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.

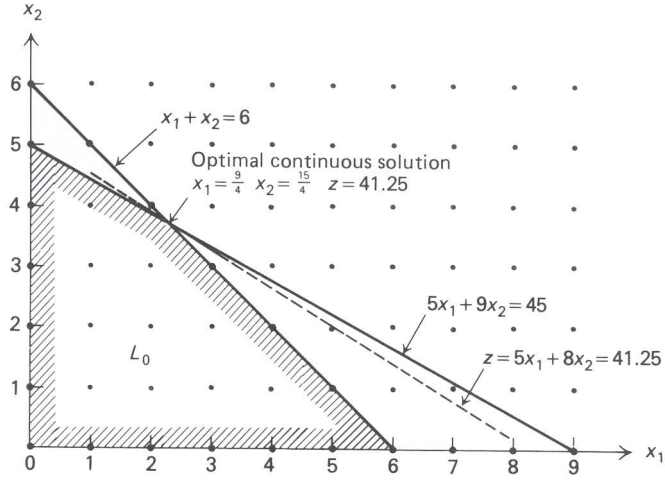

Figure 9.8 An integer programming example.

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.

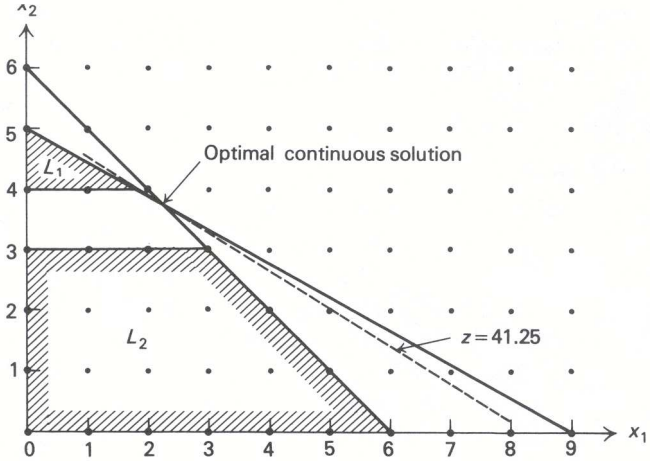

Figure 9.9 Subdividing the feasible region.

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 ≤ 3 or an integer ≥ 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

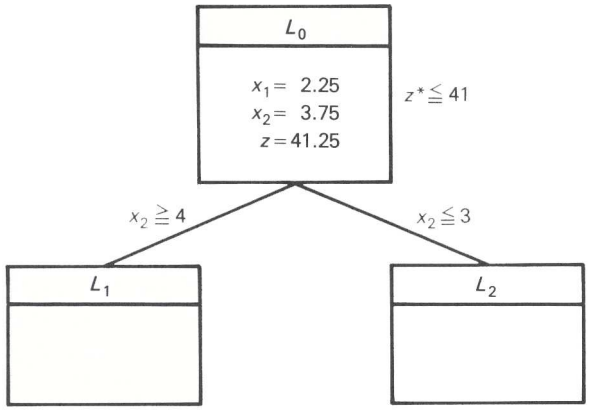

Figure 9.10 Enumeration tree.

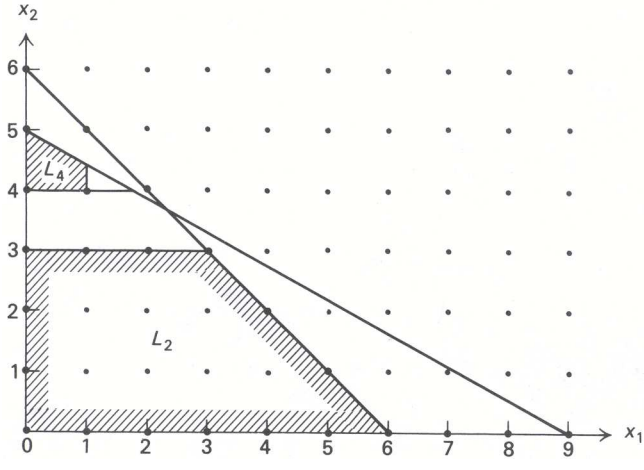

Figure 9.11 Subdividing the region L_1.

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

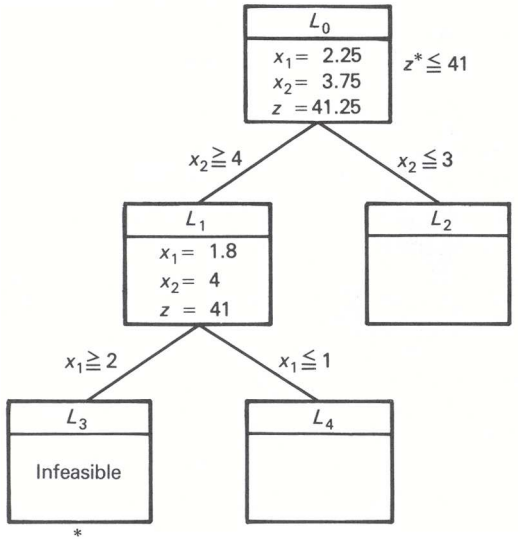

Figure 9.12

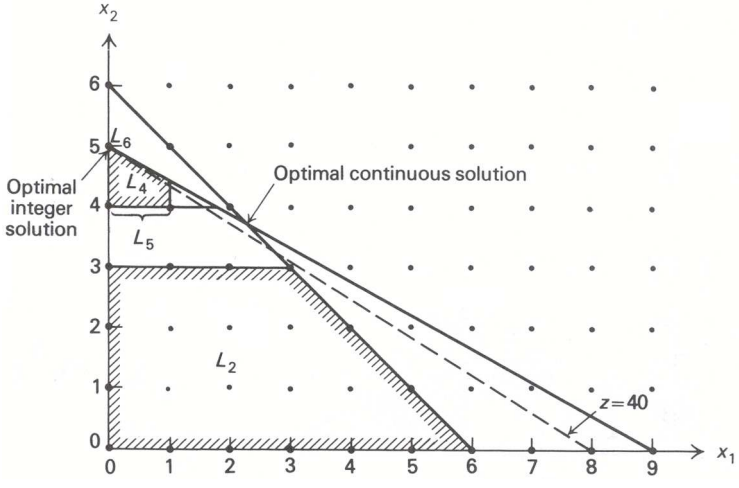

Figure 9.13 Final subdivisions for the example.

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,

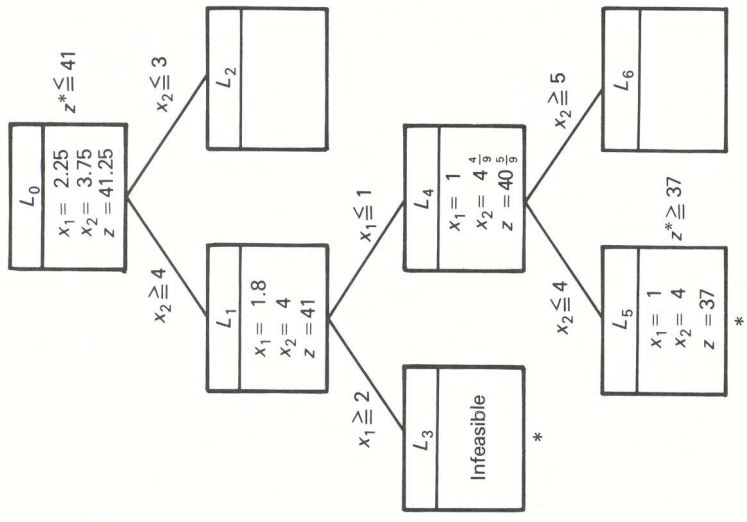

Figure 9.14

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.

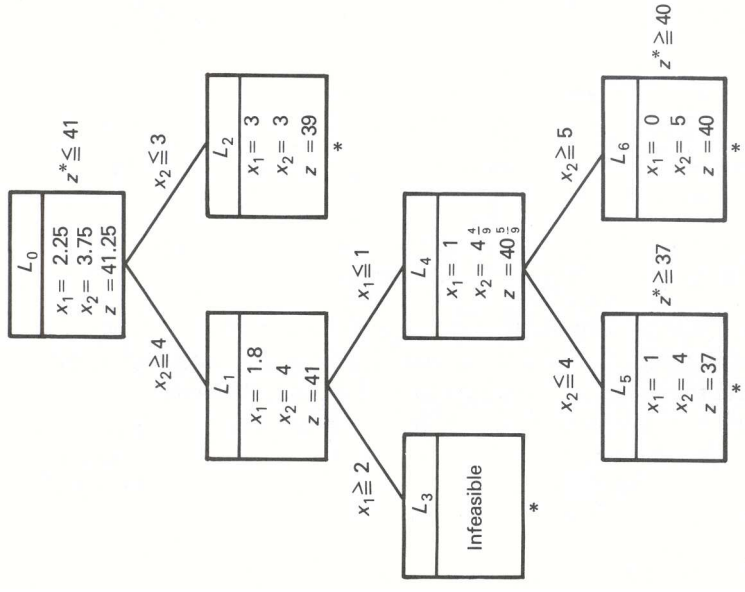

Figure 9.15

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:

$$\begin{array}{rrl} (-z) & -\frac{5}{4}s_1 - \frac{3}{4}s_2 = -41\frac{1}{4} & \left.\begin{array}{l}\\\\\\\end{array}\right\} \text{Constraints from the optimal canonical form} \\ x_1 & +\frac{9}{4}s_1 - \frac{1}{4}s_2 = \frac{9}{4} & \\ & \textcircled{x_2} - \frac{5}{4}s_1 + \frac{1}{4}s_2 = \frac{15}{4} & \\ & -x_2 + s_3 = -4, & \text{Added constraint} \\ & x_1,\ x_2,\ s_1,\ s_2,\ s_3 \geq 0, & \end{array}$$

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:

$$\begin{array}{rrrrrrrrr} (-z) & & -\frac{5}{4}s_1 & -\frac{3}{4}s_2 & & = & -41\frac{1}{4}, \\ & x_1 & & & +\frac{9}{4}s_1 & -\frac{1}{4}s_2 & = & \frac{9}{4}, \\ & & x_2 & -\frac{5}{4}s_1 & +\frac{1}{4}s_2 & = & \frac{15}{4}, \\ & & & \textcircled{-\frac{5}{4}s_1} & +\frac{1}{4}s_2 & +s_3 & = & -\frac{1}{4}, \\ & & & x_1, x_2, s_1, s_2, s_3 \geq 0. & & & & \end{array}$$

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:

$$\begin{array}{rrrrrrr} (-z) & & & -s_2 & -s_3 & = & -41, \\ & x_1 & & +\frac{1}{5}s_2 & +\frac{9}{5}s_3 & = & \frac{9}{5}, \\ & & x_2 & & -s_3 & = & 4, \\ & & & s_1 - \frac{1}{5}s_2 & -\frac{4}{5}s_3 & = & \frac{1}{5}, \\ & & & x_1,\ x_2,\ s_1,\ s_2,\ s_3 \geq 0. & & & \end{array}$$

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.

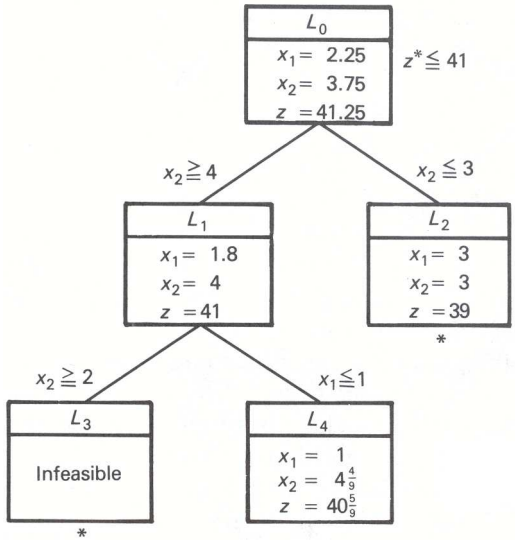

Figure 9.16

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 $\overline{z}$ 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* $\underline{z} < z^* < \overline{z}$ 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.

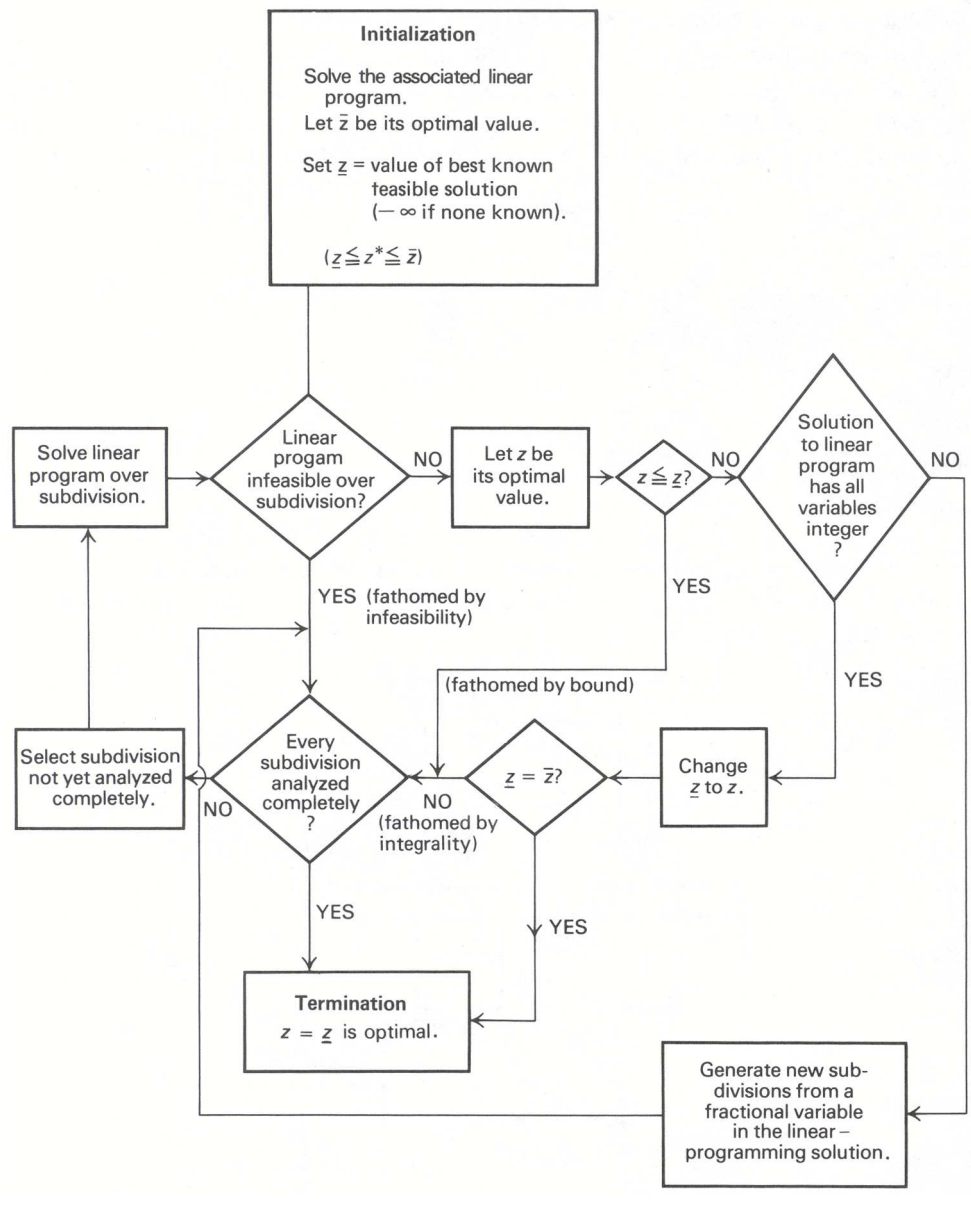

Figure 9.17 Branch-and-bound for integer-programming maximization.

† 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.''

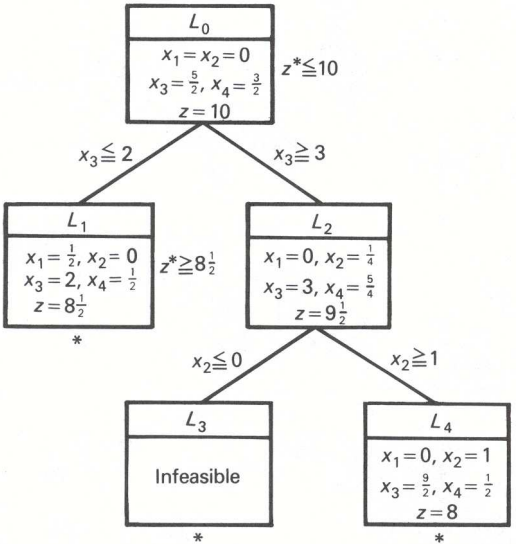
L_0
$x_1 = x_2 = 0$
$x_3 = \frac{5}{2}$, $x_4 = \frac{3}{2}$
$z = 10$
$z^* \leqq 10$
$x_3 \leqq 2$
$x_3 \geqq 3$
L_1
$x_1 = \frac{1}{2}$, $x_2 = 0$
$x_3 = 2$, $x_4 = \frac{1}{2}$
$z = 8\frac{1}{2}$
$z^* \geqq 8\frac{1}{2}$
*
L_2
$x_1 = 0$, $x_2 = \frac{1}{4}$
$x_3 = 3$, $x_4 = \frac{5}{4}$
$z = 9\frac{1}{2}$
$x_2 \leqq 0$
$x_2 \geqq 1$
L_3
Infeasible
*
L_4
$x_1 = 0$, $x_2 = 1$
$x_3 = \frac{9}{2}$, $x_4 = \frac{1}{2}$
$z = 8$
*

Figure 9.18

Branch and Bound

We will explain branch and bound by using the capital budgeting example from the previous section. In that problem, the model is

$$\begin{array}{ll} \text{Maximize} & 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ \text{subject to} & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ & x_j \in \{0,1\}\ j = 1, \ldots 4. \end{array}$$

The linear relaxation solution is $x_1 = 1, x_2 = 1, x_3 = 0.5, x_4 = 0$ with a value of 22. We know that no integer solution will have value more than 22. Unfortunately, since x_3 is not integer, we do not have an integer solution yet.

We want to force x_3 to be integer. To do so, we *branch* on x_3, creating two new problems. In one, we will add the constraint $x_3 = 0$. In the other, we add the constraint $x_3 = 1$. This is illustrated in Figure 2.

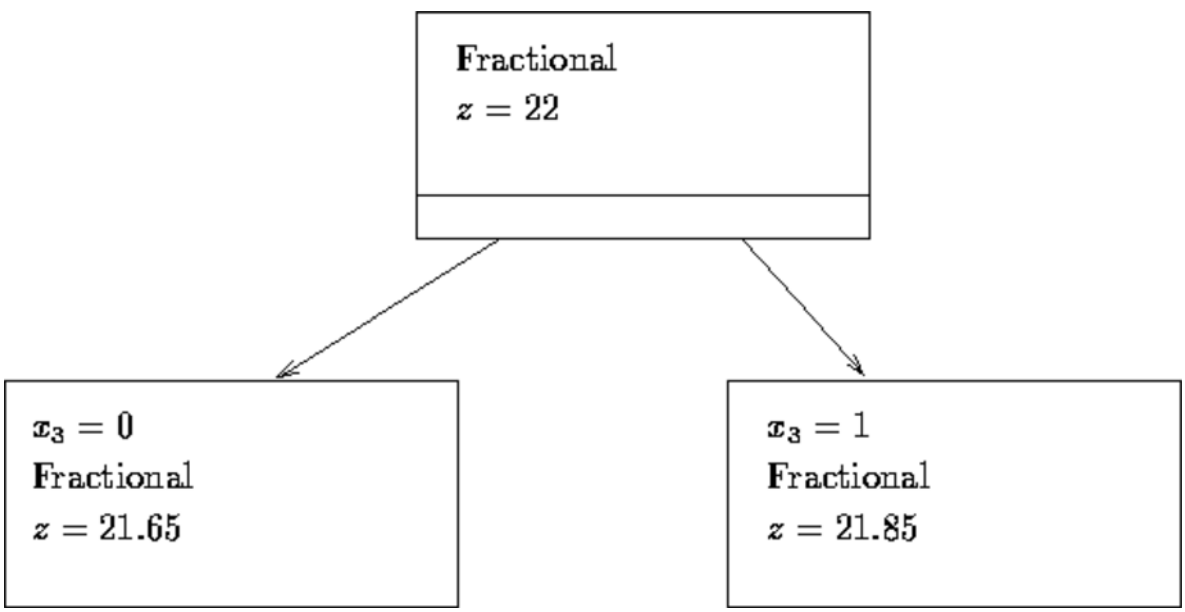

Figure 2: First Branching

Note that any optimal solution to the overall problem must be feasible to one of the subproblems. If we solve the linear relaxations of the subproblems, we get the following solutions:

- $x_3 = 0$: objective 21.65, $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0.667$;
- $x_3 = 1$: objective 21.85, $x_1 = 1, x_2 = 0.714, x_3 = 1, x_4 = 0$.

At this point we know that the optimal integer solution is no more than 21.85 (we actually know it is less than or equal to 21 (Why?)), but we still do not have any feasible integer solution. So, we will take a subproblem and branch on one of its variables. In general, we will choose the subproblem as follows:

- We will choose an *active* subproblem, which so far only means one we have not chosen before, and
- We will choose the subproblem with the highest solution value (for maximization) (lowest for

minimization).

In this case, we will choose the subproblem with $x_3 = 1$, and branch on x_2. After solving the resulting subproblems, we have the branch and bound tree in Figure 3.

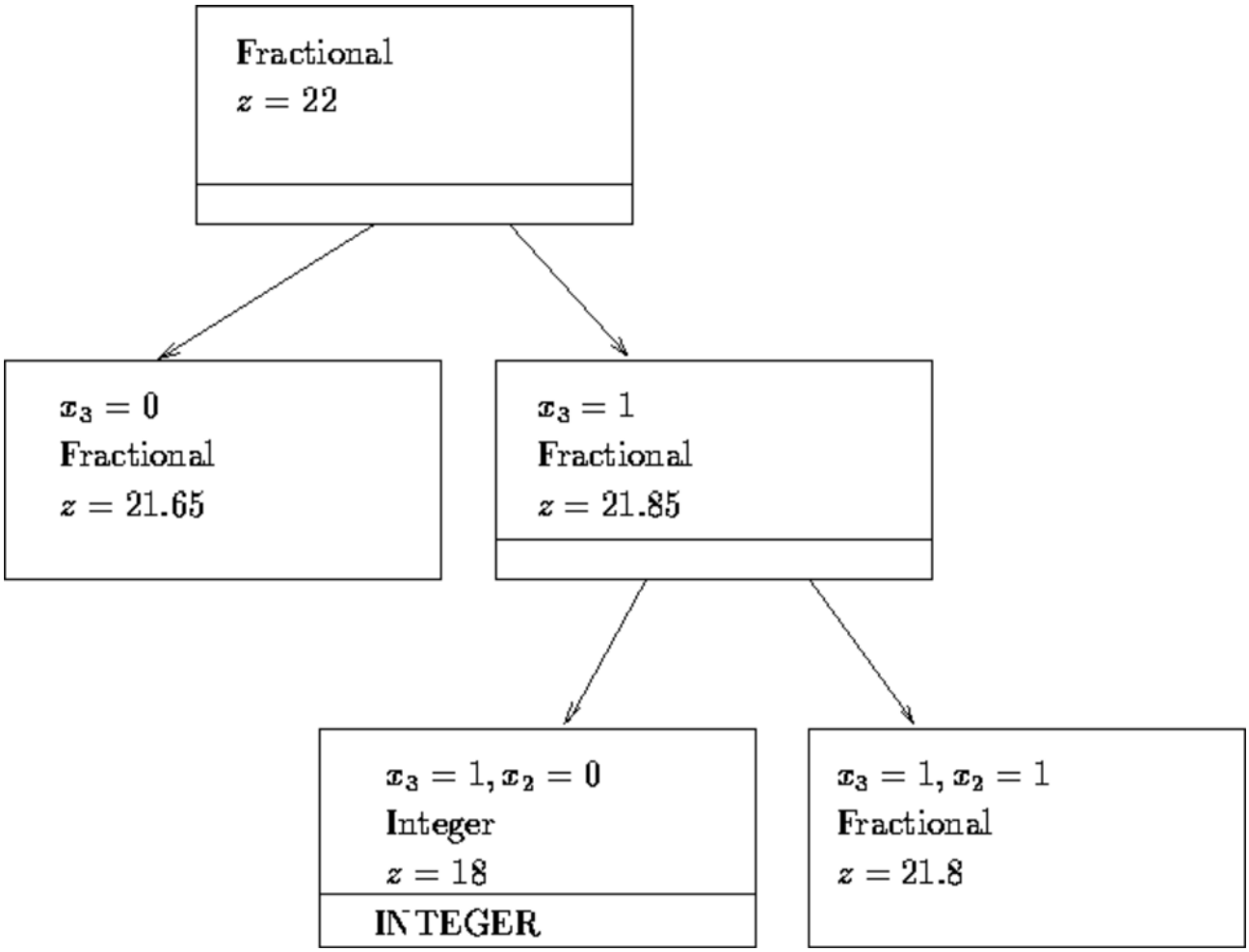

Figure 3: Second Branching

The solutions are:

- $x_3 = 1, x_2 = 0$: objective 18, $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$;
- $x_3 = 1, x_2 = 1$: objective 21.8, $x_1 = 0.6, x_2 = 1, x_3 = 1, x_4 = 0$.

We now have a feasible integer solution with value 18. Furthermore, since the $x_3 = 1, x_2 = 0$ problem gave an integer solution, no further branching on that problem is necessary. It is not active due to integrality of solution. There are still active subproblems that might give values more than 18. Using our rules, we will branch on problem $x_3 = 1, x_2 = 1$ by branching on x_1 to get Figure 4.

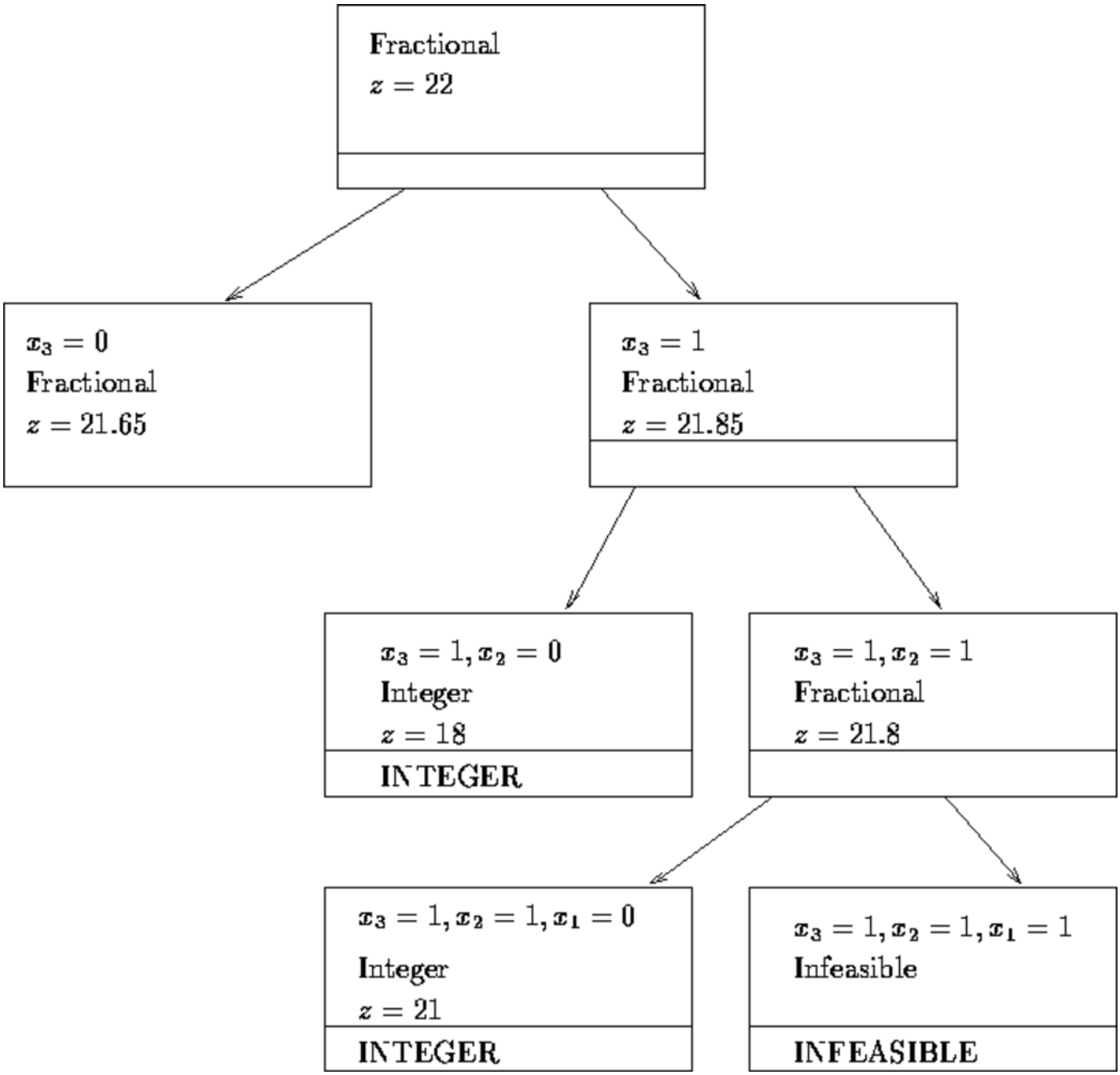

Figure 4: Third Branching

The solutions are:

- $x_3 = 1, x_2 = 1, x_1 = 0$: objective 21, $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 1$;
- $x_3 = 1, x_2 = 1, x_1 = 1$: infeasible.

Our best integer solution now has value 21. The subproblem that generates that is not active due to integrality of solution. The other subproblem generated is not active due to infeasibility. There is still a subproblem that is active. It is the subproblem with solution value 21.65. By our ``round-down" result, there is no better solution for this subproblem than 21. But we already have a solution with value 21. It is not useful to search for another such solution. We can *fathom* this subproblem based on the above bounding argument and mark it not active. There are no longer any active subproblems, so the optimal solution value is 21.

We have seen all parts of the branch and bound algorithm. The essence of the algorithm is as follows:

1. Solve the linear relaxation of the problem. If the solution is integer, then we are done. Otherwise

create two new subproblems by branching on a fractional variable.

2. A subproblem is not active when any of the following occurs:
    1. You used the subproblem to branch on,
    2. All variables in the solution are integer,
    3. The subproblem is infeasible,
    4. You can fathom the subproblem by a bounding argument.

3. Choose an active subproblem and branch on a fractional variable. Repeat until there are no active subproblems.

That's all there is to branch and bound! Depending on the type of problem, the branching rule may change somewhat. For instance, if *x* is restricted to be integer (but not necessarily 0 or 1), then if *x*=4.27 your would branch with the constraints $x \leq 4$ and $x \geq 5$ (not on *x*=4 and *x*=5).

In the worst case, the number of subproblems can get huge. For many problems in practice, however, the number of subproblems is quite reasonable.

For an example of a huge number of subproblems, try the following in LINGO:

```
model:
  sets:
    a /1..17/: x;
  endsets

  max = -x0 + @sum(a: 2 * x);
  x0 + @sum(a: 2 * x) < 17;
  @for (a: @bin(x));
end
```

Note that this problem has only 18 variables and only a single constraint. LINDO looks at 48,619 subproblems, taking about 20 minutes on a Sun Sparc workstation, before deciding the optimal objective is 16. LINGO on a 16MHz 386 PC (with math coprocessor) looks at 48,000+ subproblems and takes about five hours. CPLEX on a Sun SPARC 10 takes about 50 seconds to examine 61,497 subproblems (counting those that are fathomed without solving the LP). The 100 variable version of this problem would take about 10^{29} subproblems or about 3×10^{18} years (at 1000 subproblems per second). Luckily, most problems take far less time.

Exercise 5 (Optional) *Solve the following problem by the branch and bound algorithm. For convenience, always select x_1 as the branching variable when both x_1 and x_2 are fractional.*

$$\begin{aligned} \textit{Maximize} \quad & x_1 + x_2 \\ \textit{subject to} \quad & 2x_1 + 5x_2 \leq 16 \\ & 6x_1 + 5x_2 \leq 30 \\ & x_1, x_2 \geq 0 \textit{ and integer.} \end{aligned}$$

Exercise 6 (Optional) *Repeat the preceeding exercise assuming that x_1 only is restricted to integer values.*

Exercise 7 (Optional) *Consider the following* cargo-loading *problem, where five items are to be loaded on a vessel. The weights w_i and the volume v_i per unit of the different items as well as their corresponding values r_i are tabulated as follows.*

Item i	w_i	v_i	r_i
1	*5*	*1*	*4*
2	*8*	*8*	*7*
3	*3*	*6*	*6*
4	*2*	*5*	*5*
5	*7*	*4*	*4*

The maximum cargo weight and volume are given by $W = 112$ and $V = 109$, respectively. It is required to determine the most valuable cargo load in discrete units of each item. Formulate the problem as an integer program and solve by LINDO.

(Ova stranica je ostavljena prazna)