infinite number of times and the constraints may never be satisfied. Thus, to insure a finite algorithm, we must assume that all x_j values have finite upper bounds m_j.

Let $x_j \leq m_j$ for $j = 1, 2, \ldots, n$ be an additional constraint in (1). The bounded variable linear programming technique is used to transform (1) to (2). Some of the nonbasic variables may be at their upper bounds in the continuous solution of (1). These variables have corresponding $\bar{c}_j$ values that are nonpositive. We also have $\bar{c}_j \geq 0$ for nonbasic variables at the zero value. For ease of computation, we make the transformation $x_j' = m_j - x_j$ for those nonbasic variables at their upper bound. When the optimal solution is obtained, we use the reverse transformation to calculate the x_j value. Thus we may assume a form like (2) with all $\bar{c}_j \geq 0$ and all nonbasic variables at zero value.

When the variables are bounded, the dynamic programming enumeration is performed so that the upper bounds m_j are not violated.

If we define M as a vector with ith component m_{t+i}, we stop the procedure when $F(\alpha, \beta_0)$ is calculated with $0 \leq \alpha_0 - \alpha \leq M$.

Thus we are led to

Algorithm 1

1. Define a solution vector $S(z^*, x_1^*, x_2^*, \ldots, x_t^*)$, where x_j^* for $j = 1, 2, \ldots, t$ is a feasible integer solution to the problem with objective value z^*. If no feasible solution is apparent, take $z^* = \infty$. Go to 2.
2. List the values of the problem as

1	2	3	...	t
$\bar{c}_1$	$\bar{c}_2$	$\bar{c}_3$	...	$\bar{c}_t$
α_1	α_2	α_3	...	α_t
β_1	β_2	β_3	...	β_t

Go to 3.

3. In the newly listed section, define J as the set of indices j where $\beta_j = \beta_0$ and $0 \leq \alpha_0 - \alpha_j \leq M$. If J has no members, go to 4. Otherwise, find index r from $\bar{c}_r = \min_{j \in J} \bar{c}_j$. If $\bar{c}_r \geq z^*$, go to 4. Otherwise, take $z^* = \bar{c}_r$ and form $S(z^*, x_1^*, x_2^*, \ldots, x_t^*)$, where the x_j^* are the nonzero x_j values found below the section. The other x_j^* values equal zero. Increase x_r^* by one and go to 4.

4. Given the list, the solution $S(z^*, x_1^*, x_2^*, \ldots, x_t^*)$ is the minimal integer solution if there are no unmarked columns with $\bar{c}_j < z^*$. Go to 6. Otherwise, find index r from $\bar{c}_r = \min \bar{c}_j$ for all unmarked columns in all sections. Go to 5.

5. Add a new section of columns to the list as follows, if possible:
   (a) Calculate $c_j' = \bar{c}_r + \bar{c}_j$, $\alpha_j' = \alpha_r + \alpha_j$, and $\beta_j' \equiv \beta_r + \beta_j \bmod 1$ for the indices j of the unmarked columns in the section containing column r. Mark the r column. The $\bar{c}_j$, α_j, and β_j values are taken from the list in step 2.
   (b) Add columns with values c_j', α_j', and β_j' headed by indices j if $c_j' < z^*$. Include a column headed by r only if $x_r + 1 < m_r$ for the x_r value found below the section containing the newly

marked r column. If no new columns are added, go to 3. Designate the c_j', α_j', and β_j' values as new $\bar{c}_j$, α_j, and β_j values, respectively.

(c) Underneath the section added, write the x_j values from the section containing the newly marked r column. The non-appearance of a variable means that it has zero value. Increase x_r by one for the new section. Go to 3.

6. The minimal solution to the integer program is $z = \bar{z}_0 + z^*$, $x = \alpha_0 - \sum_{j=1}^{t} \alpha_j x_j^*$. End.

Example

Consider the problem first given on page 110: Find integers $x_j \geq 0$ that minimize z when

$$\begin{aligned} 3x_1 - x_2 + 12x_3 + x_4 &= z, \\ x_1 + x_2 + 3x_3 - 3x_4 &= 4, \\ 3x_1 - 3x_2 + 11x_3 + x_4 &= 2. \end{aligned}$$

Using the simplex algorithm, we transform the equations to

$$\begin{aligned} -z \quad + \tfrac{5}{3}x_3 + \tfrac{10}{3}x_4 &= -\tfrac{16}{3}, \\ x_1 \quad + \tfrac{10}{3}x_3 - \tfrac{4}{3}x_4 &= \tfrac{7}{3}, \\ x_2 - \tfrac{1}{3}x_3 - \tfrac{5}{3}x_4 &= \tfrac{5}{3}, \end{aligned}$$

and develop the congruences

$$\begin{aligned} \tfrac{1}{3}x_3 + \tfrac{2}{3}x_4 &\equiv \tfrac{1}{3} \mod 1, \\ \tfrac{2}{3}x_3 + \tfrac{1}{3}x_4 &\equiv \tfrac{2}{3} \mod 1. \end{aligned}$$

1. $z^* = \infty$.
2. The problem is listed in Tableau E1; $\bar{z}_0 = \frac{16}{3}$, $\alpha_0 = (\frac{7}{3}, \frac{5}{3}$, and $\beta_0 = (\frac{1}{3}, \frac{2}{3})$.
3. J has no members.
4. $r = 3$ in Tableau E1, $\bar{c}_r = \frac{5}{3}$.
5. We form Tableau E2. Mark column 3 of Tableau E1.
3. J has no members.

3*	4*	3*	4	4	3
$\frac{5}{3}$	$\frac{10}{3}$	$\frac{10}{3}$	5	$\frac{20}{3}$	5
$\frac{10}{3}$	$-\frac{4}{3}$	$\frac{20}{3}$	2	$-\frac{8}{3}$	10
$-\frac{1}{3}$	$-\frac{5}{3}$	$-\frac{2}{3}$	-2	$-\frac{10}{3}$	-1
$\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0	$\frac{1}{3}$	0
$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{2}{3}$	0
		$x_3 = 1$		$x_4 = 1$	$x_3 = 2$
E1		E2		E3	E4

4. $r = 4$ in Tableau E1, $\bar{c}_r = \frac{10}{3}$.
5. We form Tableau E3. Mark column 4 of Tableau E1.
3. $J = (4)$ in Tableau E3. $r = 4$ and $\bar{c}_r = \frac{20}{3}$; we have a feasible solution $z^* = \frac{20}{3}$, $x_3^* = 0$, $x_4^* = 2$.
4. $r = 3$ in Tableau E2, $\bar{c}_r = \frac{10}{3}$.
5. We form Tableau E4. Mark column 3 of Tableau E2. No further tableaus are formed and the current feasible solution is minimal.
6. The solution to the original problem is then $z = 12$, $x_1 = 5$, $x_2 = 5$, $x_3 = 0$, $x_4 = 2$.

2 REDUCING THE NUMBER OF CONGRUENCES

Algorithm 1 required that calculations be made for m congruences in (8). Ordinarily, it is not necessary to include all the congruences in a problem; all solutions of one congruence may satisfy a second congruence, which, then, need not be included. We now show how to determine the congruences to be included.

On the basis of results in Chapter 5, we know that the constants of the optimal format (2) are in the form i/D. Numerator i is an integer and denominator D is the absolute value of the determinant of the coefficient matrix of basic variables $x_{t+1}, x_{t+2}, \ldots, x_n$ in the original problem Eq. (1). Furthermore, the constants in congruence (6) are also

in the form i/D, since they are the fractional parts of the constants in an equation in (2). Hence, congruence (6) may be written as

$$\sum_{j=1}^{t} \frac{h_j}{D} x_j \equiv \frac{h_0}{D} \mod 1, \tag{13}$$

where $f_j = h_j/D$ and h_j is an integer. Other congruences developed from (2) may have the same integer solution as in (13).

A congruence that has the same integer solution as in (13) may be generated by forming

$$\sum_{j=1}^{t} g_j x_j \equiv g_0 \mod 1, \tag{14}$$

where

$$g_j \equiv \frac{kh_j}{D} \mod 1, \qquad j = 0, 1, \ldots, t; \tag{15}$$

k is a positive integer. The fact that all integer solutions of (13) also satisfy (14) is seen from

Theorem 1

If integer x_j^* solves (13), then x_j^* solves (14).

Proof

If x_j^* solves (13), we have

$$\sum_{j=1}^{t} \frac{h_j}{D} x_j^* - \frac{h_0}{D} = w, \tag{16}$$

where w is an integer. From (15) we obtain

$$g_j - \frac{kh_j}{D} = y_j, \qquad j = 0, 1, \ldots, t, \tag{17}$$

where y_j is an integer. Multiplying (16) through by positive integer k and using (17), we obtain

$$\sum_{j=1}^{t} g_j x_j^* - g_0 = kw - y_0 + \sum_{j=1}^{t} y_j x_j^*. \tag{18}$$

Since the right side of (18) is an integer, it follows that

$$\sum_{j=1}^{t} g_j x_j^* \equiv g_0 \mod 1.$$

Thus x_j^* solves (14), thereby proving the theorem.

Since any solution of congruence (13) satisfies (14), we need not include the latter in (8). Observe that some solutions to congruence (14) may not satisfy (13); care must be taken in picking the congruence to keep when both are developed from (2).

Sometimes it may happen that the congruences are equivalent. Two congruences are *equivalent congruences* if they have exactly the same solutions. If k and D are relatively prime, then (13) and (14) are equivalent; either congruence may be used. Thus

Theorem 2

If $\gcd(k, D) = 1$, then (13) and (14) are equivalent congruences.

Proof

Since the converse was proven in Theorem 1, we need prove only that if x_j^* solves (14), then it solves (13). Let x_j^* satisfy (14); then

$$\sum_{j=1}^{t} g_j x_j^* - g_0 = w,$$

where w is an integer. Using (17), we obtain

$$k\left(\sum_{j=1}^{t} h_j x_j^* - h_0\right) = Dy, \tag{19}$$

where integer y is given by

$$y = w - \sum_{j=1}^{t} y_j x_j^* + y_0.$$

Looking at (19), we see that k must divide y because $\gcd(k, D) = 1$. Thus, $y = ky^*$, where y^* is an integer and

$$\sum_{j=1}^{t} h_j x_j^* - h_0 = Dy^*. \tag{20}$$

Dividing both sides of (20) by D, we see that x_j^* satisfies (13), thereby proving the theorem.

In a given problem it is possible for a single congruence to generate all the other congruences. It is therefore desirable to find the congruence of (8) that generates the largest number of congruences. We have

Theorem 3

If h_j and D are relatively prime, then kh_j/D is incongruent to h_j/D mod 1 for $k = 2, 3, \ldots, D$.

Proof

Suppose

$$\frac{kh_j}{D} \equiv \frac{h_j}{D} \mod 1.$$

There exists an integer y so that

$$\frac{kh_j}{D} - \frac{h_j}{D} = y$$

and

$$h_j(k-1) = yD.$$

Since h_j and D are relatively prime, h_j must divide y, or $y = h_j r$ for some integer r. Thus

$$k - 1 = rD$$

or k must satisfy

$$k \equiv 1 \mod D.$$

Therefore, $k = 2, 3, \ldots, D$ produces kh_j/D noncongruent to h_j/D mod 1.

From Theorem 3 we know that if a congruence of (8) has any constant with numerator and denominator D relatively prime, then the congruence can generate $D - 1$ other congruences. These other congruences, if they appear in (8), are not needed and may be eliminated.

It should be noted that a congruence with the desired property may not necessarily generate every other congruence. More significantly, a congruence without a constant having a numerator and denominator relatively prime cannot generate a congruence with a constant having that property.

Instead of developing all the congruences of (8) first and then searching for a congruence having a constant with the required property, it is better, from a computational standpoint, to find an equation of (2) having a constant with relatively prime numerator and denominator. The congruence developed from the equation will then have a constant with the required property.

In hand-solved problems, the constants of (2) can be inspected easily to determine the equation having a constant with numerator and denominator relatively prime. But when an electronic computer is used to solve extremely large problems, it is not practical to evaluate each constant until one is found having the relatively prime property.

To offset this difficulty, we offer a method to facilitate the computations. In solving Eq. (1) as a linear program, we shall assume that the inverse of the basic variables is available. Following Section 5 of Chapter 2, the inverse is

$$B^{-1} = \begin{bmatrix} 1 & -\pi_1 & -\pi_2 & \cdots & -\pi_m \\ 0 & b_{11} & b_{12} & \cdots & b_{1m} \\ 0 & b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & b_{m1} & b_{m2} & \cdots & b_{mm} \end{bmatrix}.$$

Thus, $\bar{c}_j$, $\bar{z}_0$, $\bar{a}_{ij}$, and $\bar{b}_i$ in (2) are given by

$$\bar{c}_j = c_j - \sum_{i=1}^{m} \pi_i a_{ij},$$

$$\bar{z}_0 = \sum_{i=1}^{m} \pi_i b_i,$$

$$\bar{a}_{ij} = \sum_{k=1}^{m} b_{ik} a_{kj},$$

$$\bar{b}_i = \sum_{k=1}^{m} b_{ik} b_k.$$

Let us examine, for example, the relationship for $\bar{a}_{ij}$. Since each $\bar{a}_{ij} = h_{ij}/D$ and $b_{ik} = f_{ik}/D$, where the h_{ij} and f_{ik} are integers, we have

$$h_{ij} = \sum_{k=1}^{m} f_{ik} a_{kj} \tag{21}$$

and arrive at

Theorem 4

If h_{ij} and D are relatively prime for some i and j, then $\gcd(f_{i1}, f_{i2}, \ldots, f_{im}, D) = 1$.

Proof

If h_{ij} and D are relatively prime, then there exist integers x and y so that

$$h_{ij} x + Dy = 1.$$

Using (21), we obtain

$$x \sum_{k=1}^{m} f_{ik} a_{kj} + Dy = 1.$$

For the given j value we have $y_k = x a_{kj}$ and y_k is an integer. Thus,

$$\sum_{k=1}^{m} f_{ik} y_k + Dy = 1$$

and $\gcd(f_{i1}, f_{i2}, \ldots, f_{im}, D) = 1$.

The converse of Theorem 4 is not true. We shall use the theorem, however, in Algorithm 2 to develop the necessary congruences in (8). We must first solve (1) as a linear program by the inverse matrix method. The value D is calculated during the computation as the product of the pivot elements. When (2) is achieved and B^{-1} is available, we use the Euclidean algorithm of Chapter 6 to calculate the values

$$D_i = \gcd(f_{i1}, f_{i2}, \ldots, f_{im}, D), \qquad i = 1, 2, \ldots, m,$$

and determine index s from

$$D_s = \min D_i;$$

in case of ties we arbitrarily select the smallest index. We then form a single congruence, as in (6), from constraint s of (2). The requisite procedure is contained in

Algorithm 2

1. Same as Algorithm 1.
2. Same as Algorithm 1, except that, initially, only the congruence from constraint s makes up the β_j.
3. (a) In the newly listed section, define J as the set of indices j where $\beta_j = \beta_0$ and $0 \leq \alpha_0 - \alpha_j \leq M$. If J has no members, go to 4. Otherwise, go to 3(b).

   (b) Find index r from $\bar{c}_r = \min_{j \in J} \bar{c}_j$. If $\bar{c}_r \geq z^*$, go to 4. Otherwise, calculate $x = \alpha_0 - \alpha_r$. If the components of x are integers, take $z^* = \bar{c}_r$ and form $S(z^*, x_1^*, x_2^*, \ldots, x_t^*)$, where the x_j^* are the nonzero x_j values found below the section. The other x_j^* values equal zero. Increase x_r^* by one and go to 4. If any component of x is fractional, define I as the set of indices i where component i of x is fractional. Find index s from $D_s = \min_{i \in I} D_i$. Form a new congruence

$$\sum_{j=1}^{t} f_j x_j \equiv f_0 \mod 1$$

   from constraint s of (2). Each β_j for $j = 0, 1, \ldots, t$ is enlarged by the additional component f_j. Modify the list in step 2 accordingly. In addition, calculate a new β_j component in each section having unmarked columns by first calculating

$$f \equiv \sum_{j=1}^{t} f_j x_j' \mod 1,$$

   where the x_j' are the nonzero x_j values found below the section. The other x_j' values equal zero. The additional component of β_j for the unmarked j column of the section is then

$$f_j' \equiv f + f_j \mod 1.$$

   Go to 3(a).

4, 5, and 6. Same as Algorithm 1.

Example

Consider the problem on page 161. If it is solved as a linear program by the inverse matrix method, we obtain a value of $D = 6$ and inverse matrix

$$B^{-1} = \begin{bmatrix} 1 & -1 & -\frac{2}{3} \\ 0 & \frac{1}{2} & \frac{1}{6} \\ 0 & \frac{1}{2} & -\frac{1}{6} \end{bmatrix}.$$

From the second row of B^{-1} we calculate $D_1 = \gcd(3, 1, 6) = 1$; from the third row $D_2 = 1$. We develop the congruence

$$\tfrac{1}{3}x_3 + \tfrac{2}{3}x_4 \equiv \tfrac{1}{3} \mod 1.$$

The problem is then solved with this congruence only.

3 AN ACCELERATED DYNAMIC PROGRAMMING SOLUTION

In Eq. (2) it may occur that a positive value for one of the variables may not allow a feasible solution to be found. It may also be that some of the variables must have positive values to obtain a feasible solution. Similar conditions may exist when Algorithm 1 is applied in the search for the solution to (2). When these situations arise, we can accelerate the dynamic programming enumeration, just as we did in Chapter 4.

Consider (9) with $x_j \le m_j$ as listed in step 2 of Algorithm 1. Since $x_{t+i} \le m_{t+i}$, we have from (2) that

$$\sum_{j=1}^{t} \bar{a}_{ij} x_j \ge \bar{b}_i - m_{t+i}, \qquad i = 1, 2, \ldots, m.$$

Form an index set K with initial members $j = 1, 2, \ldots, t$. Then form

$$U_i = \sum_{k \in K} \bar{a}_{ik} y_k, \qquad i = 1, 2, \ldots, m; \tag{22}$$

y_k for $k \in K$ is a nonnegative integer that represents the subsequent increase in x_k. Thus, $y_k \le m_k'$, where $m_k' = m_k - x_k'$. Initially all $x_k' = 0$.

We require that $U_i \geq \bar{b}_i - m_{t+i}$. Hence, we may be able to determine from (22) whether a positive y_k value produces infeasibility or whether y_k must have a positive value to produce feasibility. Take

$$U_i^* = \sum_{k \in K_i^+} \bar{a}_{ik} m_k', \qquad i = 1, 2, \ldots, m, \tag{23}$$

where K_i^+ is the set of indices in K with $\bar{a}_{ik} > 0$. The development of U_i^* in (23) is similar to the development of the U_i^* in (9) of Chapter 4. Therefore, rules A1–A3 of Chapter 4 apply here with a_{ij} replaced by $\bar{a}_{ij}$ and b_i replaced by $\bar{b}_i - m_{t+i}$.

We can now augment the rules of Chapter 4 by calculating

$$U_i' = \sum_{k \in K_i^-} \bar{a}_{ik} m_k', \qquad i = 1, 2, \ldots, t, \tag{24}$$

where K_i^- is the set of indices in K with $\bar{a}_{ik} < 0$ and U_i' is the minimum value of U_i. If K_i^- has no members, then $U_i' = 0$. Since $x_{t+i} \geq 0$, we require that $U_i' \leq \bar{b}_i$. The following rules apply where K, c, the m_k', d_i, and x_i' are the result of A1–A3:

A4. If any $U_i' > \bar{b}_i - d_i$, then no y_k values can produce feasibility. No feasible solution to the problem can be found. Stop.

A5. If $U_i' + \bar{a}_{ij} > \bar{b}_i - d_i$ for $j \in K$, then $y_j = 0$ since $y_j \geq 1$ produces infeasibility. Mark column j and remove index j from K. Write a new form for the U_i in (22) and calculate the U_i^* in (23). Return to rule A1.

A6. (a) If j is the only index in K_i^- and $d_i > \bar{b}_i$, then $\bar{a}_{ij} y_j \leq \bar{b}_i - d_i$ requires that $y_j \geq \theta = \{(\bar{b}_i - d_i)/\bar{a}_{ij}\}$.

(b) If $U_i' - \bar{a}_{ij} m_j' > \bar{b}_i - d_i$ for $j \in K$, then $y_j \geq \theta$, where θ is the smallest integer with the property that $U_i' - \bar{a}_{ij} m_j' + \bar{a}_{ij} \theta \leq \bar{b}_i - d_i$.

(c) When θ is found in (a) or (b), increase x_j' by θ, decrease m_j' by θ, replace c by $c + \bar{c}_j \theta$ and d_i by $d_i + \bar{a}_{ij} \theta$. If $m_j' = 0$, remove index j from K. Write a new form for the U_i and calculate U_i^*. Return to rule A1.

Each time rule A4 is applied, the U_i' calculation from (24) must be done again. If as a result of adhering to rules A1–A6 all $x_j' = 0$, continue Algorithm 1 in step 3. If, on the other hand, some x_j' is positive,

we define vector α with d_i as component i and calculate

$$\beta \equiv \sum_{j=1}^{t} \beta_j x_j' \mod 1.$$

If $\beta = \beta_0$ and $0 \leq \alpha_0 - \alpha \leq M$, then program (2) is solved with $z = \bar{z}_0 + c$, $x_j = x_j'$ for $j = 1, 2, \ldots, t$ and

$$x = \alpha_0 - \sum_{j=1}^{t} \alpha_j x_j'$$

for the basic variables. Otherwise, add a new section of columns to the list. Each column is headed by an index $k \in K$. The column values are given by $c_k' = c + \bar{c}_k$, $\alpha_k' = \alpha + \alpha_k$ and $\beta_k' \equiv \beta + \beta_k \mod 1$. The $\bar{c}_k$, α_k, and β_k values are taken from the list in step 2. Underneath the section write the $x_j = x_j'$ values. Mark all unmarked columns on the list from step 2 and go to 3.

We are also able to modify the algorithm to ascertain whether a current stage of the enumeration will lead to a feasible α or whether some of the x_j require specific values in order to produce a feasible α.

When index r is found from $\bar{c}_r = \min \bar{c}_j$ in step 4 (with $c = \bar{c}_r$), we define the set of columns K as on page 93. Define the column values for α_r as $d_1, d_2, \ldots, d_m$. The possible values of z and components V_i of α that can be enumerated at this stage are given by

$$z = c + \sum_{k \in K} \bar{c}_k y_k,$$

$$V_i = d_i + U_i, \qquad i = 1, 2, \ldots, m,$$

where U_i is given by (22) for the present set K; $y_k \leq m_k'$, where $m_k' = m_k - x_k'$. The x_k' are the x_k values that appear below the section containing column r with $x_r' = x_r + 1$.

Since $x_{t+i} \leq m_{t+i}$, we require that $V_i \geq \bar{b}_i - m_{t+i}$. Consider

$$V_i^* = d_i + U_i^*, \qquad i = 1, 2, \ldots, m,$$

where U_i^* is given by (23) with K_i^+ as the set of indices in current K with $\bar{a}_{ik} > 0$. Then V_i^* is the maximum value V_i can attain when we perform the enumeration starting with the section containing column r. The development of V_i^* is similar to the development of the V_i^*

of Chapter 4. Rules B1–B3 of Chapter 4 apply here with a_{ij} replaced by $\bar{a}_{ij}$ and b_i replaced by $\bar{b}_i - m_{t+i}$.

We can add other rules following rule B3. Since $x_{t+i} \geq 0$, we require that $V_i \leq \bar{b}_i$. Consider

$$V_i' = d_i + U_i',$$

where U_i' is given by (24) and K_i^- is the set of indices in current K with $\bar{a}_{ik} < 0$. Then, V_i' is the minimum value that V_i can attain when we perform the enumeration starting with the section containing column r. The following rules then apply:

B4. If any $V_i' > \bar{b}_i$, then no y_k value can produce feasibility. Mark column r and continue the algorithm in step 4.

B5. If $V_i' + \bar{a}_{ij} > \bar{b}_i$ for $j \in K$, then $y_j = 0$ since $y_j \geq 1$ produces infeasibility. Remove index j from K and write new forms for the U_i in (22). Calculate the V_i^* and return to rule B1.

B6. (a) If j is the only index in K_i^- and $d_i > \bar{b}_i$, then $d_i + \bar{a}_{ij} y_j \leq \bar{b}_i$ requires that $y_j \geq \theta = \{(\bar{b}_i - d_i)/\bar{a}_{ij}\}$.

(b) If $V_i' - \bar{a}_{ij} m_j' > \bar{b}_i$ for $j \in K$, then $y_j \geq \theta$ where θ is the smallest integer with the property that $V_i' - \bar{a}_{ij} m_j' + \bar{a}_{ij}\theta \leq \bar{b}_i$.

(c) When θ is found in (a) or (b), increase x_j' by θ, decrease m_j' by θ, replace c by $c + \bar{c}_j\theta$ and d_i by $d_i + \bar{a}_{ij}\theta$. If $m_j' = 0$, remove index j from K. If $c + \bar{c}_k \geq z^*$, remove k from K. Write new forms for the U_i and calculate the V_i^*. Return to rule B1.

Each time that rule B4 is applied, the V_i' calculation must be done again. When the application of the rules is completed, we return to the algorithm in a modified step 5. The modifications consist of the following:

5. (a) Mark column r. Take value d_i as component i of α. Calculate

$$\beta \equiv \sum_{j=1}^{t} \beta_j x_j' \mod 1.$$

If $\beta = \beta_0$ and $0 \leq \alpha_0 - \alpha \leq M$, take $z^* = c$ and form $S(z^*, x_1', x_2', \ldots, x_n')$; go to 4. Otherwise, go to 5(b).

(b) If set K has no elements, go to 4. Otherwise, add a new section

of columns to the list. Each column is headed by an index $k \in K$. The column values are given by $c_k' = c + \bar{c}_k$, $\alpha_k' = \alpha + \alpha_k$ and $\beta_k' \equiv \beta + \beta_k \bmod 1$. The $\bar{c}_k$, α_k, β_k values are taken from the list in step 2. Underneath the section write the $x_j = x_j'$ values. Go to 3.

Example

Find nonnegative integers $x_j \leq 1$ that minimize z when

$$\begin{aligned} -z \quad &+ \tfrac{1}{3}x_3 \quad + \tfrac{15}{9}x_5 + x_6 = -\tfrac{1}{3}, \\ x_1 \quad &+ \tfrac{1}{9}x_3 + \tfrac{5}{3}x_4 + \tfrac{2}{9}x_5 - \tfrac{1}{3}x_6 = \tfrac{8}{9}, \\ x_2 &+ \tfrac{11}{9}x_3 - \tfrac{2}{3}x_4 - \tfrac{5}{9}x_5 + \tfrac{1}{3}x_6 = \tfrac{7}{9}. \end{aligned}$$

The equations are in an optimal canonical format. The continuous solution is $z = \frac{1}{3}$, $x_1 = \frac{8}{9}$, $x_2 = \frac{2}{9}$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$. We develop the congruences

$$\begin{aligned} \tfrac{1}{9}x_3 + \tfrac{2}{3}x_4 + \tfrac{2}{9}x_5 + \tfrac{2}{3}x_6 &\equiv \tfrac{8}{9} \mod 1, \\ \tfrac{2}{9}x_3 + \tfrac{1}{3}x_4 + \tfrac{4}{9}x_5 + \tfrac{1}{3}x_6 &\equiv \tfrac{7}{9} \mod 1. \end{aligned}$$

1. $z^* = \infty$.

2. The problem is listed in Tableau E1: $\bar{z}_0 = \frac{1}{3}$, $\alpha_0 = (\frac{8}{9}, \frac{7}{9})$, $\beta_0 = (\frac{8}{9}, \frac{7}{9})$ and all $m_j = 1$.

3*	4*	5	6	6	6
$\frac{1}{3}$	0	$\frac{15}{9}$	1	3	$\frac{24}{9}$
$\frac{1}{9}$	$\frac{5}{3}$	$\frac{2}{9}$	$-\frac{1}{3}$	0	$-\frac{1}{9}$
$\frac{11}{9}$	$-\frac{2}{3}$	$-\frac{5}{9}$	$\frac{1}{3}$	1	$-\frac{2}{9}$
$\frac{1}{9}$	$\frac{2}{3}$	$\frac{2}{9}$	$\frac{2}{3}$	0	$\frac{8}{9}$
$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	$\frac{1}{3}$	0	$\frac{7}{9}$
				$x_3 = 1$ $x_5 = 1$	$x_5 = 1$
		E1		E2	E3

Tableaus

3. J has no members. We obtain $U_1' = -\frac{1}{3}$, $U_2' = -\frac{11}{9}$ with $K = (3, 4, 5, 6)$. For rule A5, $U_1' + \bar{a}_{14} = \frac{4}{3} > \frac{8}{9}$; we mark column 4 and remove 4 from K. No other rules apply.

4. $r = 3$ in Tableau E1, $\bar{c}_r = \frac{1}{3}$; $d_1 = \frac{1}{9}$, $d_2 = \frac{11}{9}$, and $V_1' = -\frac{2}{9}$, $V_2' = \frac{6}{9}$ with $K = (5, 6)$. Since $V_2' - \bar{a}_{25} = \frac{11}{9} > \frac{7}{9}$, then $y_5 = 1$ and $K = (6)$. No other rules apply. We mark column 3 and form Tableau E2.

3. J has no members.

4. $r = 5$ in Tableau E1, $\bar{c}_r = \frac{15}{9}$. No rules apply. We mark column 5 and form Tableau E3.

3. $J = (6)$. The optimal solution becomes apparent from Tableau E3; it is $z = 3$, $x_5 = 1$, $x_6 = 1$, $x_3 = 0$, $x_4 = 0$, $x_1 = 1$, $x_2 = 1$.

Problems

1. Solve the problem on page 161 by the accelerated enumeration procedure.
2. Use the inverse matrix method to develop a single congruence in:

$$\begin{aligned} 16x_1 + 15x_2 + 12x_3 &= z, \\ 8x_1 + 5x_2 + 2x_3 - x_4 &= 4, \\ 2x_1 + 3x_2 + 3x_3 - x_4 &= 3. \end{aligned}$$

   Solve the problem.
3. Use the accelerated enumeration method to find nonnegative $x_j \leq 2$ and min z for

$$\begin{aligned} -z + \tfrac{1}{9}x_3 + \tfrac{11}{9}x_4 + \tfrac{2}{3}x_5 &= -\tfrac{2}{3}, \\ x_1 - \tfrac{7}{9}x_3 + \tfrac{4}{9}x_4 + \tfrac{1}{3}x_5 &= \tfrac{2}{3}, \\ x_2 + \tfrac{11}{9}x_3 - \tfrac{5}{9}x_4 + \tfrac{1}{3}x_5 &= \tfrac{2}{3}. \end{aligned}$$

4. Discuss means of saving computer storage space by calculating α_j and β_j values only when needed. Show how a backtracking operation can be used to obtain the x_j values.

References

1. Gomory, R. E., On the relation between integer and non-integer solutions to linear programs, *Proc. Nat. Acad. Sci.* **53**, 260 (1965).
2. Greenberg, H., A dynamic programming solution to integer linear programs, *J. Math. Anal. Appl.* **26** (2), 454 (1969).

8 BRANCH AND BOUND PROCEDURES

The branch and bound procedure for the solution to integer programs, as originated by Land and Doig [3], works well on problems having few variables. For problems with many variables, however, it requires extremely large computer storage capacity. We shall present an extension of the method into a branch search scheme that overcomes these large storage requirements. We also provide a method for solving the mixed integer problem.

1 A BRANCH AND BOUND METHOD

We wish to solve the integer programming problem: Find integers $x_j \geq 0$ for $j = 1, 2, \ldots, n$ that minimize z when

$$\begin{aligned} &\sum_{j=1}^{n} c_j x_j = z, \\ &\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, m, \\ &x_j \leq m_j, \qquad j = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

where the c_j, a_{ij}, b_i, and m_j are given integers. Any or all of the m_j may be infinite.

We are able to solve (1) by enumerating solutions in a tree search procedure. Consider (1) as a linear program by relaxing the integer constraints on the x_j. The resulting continuous problem may be solved by the simplex method, which converts the equations of (1) to an optimality format and equivalent problem: Find nonnegative integers $x_j \leq m_j$ that minimize z when

$$
\begin{aligned}
&-z + \sum_{j=m+1}^{n} \bar{c}_j x_j = -\bar{z}_0, \\
&x_i + \sum_{j=m+1}^{n} \bar{a}_{ij} x_j = \bar{b}_i, \qquad i = 1, 2, \ldots, m,
\end{aligned} \tag{2}
$$

where the first m variables and the last $n - m$ variables have been arbitrarily selected as the basic and nonbasic variables, respectively. In addition, $\bar{c}_j \geq 0$ for variables at their lower bound zero and $\bar{c}_j \leq 0$ for variables at their upper bound m_j. The values of the objective function and basic variables are given by

$$
\begin{aligned}
z_0' &= \bar{z}_0 + \sum_{j \in U} \bar{c}_j m_j, \\
b_i' &= \bar{b}_i - \sum_{j \in U} \bar{a}_{ij} m_j;
\end{aligned}
$$

the objective value is z_0', the ith basic variable has nonnegative value $b_i' \leq m_i$, and U is the index set of nonbasic variables at their upper bounds.

If the continuous solution has all b_i' as integer, program (1) is solved. If any of the b_i' are fractional, we start the tree enumeration. We consider the zero node of a tree as corresponding to the fractional solution with objective value z_0'. Integer x_i must satisfy $x_i \leq [b_i']$ or $x_i \geq [b_i'] + 1$. We branch to level one of the tree by adding either the upper or the lower bound to the constraints of (2); then we find a new continuous minimum for z. Whichever bound we choose, we pursue the single branch until it becomes necessary to backtrack. Then we follow the other branch. The minimal integer solution occurs for one of the branches. This type of single-branch procedure originally appears in Dakin [1].

Let us assume that after some minimization we are at level r of the tree. We have a canonical form like (2) with the additional constraints $0 \le \bar{p}_j \le x_j \le \bar{m}_j$; $\bar{p}_j$ and $\bar{m}_j$ are lower and upper bounds, respectively. Initially $\bar{p}_j = 0$ and $\bar{m}_j = m_j$. The values of the objective function and basic variables are given by

$$
\begin{aligned}
z_0' &= \bar{z}_0 + \sum_{j \in U} \bar{c}_j \bar{m}_j + \sum_{j \in L} \bar{c}_j \bar{p}_j, \\
b_i' &= \bar{b}_i - \sum_{j \in U} \bar{a}_{ij} \bar{m}_j - \sum_{j \in L} \bar{a}_{ij} \bar{p}_j;
\end{aligned} \tag{3}
$$

the objective value is z_0', the ith basic variable has value b_i' where $\bar{p}_i \le b_i' \le \bar{m}_i$, and L is the index set of nonbasic variables at their lower bounds. We consider the node at level r as corresponding to the current fractional solution.

The optimality conditions for (2) with the current bounds are given in

Theorem 1

Any values of x_j for $j = 1, 2, \ldots, n$ satisfying the constraint equations in (2) and $\bar{p}_j \le x_j \le \bar{m}_j$ are a minimal solution for z if $\bar{c}_j \ge 0$ for variables at their lower bound $\bar{p}_j$ and $\bar{c}_j \le 0$ for variables at their upper bound $\bar{m}_j$.

Proof

We have optimality by the conditions of the theorem, since any increase in the variables at their lower bound or any decrease in the variables at their upper bound can only increase z.

Suppose some of the b_i' are fractional. We select some fractional b_i' and *branch* to level $r + 1$ of the tree. We accomplish this by adding either $x_i \le \bar{m}_i = [b_i']$ or $x_i \ge \bar{p}_i = [b_i'] + 1$ to the constraints of (2) and then reminimizing. Whichever bound we choose, we pursue the single branch until it becomes necessary to follow the other branch. We must decide which fractional-valued basic variable to use and, at the same time, whether to use an upper or lower bound for the variable.

The choice of either an upper or a lower bound will cause the objective value to be greater than or equal to z_0' because another constraint has been added to the problem. Thus, z_0' is a lower *bound* to all objective values for nodes that lead from the current one.

We can calculate the change in the objective function for the newly constrained problem. Let us suppose that x_i, a basic variable, has a fractional value after the optimal canonical form (2) has been achieved. We determine

$$D_i = \min_j \frac{\bar{c}_j}{\bar{a}_{ij}} \geq 0, \tag{4}$$

where D_i is taken as infinite if no proper $\bar{a}_{ij}$ exists to produce a finite value. We also calculate

$$I_i = \min_j \frac{-\bar{c}_j}{\bar{a}_{ij}} \geq 0, \tag{5}$$

where I_i is taken as infinite if no proper $\bar{a}_{ij}$ exists to produce a finite value.[1] We can now determine the effect on the objective function of either a decrease or an increase in the value of x_i. Equations (4) and (5) each define a variable with index j that can be eliminated from the objective function of (2) and produce an optimality format for z. Thus, we are able to calculate the smallest change in the objective value that results when x_i is decreased or increased.[2] If x_i is to be decreased, we use (4) with the z and x_i equations of (2) to form

$$z = \bar{z}_0 + D_i \bar{b}_i - D_i x_i + \sum_{j=m+1}^{n} (\bar{c}_j - D_i \bar{a}_{ij}) x_j; \tag{6}$$

if we take $x_i \leq \overline{m}_i$, we must have $z \geq U_i$, where

$$U_i = z_0' + D_i(b_i' - \overline{m}_i).$$

[1] The index i on D_i or I_i, will more generally be s_i, the index of the basic variable in equation i.

[2] This is the cost ranging procedure of linear programming. (See Driebeck [2] for its original application in integer programming.)

If x_i is to be increased, we use (5) to form

$$z = \bar{z}_0 - I_i \bar{b}_i + I_i x_i + \sum_{j=m+1}^{n} (\bar{c}_j + I_i \bar{a}_{ij}) x_j; \tag{7}$$

if we take $x_i \geq \bar{p}_i$, then we have $z \geq V_i$, where

$$V_i = z_0' + I_i(\bar{p}_i - b_i').$$

Note that if no $\bar{a}_{ij}$ with the proper sign exists to produce a finite value of D_i, then it is not possible to reduce the value of x_i; we take $x_i \geq \bar{p}_i$. Similarly, if no proper $\bar{a}_{ij}$ exists to produce a finite value of I_i, it is not possible to increase the value of x_i and we take $x_i \leq \bar{m}_i$. If $U_i + 1 > z^*$, we take $x_i \geq \bar{p}_i$, while if $V_i + 1 > z^*$, we take $x_i \leq \bar{m}_i$, where z^* is the objective value of a feasible integer solution.

At this point we can decide which basic variable to pick for branching. If U_i is large for basic variable x_i, we may not need the $x_i \leq \bar{m}_i$ branch if we use $x_i \geq \bar{p}_i$. There is a chance of obtaining a feasible integer solution with objective z^* with $U_i + 1 > z^*$. (This type of strategy was developed by Little, and co-workers [4] for solving the traveling salesman problem.) Similar possibilities exist if V_i is large. The branching variable x_k is obtained by selecting the index k from

$$W_k = \max(U_i, V_i) \tag{8}$$

for indices i where x_i is fractional. Therefore, if the maximum occurs for U_k, we use $x_k \geq \bar{p}_k = [b_k'] + 1$; if it occurs for V_k, we use $x_k \leq \bar{m}_k = [b_k']$.

Suppose now that a feasible integer solution to (1) is available with objective value z^*. This solution is an upper bound to the optimal one. There is no need to branch from node level r to $r + 1$ if:

1. $z_0' + 1 > z^*$,
2. all the b_i' are feasible integers and $z_0' < z^*$ (we have found a feasible integer solution with new z^*),
3. no feasible solution exists at the node.

In any of these three cases, we backtrack to a node at level $t < r$ that has only one branch and has $W_k \leq z^* - 1$. We eliminate all $\bar{m}_j$ and $\bar{p}_j$ bounds along the path from the r level to the t level node. We

then branch to level $t + 1$ by imposing $x_k \leq \bar{m}_k$ if $x_k \geq \bar{p}_k$ initially, and vice versa.

We must find the new canonical form and variable value for level $r + 1$. In the canonical form (2) at level r, the equation for basic variable x_k is

$$x_k + \sum_{j=m+1}^{n} \bar{a}_{kj} x_j = \bar{b}_k. \tag{9}$$

The current value of x_k is b_k'. If we use $x_k \leq \bar{m}_k$, then $b_k' > \bar{m}_k$. We replace (9) in (2) by

$$x_k + \sum_{j=m+1}^{n} \bar{a}_{kj} x_j + x_{n+k} = \bar{b}_k, \tag{10}$$

where $x_{n+k} \geq 0$ is an artificial variable. The variable x_k is given the value $\bar{m}_k$ and x_{n+k} has value $x_{n+k}' = b_k' - \bar{m}_k$. The simplex method is initiated again by adding the w-objective function

$$-w - x_k - \sum_{j=m+1}^{n} \bar{a}_{kj} x_j = -\bar{b}_k \tag{11}$$

to (2). The w-objective value is $w_0' = b_k' - \bar{m}_k$. If we use $x_k \geq \bar{p}_k$, then $b_k' < \bar{p}_k$. We replace (9) in (2) by

$$-x_k - \sum_{j=m+1}^{n} \bar{a}_{kj} x_j + x_{n+k} = -\bar{b}_k, \tag{12}$$

where $x_{n+k} \geq 0$ is an artificial variable. The variable x_k is given the value $\bar{p}_k$ and x_{n+k} has value $x_{n+k}' = \bar{p}_k - b_k'$. The simplex method is initiated again by adding the w-objective function

$$-w + x_k + \sum_{j=m+1}^{n} \bar{a}_{kj} x_j = \bar{b}_k \tag{13}$$

to (2). The w-objective value is $w_0' = \bar{p}_k - b_k'$. At the end of the minimization phase we have the canonical form for level $r + 1$.

To convert the canonical form at level r to a canonical form at level $t + 1$, we first remove the bounds along the path from the r level to the t level node. The bounds for the t level are given by the branchings from the zero level node to the t level node. Let us impose these bounds as the $\bar{p}_j$ and $\bar{m}_j$. All current variable values at level r satisfy $\bar{p}_j \leq x_j \leq \bar{m}_j$; the

current canonical form is feasible. To obtain the canonical form for level t, we simply reminimize as a Phase II simplex problem. When the canonical form is achieved, we must branch with the x_k variable. If x_k is not a basic variable, due to a nonunique solution, we make x_k basic (the corresponding $\bar{c}_k$ is zero). We then achieve the form given in (9). We impose the appropriate bound to obtain (10) and (11) or (12) and (13), and reminimize to get the canonical form for level $t+1$.

The two phases of the simplex method for obtaining a new continuous solution follow from the methods of Chapter 2. In any case, assume that the w equation from (11) or (12) is of the form

$$-w + \sum_{j=1}^{n} \bar{d}_j x_j = -\bar{w}_0 .$$

The index s is obtained from

$$d_s' = \min \begin{cases} \bar{d}_j, & \text{for } x_j \text{ not at its upper bound,} \\ -\bar{d}_j, & \text{for } x_j \text{ not at its lower bound} \end{cases} \tag{14}$$

in Phase I.

Let us assume that $\bar{d}_s$ is negative and that nonbasic x_s has value x_s'. The effect of increasing x_s by θ is seen by changing w_0', z_0', x_{n+k}', and the b_i' for $i \neq k$ to the new values

$$\begin{aligned} w_0'' &= w_0' + \bar{d}_s \theta, \\ z_0'' &= z_0' + \bar{c}_s \theta, \\ x_{n+k}'' &= x_{n+k}' - \bar{a}_{ks} \theta, \\ b_i'' &= b_i' - \bar{a}_{is} \theta. \end{aligned} \tag{15}$$

The greatest x_s can be increased and still maintain feasibility is

$$\theta^* = \min \begin{cases} \bar{m}_s - x_s', & \\ \dfrac{\bar{m}_i - b_i'}{-\bar{a}_{is}}, & \bar{a}_{is} < 0, \quad i \neq k, \\ \dfrac{b_i' - \bar{p}_i}{\bar{a}_{is}}, & \bar{a}_{is} > 0, \quad i \neq k, \\ \dfrac{x_{n+k}'}{\bar{a}_{ks}}, & \bar{a}_{ks} > 0. \end{cases} \tag{16}$$

Suppose that $\bar{d}_s$ is positive and that nonbasic x_s has value x_s'. The effect of decreasing x_s by γ is seen by changing w_0', z_0', x'_{n+k}, and the b_i' for $i \neq k$ to the new values

$$
\begin{aligned}
w_0'' &= w_0' - \bar{d}_s\gamma, \\
z_0'' &= z_0' - \bar{c}_s\gamma, \\
x''_{n+k} &= x'_{n+k} + \bar{a}_{ks}\gamma, \\
b_i'' &= b_i' + \bar{a}_{is}\gamma.
\end{aligned} \tag{17}
$$

The greatest x_s can be reduced and still maintain feasibility is

$$
\gamma^* = \min \begin{cases} x_s' - \bar{p}_s, & \\ \dfrac{b_i' - \bar{p}_i}{-\bar{a}_{is}}, & \bar{a}_{is} < 0, \quad i \neq k, \\ \dfrac{\bar{m}_i - b_i'}{\bar{a}_{is}}, & \bar{a}_{is} > 0, \quad i \neq k, \\ \dfrac{x'_{n+k}}{-\bar{a}_{ks}}, & \bar{a}_{ks} < 0. \end{cases} \tag{18}
$$

After selecting the nonbasic variable x_s by the choice rule of (14), we have

Case I.

Here $x_s = x_s' < \bar{m}_s$ and $\bar{d}_s < 0$. In the next iteration $x_s = x_s' + \theta^*$. The variable values are given in (15) with $\theta = \theta^*$; all other nonbasic variables retain their same values. If $\theta^* = \bar{m}_s - x_s'$, then x_s assumes its upper bound value and no pivot operation is performed. Otherwise, x_s is made basic; we use choice rule (16) to select some basic variable x_r to become nonbasic. The usual pivot operation is then performed with $\bar{a}_{rs}$ as pivot element.

Case II.

Here $x_s = x_s' > \bar{p}_s$ and $\bar{d}_s > 0$. In the next iteration $x_s = x_s' - \gamma^*$. The variable values are given in (17) with $\gamma = \gamma^*$; all other nonbasic variables retain their same values. If $\gamma^* = x_s' - \bar{p}_s$, then x_s assumes its

lower bound value and no pivot operation is performed. Otherwise, x_s is made basic; we use choice rule (18) to select some variable x_r to become nonbasic. Pivoting then occurs.

If $w = w_0'' > 0$, we repeat the procedure starting with choice rule (14). If w cannot be reduced to zero, the branching is infeasible. When $w = 0$, we initiate Phase II of the simplex algorithm by finding index s from

$$c_s' = \min \begin{cases} \bar{c}_j, & \text{for } x_j \text{ not at its upper bound,} \\ -\bar{c}_j, & \text{for } x_j \text{ not at its lower bound.} \end{cases} \tag{19}$$

The analysis follows as in Eq. (15)–(18) where the w and x_{n+k} rows do not appear. Case I is the same for $x_s = x_s' < \bar{m}_s$ and $\bar{c}_s < 0$. Case II is the same for $x_s = x_s' > \bar{p}_s$ and $\bar{c}_s > 0$. We have the newest continuous solution and canonical form when $c_s' \geq 0$ in (19).

The branch search procedure is summarized in

Algorithm 1

1. Define solution vector $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$ where x_j^* for $j = 1, 2, \ldots, n$ is a feasible integer solution to (1) with objective value z^*. If no feasible solution is apparent, take $z^* = \infty$. Let r refer to the rth branching variable, where $I(r)$ is the index of the variable, $N(r)$ is the number of branchings [initially $N(r) = 0$], $B(r)$ is the bound for the variable (this bound will usually be a temporary one), and $G(r) = 0$ indicates that $x_k \leq B(r)$ for $k = I(r)$; $G(r) = 1$ indicates that $x_k \geq B(r)$ for $k = I(r)$. Initially all $\bar{p}_j = p_j = 0$ and $\bar{m}_j = m_j$.

Solve (1) as a linear program. If the solution is all integer, the problem is solved. If not, set $r = 0$ and go to 2, maintaining the canonical form and variable values to the solution to (1).

2. Set $r = r + 1$. Select index k and determine W_k from (8). If $W_k = U_k$ in the calculation, set $B(r) = [b_k'] + 1$, $G(r) = 1$, and $\bar{p}_k = B(r)$. Otherwise, $W_k = V_k$; set $B(r) = [b_k']$, $G(r) = 0$, and $\bar{m}_k = B(r)$. In any case, set $I(r) = k$, $N(r) = 1$, $W(r) = W_k$, and go to 3.

3. Solve the linear program from the current canonical form with the new bounds on the variables. We do one of the following:

   (a) If the solution produces an objective value z with $z + 1 > z^*$, go to 4.

(b) If the current problem has an infeasible solution, go to 4.

(c) If the solution produces an objective value z with $z < z^*$ and the solution $x_1^*, x_2^*, \ldots, x_n^*$ is all integer, redefine $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$ as a new feasible integer solution where $z^* = z$. Go to 4.

(d) If the solution produces an objective value z with $z + 1 \leq z^*$ and the solution is fractional, go to 2.

4. *Backtrack*. Set $\bar{m}_j = m_j$ and $\bar{p}_j = p_j$ for $j = 1, 2, \ldots, n$. Go to 4(a).

(a) If $W(r) > z^* - 1$, go to 4(f). Otherwise, go to 4(b).

(b) If $N(r) = 1$ and $r = 1$, go to 4(e). If $N(r) = 1$ and $r > 1$, let $t = 1$ and go to 4(c). Otherwise, $N(r) = 2$; go to 4(f).

(c) If $G(t) = 0$, let $\bar{m}_k = B(t)$ for $k = I(t)$. Otherwise, $G(t) = 1$; let $\bar{p}_k = B(t)$ for $k = I(t)$. Let $t = t + 1$ and go to 4(d).

(d) If $t < r$, go to 4(c). Otherwise, $t = r$. Solve the linear program from the current canonical form with the new bounds on the variables. Go to 4(e).

(e) If $G(r) = 0$, set $B(r) = B(r) + 1$ and $N(r) = 2$. Thus, $\bar{p}_k = B(r)$ for $k = I(r)$. Go to 3. Otherwise, $G(r) = 1$; set $B(r) = B(r) - 1$ and $N(r) = 2$. Thus, $\bar{m}_k = B(r)$ for $k = I(r)$ Go to 3.

(f) If $r = 1$, the feasible solution $S(z^*, x_1^*, x_2^*, \ldots, x_n^*)$ is optimal. Stop. Otherwise, set $N(r) = 0$, $r = r - 1$ and go to 4(a).

Example

Find integers $x_j \geq 0$ that minimize z when

$$\begin{aligned} 3x_1 + 4x_2 &= z, \\ 2x_1 + x_2 - x_3 &= 1, \\ x_1 + 3x_2 - x_4 &= 4. \end{aligned}$$

1. We solve the problem as a linear program to obtain

$$\begin{aligned} -z + \tfrac{5}{3}x_1 + \tfrac{4}{3}x_4 &= -\tfrac{16}{3}, \\ \tfrac{1}{3}x_1 + x_2 - \tfrac{1}{3}x_4 &= \tfrac{4}{3}, \\ -\tfrac{5}{3}x_1 + x_3 - \tfrac{1}{3}x_4 &= \tfrac{1}{3}. \end{aligned}$$

The solution is $z = \frac{16}{3}$, $x_1 = 0$, $x_2 = \frac{4}{3}$, $x_3 = \frac{1}{3}$, $x_4 = 0$.

2. $r = 1$. $D_2 = 5$, $D_3 = \infty$. We set $x_3 \geq 1$ and $W(1) = \infty$.

3. The new canonical form is

$$\begin{aligned} -z \quad &+ x_3 + x_4 = -5, \\ x_2 + \tfrac{1}{5}x_3 - \tfrac{2}{5}x_4 &= \tfrac{7}{5}, \\ x_1 \quad - \tfrac{3}{5}x_3 + \tfrac{1}{5}x_4 &= -\tfrac{1}{5}, \end{aligned}$$

where we have the solution $z = 6$, $x_1 = \frac{2}{5}$, $x_2 = \frac{6}{5}$, $x_3 = 1$, $x_4 = 0$.

2. $r = 2$. $D_2 = 5$, $D_1 = 5$, $I_2 = \frac{5}{2}$, $I_1 = \frac{5}{3}$; $U_2 = 7$, $U_1 = 8$, $V_2 = 8$, $V_1 = 7$. We take $W_2 = V_2 = 8$. We set $x_2 \leq 1$ and $W(2) = 8$.

3. The new canonical form is

$$\begin{aligned} -z \quad - 5x_2 \quad &+ 3x_4 = -12, \\ 5x_2 + x_3 - 2x_4 &= 7, \\ x_1 + 3x_2 \quad - x_4 &= 4, \end{aligned}$$

where we have solution $z^* = 7$, $x_1^* = 1$, $x_2^* = 1$, $x_3^* = 2$, $x_4^* = 0$.

4. *Backtrack*. $W(2) > 6$, $W(1) > 6$. The current feasible solution is minimal.

2 TIGHTENING THE BOUNDS

In Algorithm 1 the upper bounds m_j and the lower bounds $p_j = 0$ remain fixed. The $\bar{m}_j$ and $\bar{p}_j$ are regarded as temporary bounds. Whenever a continuous linear programming solution is found in steps 1, 3, or 4, further bounds may be placed on the variables. This tightening of the bounds will tend to improve the efficiency of the algorithm.

The improvement is possible when a feasible integer solution is available with objective value z^*. Thus, if we are interested in obtaining a smaller objective value, we need consider only values of the objective function given by

$$z = \bar{z}_0 + \sum_{j=m+1}^{n} \bar{c}_j x_j \leq z^* - 1.$$

We can obtain bounds on the nonbasic variables x_s where $\bar{c}_s \neq 0$.

If $\bar{c}_s < 0$, we achieve $x_s \geq \{v_s\}$ where

$$v_s = m_s + \frac{z^* - 1 - z_0'}{\bar{c}_s}.$$

If $\bar{c}_s > 0$, we have $x_s \leq [u_s]$ where

$$u_s = p_s + \frac{z^* - 1 - z_0'}{\bar{c}_s}.$$

Thus, the lower bound on the variable x_s can be increased to $\{v_s\}$ if the present bound is smaller (when $\bar{c}_s < 0$). Also the upper bound on x_s can be reduced to $[u_s]$ if the present bound is larger (when $\bar{c}_s > 0$). Similar results hold for the bounds $\bar{m}_j$ and $\bar{p}_j$ during the branching process, if we define $\bar{u}_s$ and $\bar{v}_s$ by replacing m_s and p_s by $\bar{m}_s$ and $\bar{p}_s$, respectively.

In addition, we can use (6) and (7) to tighten the bounds on the x_i by requiring that $z \leq z^* - 1$. Using (6) we obtain

$$z \geq z_0' + D_i b_i' - D_i x_i;$$

thus when $D_i > 0$ we have $x_i \geq \{v_i\}$ where

$$v_i = b_i' - \frac{z^* - 1 - z_0'}{D_i}.$$

Using (7), we obtain

$$z \geq z_0' - I_i b_i' + I_i x_i;$$

thus, when $I_i > 0$, we have $x_i \leq [u_i]$ where

$$u_i = b_i' + \frac{z^* - 1 - z_0'}{I_i}.$$

We replace bounds $\bar{m}_i$ and $\bar{p}_i$ by these new bounds if the latter are tighter.

3 THE MIXED INTEGER PROBLEM

The algorithm in Section 1 is modified slightly to handle the mixed integer case where only some of the x_j variables are restricted to be integers. The remaining variables can be integers or fractions and are

never picked to be the branching variables. The enumeration is performed for the integer-restricted variables only; eventually a feasible solution results for both the integer and fractional variables.

If the objective value is also constrained to be an integer, it appears simplest to replace the objective function in (1) by

$$x_{n+1} - x_{n+2} = z$$

and include the additional constraint

$$\sum_{j=1}^{n} c_j x_j - x_{n+1} + x_{n+2} = 0,$$

where x_n and x_{n+1} are nonnegative integers.

If the objective value z can be fractional, we replace steps 3(a) and 4(a) of the algorithm by

3. (a) If the solution produces an objective value z with $z \geq z^*$, go to 4.
4. (a) If $W(r) \geq z^*$, go to 4(f). Otherwise, go to 4(b).

Example

Find $x_1 \geq 0$, integer $x_2 \geq 0$, $x_3 \geq 0$, and integer $x_4 \geq 0$ that minimize z when

$$\begin{aligned} 5x_1 - x_2 - x_3 + 2x_4 &= z, \\ 9x_1 - 2x_2 - 5x_3 + 4x_4 &= 2, \\ x_2 + x_3 + x_4 &= 4. \end{aligned}$$

1. We solve the problem as a linear program to obtain

$$\begin{aligned} -z + \tfrac{1}{2}x_1 + \tfrac{3}{2}x_3 &= -\tfrac{1}{2}, \\ -\tfrac{3}{2}x_1 + x_2 + \tfrac{3}{2}x_3 &= \tfrac{5}{2}, \\ \tfrac{3}{2}x_1 - \tfrac{1}{2}x_3 + x_4 &= \tfrac{3}{2}. \end{aligned}$$

The solution is $z = \frac{1}{2}$, $x_1 = 0$, $x_2 = \frac{5}{2}$, $x_3 = 0$, $x_4 = \frac{3}{2}$.

2. $r = 1$. $D_2 = 1$, $D_4 = \frac{1}{3}$, $I_2 = \frac{1}{3}$, $I_4 = 3$; $U_2 = 1$, $U_4 = \frac{2}{3}$, $V_2 = \frac{2}{3}$, $V_4 = 2$. $W_4 = V_4 = 2$. We set $x_4 \leq 1$ and $W(1) = 2$.

3. The new canonical form is

$$\begin{aligned} -z \quad &+ \tfrac{5}{3}x_3 - \tfrac{1}{3}x_4 = -1, \\ x_2 + \; &x_3 + \; x_4 = \; 4, \\ x_1 \quad &- \tfrac{1}{3}x_3 + \tfrac{2}{3}x_4 = \; 1, \end{aligned}$$

where we have solution $z^* = \frac{2}{3}$, $x_1^* = \frac{1}{3}$, $x_2^* = 3$, $x_3^* = 0$, $x_4^* = 1$.

4. *Backtrack.* $W(1) \geq \frac{2}{3}$. The current feasible solution is minimal.

Problems

1. Use the branch and bound technique to solve the integer problem:

$$\begin{aligned} -z \quad &+ \tfrac{5}{3}x_3 + \tfrac{10}{3}x_4 = -\tfrac{16}{3}, \\ x_1 + \tfrac{10}{3}x_3 - \; &\tfrac{4}{3}x_4 = -\tfrac{7}{3}, \\ x_2 - \; \tfrac{1}{3}x_3 - \; &\tfrac{5}{3}x_4 = \; \tfrac{5}{3}. \end{aligned}$$

2. Find the minimum z for the integer problem:

$$\begin{aligned} -z \quad &- \tfrac{1}{3}x_1 + \tfrac{2}{3}x_4 = -\tfrac{8}{3}, \\ x_3 + \tfrac{4}{3}x_1 - \; &\tfrac{2}{3}x_4 = \; \tfrac{5}{3}, \\ x_2 - \tfrac{1}{3}x_1 + \; &\tfrac{2}{3}x_4 = \; \tfrac{1}{3}; \end{aligned}$$

$$0 \leq x_j \leq 1 \quad \text{for} \quad j = 1, 2, 3, 4.$$

3. Solve the mixed integer problem on page 189 where z is constrained to be an integer.
4. Minimize z for integers $x_j \geq 0$ in

$$\begin{aligned} -z \quad &+ \tfrac{4}{3}x_1 + \tfrac{16}{3}x_3 = -3, \\ x_2 + \tfrac{1}{6}x_1 + \; &\tfrac{1}{6}x_3 = \; \tfrac{1}{2}, \\ x_4 + \tfrac{1}{2}x_1 + \; &\tfrac{1}{2}x_3 = \; \tfrac{1}{2}. \end{aligned}$$

References

1. Dakin, R. J., A tree-search algorithm for mixed integer programming problems, *Comput. J.*, **8** (3), 250 (1965).
2. Driebeck, N. J., An algorithm for the solution of mixed integer programming problems, *Management Sci.* **12**, 576 (1966).
3. Land, A. H., and Doig, A. G., An automatic method of solving discrete programming problems, *Econometrica* **28**, 497 (1960).
4. Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C., An algorithm for the traveling salesman problem, *Operations Res.* **11**, 972 (1963).

AUTHOR INDEX

Numbers in parentheses are reference numbers and indicate that an author's work is referred to although his name is not cited in the text. Numbers in italics show the page on which the complete reference is listed.

SUBJECT INDEX