# Integer Programming

# 9

The linear-programming models that have been discussed thus far all have been *continuous*, in the sense that decision variables are allowed to be fractional. Often this is a realistic assumption. For instance, we might easily produce $102\frac{3}{4}$ gallons of a divisible good such as wine. It also might be reasonable to accept a solution giving an hourly production of automobiles at $58\frac{1}{2}$ if the model were based upon average hourly production, and the production had the interpretation of *production rates*.

At other times, however, fractional solutions are not realistic, and we must consider the optimization problem:

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

subject to:

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

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n),$$

$$x_j \text{ integer} \qquad (\text{for some or all } j = 1, 2, \ldots, n).$$

This problem is called the (linear) *integer-programming problem*. It is said to be a *mixed* integer program when some, but not all, variables are restricted to be integer, and is called a *pure* integer program when *all* decision variables must be integers. As we saw in the preceding chapter, if the constraints are of a network nature, then an integer solution can be obtained by ignoring the integrality restrictions and solving the resulting linear program. In general, though, variables will be fractional in the linear-programming solution, and further measures must be taken to determine the integer-programming solution.

The purpose of this chapter is twofold. First, we will discuss integer-programming formulations. This should provide insight into the scope of integer-programming applications and give some indication of why many practitioners feel that the integer-programming model is one of the most important models in management science. Second, we consider basic approaches that have been developed for solving integer and mixed-integer programming problems.

### 9.1 SOME INTEGER-PROGRAMMING MODELS

Integer-programming models arise in practically every area of application of mathematical programming. To develop a preliminary appreciation for the importance of these models, we introduce, in this section, three areas where integer programming has played an important role in supporting managerial decisions. We do not provide the most intricate available formulations in each case, but rather give basic models and suggest possible extensions.

**Capital Budgeting** In a typical capital-budgeting problem, decisions involve the selection of a number of potential investments. The investment decisions might be to choose among possible plant locations, to select a configuration of capital equipment, or to settle upon a set of research-and-development projects. Often it makes no sense to consider partial investments in these activities, and so the problem becomes a *go–no-go* integer program, where the decision variables are taken to be $x_j = 0$ or 1, indicating that the $j$th investment is rejected or accepted. Assuming that $c_j$ is the contribution resulting from the $j$th investment and that $a_{ij}$ is the amount of resource $i$, such as cash or manpower, used on the $j$th investment, we can state the problem formally as:

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

subject to:

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

$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n).$$

The objective is to maximize total contribution from all investments without exceeding the limited availability $b_i$ of any resource.

One important special scenario for the capital-budgeting problem involves cash-flow constraints. In this case, the constraints

$$\sum_{j=1}^{n} a_{ij} x_i \le b_i$$

reflect incremental cash balance in each period. The coefficients $a_{ij}$ represent the net cash flow from investment $j$ in period $i$. If the investment requires additional cash in period $i$, then $a_{ij} > 0$, while if the investment *generates* cash in period $i$, then $a_{ij} < 0$. The righthand-side coefficients $b_i$ represent the incremental exogenous cash flows. If additional funds are made available in period $i$, then $b_i > 0$, while if funds are withdrawn in period $i$, then $b_i < 0$. These constraints state that the funds required for investment must be less than or equal to the funds generated from prior investments plus exogenous funds made available (or *minus* exogenous funds withdrawn).

The capital-budgeting model can be made much richer by including logical considerations. Suppose, for example, that investment in a new product line is contingent upon previous investment in a new plant. This *contingency* is modeled simply by the constraint

$$x_j \ge x_i,$$

which states that if $x_i = 1$ and project $i$ (new product development) is accepted, then necessarily $x_j = 1$ and project $j$ (construction of a new plant) must be accepted. Another example of this nature concerns conflicting projects. The constraint

$$x_1 + x_2 + x_3 + x_4 \le 1,$$

for example, states that only one of the first four investments can be accepted. Constraints like this commonly are called *multiple-choice constraints*. By combining these logical constraints, the model can incorporate many complex interactions between projects, in addition to issues of resource allocation.

The simplest of all capital-budgeting models has just one resource constraint, but has attracted much attention in the management-science literature. It is stated as:

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

subject to:

$$\sum_{j=1}^{n} a_j x_j \le b,$$
$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n).$$

Usually, this problem is called the 0–1 *knapsack* problem, since it is analogous to a situation in which a hiker must decide which goods to include on his trip. Here $c_j$ is the ''value'' or utility of including good $j$, which weighs $a_j > 0$ pounds; the objective is to maximize the ''pleasure of the trip,'' subject to the weight limitation that the hiker can carry no more than $b$ pounds. The model is altered somewhat by allowing more than one unit of any good to be taken, by writing $x_j \ge 0$ and $x_j$-integer in place of the 0–1 restrictions on the variables. The knapsack model is important because a number of integer programs can be shown to be equivalent to it, and further, because solution procedures for knapsack models have motivated procedures for solving general integer programs.

**Warehouse Location** In modeling distribution systems, decisions must be made about tradeoffs between transportation costs and costs for operating distribution centers. As an example, suppose that a manager must decide which of $n$ warehouses to use for meeting the demands of $m$ customers for a good. The decisions to be made are which warehouses to operate and how much to ship from any warehouse to any customer. Let

$$y_i = \begin{cases} 1 & \text{if warehouse } i \text{ is opened,} \\ 0 & \text{if warehouse } i \text{ is not opened;} \end{cases}$$
$$x_{ij} = \text{Amount to be sent from warehouse } i \text{ to customer } j.$$

The relevant costs are:

$f_i$ = Fixed operating cost for warehouse $i$, ifopened (for example, a cost to lease the warehouse),

$c_{ij}$ = Per-unit operating cost at warehouse $i$ plus the transportation cost for shipping from warehouse $i$ to customer $j$.

There are two types of constraints for the model:

i) the demand $d_j$ of each customer must be filled from the warehouses; and

ii) goods can be shipped from a warehouse only if it is opened.

The model is:

$$\text{Minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i, \tag{1}$$

subject to:

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

$$\sum_{j=1}^{n} x_{ij} - y_i \left( \sum_{j=1}^{n} d_j \right) \le 0 \qquad (i = 1, 2, \ldots, m), \tag{3}$$

$$x_{ij} \ge 0 \qquad (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n),$$
$$y_i = 0 \quad \text{or} \quad 1 \qquad (i = 1, 2, \ldots, m).$$

The objective function incorporates transportation and variable warehousing costs, in addition to fixed costs for operating warehouses. The constraints (2) indicate that each customer's demand must be met. The summation over the shipment variables $x_{ij}$ in the $i$th constraint of (3) is the amount of the good shipped from warehouse $i$. When the warehouse is not opened, $y_i = 0$ and the constraint specifies that nothing can be shipped from the warehouse. On the other hand, when the warehouse is opened and $y_i = 1$, the constraint simply states that the amount to be shipped from warehouse $i$ can be no larger than the total demand, which is always true. Consequently, constraints (3) imply restriction (ii) as proposed above.

Although oversimplified, this model forms the core for sophisticated and realistic distribution models incorporating such features as:

1. multi-echelon distribution systems from plant to warehouse to customer;
2. capacity constraints on both plant production and warehouse throughput;
3. economies of scale in transportation and operating costs;
4. service considerations such as maximum distribution time from warehouses to customers;
5. multiple products; or
6. conditions preventing splitting of orders (in the model above, the demand for any customer can be supplied from several warehouses).

These features can be included in the model by changing it in several ways. For example, warehouse capacities are incorporated by replacing the term involving $y_i$ in constraint (3) with $y_i K_i$, where $K_i$ is the throughput capacity of warehouse $i$; multi-echelon distribution may require triple-subscripted variables $x_{ijk}$ denoting the amount to be shipped, from plant $i$ to customer $k$ through warehouse $j$. Further examples of how the simple warehousing model described here can be modified to incorporate the remaining features mentioned in this list are given in the exercises at the end of the chapter.

**Scheduling** The entire class of problems referred to as sequencing, scheduling, and routing are inherently integer programs. Consider, for example, the scheduling of students, faculty, and classrooms in such a way that the number of students who cannot take their first choice of classes is minimized. There are constraints on the number and size of classrooms available at any one time, the availability of faculty members at particular times, and the preferences of the students for particular schedules. Clearly, then, the $i$th student *is* scheduled for the $j$th class during the $n$th time period or *not*; hence, such a variable is either zero or one. Other examples of this class of problems include line-balancing, critical-path scheduling with resource constraints, and vehicle dispatching.

As a specific example, consider the scheduling of airline flight personnel. The airline has a number of routing ''legs'' to be flown, such as 10 A.M. New York to Chicago, or 6 P.M.Chicago to Los Angeles. The airline must schedule its personnel crews on routes to cover these flights. One crew, for example, might be scheduled to fly a route containing the two legs just mentioned. The decision variables, then, specify the scheduling of the crews to routes:

$$x_j = \begin{cases} 1 & \text{if a crew is assigned to route } j, \\ 0 & \text{otherwise.} \end{cases}$$

Let

$$a_{ij} = \begin{cases} 1 & \text{if leg } i \text{ is included on route } j, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$c_j = \text{Cost for assigning a crew to route } j.$$

The coefficients $a_{ij}$ define the acceptable combinations of legs and routes, taking into account such characteristics as sequencing of legs for making connections between flights and for including in the routes ground time for maintenance. The model becomes:

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

subject to:

$$\sum_{j=1}^{n} a_{ij}x_j = 1 \qquad (i = 1, 2, \ldots, m), \tag{4}$$

$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n).$$

The $i$th constraint requires that one crew must be assigned on a route to fly leg $i$. An alternative formulation permits a crew to ride as passengers on a leg. Then the constraints (4) become:

$$\sum_{j=1}^{n} a_{ij}x_j \geq 1 \qquad (i = 1, 2, \ldots, m). \tag{5}$$

If, for example,

$$\sum_{j=1}^{n} a_{1j}x_j = 3,$$

then two crews fly as passengers on leg 1, possibly to make connections to other legs to which they have been assigned for duty.

These airline-crew scheduling models arise in many other settings, such as vehicle delivery problems, political districting, and computer data processing. Often model (4) is called a *set-partitioning problem*, since the set of legs will be divided, or partitioned, among the various crews. With constraints (5), it is called a *set-covering problem*, since the crews then will cover the set of legs.

Another scheduling example is the so-called *traveling salesman problem*. Starting from his home, a salesman wishes to visit each of $(n-1)$ other cities and return home at minimal cost. He must visit each city exactly once and it costs $c_{ij}$ to travel from city $i$ to city $j$. What route should he select? If we let

$$x_{ij} = \begin{cases} 1 & \text{if he goes from city } i \text{ to city } j, \\ 0 & \text{otherwise,} \end{cases}$$

we may be tempted to formulate his problem as the assignment problem:

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

subject to:

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

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

$$x_{ij} \geq 0 \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

The constraints require that the salesman must enter and leave each city exactly once. Unfortunately, the assignment model can lead to infeasible solutions. It is possible in a six-city problem, for example, for the assignment solution to route the salesman through two disjoint subtours of the cities instead of on a single trip or tour. (See Fig. 9.1.)

Consequently, additional constraints must be included in order to eliminate subtour solutions. There are a number of ways to accomplish this. In this example, we can avoid the subtour solution of Fig. 9.1 by including the constraint:

$$x_{14} + x_{15} + x_{16} + x_{24} + x_{25} + x_{26} + x_{34} + x_{35} + x_{36} \geq 1.$$

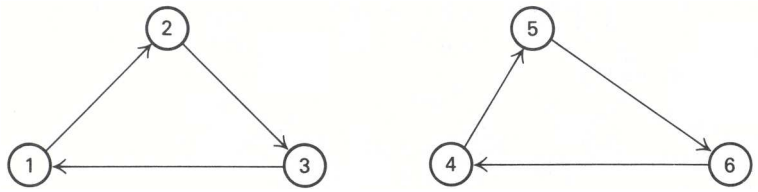


**Figure 9.1** Disjoint subtours.

This inequality ensures that at least one leg of the tour connects cities 1, 2, and 3 with cities 4, 5, and 6. In general, if a constraint of this form is included for each way in which the cities can be divided into two groups, then subtours will be eliminated. The problem with this and related approaches is that, with $n$ cities, $(2^n - 1)$ constraints of this nature must be added, so that the formulation becomes a very large integer-programming problem. For this reason the traveling salesman problem generally is regarded as difficult when there are many cities.

The traveling salesman model is used as a central component of many vehicular routing and scheduling models. It also arises in production scheduling. For example, suppose that we wish to sequence $(n - 1)$ jobs on a single machine, and that $c_{ij}$ is the cost for setting up the machine for job $j$, given that job $i$ has just been completed. What scheduling sequence for the jobs gives the lowest total setup costs? The problem can be interpreted as a traveling salesman problem, in which the ''salesman'' corresponds to the machine which must ''visit'' or perform each of the jobs. ''Home'' is the initial setup of the machine, and, in some applications, the machine will have to be returned to this initial setup after completing all of the jobs. That is, the ''salesman'' must return to ''home'' after visiting the ''cities.''

## 9.2 FORMULATING INTEGER PROGRAMS

The illustrations in the previous section not only have indicated specific integer-programming applications, but also have suggested how integer variables can be used to provide broad modeling capabilities beyond those available in linear programming. In many applications, integrality restrictions reflect natural indivisibilities of the problem under study. For example, when deciding how many nuclear aircraft carriers to have in the U.S. Navy, fractional solutions clearly are meaningless, since the optimal number is on the order of one or two. In these situations, the decision variables are inherently integral by the nature of the decision-making problem.

This is not necessarily the case in every integer-programming application, as illustrated by the capital-budgeting and the warehouse-location models from the last section. In these models, integer variables arise from (i) logical conditions, such as *if* a new product is developed, *then* a new plant must be constructed, and from (ii) non-linearities such as fixed costs for opening a warehouse. Considerations of this nature are so important for modeling that we devote this section to analyzing and consolidating specific integer-programming formulation techniques, which can be used as tools for a broad range of applications.

### Binary (0–1) Variables

Suppose that we are to determine whether or not to engage in the following activities: (i) to build a new plant, (ii) to undertake an advertising campaign, or (iii) to develop a new product. In each case, we must make a *yes–no* or so-called *go–no–go* decision. These choices are modeled easily by letting $x_j = 1$ if we engage in the $j$th activity and $x_j = 0$ otherwise. Variables that are restricted to 0 or 1 in this way are termed *binary*, *bivalent*, *logical*, or *0–1 variables*. Binary variables are of great importance because they occur regularly in many model formulations, particularly in problems addressing long-range and high-cost strategic decisions associated with capital-investment planning.

If, further, management had decided that *at most one* of the above three activities can be pursued, the

following constraint is appropriate:

$$\sum_{j=1}^{3} x_j \leq 1.$$

As we have indicated in the capital-budgeting example in the previous section, this restriction usually is referred to as a *multiple-choice* constraint, since it limits our choice of investments to be *at most one* of the three available alternatives.

Binary variables are useful whenever variables can assume one of two values, as in batch processing. For example, suppose that a drug manufacturer must decide whether or not to use a fermentation tank. If he uses the tank, the processing technology requires that he make $B$ units. Thus, his production $y$ must be 0 or $B$, and the problem can be modeled with the binary variable $x_j = 0$ or 1 by substituting $Bx_j$ for $y$ everywhere in the model.

#### Logical Constraints

Frequently, problem settings impose logical constraints on the decision variables (like timing restrictions, contingencies, or conflicting alternatives), which lend themselves to integer-programming formulations. The following discussion reviews the most important instances of these logical relationships.

*Constraint Feasibility*

Possibly the simplest logical question that can be asked in mathematical programming is whether a given choice of the decision variables satisfies a constraint. More precisely, *when* is the general constraint

$$f(x_1, x_2, \ldots, x_n) \leq b \tag{6}$$

satisfied?

We introduce a binary variable $y$ with the interpretation:

$$y = \begin{cases} 0 & \text{if the constraint is known to be satisfied,} \\ 1 & \text{otherwise,} \end{cases}$$

and write

$$f(x_1, x_2, \ldots, x_n) - By \leq b, \tag{7}$$

where the constant $B$ is chosen to be large enough so that the constraint always is satisfied if $y = 1$; that is,

$$f(x_1, x_2, \ldots, x_n) \leq b + B,$$

for every possible choice of the decision variables $x_1, x_2, \ldots, x_n$ at our disposal. Whenever $y = 0$ gives a feasible solution to constraint (7), we know that constraint (6) must be satisfied. In practice, it is usually very easy to determine a large number to serve as $B$, although generally it is best to use the smallest possible value of $B$ in order to avoid numerical difficulties during computations.

*Alternative Constraints*

Consider a situation with the *alternative* constraints:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &\leq b_1, \\ f_2(x_1, x_2, \ldots, x_n) &\leq b_2. \end{aligned}$$

At least one, but not necessarily both, of these constraints must be satisfied. This restriction can be modeled by combining the technique just introduced with a multiple-choice constraint as follows:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) - B_1 y_1 &\leq b_1, \\ f_2(x_1, x_2, \ldots, x_n) - B_2 y_2 &\leq b_2, \\ y_1 + y_2 &\leq 1, \\ y_1, y_2 \text{ binary.} & \end{aligned}$$

The variables $y_1$ and $y_2$ and constants $B_1$ and $B_2$ are chosen as above to indicate when the constraints are satisfied. The multiple-choice constraint $y_1 + y_2 \leq 1$ implies that at least one variable $y_j$ equals 0, so that, as required, at least one constraint must be satisfied.

We can save one integer variable in this formulation by noting that the multiple-choice constraint can be replaced by $y_1 + y_2 = 1$, or $y_2 = 1 - y_1$, since this constraint also implies that either $y_1$ or $y_2$ equals 0. The resulting formulation is given by:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) - B_1 y_1 &\leq b_1, \\ f_2(x_1, x_2, \ldots, x_n) - B_2(1 - y_1) &\leq b_2, \end{aligned}$$

$$y_1 = 0 \quad \text{or} \quad 1.$$

As an illustration of this technique, consider again the custom-molder example from Chapter 1. That example included the constraint

$$6x_1 + 5x_2 \leq 60, \tag{8}$$

which represented the production capacity for producing $x_1$ hundred cases of six-ounce glasses and $x_2$ hundred cases of ten-ounce glasses. Suppose that there were an alternative production process that could be used, having the capacity constraint

$$4x_1 + 5x_2 \leq 50. \tag{9}$$

Then the decision variables $x_1$ and $x_2$ must satisfy *either* (8) or (9), depending upon which production process is selected. The integer-programming formulation replaces (8) and (9) with the constraints:

$$\begin{aligned} 6x_1 + 5x_2 - 100y &\leq 60, \\ 4x_1 + 5x_2 - 100(1 - y) &\leq 50, \end{aligned}$$

$$y = 0 \quad \text{or} \quad 1.$$

In this case, both $B_1$ and $B_2$ are set to 100, which is large enough so that the constraint is not limiting for the production process *not* used.

*Conditional Constraints*

These constraints have the form:

$$f_1(x_1, x_2, \ldots, x_n) > b_1 \quad \text{implies that } f_2(x_1, x_2, \ldots, x_n) \leq b_2.$$

Since this implication is not satisfied *only when both* $f_1(x_1, x_2, \ldots, x_n) > b_1$ *and* $f_2(x_1, x_2, \ldots, x_n) > b_2$, the conditional constraint is logically equivalent to the alternative constraints

$$f_1(x_1, x_2, \ldots, x_n) \leq b_1 \quad \text{and/or} \quad f_2(x_1, x_2, \ldots, x_n) \leq b_2,$$

where *at least one* must be satisfied. Hence, this situation can be modeled by alternative constraints as indicated above.

*k-Fold Alternatives*

Suppose that we must satisfy at least $k$ of the constraints:

$$f_j(x_1, x_2, \ldots, x_n) \leq b_j \quad (j = 1, 2, \ldots, p).$$

For example, these restrictions may correspond to manpower constraints for $p$ potential inspection systems for quality control in a production process. If management has decided to adopt at least $k$ inspection systems, then the $k$ constraints specifying the manpower restrictions for these systems must be satisfied, and the

remaining constraints can be ignored. Assuming that $B_j$ for $j = 1, 2, \ldots, p$, are chosen so that the ignored constraints will not be binding, the general problem can be formulated as follows:

$$f_j(x_1, x_2, \ldots, x_n) - B_j(1 - y_j) \leq b_j \qquad (j = 1, 2, \ldots, p),$$

$$\sum_{j=1}^{p} y_j \geq k,$$

$$y_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, p).$$

That is, $y_j = 1$ if the $j$th constraint is to be satisfied, and at least $k$ of the constraints must be satisfied. If we define $y'_j \equiv 1 - y_j$, and substitute for $y_j$ in these constraints, the form of the resulting constraints is analogous to that given previously for modeling alternative constraints.

*Compound Alternatives*

The feasible region shown in Fig. 9.2 consists of three disjoint regions, each specified by a system of inequalities. The feasible region is defined by alternative sets of constraints, and can be modeled by the system:

$$\left.\begin{aligned} f_1(x_1, x_2) - B_1 y_1 &\leq b_1 \\ f_2(x_1, x_2) - B_2 y_1 &\leq b_2 \end{aligned}\right\} \text{ Region 1 constraints}$$

$$\left.\begin{aligned} f_3(x_1, x_2) - B_3 y_2 &\leq b_3 \\ f_4(x_1, x_2) - B_4 y_2 &\leq b_4 \end{aligned}\right\} \text{ Region 2 constraints}$$

$$\left.\begin{aligned} f_5(x_1, x_2) - B_5 y_3 &\leq b_5 \\ f_6(x_1, x_2) - B_6 y_3 &\leq b_6 \\ f_7(x_1, x_2) - B_7 y_3 &\leq b_7 \end{aligned}\right\} \text{ Region 3 constraints}$$

$$y_1 + y_2 + y_3 \leq 2,$$
$$x_1 \geq 0, \quad x_2 \geq 0,$$
$$y_1, y_2, y_3 \quad \text{binary}.$$

Note that we use the same binary variable $y_j$ for eachconstraint defining one of the regions, and that the

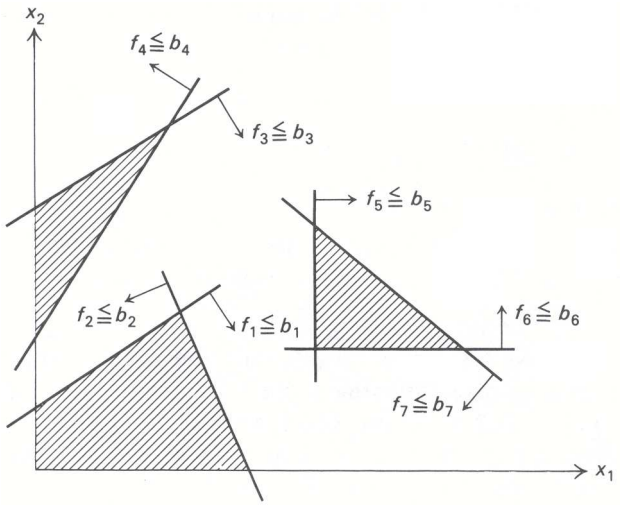


**Figure 9.2** An example of compound alternatives.

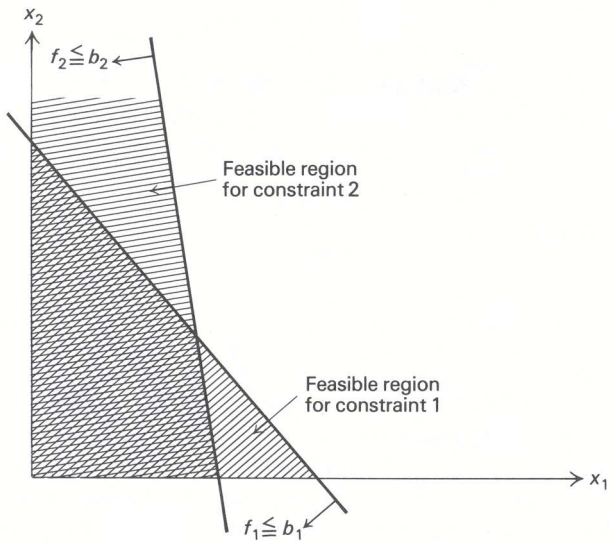


**Figure 9.3** Geometry of alternative constraints.

constraint $y_1 + y_2 + y_3 \leq 2$ implies that the decision variables $x_1$ and $x_2$ lie in *at least one* of the required regions. Thus, for example, if $y_3 = 0$, then each of the constraints

$$f_5(x_1, x_2) \leq b_5, \quad f_6(x_1, x_2) \leq b_6, \quad \text{and} \quad f_7(x_1, x_2) \leq b_7$$

is satisfied.

The regions do not have to be disjoint before we can apply this technique. Even the simple alternative constraint

$$f_1(x_1, x_2) \leq b_1 \quad \text{or} \quad f_2(x_1, x_2) \leq b_2$$

shown in Fig. 9.3 contains overlapping regions.

#### Representing Nonlinear Functions

Nonlinear functions can be represented by integer-programming formulations. Let us analyze the most useful representations of this type.

i) *Fixed Costs*

Frequently, the objective function for a minimization problem contains fixed costs (preliminary design costs, fixed investment costs, fixed contracts, and so forth). For example, the cost of producing $x$ units of a specific product might consist of a fixed cost of setting up the equipment and a variable cost per unit produced on the equipment. An example of this type of cost is given in Fig. 9.4.

Assume that the equipment has a capacity of $B$ units. Define $y$ to be a binary variable that indicates when the fixed cost is incurred, so that $y = 1$ when $x > 0$ and $y = 0$ when $x = 0$. Then the contribution to cost due to $x$ may be written as

$$Ky + cx,$$

with the constraints:

$$\begin{aligned} x &\leq By, \\ x &\geq 0, \\ y &= 0 \quad \text{or} \quad 1. \end{aligned}$$

As required, these constraints imply that $x = 0$ when the fixed cost is not incurred, i.e., when $y = 0$. The constraints themselves do not imply that $y = 0$ if $x = 0$. But when $x = 0$, the minimization will clearly

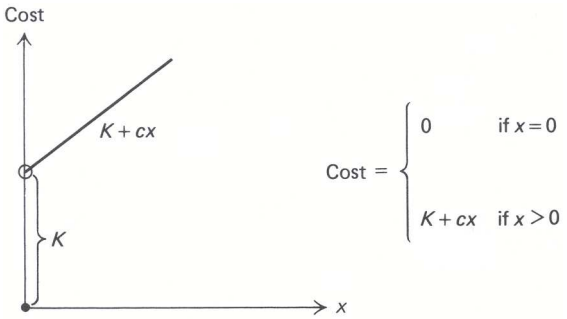


**Figure 9.4** A fixed cost.

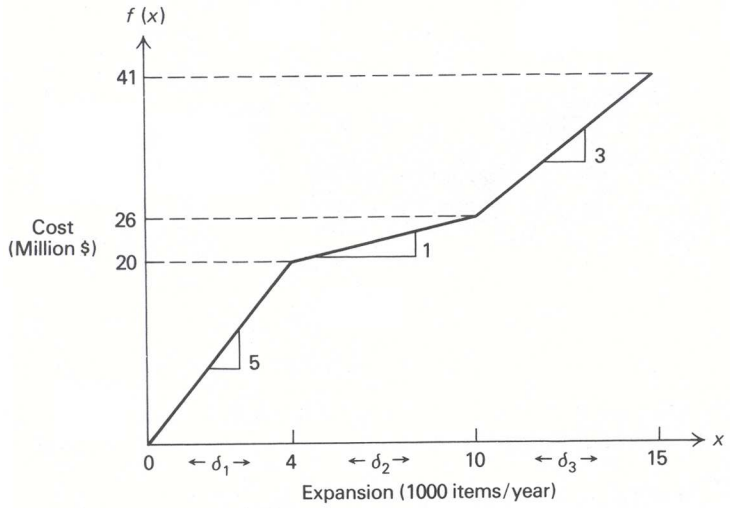


**Figure 9.5** Modeling a piecewise linear curve.

select $y = 0$, so that the fixed cost is not incurred. Finally, observe that if $y = 1$, then the added constraint becomes $x \leq B$, which reflects the capacity limit on the production equipment.

ii) *Piecewise Linear Representation*

Another type of nonlinear function that can be represented by integer variables is a piecewise linear curve. Figure 9.5 illustrates a cost curve for plant expansion that contains three linear segments with variable costs of 5, 1, and 3 million dollars per 1000 items of expansion.

To model the cost curve, we express any value of $x$ as the sum of three variables $\delta_1, \delta_2, \delta_3$, so that the cost for each of these variables is linear. Hence,

$$x = \delta_1 + \delta_2 + \delta_3,$$

where

$$\begin{aligned} 0 \leq \delta_1 \leq 4, \\ 0 \leq \delta_2 \leq 6, \\ 0 \leq \delta_3 \leq 5; \end{aligned} \tag{10}$$

and the total variable cost is given by:

$$\text{Cost} = 5\delta_1 + \delta_2 + 3\delta_3.$$

Note that we have defined the variables so that:

$\delta_1$ corresponds to the amount by which $x$ exceeds 0, but is less than or equal to 4;
$\delta_2$ is the amount by which $x$ exceeds 4, but is less than or equal to 10; and
$\delta_3$ is the amount by which $x$ exceeds 10, but is less than or equal to 15.

If this interpretation is to be valid, we must also require that $\delta_1 = 4$ whenever $\delta_2 > 0$ and that $\delta_2 = 6$ whenever $\delta_3 > 0$. Otherwise, when $x = 2$, say, the cost would be minimized by selecting $\delta_1 = \delta_3 = 0$ and $\delta_2 = 2$, since the variable $\delta_2$ has the smallest variable cost. However, these restrictions on the variables are simply conditional constraints and can be modeled by introducing binary variables, as before.

If we let

$$w_1 = \begin{cases} 1 & \text{if } \delta_1 \text{ is at its upper bound,} \\ 0 & \text{otherwise,} \end{cases}$$

$$w_2 = \begin{cases} 1 & \text{if } \delta_2 \text{ is at its upper bound,} \\ 0 & \text{otherwise,} \end{cases}$$

then constraints (10) can be replaced by

$$\begin{aligned} 4w_1 \le \delta_1 &\le 4, \\ 6w_2 \le \delta_2 &\le 6w_1, \\ 0 \le \delta_3 &\le 5w_2, \\ w_1 \quad \text{and} \quad w_2 &\text{ binary,} \end{aligned} \tag{11}$$

to ensure that the proper conditional constraints hold. Note that if $w_1 = 0$, then $w_2 = 0$, to maintain feasibility for the constraint imposed upon $\delta_2$, and (11) reduces to

$$0 \le \delta_1 \le 4, \quad \delta_2 = 0, \quad \text{and} \quad \delta_3 = 0.$$

If $w_1 = 1$ and $w_2 = 0$, then (11) reduces to

$$\delta_1 = 4, \quad 0 \le \delta_2 \le 6, \quad \text{and} \quad \delta_3 = 0.$$

Finally, if $w_1 = 1$ and $w_2 = 1$, then (11) reduces to

$$\delta_1 = 4, \quad \delta_2 = 6, \quad \text{and} \quad 0 \le \delta_3 \le 5.$$

Hence, we observe that there are three feasible combinations for the values of $w_1$ and $w_2$:

$$\begin{aligned} &w_1 = 0, \quad w_2 = 0 \quad \text{corresponding to } 0 \le x \le 4 \quad \text{since } \delta_2 = \delta_3 = 0; \\ &w_1 = 1, \quad w_2 = 0 \quad \text{corresponding to } 4 \le x \le 10 \quad \text{since } \delta_1 = 4 \text{ and } \delta_3 = 0; \end{aligned}$$

and

$$w_1 = 1, \quad w_2 = 1 \quad \text{corresponding to } 10 \le x \le 15 \quad \text{since } \delta_1 = 4 \text{ and } \delta_2 = 6.$$

The same general technique can be applied to piecewise linear curves with any number of segments. The general constraint imposed upon the variable $\delta_j$ for the $j$th segment will read:

$$L_j w_j \le \delta_j \le L_j w_{j-1},$$

where $L_j$ is the length of the segment.

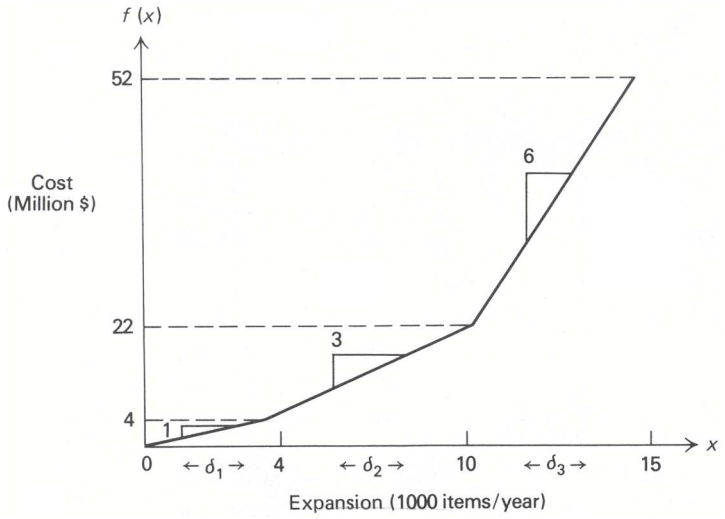


**Figure 9.6** Diseconomies of scale.

iii) *Diseconomies of Scale*
An important special case for representing nonlinear functions arises when only diseconomies of scale apply—that is, when marginal costs are increasing for a minimization problem or marginal returns are decreasing for a maximization problem. Suppose that the expansion cost in the previous example now is specified by Fig. 9.6.

In this case, the cost is represented by

$$\text{Cost} = \delta_1 + 3\delta_2 + 6\delta_3,$$

subject only to the linear constraints without integer variables,

$$\begin{aligned} 0 \le \delta_1 \le 4 \\ 0 \le \delta_2 \le 6, \\ 0 \le \delta_3 \le 5. \end{aligned}$$

The conditional constraints involving binary variables in the previous formulation can be ignored if the cost curve appears in a minimization objective function, since the coefficients of $\delta_1$, $\delta_2$, and $\delta_3$ imply that it is always best to set $\delta_1 = 4$ before taking $\delta_2 > 0$, and to set $\delta_2 = 6$ before taking $\delta_3 > 0$. As a consequence, the integer variables have been avoided completely.

This representation without integer variables is not valid, however, if economies of scale are present; for example, if the function given in Fig. 9.6 appears in a maximization problem. In such cases, it would be best to select the third segment with variable $\delta_3$ before taking the first two segments, since the returns are higher on this segment. In this instance, the model requires the binary-variable formulation of the previous section.

iv) *Approximation of Nonlinear Functions*
One of the most useful applications of the piecewise linear representation is for approximating nonlinear functions. Suppose, for example, that the expansion cost in our illustration is given by the heavy curve in Fig. 9.7.

If we draw linear segments joining selected points on the curve, we obtain a *piecewise linear approximation*, which can be used instead of the curve in the model. The piecewise approximation, of course, is represented by introducing integer variables as indicated above. By using more points on the curve, we can make the approximation as close as we desire.

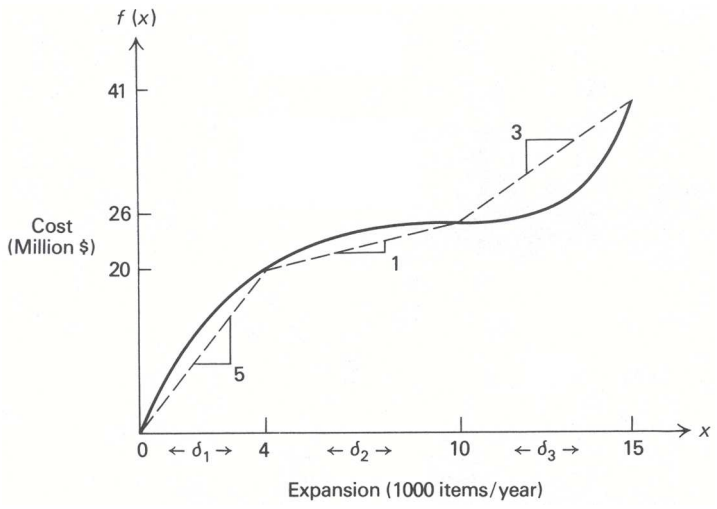


**Figure 9.7** Approximation of a nonlinear curve.

### 9.3 A SAMPLE FORMULATION †

Proper placement of service facilities such as schools, hospitals, and recreational areas is essential to efficient urban design. Here we will present a simplified model for firehouse location. Our purpose is to show formulation devices of the previous section arising together in a meaningful context, rather than to give a comprehensive model for the location problem *per se*. As a consequence, we shall ignore many relevant issues, including uncertainty.

Assume that population is concentrated in $I$ districts within the city and that district $i$ contains $p_i$ people. Preliminary analysis (land surveys, politics, and so forth) has limited the potential location of firehouses to $J$ sites. Let $d_{ij} \geq 0$ be the distance from the center of district $i$ to site $j$. We are to determine the ''best'' site selection and assignment of districts to firehouses. Let

$$y_j = \begin{cases} 1 & \text{if site } j \text{ is selected,} \\ 0 & \text{otherwise;} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1 & \text{if district } i \text{ is assigned to site } j, \\ 0 & \text{otherwise.} \end{cases}$$

The basic constraints are that every district should be assigned to exactly one firehouse, that is,

$$\sum_{j=1}^{J} x_{ij} = 1 \qquad (i = 1, 2, \ldots, I),$$

and that no district should be assigned to an unused site, that is, $y_j = 0$ implies $x_{ij} = 0$ $(i = 1, 2, \ldots, I)$. The latter restriction can be modeled as alternative constraints, or more simply as:

$$\sum_{i=1}^{I} x_{ij} \leq y_j I \qquad (j = 1, 2, \ldots, J).$$

Since $x_{ij}$ are binary variables, their sum never exceeds $I$, so that if $y_j = 1$, then constraint $j$ is nonbinding. If $y_j = 0$, then $x_{ij} = 0$ for all $i$.

† This section may be omitted without loss of continuity.

Next note that $d_i$, the distance from district $i$ to its assigned firehouse, is given by:

$$d_i = \sum_{j=1}^{J} d_{ij} x_{ij},$$

since one $x_{ij}$ will be 1 and all others 0.

Also, the total population serviced by site $j$ is:

$$s_j = \sum_{i=1}^{I} p_i x_{ij}.$$

Assume that a central district is particularly susceptible to fire and that either sites 1 and 2 or sites 3 and 4 can be used to protect this district. Then one of a number of similar restrictions might be:

$$y_1 + y_2 \geq 2 \qquad \text{or} \qquad y_3 + y_4 \geq 2.$$

We let $y$ be a binary variable; then these alternative constraints become:

$$\begin{aligned} y_1 + y_2 &\geq 2y, \\ y_3 + y_4 &\geq 2(1 - y). \end{aligned}$$

Next assume that it costs $f_j(s_j)$ to build a firehouse at site $j$ to service $s_j$ people and that a total budget of $B$ dollars has been allocated for firehouse construction. Then

$$\sum_{j=1}^{J} f_j(s_j) \leq B.$$

Finally, one possible social-welfare function might be to minimize the distance traveled to the district farthest from its assigned firehouse, that is, to:

Minimize $D$,

where

$$D = \max d_i;$$

or, equivalently,[‡] to

Minimize $D$,

subject to:

$$D \geq d_i \quad (i = 1, 2, \ldots, I).$$

Collecting constraints and substituting above for $d_i$ in terms of its defining relationship

$$d_i = \sum_{j=1}^{J} d_{ij} x_{ij},$$

we set up the full model as:

Minimize $D$,

[‡] The inequalities $D \geq d_i$ imply that $D \geq \max d_i$. The minimization of $D$ then ensures that it will actually be the maximum of the $d_i$.

subject to:

$$
\begin{aligned}
D- \quad & \sum_{j=1}^{J} d_{ij}x_{ij} \geq 0 \qquad (i = 1, 2, \ldots, I), \\
& \sum_{j=1}^{J} x_{ij} = 1 \qquad (i = 1, 2, \ldots, I), \\
& \sum_{i=1}^{I} x_{ij} \leq y_j I \qquad (j = 1, 2, \ldots, J), \\
S_j- \quad & \sum_{i=1}^{I} p_i x_{ij} = 0 \qquad (j = 1, 2, \ldots, J), \\
& \sum_{j=1}^{J} f_j(s_j) \leq B, \\
& y_1 + y_2 - 2y \geq 0, \\
& y_3 + y_4 + 2y \geq 2,
\end{aligned}
$$

$$
x_{ij},\ y_j,\ y \quad \text{binary} \qquad (i = 1, 2, \ldots, I;\ j = 1, 2, \ldots, J).
$$

At this point we might replace each function $f_j(s_j)$ by an integer-programming approximation to complete the model. Details are left to the reader. Note that if $f_j(s_j)$ contains a fixed cost, then new fixed-cost variables need not be introduced—the variable $y_j$ serves this purpose.

The last comment, and the way in which the conditional constraint ''$y_j = 0$ implies $x_{ij} = 0$ $(i = 1, 2, \ldots, I)$'' has been modeled above, indicate that the formulation techniques of Section 9.2 should not be applied without thought. Rather, they provide a common framework for modeling and should be used in conjunction with good modeling ''common sense.'' In general, it is best to introduce as few integer variables as possible.

## 9.4 SOME CHARACTERISTICS OF INTEGER PROGRAMS—A SAMPLE PROBLEM

Whereas the simplex method is effective for solving linear programs, there is no single technique for solving integer programs. Instead, a number of procedures have been developed, and the performance of any particular technique appears to be highly problem-dependent. Methods to date can be classified broadly as following one of three approaches:

i) enumeration techniques, including the branch-and-bound procedure;
ii) cutting-plane techniques; and
iii) group-theoretic techniques.

In addition, several composite procedures have been proposed, which combine techniques using several of these approaches. In fact, there is a trend in computer systems for integer programming to include a number of approaches and possibly utilize them all when analyzing a given problem. In the sections to follow, we shall consider the first two approaches in some detail. At this point, we shall introduce a specific problem and indicate some features of integer programs. Later we will use this example to illustrate and motivate the solution procedures. Many characteristics of this example are shared by the integer version of the custom-molder problem presented in Chapter 1.

The problem is to determine $z^*$ where:

$$z^* = \max z = 5x_1 + 8x_2,$$

subject to:

$$\begin{aligned} x_1 + \ x_2 &\leq \ 6, \\ 5x_1 + 9x_2 &\leq 45, \end{aligned}$$

$$x_1, x_2 \geq 0 \quad \text{and} \quad \text{integer}.$$

The feasible region is sketched in Fig. 9.8. Dots in the shaded region are feasible integer points.

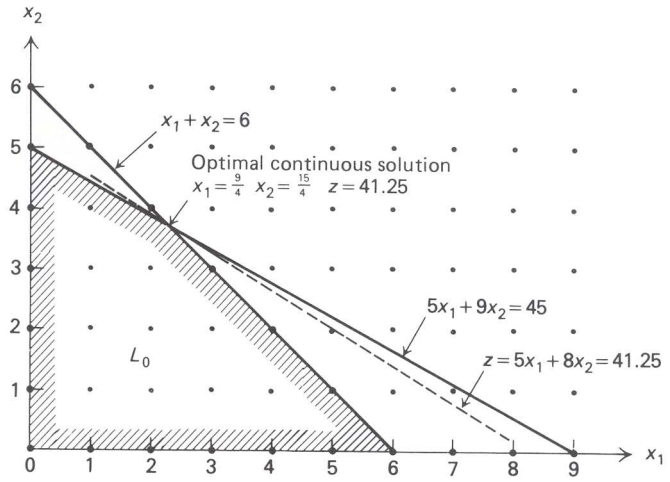


**Figure 9.8** An integer programming example.

If the integrality restrictions on variables are dropped, the resulting problem is a linear program. We will call it the *associated linear program*. We may easily determine its optimal solution graphically. Table 9.1 depicts some of the features of the problem.

**Table 9.1** Problem features.

| | *Continuous optimum* | *Round off* | *Nearest feasible point* | *Integer optimum* |
|---|---|---|---|---|
| $x_1$ | $\frac{9}{4} = 2.25$ | 2 | 2 | 0 |
| $x_2$ | $\frac{15}{4} = 3.75$ | 4 | 3 | 5 |
| $z$ | 41.25 | Infeasible | 34 | 40 |

Observe that the optimal integer-programming solution is not obtained by *rounding* the linear-programming solution. The closest point to the optimal linear-program solution is not even feasible. Also, note that the nearest feasible integer point to the linear-program solution is far removed from the optimal integer point. Thus, it is not sufficient simply to round linear-programming solutions. In fact, by scaling the righthand-side and cost coefficients of this example properly, we can construct a problem for which the optimal integer-programming solution lies as far as we like from the rounded linear-programming solution, in either $z$ value or distance on the plane.

In an example as simple as this, almost any solution procedure will be effective. For instance, we could easily enumerate all the integer points with $x_1 \leq 9,\ x_2 \leq 6$, and select the best feasible point. In practice, the number of points to be considered is likely to prohibit such an exhaustive enumeration of potentially feasible points, and a more sophisticated procedure will have to be adopted.

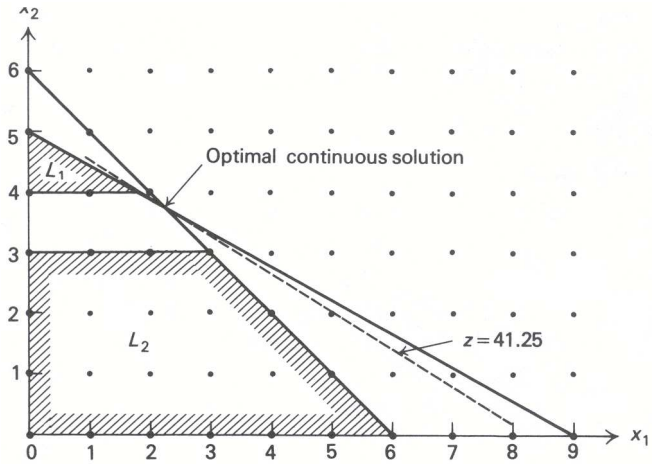


**Figure 9.9** Subdividing the feasible region.

## 9.5 BRANCH-AND-BOUND

*Branch-and-bound* is essentially a strategy of ''divide and conquer.'' The idea is to partition the feasible region into more manageable subdivisions and then, if required, to further partition the subdivisions. In general, there are a number of ways to divide the feasible region, and as a consequence there are a number of branch-and-bound algorithms. We shall consider one such technique, for problems with only binary variables, in Section 9.7. For historical reasons, the technique that will be described next usually is referred to as *the* branch-and-bound procedure.

### Basic Procedure

An integer linear program is a linear program further constrained by the integrality restrictions. Thus, in a maximization problem, the value of the objective function, at the linear-program optimum, will always be an upper bound on the optimal integer-programming objective. In addition, any integer feasible point is always a lower bound on the optimal linear-program objective value.

The idea of branch-and-bound is to utilize these observations to systematically subdivide the linear-programming feasible region and make assessments of the integer-programming problem based upon these subdivisions. The method can be described easily by considering the example from the previous section. At first, the linear-programming region is not subdivided: The integrality restrictions are dropped and the associated linear program is solved, giving an optimal value $z^0$. From our remark above, this gives the upper bound on $z^*$, $z^* \leq z^0 = 41\frac{1}{4}$. Since the coefficients in the objective function are integral, $z^*$ must be integral and this implies that $z^* \leq 41$.

Next note that the linear-programming solution has $x_1 = 2\frac{1}{4}$ and $x_2 = 3\frac{3}{4}$. Both of these variables must be integer in the optimal solution, and we can divide the feasible region in an attempt to *make* either integral. We know that, in any integer programming solution, $x_2$ must be either an integer $\leq 3$ or an integer $\geq 4$. Thus, our first subdivision is into the regions where $x_2 \leq 3$ and $x_2 \geq 4$ as displayed by the shaded regions $L_1$ and $L_2$ in Fig. 9.9. Observe that, by making the subdivisions, we have excluded the old linear-program solution. (If we selected $x_1$ instead, the region would be subdivided with $x_1 \leq 2$ and $x_1 \geq 3$.)

The results up to this point are pictured conveniently in an *enumeration tree* (Fig. 9.10). Here $L_0$ represents the associated linear program, whose optimal solution has been included within the $L_0$ box, and the upper bound on $z^*$ appears to the right of the box. The boxes below correspond to the new subdivisions; the constraints that subdivide $L_0$ are included next to the lines joining the boxes. Thus, the constraints of $L_1$ are those of $L_0$ together with the constraint $x_2 \geq 4$, while the constraints of $L_2$ are those of $L_0$ together with the constraint $x_2 \leq 3$.

The strategy to be pursued now may be apparent: Simply treat each subdivision as we did the original problem. Consider $L_1$ first. Graphically, from Fig. 9.9 we see that the optimal linear-programming solution

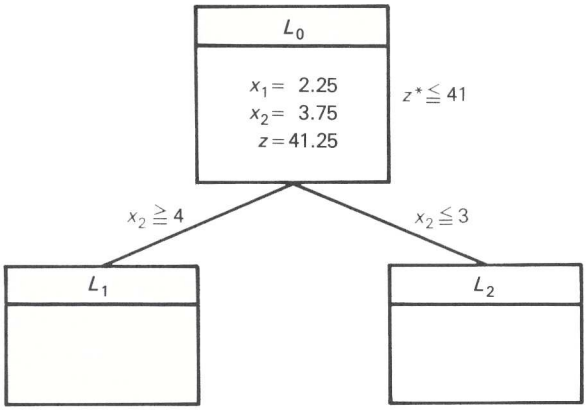


**Figure 9.10** Enumeration tree.

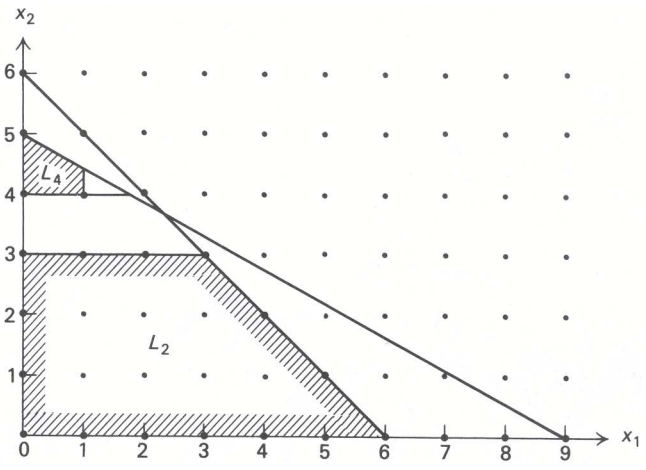


**Figure 9.11** Subdividing the region $L_1$.

lies on the second constraint with $x_2 = 4$, giving $x_1 = \frac{1}{5}(45 - 9(4)) = \frac{9}{5}$ and an objective value $z = 5(\frac{9}{5}) + 8(4) = 41$. Since $x_1$ is not integer, we subdivide $L_1$ further, into the regions $L_3$ with $x_1 \geq 2$ and $L_4$ with $x_1 \leq 1$. $L_3$ is an infeasible problem and so this branch of the enumeration tree no longer needs to be considered.

The enumeration tree now becomes that shown in Fig. 9.12. Note that the constraints of any subdivision are obtained by tracing back to $L_0$. For example, $L_4$ contains the original constraints together with $x_2 \geq 4$ and $x_1 \leq 2$. The asterisk $(*)$ below box $L_3$ indicates that the region need not be subdivided or, equivalently, that the tree will not be extended from this box.

At this point, subdivisions $L_2$ and $L_4$ must be considered. We may select one arbitrarily; however, in practice, a number of useful heuristics are applied to make this choice. For simplicity, let us select the subdivision most recently generated, here $L_4$. Analyzing the region, we find that its optimal solution has

$$x_1 = 1, \qquad x_2 = \tfrac{1}{9}(45 - 5) = \tfrac{40}{9}.$$

Since $x_2$ is not integer, $L_4$ must be further subdivided into $L_5$ with $x_2 \leq 4$, and $L_6$ with $x_2 \geq 5$, leaving $L_2$, $L_5$ and $L_6$ yet to be considered.

Treating $L_5$ first (see Fig. 9.13), we see that its optimum has $x_1 = 1$, $x_2 = 4$, and $z = 37$. Since this is the best linear-programming solution for $L_5$ and the linear program contains every integer solution in $L_5$, no integer point in that subdivision can give a larger objective value than this point. Consequently, other points

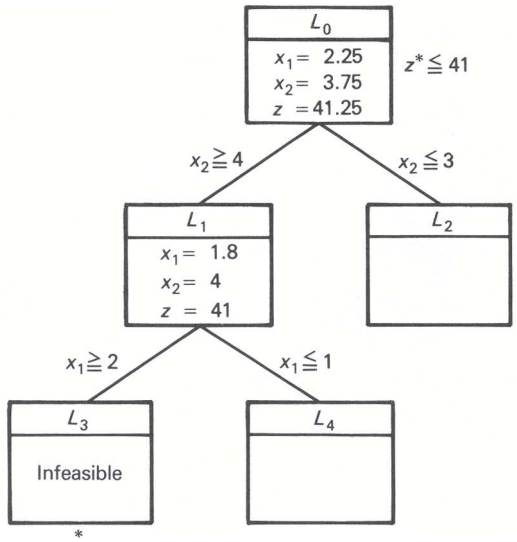


**Figure 9.12**

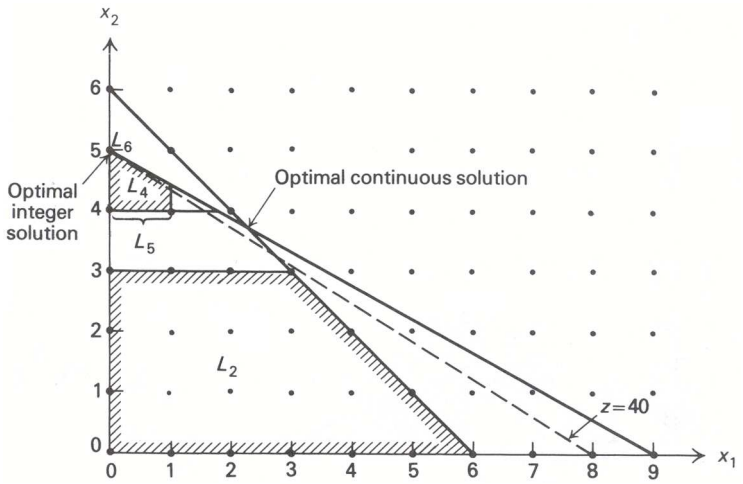


**Figure 9.13** Final subdivisions for the example.

in $L_5$ need never be considered and $L_5$ need not be subdivided further. In fact, since $x_1 = 1, x_2 = 4, z = 37$, is a feasible solution to the original problem, $z^* \geq 37$ and we now have the bounds $37 \leq z^* \leq 41$. Without further analysis, we could terminate with the integer solution $x_1 = 1, x_2 = 4$, knowing that the objective value of this point is within 10 percent of the true optimum. For convenience, the lower bound $z^* \geq 37$ just determined has been appended to the right of the $L_5$ box in the enumeration tree (Fig. 9.14).

Although $x_1 = 1,\ x_2 = 4$ is the best integer point in $L_5$, the regions $L_2$ and $L_6$ might contain better feasible solutions, and we must continue the procedure by analyzing these regions. In $L_6$, the only feasible point is $x_1 = 0, x_2 = 5$, giving an objective value $z = +40$. This is better than the previous integer point and thus the lower bound on $z^*$ improves, so that $40 \leq z^* \leq 41$. We could terminate with this integer solution knowing that it is within 2.5 percent of the true optimum. However, $L_2$ *could* contain an even better integer solution.

The linear-programming solution in $L_2$ has $x_1 = x_2 = 3$ and $z = 39$. This is the best integer point in $L_2$ but is not as good as $x_1 = 0,\ x_2 = 5$, so the later point (in $L_6$) must indeed be optimal. It is interesting to note that, even if the solution to $L_2$ did not give $x_1$ and $x_2$ integer, but had $z < 40$, then no feasible (and, in particular, no integer point) in $L_2$ could be as good as $x_1 = 0,\ x_2 = 5$, with $z = 40$. Thus, again $x_1 = 0,\ x_2 = 5$ would be known to be optimal. This observation has important computational implications,

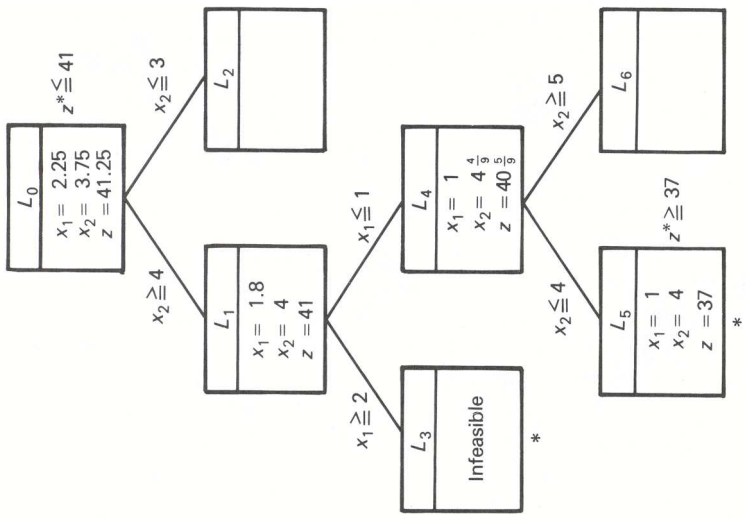


**Figure 9.14**

since it is not necessary to drive every branch in the enumeration tree to an integer or infeasible solution, but only to an objective value below the best integer solution.

The problem now is solved and the entire solution procedure can be summarized by the enumeration tree in Fig. 9.15.

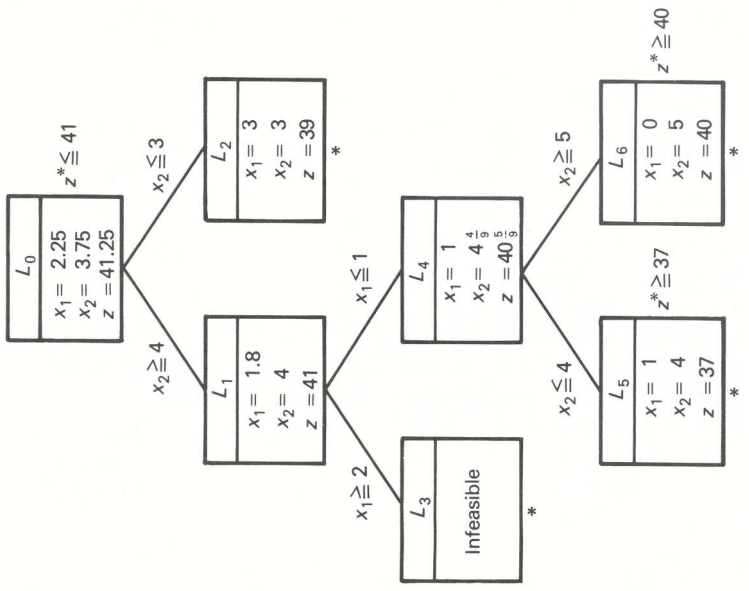


**Figure 9.15**

#### Further Considerations

There are three points that have yet to be considered with respect to the branch-and-bound procedure:

i) Can the linear programs corresponding to the subdivisions be solved efficiently?
ii) What is the best way to subdivide a given region, and which unanalyzed subdivision should be considered next?

iii) Can the upper bound ($z = 41$, in the example) on the optimal value $z^*$ of the integer program be improved while the problem is being solved?

The answer to the first question is an unqualified *yes*. When moving from a region to one of its subdivisions, we add one constraint that is not satisfied by the optimal linear-programming solution over the parent region. Moreover, this was one motivation for the dual simplex algorithm, and it is natural to adopt that algorithm here.

Referring to the sample problem will illustrate the method. The first two subdivisions $L_1$ and $L_2$ in that example were generated by adding the following constraints to the original problem:

$$\begin{array}{lllll} \text{For subdivision 1:} & x_2 \geq 4 & \text{or} & x_2 - s_3 = 4 & (s_3 \geq 0); \\ \text{For subdivision 2:} & x_2 \leq 3 & \text{or} & x_2 + s_4 = 3 & (s_4 \geq 0). \end{array}$$

In either case we add the new constraint to the optimal linear-programming tableau. For subdivision 1, this gives:

$$\begin{array}{rrl} (-z) & -\frac{5}{4}s_1 - \frac{3}{4}s_2 = -41\frac{1}{4} & \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} \text{Constraints from the optimal canonical form} \\ x_1 & +\frac{9}{4}s_1 - \frac{1}{4}s_2 = \frac{9}{4} & \\ \textcircled{x_2} & -\frac{5}{4}s_1 + \frac{1}{4}s_2 = \frac{15}{4} & \end{array}$$

$$-x_2 + s_3 = -4, \qquad \text{Added constraint}$$

$$x_1,\ x_2,\ s_1,\ s_2,\ s_3 \geq 0,$$

where $s_1$ and $s_2$ are slack variables for the two constraints in the original problem formulation. Note that the new constraint has been multiplied by $-1$, so that the slack variable $s_3$ can be used as a basic variable. Since the basic variable $x_2$ appears with a nonzero coefficient in the new constraint, though, we must pivot to isolate this variable in the second constraint to re-express the system as:

$$\begin{array}{rrrrrrrrr} (-z) & & -\frac{5}{4}s_1 & -\frac{3}{4}s_2 & & = & -41\frac{1}{4}, & & \\ & x_1 & & & +\frac{9}{4}s_1 & -\frac{1}{4}s_2 & & = & \frac{9}{4}, \\ & x_2 & -\frac{5}{4}s_1 & +\frac{1}{4}s_2 & & = & \frac{15}{4}, & & \\ & & \textcircled{-\frac{5}{4}s_1} & +\frac{1}{4}s_2 & +s_3 & = & -\frac{1}{4}, & & \end{array}$$

$$x_1, x_2, s_1, s_2, s_3 \geq 0.$$

These constraints are expressed in the proper form for applying the dual simplex algorithm, which will pivot next to make $s_1$ the basic variable in the third constraint. The resulting system is given by:

$$\begin{array}{rrrrrrr} (-z) & & - s_2 & - s_3 & = & -41, \\ & x_1 & +\frac{1}{5}s_2 & +\frac{9}{5}s_3 & = & \frac{9}{5}, \\ & x_2 & & - s_3 & = & 4, \\ & s_1 & -\frac{1}{5}s_2 & -\frac{4}{5}s_3 & = & \frac{1}{5}, \end{array}$$

$$x_1,\ x_2,\ s_1,\ s_2,\ s_3 \geq 0.$$

This tableau is optimal and gives the optimal linear-programming solution over the region $L_1$ as $x_1 = \frac{9}{5}$, $x_2 = 4$, and $z = 41$. The same procedure can be used to determine the optimal solution in $L_2$.

When the linear-programming problem contains many constraints, this approach for recovering an optimal solution is very effective. After adding a new constraint and making the slack variable for that constraint basic, we always have a starting solution for the dual-simplex algorithm with only one basic variable negative. Usually, only a few dual-simplex pivoting operations are required to obtain the optimal solution. Using the primal-simplex algorithm generally would require many more computations.

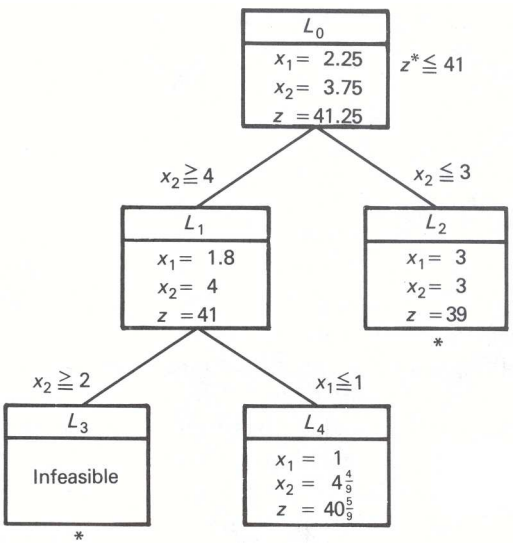


**Figure 9.16**

Issue (ii) raised above is very important since, if we can make our choice of subdivisions in such a way as to rapidly obtain a good (with luck, near-optimal) integer solution $\hat{z}$, then we can eliminate many potential subdivisions immediately. Indeed, if any region has its linear programming value $z \leq \hat{z}$, then the objective value of no integer point in that region can exceed $\hat{z}$ and the region need not be subdivided. There is no universal method for making the required choice, although several heuristic procedures have been suggested, such as selecting the subdivision with the largest optimal linear-programming value.†

Rules for determining which fractional variables to use in constructing subdivisions are more subtle. Recall that any fractional variable can be used to generate a subdivision. One procedure utilized is to look ahead one step in the dual-simplex method for every possible subdivision to see which is most promising. The details are somewhat involved and are omitted here. For expository purposes, we have selected the fractional variable arbitrarily.

Finally, the upper bound $\bar{z}$ on the value $z^*$ of the integer program can be improved as we solve the problem. Suppose for example, that subdivision $L_2$ was analyzed before subdivisions $L_5$ or $L_6$ in our sample problem. The enumeration tree would be as shown in Fig. 9.16.

At this point, the optimal solution must lie in either $L_2$ or $L_4$. Since, however, the largest value for any feasible point in either of these regions is $40\frac{5}{9}$, the optimal value for the problem $z^*$ cannot *exceed* $40\frac{5}{9}$. Because $z^*$ must be integral, this implies that $z^* \leq 40$ and the upper bound has been improved from the value 41 provided by the solution to the linear program on $L_0$. In general, the upper bound is given in this way as the largest value of any ''hanging'' box (one that has not been divided) in the enumeration tree.

#### Summary

The essential idea of branch-and-bound is to subdivide the feasible region to develop *bounds* $\underline{z} < z^* < \bar{z}$ on $z^*$. For a maximization problem, the lower bound $\underline{z}$ is the highest value of any feasible integer point encountered. The upper bound is given by the optimal value of the associated linear program or by the largest value for the objective function at any ''hanging'' box. After considering a subdivision, we must *branch* to (move to) another subdivision and analyze it. Also, if *either*

† One common method used in practice is to consider subdivisions on a last-generated–first-analyzed basis. We used this rule in our previous example. Note that data to initiate the dual-simplex method mentioned above must be stored for each subdivision that has yet to be analyzed. This data usually is stored in a list, with new information being added to the top of the list. When required, data then is extracted from the *top* of this list, leading to the last-generated–first-analyzed rule. Observe that when we subdivide a region into two subdivisions, one of these subdivisions will be analyzed next. The data required for this analysis already will be in the computer core and need not be extracted from the list.

i) the linear program over $L_j$ is infeasible;

ii) the optimal linear-programming solution over $L_j$ is integer; *or*

iii) the value of the linear-programming solution $z^j$ over $L_j$ satisfies $z^j \leq \underline{z}$ (if maximizing),

then $L_j$ need not be subdivided. In these cases, integer-programming terminology says that $L_j$ has been *fathomed*.† Case (i) is termed fathoming by infeasibility, (ii) fathoming by integrality, and (iii) fathoming by bounds.

The flow chart in Fig. 9.17 summarizes the general procedure.

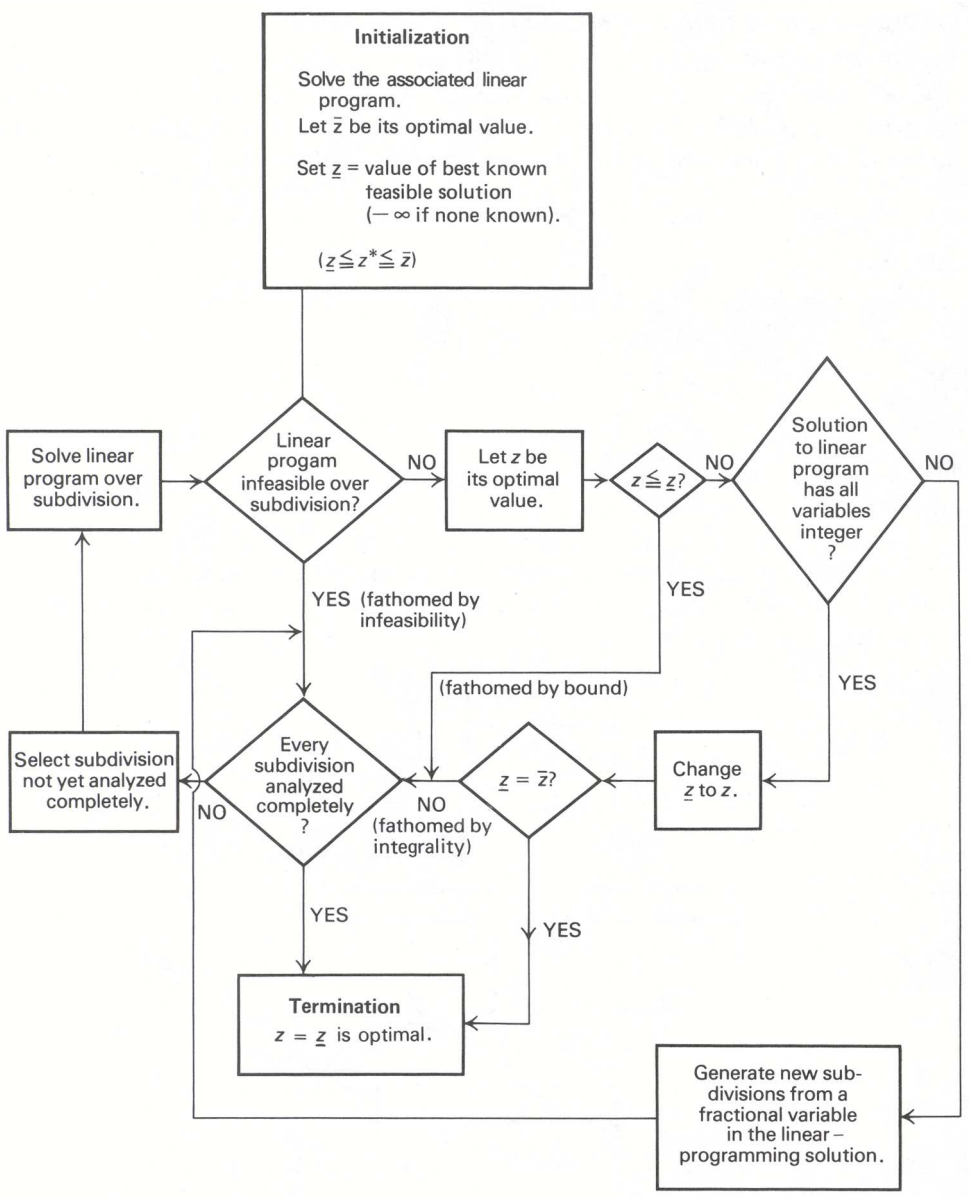


**Figure 9.17** Branch-and-bound for integer-programming maximization.

† To *fathom* is defined as ''to get to the bottom of; to understand thoroughly.'' In this chapter, *fathomed* might be more appropriately defined as ''understood enough or already considered.''

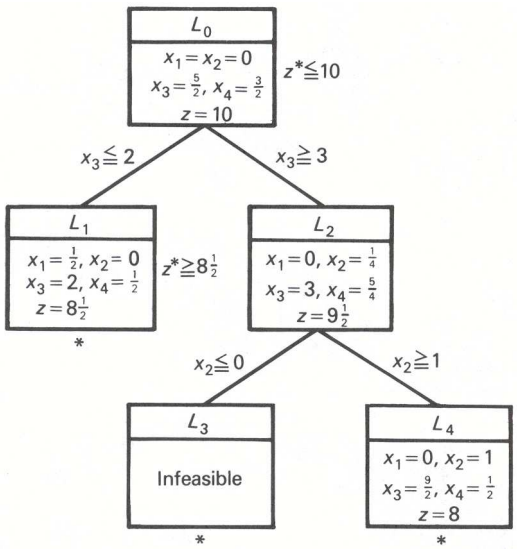


**Figure 9.18**

### 9.6 BRANCH-AND-BOUND FOR MIXED-INTEGER PROGRAMS

The branch-and-bound approach just described is easily extended to solve problems in which some, but not all, variables are constrained to be integral. Subdivisions then are generated solely by the integral variables. In every other way, the procedure is the same as that specified above. A brief example will illustrate the method.

$$z^* = \max z = -3x_1 - 2x_2 + 10,$$

subject to:

$$\begin{aligned} x_1 - 2x_2 + x_3 &= \tfrac{5}{2}, \\ 2x_1 + x_2 + x_4 &= \tfrac{3}{2}, \\ x_j \geq 0 \quad (j &= 1, 2, 3, 4), \end{aligned}$$

$$x_2 \quad \text{and} \quad x_3 \quad \text{integer.}$$

The problem, as stated, is in canonical form, with $x_3$ and $x_4$ optimal basic variables for the associated linear program.

The continuous variable $x_4$ cannot be used to generate subdivisions since any value of $x_4 \geq 0$ potentially can be optimal. Consequently, the subdivisions must be defined by $x_3 \leq 2$ and $x_3 \geq 3$. The complete procedure is summarized by the enumeration tree in Fig. 9.18.

The solution in $L_1$ satisfies the integrality restrictions, so $z^* \geq z = 8\frac{1}{2}$. The only integral variable with a fractional value in the optimal solution of $L_2$ is $x_2$, so subdivisions $L_3$ and $L_4$ are generated from this variable. Finally, the optimal linear-programming value of $L_4$ is 8, so no feasible mixed-integer solution in that region can be better than the value $8\frac{1}{2}$ already generated. Consequently, that region need not be subdivided and the solution in $L_1$ is optimal.

The dual-simplex iterations that solve the linear programs in $L_1, L_2, L_3$, and $L_4$ are given below in Tableau 1. The variables $s_j$ in the tableaus are the slack variables for the
constraints added to generate the subdivisions. The coefficients in the appended constraints are determined as we mentioned in the last section, by eliminating the basic variables $x_j$ from the new constraint that is introduced. To follow the iterations, recall that in the dual-simplex method, pivots are made on negative elements in the generating row; if all elements in this row are *positive*, as in region $L_3$, then the problem is infeasible.

**Tableau 1**

$L_1$

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_1$ |
|---|---|---|---|---|---|---|
| $(-z)$ | $-10$ | $-3$ | $-2$ | | | |
| $x_3$ | $\frac{5}{2}$ | $1$ | $-2$ | $1$ | | |
| $x_4$ | $\frac{3}{2}$ | $2$ | $1$ | | $1$ | |
| $s_1$ | $-\frac{1}{2}$ | $\textcircled{-1}$ | $2$ | | | $1$ |

(i.e., $x_3 \leqq 2$)

↑

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_1$ |
|---|---|---|---|---|---|---|
| $(-z)$ | $-8\frac{1}{2}$ | | $-8$ | | | $-3$ |
| $x_3$ | $2$ | | $0$ | $1$ | | $1$ |
| $x_4$ | $\frac{1}{2}$ | | $5$ | | $1$ | $2$ |
| $x_1$ | $\frac{1}{2}$ | $1$ | $-2$ | | | $-1$ |

$L_2$

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_2$ |
|---|---|---|---|---|---|---|
| $(-z)$ | $-10$ | $-3$ | $-2$ | | | |
| $x_3$ | $\frac{5}{2}$ | $1$ | $-2$ | $1$ | | |
| $x_4$ | $\frac{3}{2}$ | $2$ | $1$ | | $1$ | |
| $s_2$ | $-\frac{1}{2}$ | $1$ | $\textcircled{-2}$ | | | $1$ |

(i.e., $x_3 \geqq 3$)

↑

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_2$ |
|---|---|---|---|---|---|---|
| $(-z)$ | $-9\frac{1}{2}$ | $-4$ | | | | $-1$ |
| $x_3$ | $3$ | $0$ | | $1$ | | $-1$ |
| $x_4$ | $\frac{5}{4}$ | $\frac{5}{2}$ | | | $1$ | $\frac{1}{2}$ |
| $x_2$ | $\frac{1}{4}$ | $-\frac{1}{2}$ | $1$ | | | $-\frac{1}{2}$ |

## 9.7 IMPLICIT ENUMERATION

A special branch-and-bound procedure can be given for integer programs with only binary variables. The algorithm has the advantage that it requires no linear-programming solutions. It is illustrated by the following example:

$$z^* = \max z = -8x_1 - 2x_2 - 4x_3 - 7x_4 - 5x_5 + 10,$$

subject to:

$$-3x_1 - 3x_2 + x_3 + 2x_4 + 3x_5 \leq -2,$$
$$-5x_1 - 3x_2 - 2x_3 - x_4 + x_5 \leq -4,$$
$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, 5).$$

One way to solve such problems is complete enumeration. List all possible binary combinations of the variables and select the best such point that is feasible. The approach works very well on a small problem such as this, where there are only a few potential 0–1 combinations for the variables, here 32. In general, though, an $n$-variable problem contains $2^n$ 0–1 combinations; for large values of $n$, the exhaustive approach is prohibitive. Instead, one might implicitly consider every binary combination, just as every integer point was implicitly *considered*, but not necessarily evaluated, for the general problem via branch-and-bound.

Recall that in the ordinary branch-and-bound procedure, subdivisions were analyzed by maintaining the linear constraints and dropping the integrality restrictions. Here, we adopt the opposite tactic of always

**Tableau 1** (*Continued*)

$L_3$

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_2$ | $s_3$ | |
|---|---|---|---|---|---|---|---|---|
| $(-z)$ | $-9\frac{1}{2}$ | $-4$ | | | | $-1$ | | |
| $x_3$ | 3 | 0 | | 1 | | $-1$ | | |
| $x_4$ | $\frac{5}{4}$ | $\frac{5}{2}$ | | | 1 | $\frac{1}{2}$ | | |
| $x_2$ | $\frac{1}{4}$ | $-\frac{1}{2}$ | 1 | | | $-\frac{1}{2}$ | | |
| $s_3$ | $-\frac{1}{4}$ | $\frac{1}{2}$ | | | | $\frac{1}{2}$ | 1 | (i.e., $x_2 \leqq 0$) |

No pivot possible, problem infeasible

$L_4$

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_2$ | $s_4$ | |
|---|---|---|---|---|---|---|---|---|
| $(-z)$ | $-9\frac{1}{2}$ | $-4$ | | | | $-1$ | | |
| $x_3$ | 3 | 0 | | 1 | | $-1$ | | |
| $x_4$ | $\frac{5}{4}$ | $\frac{5}{2}$ | | | 1 | $\frac{1}{2}$ | | |
| $x_2$ | $\frac{1}{4}$ | $-\frac{1}{2}$ | 1 | | | $-\frac{1}{2}$ | | |
| $s_4$ | $-\frac{3}{4}$ | $-\frac{1}{2}$ | | | | $\left(-\frac{1}{2}\right)$ | 1 | (i.e., $x_2 \geqq 1$) |
| | | | | | | ↑ | | |

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $s_2$ | $s_4$ |
|---|---|---|---|---|---|---|---|
| $(-z)$ | $-8$ | $-3$ | | | | | $-2$ |
| $x_3$ | $\frac{9}{2}$ | 1 | | 1 | | | $-2$ |
| $x_4$ | $\frac{1}{2}$ | 2 | | | 1 | | 1 |
| $x_2$ | 1 | 0 | 1 | | | | $-1$ |
| $s_2$ | $\frac{3}{2}$ | 1 | | | | 1 | $-2$ |

maintaining the 0–1 restrictions, but ignoring the linear inequalities.

The idea is to utilize a branch-and-bound (or subdivision) process to fix some of the variables at 0 or 1. The variables remaining to be specified are called *free variables*. Note that, if the inequality constraints are ignored, the objective function is maximized by setting the free variables to zero, since their objective-function coefficients are negative. For example, if $x_1$ and $x_4$ are fixed at 1 and $x_5$ at 0, then the free variables are $x_2$ and $x_3$. Ignoring the inequality constraints, the resulting problem is:

$$\max [-8(1) - 2x_2 - 4x_3 - 7(1) - 5(0) + 10] = \max [-2x_2 - 4x_3 - 5],$$

subject to:

$$x_2 \quad \text{and} \quad x_3 \text{ binary.}$$

Since the free variables have negative objective-function coefficients, the maximization sets $x_2 = x_3 = 0$. The simplicity of this trivial optimization, as compared to a more formidable linear program, is what we would like to exploit.

Returning to the example, we start with *no* fixed variables, and consequently every variable is free and set to zero. The solution does not satisfy the inequality constraints, and we must subdivide to search for feasible solutions. One subdivision choice might be:

For subdivision 1 : $x_1 = 1$,

For subdivision 2 : $x_1 = 0$.

Now variable $x_1$ is fixed in each subdivision. By our observations above, if the inequalities are ignored, the optimal solution over each subdivision has $x_2 = x_3 = x_4 = x_5 = 0$. The resulting solution in subdivision 1 gives

$$z = -8(1) - 2(0) - 4(0) - 7(0) - 5(0) + 10 = 2,$$

and happens to satisfy the inequalities, so that the optimal solution to the original problem is *at least* 2, $z^* \geq 2$. Also, subdivision 1 has been fathomed: The above solution is best among all 0–1 combinations with $x_1 = 1$; thus it must be best among those satisfying the inequalities. No other feasible 0–1 combination in subdivision 1 needs to be evaluated explicitly. These combinations have been considered implicitly.

The solution with $x_2 = x_3 = x_4 = x_5 = 0$ in subdivision 2 is the same as the original solution with every variable at zero, and is infeasible. Consequently, the region must be subdivided further, say with $x_2 = 1$ or $x_2 = 0$, giving:

For subdivision 3 : $x_1 = 0, x_2 = 1;$
For subdivision 4 : $x_1 = 0, x_2 = 0.$

The enumeration tree to this point is as given in Fig. 9.19.

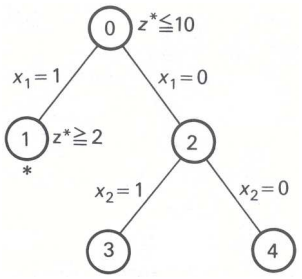


**Figure 9.19**

Observe that this tree differs from the enumeration trees of the previous sections. For the earlier procedures, the linear-programming solution used to analyze each subdivision was specified explicitly in a box. Here the 0–1 solution (ignoring the inequalities) used to analyze subdivisions is not stated explicitly, since it is known simply by setting free variables to zero. In subdivision ③, for example, $x_1 = 0$ and $x_2 = 1$ are fixed, and the free variables $x_3$, $x_4$ and$x_5$ are set to zero.

Continuing to fix variables and subdivide in this fashion produces the complete tree shown in Fig. 9.20.

The tree is not extended after analyzing subdivisions 4, 5, 7, 9, and 10, for the following reasons.

i) At ⑤, the solution $x_1 = 0,\ x_2 = x_3 = 1$ , with free variables $x_4 = x_5 = 0$, is feasible, with $z = 4$ , thus providing an improved lower bound on $z^*$.
ii) At ⑦, the solution $x_1 = x_3 = 0,\ x_2 = x_4 = 1$, and free variable $x_5 = 0$, has $z = 1 < 4$, so that no solution in that subdivision can be as good as that generated at ⑤.
iii) At ⑨ and ⑩ , every free variable is fixed. In each case, the subdivisions contain only a single point, which is infeasible, and further subdivision is not possible.
iv) At ④, the second inequality (with fixed variables $x_1 = x_2 = 0$) reads:

$$-2x_3 - x_4 + x_5 \leq -4.$$

No 0–1 values of $x_3,\ x_4$, or $x_5$ ''completing'' the fixed variables $x_1 = x_2 = 0$ satisfy this constraint, since the lowest value for the lefthand side of this equation is $-3$ when $x_3 = x_4 = 1$ and $x_5 = 0$. The subdivision then has no feasible solution and need not be analyzed further.

The last observation is completely general. If, at any point after substituting for the fixed variables, the sum of the remaining negative coefficients in any constraint exceeds the righthand side, then the region defined by these fixed variables has no feasible solution. Due to the special nature of the 0–1 problem, there are a number of other such tests that can be utilized to reduce the number of subdivisions generated. The efficiency of these tests is measured by weighing the time needed to perform them against the time saved by fewer subdivisions.

The techniques used here apply to any integer-programming problem involving only binary variables, so that implicit enumeration is an alternative branch-and-bound procedure for this class of problems. In this case, subdivisions are fathomed if any of three conditions hold:

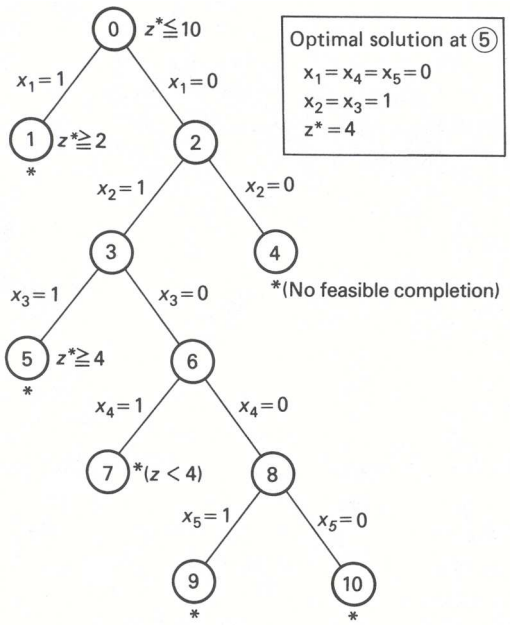


**Figure 9.20**

i) the integer program is known to be infeasible over the subdivision, for example, by the above infeasibility test;
ii) the 0–1 solution obtained by setting free variables to zero satisfies the linear inequalities; or
iii) the objective value obtained by setting free variables to zero is no larger than the best feasible 0–1 solution previously generated.

These conditions correspond to the three stated earlier for fathoming in the usual branch-and-bound procedure. If a region is not fathomed by one of these tests, implicit enumeration subdivides that region by selecting any free variable and fixing its values to 0 or 1.

Our arguments leading to the algorithm were based upon stating the original 0–1 problem in the following standard form:

1. the objective is a maximization with all coefficients negative; and
2. constraints are specified as ''less than or equal to'' inequalities.

As usual, minimization problems are transformed to maximization by multiplying cost coefficients by $-1$. If $x_j$ appears in the maximization form with a positive coefficient, then the variable substitution $x_j = 1 - x'_j$ everywhere in the model leaves the binary variable $x'_j$ with a negative objective-function coefficient. Finally, ''greater than or equal to'' constraints can be multiplied by $-1$ to become ''less than or equal to'' constraints; and general equality constraints are converted to inequalities by the special technique discussed in Exercise 17 of Chapter 2.

Like the branch-and-bound procedure for general integer programs, the way we choose to subdivide regions can have a profound effect upon computations. In implicit enumeration, we begin with the zero solution $x_1 = x_2 = \cdots = x_n = 0$ and generate other solutions by setting variables to 1. One natural approach is to subdivide based upon the variable with highest objective contribution. For the sample problem, this would imply subdividing initially with $x_2 = 1$ or $x_2 = 0$.

Another approach often used in practice is to try to drive toward feasibility as soon as possible. For instance, when $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$ are fixed in the example problem, we could subdivide based upon either $x_4$ or $x_5$. Setting $x_4$ or $x_5$ to 1 and substituting for the fixed variables, we find that the constraints become:

$$
\begin{array}{ll}
x_4 = 1, \quad x_5(\text{free}) = 0: & x_5 = 1, \quad x_4(\text{free}) = 0: \\
-3(0) - 3(1) + (0) + 2(1) + 3(0) \le -2, & -3(0) - 3(1) + (0) + 2(0) + 3(1) \le -2, \\
-5(0) - 3(1) - 2(0) - 1(1) + (0) \le -4, & -5(0) - 3(1) - 2(0) - 1(0) + (1) \le -4.
\end{array}
$$

For $x_4 = 1$, the first constraint is infeasible by 1 unit and the second constraint is feasible, giving 1 total unit of infeasibility. For $x_5 = 1$, the first constraint is infeasible by 2 units and the second by 2 units, giving 4 total units of infeasibility. Thus $x_4 = 1$ appears more favorable, and we would subdivide based upon that variable. In general, the variable giving the *least total infeasibilities* by this approach would be chosen next. Reviewing the example problem the reader will see that this approach has been used in our solution.

## 9.8 CUTTING PLANES

The cutting-plane algorithm solves integer programs by modifying linear-programming solutions until the integer solution is obtained. It does not partition the feasible region into subdivisions, as in branch-and-bound approaches, but instead works with a single linear program, which it refines by adding new constraints. The new constraints successively reduce the feasible region until an integer optimal solution is found.

In practice, the branch-and-bound procedures almost always outperform the cutting-plane algorithm. Nevertheless, the algorithm has been important to the evolution of integer programming. Historically, it was the first algorithm developed for integer programming that could be proved to converge in a finite number of steps. In addition, even though the algorithm generally is considered to be very inefficient, it has provided insights into integer programming that have led to other, more efficient, algorithms.

Again, we shall discuss the method by considering the sample problem of the previous sections:

$$z^* = \max 5x_1 + 8x_2,$$

subject to:

$$
\begin{array}{l}
x_1 + x_2 + s_1 = 6, \\
5x_1 + 9x_2 + s_2 = 45, \\
x_1, x_2, s_1, s_2 \ge 0.
\end{array}
\tag{11}
$$

$s_1$ and $s_2$ are, respectively, slack variables for the first and second constraints.

Solving the problem by the simplex method produces the following optimal tableau:

$$
\begin{array}{rrrrrr}
(-z) & & -\frac{5}{4}s_1 & -\frac{3}{4}s_2 & = & -41\frac{1}{4}, \\
x_1 & & +\frac{9}{4}s_1 & -\frac{1}{4}s_2 & = & \frac{9}{4}, \\
x_2 & & -\frac{5}{4}s_1 & +\frac{1}{4}s_2 & = & \frac{15}{4},
\end{array}
$$

$$x_1, x_2, s_1, s_2, s_3 \ge 0.$$

Let us rewrite these equations in an equivalent but somewhat altered form:

$$
\begin{array}{rrrrrrrrr}
(-z) & & -2s_1 & -s_2 & +42 & = & \frac{3}{4} & -\frac{3}{4}s_1 & -\frac{1}{4}s_2, \\
x_1 & & +2s_1 & -s_2 & -2 & = & \frac{1}{4} & -\frac{1}{4}s_1 & -\frac{3}{4}s_2, \\
x_2 & -2s_1 & & & -3 & = & \frac{3}{4} & -\frac{3}{4}s_1 & -\frac{1}{4}s_2,
\end{array}
$$

$$x_1, x_2, s_1, s_2 \ge 0.$$

These algebraic manipulations have isolated integer coefficients to one side of the equalities and fractions to the other, in such a way that the constant terms on the righthand side are all nonnegative and the slack variable coefficients on the righthand side are all nonpositive.

In any integer solution, the lefthand side of each equation in the last tableau must be integer. Since $s_1$ and $s_2$ are nonnegative and appear to the right with negative coefficients, each righthand side necessarily must be less than or equal to the fractional constant term. Taken together, these two observations show that both sides of every equation must be an integer less than or equal to zero (if an integer is less than or equal to a fraction, it necessarily must be 0 or negative). Thus, from the first equation, we may write:

$$\tfrac{3}{4} - \tfrac{3}{4}s_1 - \tfrac{1}{4}s_2 \leq 0 \quad \text{and} \quad \text{integer},$$

or, introducing a slack variable $s_3$,

$$\tfrac{3}{4} - \tfrac{3}{4}s_1 - \tfrac{1}{4}s_2 + s_3 = 0, \qquad s_3 \geq 0 \quad \text{and} \quad \text{integer}. \tag{$C_1$}$$

Similarly, other conditions can be generated from the remaining constraints:

$$\tfrac{1}{4} - \tfrac{1}{4}s_1 - \tfrac{3}{4}s_2 + s_4 = 0, \qquad s_4 \geq 0 \quad \text{and} \quad \text{integer} \tag{$C_2$}$$

$$\tfrac{3}{4} - \tfrac{3}{4}s_1 - \tfrac{1}{4}s_2 + s_5 = 0, \qquad s_5 \geq 0 \quad \text{and} \quad \text{integer}. \tag{$C_3$}$$

Note that, in this case, $(C_1)$ and $(C_3)$ are identical.

The new equations $(C_1)$, $(C_2)$, and $(C_3)$ that have been derived are called *cuts* for the following reason: Their derivation did not exclude any integer solutions to the problem, so that any integer feasible point to the original problem must satisfy the cut constraints. The linear-programming solution had $s_1 = s_2 = 0$; clearly, these do *not* satisfy the cut constraints. In each case, substituting $s_1 = s_2 = 0$ gives either $s_3$, $s_4$, or $s_5 < 0$. Thus the net effect of a cut is to cut away the optimal linear-programming solution from the feasible region without excluding any feasible integer points.

The geometry underlying the cuts can be established quite easily. Recall from (11) that slack variables $s_1$ and $s_2$ are defined by:

$$\begin{aligned} s_1 &= 6 - x_1 - x_2, \\ s_2 &= 45 - 5x_1 - 9x_2. \end{aligned}$$

Substituting these values in the cut constraints and rearranging, we may rewrite the cuts as:

$$2x_1 + 3x_2 \leq 15, \tag{$C_1$ or $C_3$}$$

$$4x_1 + 7x_2 \leq 35. \tag{$C_2$}$$

In this form, the cuts are displayed in Fig. 9.21. Notethat they exhibit the features suggested above. In each case, the added cut removes the linear-programming solution $x_1 = \frac{9}{4},\ x_2 = \frac{15}{4}$, from the feasible region, at the same time including every feasible integer solution.

The basic strategy of the cutting-plane technique is to add cuts (usually only one) to the constraints defining the feasible region and then to solve the resulting linear program. If the optimal values for the decision variables in the linear program are all integer, they are optimal; otherwise, a new cut is derived from the new optimal linear-programming tableau and appended to the constraints.

Note from Fig. 9.21 that the cut $C_1 = C_3$ leads directly to the optimal solution. Cut $C_2$ does not, and further iterations will be required if this cut is appended to the problem (without the cut $C_1 = C_3$). Also note that $C_1$ cuts deeper into the feasible region than does $C_2$. For problems with many variables, it is generally quite difficult to determine which cuts will be deep in this sense. Consequently, in applications, the algorithm frequently generates cuts that shave very little from the feasible region, and hence the algorithm's poor performance.

A final point to be considered here is the way in which cuts are generated. The linear-programming tableau for the above problem contained the constraint:

$$x_1 + \tfrac{9}{4}s_1 - \tfrac{1}{4}s_2 = \tfrac{9}{4}.$$

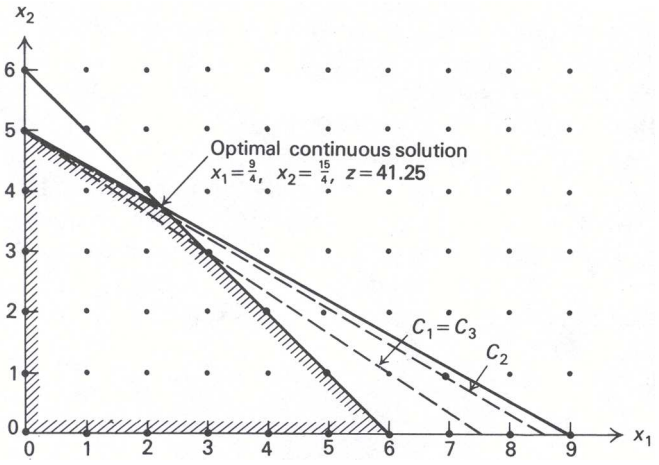


**Figure 9.21** Cutting away the linear-programming solution.

Suppose that we round *down* the fractional coefficients to integers, that is, $\frac{9}{4}$ to 2, $-\frac{1}{4}$ to $-1$, and $\frac{9}{4}$ to 2. Writing these integers to the left of the equality and the remaining fractions to the right, we obtain as before, the equivalent constraint:

$$x_1 + 2s_1 - s_2 - 2 = \tfrac{1}{4} - \tfrac{1}{4}s_1 - \tfrac{3}{4}s_2.$$

By our previous arguments, the cut is:

$$\tfrac{1}{4} - \tfrac{1}{4}s_1 - \tfrac{3}{4}s_2 \leq 0 \quad \text{and integer.}$$

Another example may help to clarify matters. Suppose that the final linear-programming tableau to a problem has the constraint

$$x_1 \qquad + \tfrac{1}{6}x_6 - \tfrac{7}{6}x_7 + 3x_8 \quad = 4\tfrac{5}{6}.$$

Then the equivalent constraint is:

$$x_1 \qquad\qquad - 2x_7 + 3x_8 - 4 = \tfrac{5}{6} - \tfrac{1}{6}x_6 - \tfrac{5}{6}x_7,$$

and the resulting cut is:

$$\tfrac{5}{6} - \tfrac{1}{6}x_6 - \tfrac{5}{6}x_7 \leq 0 \quad \text{and integer.}$$

Observe the way that fractions are determined for negative coefficients. The fraction in the cut constraint determined by the $x_7$ coefficient $-\frac{7}{6} = -1\frac{1}{6}$ is *not* $\frac{1}{6}$, but rather it is the fraction generated by rounding down to $-2$; i.e., the fraction is $-1\frac{1}{6} - (-2) = \frac{5}{6}$.

Tableau 2 shows the complete solution of the sample problem by the cutting-plane technique. Since cut $C_1 = C_3$ leads directly to the optimal solution, we have chosen to start with cut $C_2$. Note that, if the slack variable for any newly generated cut is taken as the basic variable in that constraint, then the problem is in the proper form for the dual-simplex algorithm. For instance, the cut in Tableau 2(b) generated from the $x_1$ constraint

$$x_1 + \tfrac{7}{3}s_1 - \tfrac{1}{3}s_2 = \tfrac{7}{3} \quad \text{or} \quad x_1 + 2s_1 - s_2 - 2 = \tfrac{1}{3} - \tfrac{1}{3}s_1 - \tfrac{2}{3}s_2$$

is given by:

$$\tfrac{1}{3} - \tfrac{1}{3}s_1 - \tfrac{2}{3}s_2 \leq 0 \quad \text{and} \quad \text{integer.}$$

Letting $s_4$ be the slack variable in the constraint, we obtain:

$$-\tfrac{1}{3}s_1 - \tfrac{2}{3}s_2 + s_4 = -\tfrac{1}{3}.$$

**Tableau 2** Dual Simplex Iterations for the Cutting-Plane Algorithm.

(a)

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|---|---|---|
| $(-z)$ | $-41\frac{1}{4}$ | | | $-\frac{5}{4}$ | $-\frac{3}{4}$ | |
| $x_1$ | $\frac{9}{4}$ | 1 | | $\frac{9}{4}$ | $-\frac{1}{4}$ | |
| $x_2$ | $\frac{15}{4}$ | | 1 | $-\frac{5}{4}$ | $\frac{1}{4}$ | |
| ← $s_3$ | $-\frac{1}{4}$ | | | $-\frac{1}{4}$ | $(-\frac{3}{4})$ ↑ | 1 |

(cut generated from $x_1$ constraint)

(b)

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|---|---|---|
| $(-z)$ | $-41$ | | | $-1$ | | $-1$ | |
| $x_1$ | $\frac{7}{3}$ | 1 | | $\frac{7}{3}$ | | $-\frac{1}{3}$ | |
| $x_2$ | $\frac{11}{3}$ | | 1 | $-\frac{4}{3}$ | | $\frac{1}{3}$ | |
| $s_2$ | $\frac{1}{3}$ | | | $\frac{1}{3}$ | 1 | $-\frac{4}{3}$ | |
| ← $s_4$ | $-\frac{1}{3}$ | | | $-\frac{1}{3}$ | | $(-\frac{2}{3})$ ↑ | 1 |

(cut generated from $x_1$ constraint)

(c)

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|---|---|---|
| $(-z)$ | $-40\frac{1}{2}$ | | | $-\frac{1}{2}$ | | | $-\frac{3}{2}$ | |
| $x_1$ | $\frac{5}{2}$ | 1 | | $\frac{5}{2}$ | | | $-\frac{1}{2}$ | |
| $x_2$ | $\frac{7}{2}$ | | 1 | $-\frac{3}{2}$ | | | $\frac{1}{2}$ | |
| $s_2$ | 1 | | | 1 | 1 | | $-2$ | |
| $s_3$ | $\frac{1}{2}$ | | | $\frac{1}{2}$ | | 1 | $-\frac{3}{2}$ | |
| ← $s_5$ | $-\frac{1}{2}$ | | | $(-\frac{1}{2})$ ↑ | | | $-\frac{1}{2}$ | 1 |

(cut generated from $x_1$ constraint)

(d)

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|---|---|---|
| $(-z)$ | $-40$ | | | | | | $-1$ | $-1$ |
| $x_1$ | 0 | 1 | | | | | 3 | 5 |
| $x_2$ | 5 | | 1 | | | | 2 | $-3$ |
| $s_2$ | 0 | | | | 1 | | $-3$ | 2 |
| $s_3$ | 0 | | | | | 1 | $-2$ | 1 |
| $s_1$ | 1 | | | 1 | | | 1 | $-2$ |

Since $s_1$ and $s_2$ are nonbasic variables, we may take $s_4$ to be the basic variable isolated in this constraint (see Tableau 2(b)).

By making slight modifications to the cutting-plane algorithm that has been described, we can show that an optimal solution to the integer-programming problem will be obtained, as in this example, after adding only a finite number of cuts. The proof of this fact by R. Gomory in 1958 was a very important theoretical break-through, since it showed that integer programs can be solved by *some* linear program (the associated linear program plus the added constraints). Unfortunately, the number of cuts to be added, though finite, is usually quite large, so that this result does not have important practical ramifications.

**EXERCISES**

1. As the leader of an oil-exploration drilling venture, you must determine the least-cost selection of 5 out of 10 possible sites. Label the sites $S_1, S_2, \ldots, S_{10}$, and the exploration costs associated with each as $C_1, C_2, \ldots, C_{10}$.

   Regional development restrictions are such that:

   i) Evaluating sites $S_1$ *and* $S_7$ will prevent you from exploring site $S_8$.
   ii) Evaluating site $S_3$ *or* $S_4$ prevents you from assessing site $S_5$.
   iii) Of the group $S_5, S_6, S_7, S_8$, only two sites may be assessed.

   Formulate an integer program to determine the minimum-cost exploration scheme that satisfies these restrictions.

2. A company wishes to put together an academic ''package'' for an executive training program. There are five area colleges, each offering courses in the six fields that the program is designed to touch upon.

   The package consists of 10 courses; each of the six fields must be covered.

   The tuition (basic charge), assessed when at least one course is taken, at college $i$ is $T_i$ (independent of the number of courses taken). Moreover, each college imposes an additional charge (covering course materials, instructional aids, and so forth) for *each* course, the charge depending on the college and the field of instructions.

   Formulate an integer program that will provide the company with the minimum amount it must spend to meet the requirements of the program.

3. The marketing group of A. J. Pitt Company is considering the options available for its next advertising campaign program. After a great deal of work, the group has identified a selected number of options with the characteristics shown in the accompanying table.

| | *TV* | *Trade magazine* | *Newspaper* | *Radio* | *Popular magazine* | *Promotional campaign* | *Total resource available* |
|---|---|---|---|---|---|---|---|
| *Customers reached* | 1,000,000 | 200,000 | 300,000 | 400,000 | 450,000 | 450,000 | – |
| *Cost ($)* | 500,000 | 150,000 | 300,000 | 250,000 | 250,000 | 100,000 | 1,800,000 |
| *Designers needed* (man-hours) | 700 | 250 | 200 | 200 | 300 | 400 | 1,500 |
| *Salesmen needed* (man-hours) | 200 | 100 | 100 | 100 | 100 | 1,000 | 1,200 |

   The objective of the advertising program is to maximize the number of customers reached, subject to the limitation of resources (money, designers, and salesman) given in the table above. In addition, the following constraints have to be met:

   i) If the promotional campaign is undertaken, it needs either a radio or a popular magazine campaign effort to support it.
   ii) The firm cannot advertise in both the trade and popular magazines.

   Formulate an integer-programming model that will assist the company to select an appropriate advertising campaign strategy.

4. Three different items are to be routed through three machines. Each item must be processed first on machine 1, then on machine 2, and finally on machine 3. The sequence of items may differ for each machine. Assume that the times $t_{ij}$ required to perform the work on item $i$ by machine $j$ are known and are integers. Our objective is to minimize the total time necessary to process all the items.

   a) Formulate the problem as an integer programming problem. [*Hint.* Let $x_{ij}$ be the starting time of processing item $i$ on machine $j$. Your model must prevent two items from occupying the same machine at the same time; also, an item may not start processing on machine $(j + 1)$ unless it has completed processing on machine $j$.]

b) Suppose we want the items to be processed in the same sequence on each machine. Change the formulation in part (a) accordingly.

5. Consider the problem:

$$\text{Maximize } z = x_1 + 2x_2,$$

subject to:

$$\begin{aligned} x_1 + x_2 &\le 8, \\ -x_1 + x_2 &\le 2, \\ x_1 - x_2 &\le 4, \\ x_2 &\ge 0 \quad \text{and integer}, \end{aligned}$$

$$x_1 = 0, 1, 4, \text{ or } 6.$$

a) Reformulate the problem as an equivalent integer linear program.
b) How would your answer to part (a) change if the objective function were changed to:

$$\text{Maximize } z = x_1^2 + 2x_2?$$

6. Formulate, but *do not solve,* the following mathematical-programming problems. Also, indicate the type of algorithm used in general to solve each.

a) A courier traveling to Europe can carry up to 50 kilograms of a commodity, all of which can be sold for $40 per kilogram. The round-trip air fare is $450 plus $5 per kilogram of baggage in excess of 20 kilograms (one way). Ignoring any possible profits on the return trip, should the courier travel to Europe and, if so, how much of the commodity should be taken along in order to maximize his profits?
b) A copying service incurs machine operating costs of:

$$\begin{aligned} \$0.10 &\quad \text{for copies 1 to 4,} \\ 0.05 &\quad \text{for copies 5 to 8,} \\ 0.025 &\quad \text{for copies 9 and over,} \end{aligned}$$

and has a capacity of 1000 copies per hour. One hour has been reserved for copying a 10-page article to be sold to MBA students. Assuming that all copies can be sold for $0.50 per article, how many copies of the article should be made?
c) A petrochemical company wants to maximize profit on an item that sells for $0.30 per gallon. Suppose that increased temperature increases output according to the graph in Fig. E9.1. Assuming that production costs are directly proportional to temperature as $7.50 per degree centigrade, how many gallons of the item should be produced?

7. Suppose that you are a ski buff and an entrepreneur. You own a set of hills, any or all of which can be developed. Figure E9.2 illustrates the nature of the cost for putting ski runs on any hill.

The cost includes fixed charges for putting new trails on a hill. For each hill $j$, there is a limit $d_j$ on the number of trails that can be constructed and a lower limit $t_j$ on the number of feet of trail that must be developed if the hill is used.

Use a piecewise linear approximation to formulate a strategy based on cost minimization for locating the ski runs, given that you desire to have $M$ total feet of developed ski trail in the area.

8. Consider the following word game. You are assigned a number of tiles, each containing a letter $a, b, \ldots,$ or $z$ from the alphabet. For any letter $\alpha$ from the alphabet, your assignment includes $N_\alpha$ (a nonnegative integer) tiles with the letter $\alpha$. From the letters you are to construct any of the words $w_1, w_2, \ldots, w_n$. This list might, for example, contain all words from a given dictionary.

You may construct any word at most once, and use any tile at most once. You receive $v_j \ge 0$ points for making word $w_j$ and an additional bonus of $b_{ij} \ge 0$ points for making *both* words $w_i$ and $w_j$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n$).

a) Formulate a *linear* integer program to determine your optimal choice of words.

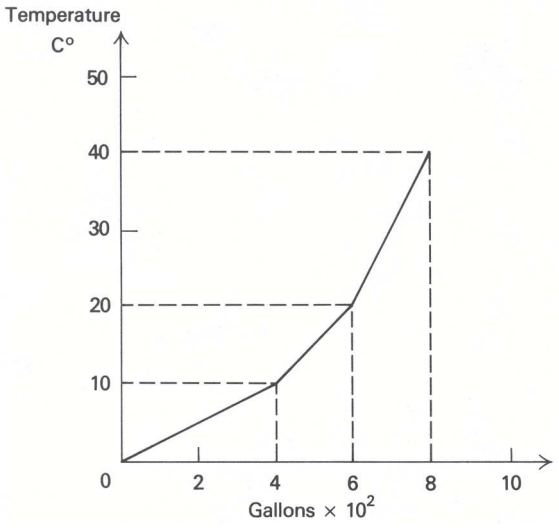


**Figure** E9.1

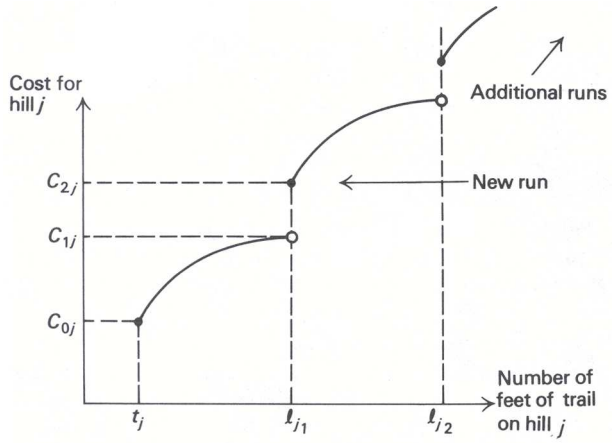


**Figure** E9.2

b) How does the formulation change if you are allowed to select 100 tiles with no restriction on your choice of letters?

9. In Section 9.1 of the text, we presented the following simplified version of the warehouse-location problem:

$$\text{Minimize} \sum_i \sum_j c_{ij} x_{ij} + \sum_i f_i y_i,$$

subject to:

$$\sum_i x_{ij} = d_j \qquad (j = 1, 2, \ldots, n)$$

$$\sum_j x_{ij} - y_i \left( \sum_j d_j \right) \leq 0 \qquad (i = 1, 2, \ldots, m)$$

$$x_{ij} \geq 0 \qquad (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n)$$

$$y_i = 0 \text{ or } 1 \qquad (i = 1, 2, \ldots, m).$$

a) The above model assumes only one product and two distribution stages (from warehouse to customer). Indicate how the model would be modified if there were three distribution stages (from plant to warehouse to customer) and $L$ different products. [*Hint.* Define a new decision variable $x_{ijkl}$ equal to the amount to be sent from plant $i$, through warehouse $j$, to customer $k$, of product $\ell$.]

b) Suppose there are maximum capacities for plants and size limits (both upper and lower bounds) for warehouses. What is the relevant model now?
c) Assume that each customer has to be served from a single warehouse; i.e., no splitting of orders is allowed. How should the warehousing model be modified? [*Hint.* Define a new variable $z_{jk}$ = fraction of demand of customer $k$ satisfied from warehouse $j$.]
d) Assume that each warehouse $i$ experiences economies of scale when shipping above a certain threshold quantity to an individual customer; i.e., the unit distribution cost is $c_{ij}$ whenever the amount shipped is between 0 and $d_{ij}$, and $c'_{ij}$ (lower than $c_{ij}$) whenever the amount shipped exceeds $d_{ij}$. Formulate the warehouse location model under these conditions.

10. Graph the following integer program:

$$\text{Maximize } z = x_1 + 5x_2,$$

subject to :

$$\begin{aligned} -4x_1 + 3x_2 &\leq 6, \\ 3x_1 + 2x_2 &\leq 18, \end{aligned}$$

$$x_1, x_2 \geq 0 \qquad \text{and integer.}$$

Apply the branch-and-bound procedure, graphically solving each linear-programming problem encountered. Interpret the branch-and-bound procedure graphically as well.

11. Solve the following integer program using the branch-and-bound technique:

$$\text{Minimize } z = 2x_1 - 3x_2 - 4x_3,$$

subject to:

$$\begin{aligned} -x_1 + x_2 + 3x_3 &\leq 8, \\ 3x_1 + 2x_2 - x_3 &\leq 10, \end{aligned}$$

$$x_1, x_2, x_3 \geq 0 \quad \text{and integer.}$$

The optimal tableau to the linear program associated with this problem is:

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|
| $x_3$ | $\frac{6}{7}$ | $-\frac{5}{7}$ | | 1 | $\frac{2}{7}$ | $-\frac{1}{7}$ |
| $x_2$ | $\frac{38}{7}$ | $\frac{8}{7}$ | 1 | | $\frac{1}{7}$ | $\frac{3}{7}$ |
| $(-z)$ | $\frac{138}{7}$ | $\frac{18}{7}$ | | | $\frac{11}{7}$ | $\frac{5}{7}$ |

The variables $x_4$ and $x_5$ are slack variables for the two constraints.

12. Use branch-and-bound to solve the mixed-integer program:

$$\text{Maximize } z = -5x_1 - x_2 - 2x_3 + 5,$$

subject to:

$$\begin{aligned} -2x_1 + x_2 - x_3 + x_4 &= \tfrac{7}{2}, \\ 2x_1 + x_2 + x_3 + x_5 &= 2, \end{aligned}$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, 5)$$

$x_3$ and $x_5$ integer.

13. Solve the mixed-integer programming knapsack problem:

$$\text{Maximize } z = 6x_1 + 4x_2 + 4x_3 + x_4 + x_5,$$

subject to:

$$\begin{aligned} 2x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 &= 7, \\ x_j \geq 0 \qquad (j &= 1, 2, \ldots, 5), \end{aligned}$$

$x_1$ and $x_2$ integer.

14. Solve the following integer program using implicit enumeration:

$$\text{Maximize } z = 2x_1 - x_2 - x_3 + 10,$$

subject to:

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &\leq 9, \\ 2x_2 + x_3 &\geq 4, \\ 3x_1 + 3x_2 + 3x_3 &= 6, \\ x_j = 0 \quad \text{or} \quad 1 \qquad (j &= 1, 2, 3). \end{aligned}$$

15. A college intramural four-man basketball team is trying to choose its starting line-up from a six-man roster so as to maximize average height. The roster follows:

| *Player* | *Number* | *Height** | *Position* |
|---|---|---|---|
| Dave | 1 | 10 | Center |
| John | 2 | 9 | Center |
| Mark | 3 | 6 | Forward |
| Rich | 4 | 6 | Forward |
| Ken | 5 | 4 | Guard |
| Jim | 6 | −1 | Guard |

* In inches over $5'\ 6''$.

The starting line-up must satisfy the following constraints:

i) At least one guard must start.
ii) Either John or Ken must be held in reserve.
iii) Only one center can start.
iv) If John or Rich starts, then Jim cannot start.

a) Formulate this problem as an integer program.
b) Solve for the optimal starting line-up, using the implicit enumeration algorithm.

16. Suppose that one of the following constraints arises when applying the implicit enumeration algorithm to a 0−1 integer program:

$$-2x_1 - 3x_2 + x_3 + 4x_4 \leq -6, \tag{1}$$

$$-2x_1 - 6x_2 + x_3 + 4x_4 \leq -5, \tag{2}$$

$$-4x_1 - 6x_2 - x_3 + 4x_4 \leq -3. \tag{3}$$

In each case, the variables on the lefthand side of the inequalities are free variables and the righthand-side coefficients include the contributions of the fixed variables.

a) Use the feasibility test to show that constraint (1) contains no feasible completion.
b) Show that $x_2 = 1$ and $x_4 = 0$ in any feasible completion to constraint (2). State a general rule that shows when a variable $x_j$, like $x_2$ or $x_4$ in constraint (2), must be either 0 or 1 in any feasible solution. [*Hint.* Apply the usual feasibility test after setting $x_j$ to 1 or 0.]
c) Suppose that the objective function to minimize is:

$$z = 6x_1 + 4x_2 + 5x_3 + x_4 + 10,$$

and that $\underline{z} = 17$ is the value of an integer solution obtained previously. Show that $x_3 = 0$ in a feasible completion to constraint (3) that is a potential improvement upon $\underline{z}$ with $z < \underline{z}$. (Note that either $x_1 = 1$ or $x_2 = 1$ in any feasible solution to constraint (3) having $x_3 = 1$.)
d) How could the tests described in parts (b) and (c) be used to reduce the size of the enumeration tree encountered when applying implicit enumeration?

17. Although the constraint

$$x_1 - \quad x_2 \leq -1$$

and the constraint

$$-x_1 + 2x_2 \leq -1$$

to a zero–one integer program both have feasible solutions, the system composed of both constraints is infeasible. One way to exhibit the inconsistency in the system is to add the two constraints, producing the constraint

$$x_2 \leq -1,$$

which has no feasible solution with $x_2 = 0$ or 1.

More generally, suppose that we multiply the $i$th constraint of a system

$$\begin{array}{rcl} & & \text{Multipliers} \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1, & & u_1 \\ \vdots \qquad\qquad \vdots & & \vdots \\ a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i, & & u_i \\ \vdots \qquad\qquad \vdots & & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m, & & u_m \end{array}$$

$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n),$$

by nonnegative constraints $u_i$ and add to give the composite, or *surrogate*, constraint:

$$\left(\sum_{i=1}^{m} u_i a_{i1}\right) x_1 + \left(\sum_{i=1}^{m} u_i a_{i2}\right) x_2 + \cdots + \left(\sum_{i=1}^{m} u_i a_{in}\right) x_n \leq \sum_{i=1}^{m} u_i b_i.$$

a) Show that any feasible solution $x_j = 0$ or 1 $(j = 1, 2, \ldots, n)$ for the system of constraints must also be feasible to the surrogate constraint.
b) How might the fathoming tests discussed in the previous exercise be used in conjunction with surrogate constraints in an implicit enumeration scheme?
c) A ''best'' surrogate constraint might be defined as one that eliminates as many nonoptimal solutions $x_j = 0$ or 1 as possible. Consider the objective value of the integer program when the system constraints are replaced by the surrogate constraint; that is the problem:

$$v(u) = \text{Min } c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

subject to:

$$\left(\sum_{i=1}^{m} u_i a_{i1}\right) x_1 + \left(\sum_{i=1}^{m} u_i a_{i2}\right) x_2 + \cdots + \left(\sum_{i=1}^{m} u_i a_{in}\right) x_n \leq \sum_{i=1}^{m} u_i b_i,$$

$$x_j = 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n).$$

Let us say that a surrogate constraint with multipliers $u_1, u_2 \ldots, u_m$ is *stronger* than another surrogate constraint with multipliers $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m$, if the value $v(u)$ of the surrogate constraint problem with multipliers $u_1, u_2, \ldots, u_m$ is larger than the value of $v(\bar{u})$ with multipliers $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m$. (The larger we make $v(u)$, the more nonoptimal solutions we might expect to eliminate.)

Suppose that we estimate the value of $v(u)$ defined above by replacing $x_j = 0$ or 1 by $0 \le x_j \le 1$ for $j = 1, 2, \ldots, n$. Then, to estimate the strongest surrogate constraint, we would need to find those values of the multipliers $u_1, u_2, \ldots, u_m$ to maximize $v(u)$, where

$$v(u) = \text{Min } c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

subject to:

$$0 \le x_j \le 1 \tag{1}$$

and the surrogate constraint.

Show that the optimal shadow prices to the linear program

$$\text{Maximize } c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

subject to:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \le b_i \quad (i = 1, 2, \ldots, m),$$

$$0 \le x_j \le 1 \qquad (j = 1, 2, \ldots, n),$$

solve the problem of maximizing $v(u)$ in (1).

18. The following tableau specifies the solution to the linear program associated with the integer program presented below.

| *Basic variables* | *Current values* | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|---|
| $x_1$ | $\frac{16}{5}$ | 1 | | $\frac{2}{5}$ | $-\frac{1}{5}$ |
| $x_2$ | $\frac{23}{5}$ | | 1 | $\frac{1}{5}$ | $\frac{2}{5}$ |
| $(-z)$ | $-\frac{133}{5}$ | | | $-\frac{11}{5}$ | $-\frac{2}{5}$ |

$$\text{Maximize } z = 4x_1 + 3x_2,$$

subject to:

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 11, \\ -x_1 + 2x_2 + x_4 &= 6, \\ x_j \ge 0 \text{ and integer} &\qquad (j = 1, 2, 3, 4). \end{aligned}$$

a) Derive cuts from each of the rows in the optimal linear-programming tableau, including the objective function.
b) Express the cuts in terms of the variables $x_1$ and $x_2$. Graph the feasible region for $x_1$ and $x_2$ and illustrate the cuts on the graph.
c) Append the cut derived from the objective function to the linear program, and re-solve. Does the solution to this new linear program solve the integer program? If not, how would you proceed to find the optimal solution to the integer program?

19. Given the following integer linear program:

$$\text{Maximize } z = 5x_1 + 2x_2,$$

subject to:

$$\begin{aligned} 5x_1 + 4x_2 &\le 21, \\ x_1, x_2 &\ge 0 \quad \text{and integer,} \end{aligned}$$

solve, using the cutting-plane algorithm. Illustrate the cuts on a graph of the feasible region.

20. The following knapsack problem:

$$\text{Maximize } \sum_{j=1}^{n} cx_j,$$

subject to:

$$\begin{aligned} \sum_{j=1}^{n} 2x_j &\le n, \\ x_j &= 0 \quad \text{or} \quad 1 \qquad (j = 1, 2, \ldots, n), \end{aligned}$$

which has the same ''contribution'' for each item under consideration, has proved to be rather difficult to solve for most general-purpose integer-programming codes when $n$ is an *odd* number.

a) What is the optimal solution when $n$ is even? when $n$ is odd?
b) Comment specifically on why this problem might be difficult to solve on general integer-programming codes when $n$ is odd.

21. Suppose that a firm has $N$ large rolls of paper, each $W$ inches wide. It is necessary to cut $N_i$ rolls of width $W_i$ from these rolls of paper. We can formulate this problem by defining variables

$$x_{ij} = \text{ Number of smaller rolls of width } W_i \text{ cut from large roll } j.$$

We assume there are $m$ different widths $W_i$. In order to cut all the required rolls of width $W_i$, we need constraints of the form:

$$\sum_{j=1}^{N} x_{ij} = N_i \quad (i = 1, 2, \ldots, m).$$

Further, the number of smaller rolls cut from a large roll is limited by the width $W$ of the large roll. Assuming no loss due to cutting, we have constraints of the form:

$$\sum_{i=1}^{m} W_i x_{ij} \le W \quad (j = 1, 2, \ldots, N).$$

a) Formulate an objective function to minimize the number of large rolls of width $W$ used to produce the smaller rolls.
b) Reformulate the model to minimize the total trim loss resulting from cutting. Trim loss is defined to be that part of a large roll that is unusable due to being smaller than any size needed.
c) Comment on the difficulty of solving each formulation by a branch-and-bound method.
d) Reformulate the problem in terms of the collection of possible patterns that can be cut from a given large roll. (*Hint.* See Exercise 25 in Chapter 1.)
e) Comment on the difficulty of solving the formulation in (d), as opposed to the formulations in (a) or (b), by a branch-and-bound method.

22. The Bradford Wire Company produces wire screening woven on looms in a process essentially identical to that of weaving cloth. Recently, Bradford has been operating at full capacity and is giving serious consideration to a major capital investment in additional looms. They currently have a total of 43 looms, which are in production all of their available hours. In order to justify the investment in additional looms, they must analyze the utilization of their existing capacity.

Of Bradford's 43 looms, 28 are 50 inches wide, and 15 are 80 inches wide. Normally, one or two widths totaling less than the width of the loom are made on a particular loom. With the use of a ''center-tucker,'' up to three widths

can be simultaneously produced on one loom; however, in this instance the effective capacities of the 50-inch and 80-inch looms are reduced to 49 inches and 79 inches, respectively. Under no circumstance is it possible to make more than three widths simultaneously on any one loom.

Figure E9.3 gives a typical loom-loading configuration at one point in time. Loom #1, which is 50 inches wide, has two 24-inch widths being produced on it, leaving 2 inches of unused or ''wasted'' capacity. Loom #12 has only a 31-inch width on it, leaving 19 inches of the loom unused. If there were an order for a width of 19 inches or less, then it could be produced at zero marginal machine cost along with the 31-inch width already being produced on this loom. Note that loom #40 has widths of 24, 26, and 28 inches, totaling 78 inches. The ''waste'' here is considered to be only 1 inch, due to the reduced effective capacity from the use of the center-tucker. Note also that the combination of widths 24, 26, and 30 is not allowed, for similar reasons.

| | *50″ Looms* | | | *80″ Looms* | |
|---|---|---|---|---|---|
| *Loom number* | *Widths* | *Waste* | *Loom number* | *Widths* | *Waste* |
| 1 | 24–24 | 2 | 29 | 48–32 | 0 |
| 2 | 30 | 20 | 30 | 24–26–28 | 1 |
| 3 | 40 | 10 | 31 | 48–32 | 0 |
| 4 | $30\frac{1}{4}$ | $19\frac{3}{4}$ | 32 | 48–32 | 0 |
| 5 | 48 | 2 | 33 | 36–36 | 8 |
| 6 | $32–14\frac{1}{2}$ | $3\frac{1}{2}$ | 34 | 36–36 | 8 |
| 7 | 28 | 22 | 35 | 60–18 | 2 |
| 8 | $26\frac{1}{2}$ | $23\frac{1}{2}$ | 36 | 36–36 | 8 |
| 9 | 30 | 20 | 37 | 42–34 | 4 |
| 10 | 30 | 20 | 38 | 24–26–28 | 1 |
| 11 | 35 | 15 | 39 | 24–26–28 | 1 |
| 12 | 31 | 19 | 40 | 24–26–28 | 1 |
| 13 | $34\frac{1}{4}$ | $15\frac{3}{4}$ | 41 | 48–32 | 0 |
| 14 | 36 | 14 | 42 | 36–36 | 8 |
| 15 | 32 | 18 | 43 | 42–34 | 4 |
| 16 | 34–16 | 0 | 80″ Total | | 46 |
| 17 | $29\frac{3}{8}$ | $20\frac{5}{8}$ | | | $337\frac{3}{4}$ |
| 18 | 30 | 20 | | | |
| 19 | 36 | 14 | Grand total | | $383\frac{3}{4}$ |
| 20 | $47\frac{1}{2}$ | $2\frac{1}{2}$ | | | |
| 21 | 30 | 20 | | | |
| 22 | 20–30 | 0 | | | |
| 23 | $48\frac{7}{8}$ | $1\frac{1}{8}$ | | | |
| 24 | 28–22 | 0 | | | |
| 25 | $30–14\frac{1}{2}$ | $5\frac{1}{2}$ | | | |
| 26 | 42 | 8 | | | |
| 27 | 32–18 | 0 | | | |
| 28 | $28\frac{1}{2}$ | $21\frac{1}{2}$ | | | |
| 50″ Total | | $337\frac{3}{4}$ | | | |

**Figure** E9.3

The total of $383\frac{3}{4}$ inches of ''wasted'' loom capacity represents roughly seven and one-half 50-inch looms; and members of Bradford's management are sure that there must be a more efficient loom-loading configuration that would save some of this ''wasted'' capacity. As there are always numerous additional orders to be produced, any additional capacity can immediately be put to good use.

The two types of looms are run at different speeds and have different efficiencies. The 50-inch looms are operated at 240 pics per second, while the 80-inch looms are operated at 214 pics per second. (There are 16 pics to the inch). Further, the 50-inch looms are more efficient, since their ''up time'' is about 85% of the total time, as compared to 65% for the 80-inch looms.

The problem of scheduling the various orders on the looms sequentially over time is difficult. However, as a first cut at analyzing how efficiently the looms are currently operating, the company has decided to examine the loom-loading configuration at one point in time as given in Fig. E9-3. If these same orders can be rearranged in

such a way as to free up one or two looms, then it would be clear that a closer look at the utilization of existing equipment would be warranted before additional equipment is purchased.

a) Since saving an 80-inch loom is not equivalent to saving a 50-inch loom, what is an appropriate objective function to minimize?
b) Formulate an integer program to minimize the objective function suggested in part (a).

23. In the export division of Lowell Textile Mills, cloth is woven in lengths that are multiples of the piece-length required by the customer. The major demand is for 18-meter piece-lengths, and the cloth is generally woven in 54-meter lengths.

Cloth cannot be sold in lengths greater than the stipulated piece-length. Lengths falling short are classified into four groups. For 18-meter piece-lengths, the categories and the contribution per meter are as follows:

| *Category* | *Technical term* | *Length* (Meters) | *Contribution/ Meter* |
|---|---|---|---|
| A | First sort | 18 | 1.00 |
| B | Seconds | 11–17 | 0.90 |
| C | Short lengths | 6–10 | 0.75 |
| D | Rags | 1–5 | 0.60 |
| J | Joined parts∗ | 18 | 0.90 |

∗ Joined parts consist of lengths obtained by joining two pieces such that the shorter piece is at least 6 meters long.

The current cutting practice is as follows. Each woven length is inspected and defects are flagged prominently. The cloth is cut at these defects and, since the cloth is folded in exact meter lengths, the lengths of each cut piece is known. The cut pieces are then cut again, if necessary, to obtain as many pieces of first sort as possible. The short lengths are joined wherever possible to make joined parts.

Since the process is continuous, it is impossible to pool all the woven lengths and then decide on a cutting pattern. Each woven length has to be classified for cutting before it enters the cutting room.

As an example of the current practice, consider a woven length of 54 meters with two defects, one at 19 meters and one at 44 meters. The woven length is first cut at the defects, giving three pieces of lengths 19, 25, and 10 meters each. Then further cuts are made as follows: The resulting contribution is

| *Piece length* | *Further cuts* |
|---|---|
| 25 | 18 + 7 |
| 19 | 18 + 1 |
| 10 | 10 |

$$2 \times 18 \times 1.00 + (7 + 10) \times 0.75 + 1 \times 0.60 = 49.35.$$

It is clear that this cutting procedure can be improved upon by the following alternative cutting plan: By joining 7

| *Piece length* | *Further cuts* |
|---|---|
| 25 | 18 + 7 |
| 19 | 8 + 11 |
| 10 | 10 |

+ 11 and 8 + 10 to make two joined parts, the resulting contribution is:

$$18 \times 1.00 + 18 \times 2 \times 0.90 = 50.40.$$

Thus with one woven length, contribution can be increased by $1.05 by merely changing the cutting pattern. With a daily output of 1000 such woven lengths (two defects on average), substantial savings could be realized by improved cutting procedures.

a) Formulate an integer program to maximize the contribution from cutting the woven length described above.
b) Formulate an integer program to maximize the contribution from cutting a general woven length with no more than four defects. Assume that the defects occur at integral numbers of meters.
c) How does the formulation in (b) change when the defects are *not* at integral numbers of meters?

24. Custom Pilot Chemical Company is a chemical manufacturer that produces batches of specialty chemicals to order. Principal equipment consists of eight interchangeable reactor vessels, five interchangeable distillation columns, four large interchangeable centrifuges, and a network of switchable piping and storage tanks. Customer demand comes in the form of orders for batches of one or more (usually not more than three) specialty chemicals, normally to be delivered simultaneously for further use by the customer. Customer Pilot Chemical fills these orders by means of a sort of pilot plant for each specialty chemical formed by inter-connecting the appropriate quantities of equipment. Sometimes a specialty chemical requires processing by more than one of these ''pilot'' plants, in which case one or more batches of intermediate products may be produced for further processing. There is no shortage of piping and holding tanks, but the expensive equipment (reactors, distillation columns, and centrifuges) is frequently inadequate to meet the demand.

The company's schedules are worked out in calendar weeks, with actual production always taking an integer multiple of weeks. Whenever a new order comes in, the scheduler immediately makes up a precedence tree-chart for it. The precedence tree-chart breaks down the order into ''jobs.'' For example, an order for two specialty chemicals, of which one needs an intermediate stage, would be broken down into three jobs (Fig. E9.4). Each job requires certain equipment, and each is assigned a preliminary time-in-process, or ''duration.'' The durations can always be predicted with a high degree of accuracy.

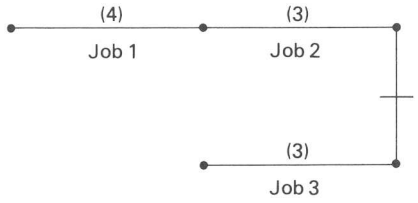


**Figure** E9.4

Currently, Custom Pilot Chemical has three orders that need to be scheduled (see the accompanying table). Orders can be started at the beginning of their arrival week and should be completed by the end of their *week due*.

| | | | | | | *Resource requirements* | | |
|---|---|---|---|---|---|---|---|---|
| *Order number* | *Job number* | *Precedence relations* | *Arrival week* | *Duration in weeks* | *Week due* | *Reactors* | *Distillation columns* | *Centri-fuges* |
| AK14 | 1 | None | 15 | 4 | 22 | 5 | 3 | 2 |
| | 2 | (1) | 15 | 3 | 22 | 0 | 1 | 1 |
| | 3 | None | 15 | 3 | 22 | 2 | 0 | 2 |
| AK15 | 1 | None | 16 | 3 | 23 | 1 | 1 | 1 |
| | 2 | None | 16 | 2 | 23 | 2 | 0 | 0 |
| | 3 | (1) | 16 | 2 | 23 | 2 | 2 | 0 |
| AK16 | 1 | None | 17 | 5 | 23 | 2 | 1 | 1 |
| | 2 | None | 17 | 1 | 23 | 1 | 3 | 0 |

For example, AK14 consists of three jobs, where Job 2 cannot be started until Job 1 has been completed. Figure E9-4 is an appropriate precedence tree for this order. Generally, the resources required are tied up for the entire duration of a job. Assume that Custom Pilot Chemical is just finishing week 14.

a) Formulate an integer program that minimizes the total completion time for all orders.

b) With a more complicated list of orders, what other objective functions might be appropriate? How would these be formulated as integer programs?

25. A large electronics manufacturing firm that produces a single product is faced with rapid sales growth. Its planning group is developing an overall capacity-expansion strategy, that would balance the cost of building new capacity, the cost of operating the new and existing facilities, and the cost associated with unmet demand (that has to be subcontracted outside at a higher cost).

The following model has been proposed to assist in defining an appropriate strategy for the firm.

*Decision variables*

| | |
|---|---|
| $Y_{it}$ | A binary integer variable that will be 1 if a facility exists at site $i$ in period $t$, and 0 otherwise. |
| $(IY)_{it}$ | A binary integer variable that will be 1 if facility is constructed at site $i$ in period $t$, and 0 otherwise. |
| $A_{it}$ | Capacity (sq ft) at site $i$ in period $t$. |
| $(IA)_{it}$ | The increase in capacity at site $i$ in period $t$. |
| $(DA)_{it}$ | The decrease in capacity at site $i$ in period $t$. |
| $P_{it}$ | Total units produced at site $i$ in period $t$. |
| $U_t$ | Total unmet demand (subcontracted) units in period $t$. |

*Forecast parameters*

| | |
|---|---|
| $D_t$ | Demand in units in period $t$. |

*Cost parameters–Capacity*

| | |
|---|---|
| $s_{it}$ | Cost of operating a facility at site $i$ in period $t$. |
| $d_{it}$ | Cost of building a facility at site $i$ in period $t$. |
| $a_{it}$ | Cost of occupying 1 sq ft of fully equipped capacity at site $i$ in period $t$. |
| $b_{it}$ | Cost of increasing capacity by 1 sq ft at site $i$ in period $t$. |
| $c_{it}$ | Cost of decreasing capacity by 1 sq ft at site $i$ in period $t$. |

*Cost parameters–Production*

| | |
|---|---|
| $o_t$ | Cost of unmet demand (subcontracting + opportunity cost) for one unit in period $t$. |
| $v_{it}$ | Tax-adjusted variable cost (material + labor + taxes) of producing one unit at site $i$ in period $t$. |

*Productivity parameters*

| | |
|---|---|
| $(pa)_{it}$ | Capacity needed (sq ft years) at site $i$ in period $t$ to produce one unit of the product. |

*Policy parameters*

| | |
|---|---|
| $\overline{F}_{it}, \underline{F}_{it}$ | Maximum, minimum facility size at site $i$ in period $t$ (if the site exists). |
| $G_{it}$ | Maximum growth (sq ft) of site $i$ in period $t$. |

*Objective function*

The model's objective function is to minimize total cost:

$$\begin{aligned}\text{Min} \sum_{t=1}^{T}\sum_{i=1}^{N} & s_{it}Y_{it} + d_{it}(IY)_{it} \\ + & \sum_{t=1}^{T}\sum_{i=1}^{N} a_{it}A_{it} + b_{it}(IA)_{it} + sc_{it}(DA)_{it} \\ + & \sum_{t=1}^{T}\sum_{i=1}^{N} v_{it}P_{it} + \sum_{t=1}^{T} o_t U_t\end{aligned}$$

*Description of constraints*

1. *Demand constraint*

$$\sum_{i=1}^{N} P_{it} + U_t = D_t$$

2. *Productivity constraint*

$$(pa)_{it} P_{it} \leq A_{it} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

3. *Facility-balance constraint*

$$Y_{it-1} + (IY)_{it} = Y_{it} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

4. *Area-balance constraint*

$$A_{it-1} + (IA)_{it} - (DA)_{it} = A_{it} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

5. *Facility-size limits*

$$\begin{aligned} A_{it} &\geq Y_{it} \underline{F}_{it} \\ A_{it} &\leq Y_{it} \overline{F}_{it} \end{aligned} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

6. *Growth constraint*

$$(IA)_{it} - (DA)_{it} \leq G_{it} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

7. *Additional constraints*

$$\begin{aligned} 0 &\leq Y_{it} \leq 1 \\ Y_{it} &\quad \text{integer} \end{aligned} \qquad \begin{cases} \text{for } i = 1, 2, \ldots, N, \\ \text{for } t = 1, 2, \ldots, T, \end{cases}$$

All variables nonnegative.

Explain the choice of decision variables, objective function, and constraints. Make a detailed discussion of the model assumptions. Estimate the number of constraints, continuous variables, and integer variables in the model. Is it feasible to solve the model by standard branch-and-bound procedures?

26. An investor is considering a total of $I$ possible investment opportunities ($i = 1, 2, \ldots, I$), during a planning horizon covering $T$ time periods ($t = 1, 2, \ldots, T$). A total of $b_{it}$ dollars is required to initiate investment $i$ in period $t$. Investment in project $i$ at time period $t$ provides an income stream $a_{i,t+1}, a_{i,t+2}, \ldots, a_{i,T}$ in succeeding time periods. This money is available for reinvestment in other projects. The total amount of money available to the investor at the beginning of the planning period is $B$ dollars.

a) Assume that the investor wants to maximize the net present value of the net stream of cash flows ($c_t$ is the corresponding discount factor for period $t$). Formulate an integer-programming model to assist in the investor's decision.
b) How should the model be modified to incorporate timing restrictions of the form:
   i) Project $j$ cannot be initiated until after project $k$ has been initiated; or
   ii) Projects $j$ and $k$ cannot both be initiated in the same period?
c) If the returns to be derived from these projects are uncertain, how would you consider the risk attitudes of the decision-maker? [*Hint.* See Exercises 22 and 23 in Chapter 3.]

27. The advertising manager of a magazine faces the following problem: For week $t$, $t = 1, 2 \ldots, 13$, she has been allocated a maximum of $n_t$ pages to use for advertising. She has received requests $r_1, r_2, \ldots, r_B$ for advertising, bid $r_k$ indicating:

i) the initial week $i_k$ to run the ad,

ii) the duration $d_k$ of the ad (in weeks),
iii) the page allocation $a_k$ of the ad (half-, quarter-, or full-page),
iv) a price offer $p_k$.

The manager must determine which bids to accept to maximize revenue, subject to the following restrictions:

i) Any ad that is accepted must be run in consecutive weeks throughout its duration.
ii) The manager cannot accept conflicting ads. Formally, subsets $T_j$ and $\bar{T}_j$ for $j = 1, 2, \ldots, n$ of the bids are given, and she may not select an ad from both $T_j$ and $\bar{T}_j$ ($j = 1, 2, \ldots, n$). For example, if $T_1 = \{r_1, r_2\}$, $\bar{T}_1 = \{r_3, r_4, r_5\}$, and bid $r_1$ or $r_2$ is accepted, then bids $r_3$, $r_4$, or $r_5$ must all be rejected; if bid $r_3$, $r_4$, or $r_5$ is accepted, then bids $r_1$ and $r_2$ must both be rejected.
iii) The manager must meet the Federal Communication Commission's balanced advertising requirements. Formally, subsets $S_j$ and $\bar{S}_j$ for $j = 1, 2, \ldots, m$ of the bids are given; if she selects a bid from $S_j$, she must also select a bid from $\bar{S}_j$ ($j = 1, 2, \ldots, m$). For example, if $S_1 = \{r_1, r_3, r_8\}$ and $S_2 = \{r_4, r_6\}$, then either request $r_4$ or $r_6$ must be accepted if any of the bids $r_1$, $r_3$, or $r_8$ are accepted.

Formulate as an integer program.

28. The $m$-traveling-salesman problem is a variant of the traveling-salesman problem in which $m$ salesmen stationed at a home base, city 1, are to make tours to visit other cities, $2, 3, \ldots, n$. Each salesman must be routed through some, but not all, of the cities and return to his (or her) home base; each city must be visited by one salesman. To avoid redundancy, the sales coordinator requires that no city (other than the home base) be visited by more than one salesman or that any salesman visit any city more than once.

Suppose that it cost $c_{ij}$ dollars for any salesman to travel from city $i$ to city $j$.

a) Formulate an integer program to determine the minimum-cost routing plan. Be sure that your formulation does not permit subtour solutions for any salesman.
b) The $m$-traveling-salesman problem can be reformulated as a single-salesman problem as follows: We replace the home base (city 1) by $m$ fictitious cities denoted $1', 2', \ldots, m'$. We link each of these fictitious cities to each other with an arc with high cost $M > \sum_{i=1}^{n} \sum_{j=1}^{n} |c_{ij}|$, and we connect each fictitious city $i'$ to every city $j$ at cost $c_{i'j} = c_{ij}$. Figure E9.5 illustrates this construction for $m = 3$.

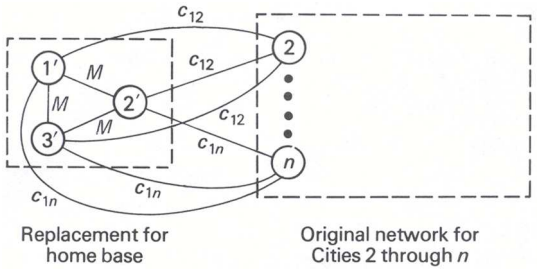


**Figure** E9.5

Suppose that we solve this reformulation as a single-traveling-salesman problem, but identify each portion of the tour that travels from one of the fictitious cities $i'$ through the original network and back to another fictitious city $j'$ as a tour for one of the $m$ salesmen. Show that these tours solve the $m$-traveling-salesman problem. [*Hint.* Can the solution to the single-salesman problem ever use any of the arcs with a high cost $M$? How many times must the single salesman leave from and return to one of the fictitious cities?]

29. Many practical problems, such as fuel-oil delivery, newspaper delivery, or school-bus routing, are special cases of the generic vehicle-routing problem. In this problem, a fleet of vehicles must be routed to deliver (or pick up) goods from a depot, node 0, to $n$ drop-points, $i = 1, 2, \ldots, n$.

Let

$Q_k$ = Loading capacity of the $k$ th vehicle in the fleet ($k = 1, 2, \ldots, m$);
$d_i$ = Number of items to be left at drop-point $i$ ($i = 1, 2, \ldots, n$);
$t_i^k$ = Time to unload vehicle $k$ at drop-point $i$ ($i = 1, 2, \ldots n$; $k = 1, 2, \ldots, m$);
$t_{ij}^k$ = Time for vehicle $k$ to travel from drop-point $i$ to drop-point $j$ ($i = 0, 1, \ldots, n$; $j = 0, 1, \ldots, n$; $k = 1, 2, \ldots, m$)
$c_{ij}^k$ = Cost for vehicle $k$ to travel from node $i$ to node $j$ ($i = 0, 1, \ldots, n$; $j = 0, 1, \ldots, n$; $k = 1, 2, \ldots, m$).

If a vehicle visits drop-point $i$, then it fulfills the entire demand $d_i$ at that drop-point. As in the traveling or $m$-traveling salesman problem, only one vehicle can visit any drop-point, and no vehicle can visit the same drop-point more than once. The routing pattern must satisfy the service restriction that vehicle $k$'s route take longer than $T_k$ time units to complete.

Define the decision variables for this problem as:

$$x_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ travels from drop-point } i \text{ to drop-point } j, \\ 0 & \text{otherwise.} \end{cases}$$

Formulate an integer program that determines the minimum-cost routing pattern to fulfill demand at the drop-points that simultaneously meets the maximum routing-time constraints and loads no vehicle beyond its capacity. How many constraints and variables does the model contain when there are 400 drop-points and 20 vehicles?

**ACKNOWLEDGMENTS**

Exercise 22 is based on the Bradford Wire Company case written by one of the authors. Exercise 23 is extracted from the Muda Cotton Textiles case written by Munakshi Malya and one of the authors.
Exercise 24 is based on the Custom Pilot Chemical Co. case, written by Richard F. Meyer and Burton Rothberg, which in turn is based on ''Multiproject Scheduling with Limited Resources: A Zero–One Programming Approach,'' *Management Science* 1969, by A. A. B. Pritsken, L. J. Watters, and P. M. Wolfe.
In his master's thesis (Sloan School of Management, MIT, 1976), entitled ''A Capacity Expansion Planning Model with Bender's Decomposition,'' M. Lipton presents a more intricate version of the model described in Exercise 25.
The transformation used in Exercise 28 has been proposed by several researchers; it appears in the 1971 book *Distribution Management* by S. Eilson, C. Watson-Gandy, and N. Christofides, Hafner Publishing Co., 1971.