

Chapter 9

Integer Programming

Companion slides of
Applied Mathematical Programming
by Bradley, Hax, and Magnanti
(Addison-Wesley, 1977)

prepared by

José Fernando Oliveira
Maria Antónia Carravilla

What are integer-programming problems?

- 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. In these situations, the decision variables are inherently integral by the nature of the decision-making problem.

What are integer-programming problems?

- Linear programming problems in which fractional solutions are not realistic.
 - Mixed integer programs when some, but not all, variables are restricted to be integer.
 - Pure integer programs when all decision variables must be integers.
 - Binary programs when all decision variables must be either 0 or 1.

SOME INTEGER-PROGRAMMING MODELS

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.
- Assume that:
 - c_j is the contribution resulting from the j th investment,
 - a_{ij} is the amount of resource i used on the j th investment.
 - b_i is the limited availability of the i th resource.
- The objective is to maximize total contribution from all investments.

Model

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

subject to:

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

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

Note that...

- 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 $a_{ij} > 0$,
 - 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 $b_i > 0$,
 - If funds are withdrawn in period i , then $b_i < 0$.

The capital-budgeting model can be made much richer by including logical considerations

- The investment in a new product line is contingent upon previous investment in a new plant (*contingency constraints*)

$$x_j \geq x_i$$

- Conflicting projects (*multiple-choice constraints*)

$$x_1 + x_2 + x_3 + x_4 \leq 1$$

The 0-1 *knapsack* problem

- A capital budgeting problem with a single resource:

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

subject to:

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

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

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.

Decision variables and relevant data

$$y_i = \begin{cases} 1 & \text{if warehouse } i \text{ is opened,} \\ 0 & \text{if warehouse } i \text{ is not opened;} \end{cases}$$

x_{ij} = Amount to be sent from warehouse i to customer j .

f_i = Fixed operating cost for warehouse i , if opened (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:
 - the demand d_j of each customer must be filled from the warehouses,
 - goods can be shipped from a warehouse only if it is opened.

Model

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

subject to:

$$\sum_{i=1}^m x_{ij} = d_j \quad (j = 1, 2, \dots, n),$$

$$\sum_{j=1}^n x_{ij} - y_i \left(\sum_{j=1}^n d_j \right) \leq 0 \quad (i = 1, 2, \dots, m),$$

$$\begin{array}{ll} x_{ij} \geq 0 & (i = 1, 2, \dots, m; j = 1, 2, \dots, n), \\ y_i = 0 \quad \text{or} \quad 1 & (i = 1, 2, \dots, m). \end{array}$$

Scheduling

- 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.

Decision variables and relevant data

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

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

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

- 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.

Model

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

subject to:

$$\sum_{j=1}^n a_{ij} x_j = 1 \quad (i = 1, 2, \dots, m),$$
$$x_j = 0 \quad \text{or} \quad 1 \quad (j = 1, 2, \dots, n).$$

An alternative formulation
that permits a crew to ride as passengers
on a leg

$$\sum_{j=1}^n a_{ij} x_j \geq 1 \quad (i = 1, 2, \dots, m).$$

- Set-partitioning: $\sum_{j=1}^n a_{ij} x_j = 1$
- Set-covering $\sum_{j=1}^n a_{ij} x_j \geq 1$

Traveling salesperson problem

- Starting from his home, a salesperson wishes to visit each of $(n - 1)$ other cities and return home at minimal cost.
- (S)He must visit each city exactly once and it costs c_{ij} to travel from city i to city j .
- What route should (s)he select?

Decision variables

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

Model

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

subject to:

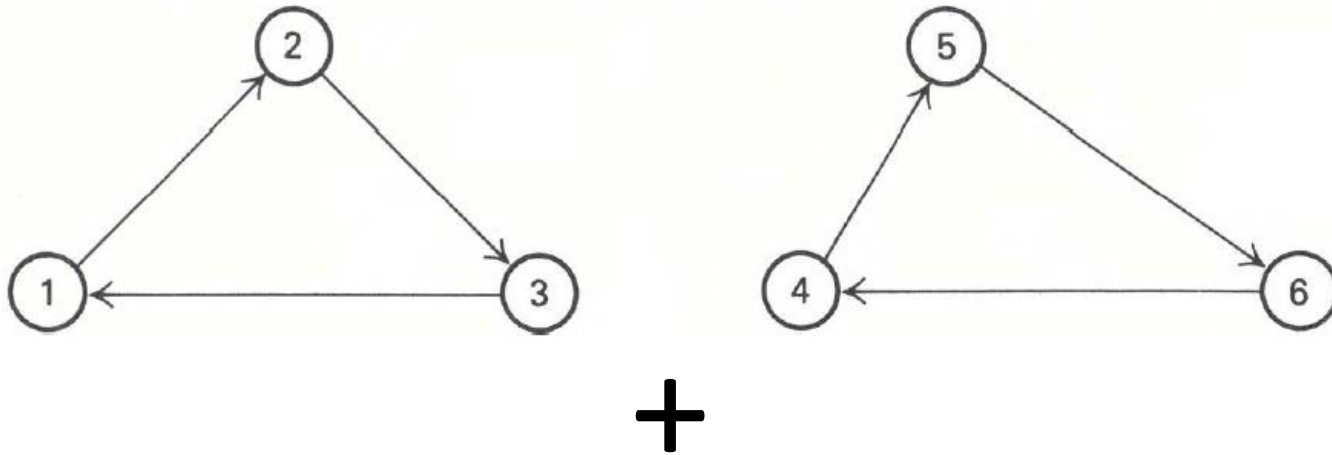
$$\sum_{i=1}^n x_{ij} = 1 \quad (j = 1, 2, \dots, n),$$

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

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

- Sub-tours are not prevented...

Sub-tours elimination



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

- With n cities, $(2^n - 1)$ constraints of this nature must be added...
 - True for this strategy and many other strategies used to eliminate sub-tours in TSP models.



FORMULATING INTEGER PROGRAMS

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.
- But can be also used as auxiliary variables...

Constraint Feasibility

- Does a given choice of the decision variables satisfies the constraint

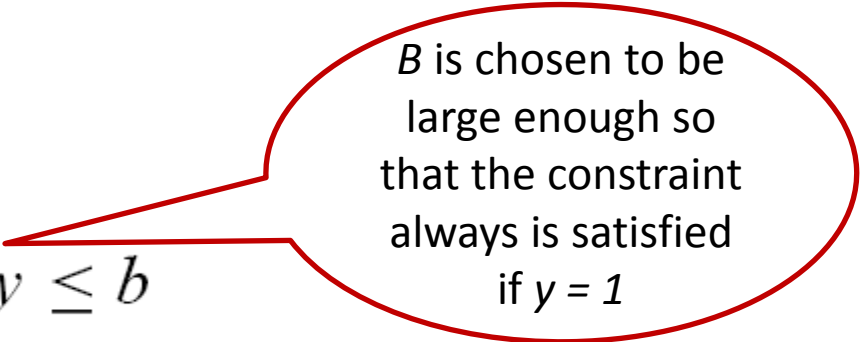
$$f(x_1, x_2, \dots, x_n) \leq b \quad ?$$

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

$$f(x_1, x_2, \dots, x_n) - By \leq b$$



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

Alternative constraints

$$f_1(x_1, x_2, \dots, x_n) \leq b_1,$$

$$f_2(x_1, x_2, \dots, x_n) \leq b_2.$$

- 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:

$$f_1(x_1, x_2, \dots, x_n) - B_1 y_1 \leq b_1,$$

$$f_2(x_1, x_2, \dots, x_n) - B_2 y_2 \leq b_2,$$

$$y_1 + y_2 \leq 1,$$

$$y_1, y_2 \text{ binary.}$$

- We can save one integer variable in this formulation:

$$\begin{aligned}f_1(x_1, x_2, \dots, x_n) - B_1 y_1 &\leq b_1, \\f_2(x_1, x_2, \dots, x_n) - B_2(1 - y_1) &\leq b_2, \\y_1 &= 0 \quad \text{or} \quad 1.\end{aligned}$$

Conditional Constraints

- These constraints have the form:

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

- If we note that:

$$(p \Rightarrow q) \Leftrightarrow (\sim p \vee q)$$

we end up with the following **alternative constraints**:

$$f_1(x_1, x_2, \dots, x_n) \leq b_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq b_2,$$

k -Fold Alternatives

- We must satisfy at least k of the constraints:

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

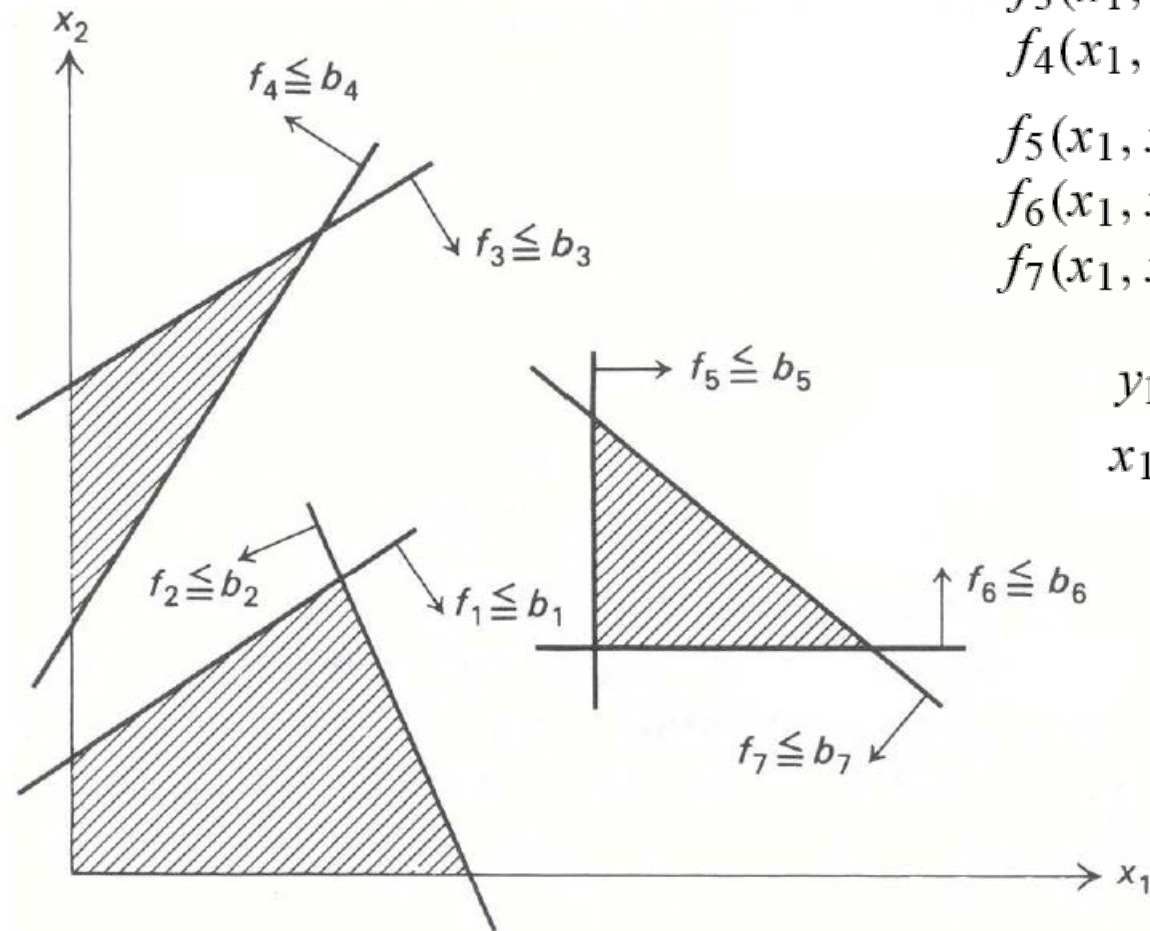
Assuming that B_j for $j = 1, 2, \dots, p$, are chosen so that the ignored constraints will not be binding, the general problem can be formulated as follows:

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

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

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

Compound Alternatives



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

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

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

$$y_1 + y_2 + y_3 \leq 2,$$

$$x_1 \geq 0, \quad x_2 \geq 0,$$

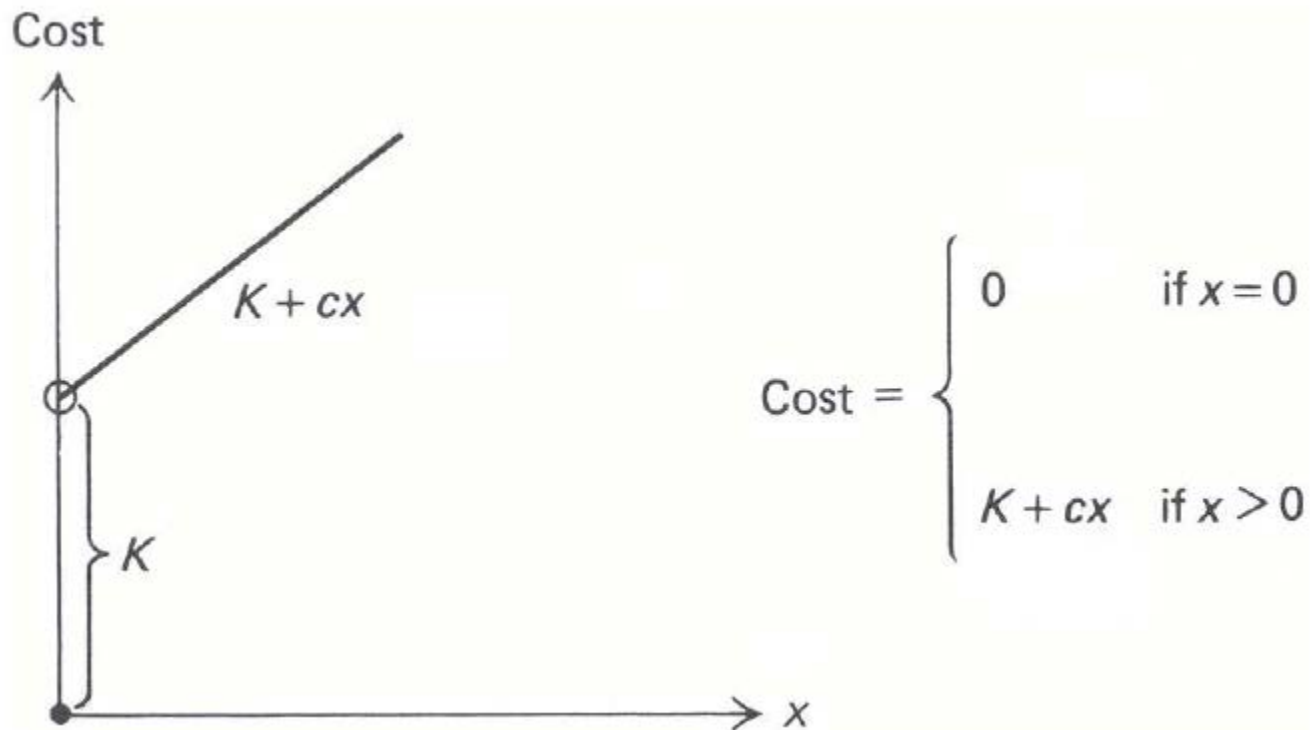
y_1, y_2, y_3 binary.

Representing Nonlinear Functions

- Nonlinear functions can be represented by integer-programming formulations...

Fixed Costs

- Frequently, the objective function for a minimization problem contains fixed costs (preliminary design costs, fixed investment costs, fixed contracts, and so forth).



- 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

$$Ky + cx,$$

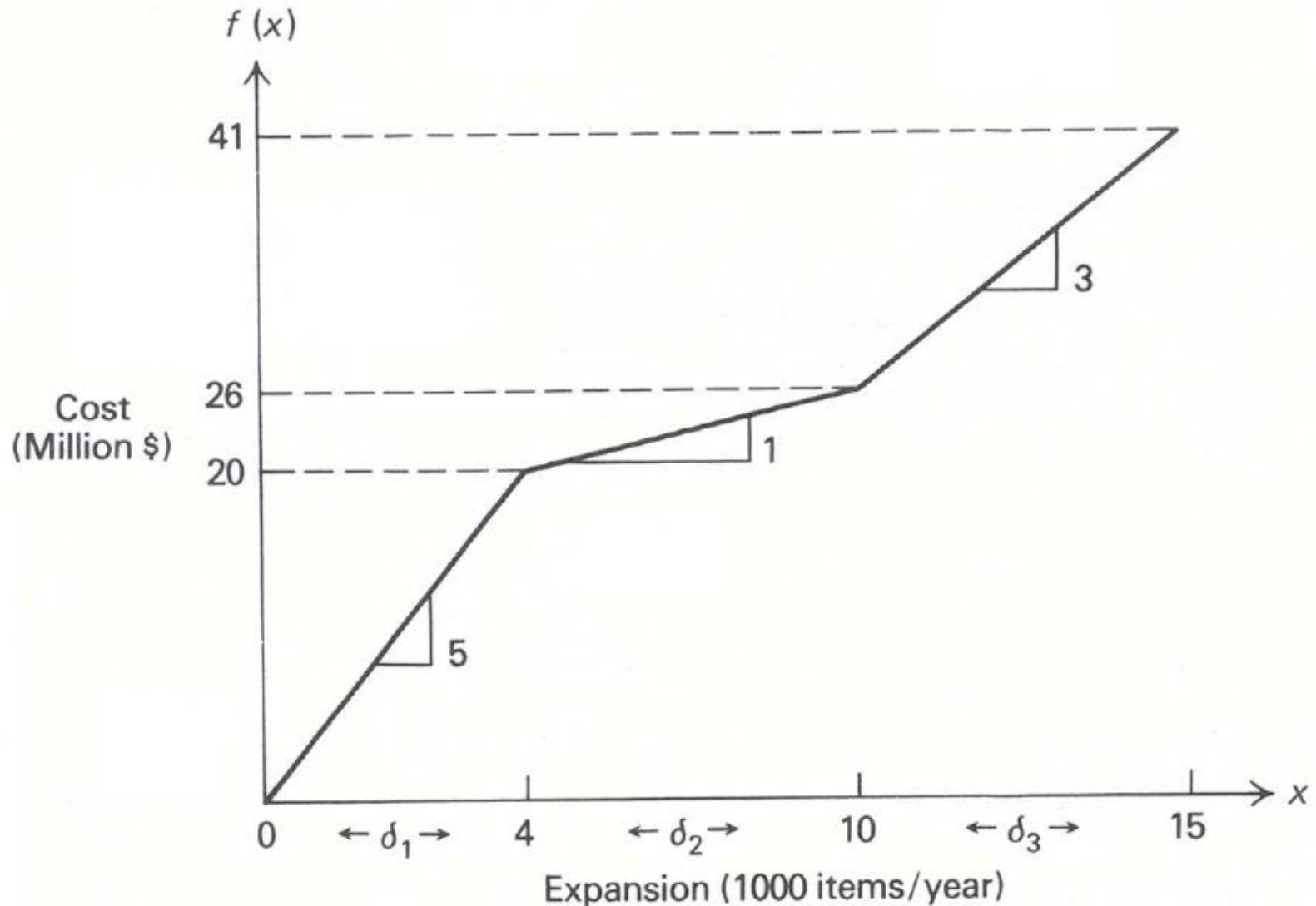
with the constraints

$$x \leq By,$$

$$x \geq 0,$$

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

Piecewise Linear Representation



- 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

$$0 \leq \delta_1 \leq 4,$$

$$0 \leq \delta_2 \leq 6,$$

$$0 \leq \delta_3 \leq 5;$$

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:
 - δ_1 corresponds to the amount by which x exceeds 0, but is less than or equal to 4;
 - δ_2 is the amount by which x exceeds 4, but is less than or equal to 10; and
 - δ_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 $\delta_3 > 0$.
- However, these restrictions on the variables are simply **conditional constraints** and can be modeled by introducing (**more**) binary variables:

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

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

$$4w_1 \leq \delta_1 \leq 4,$$

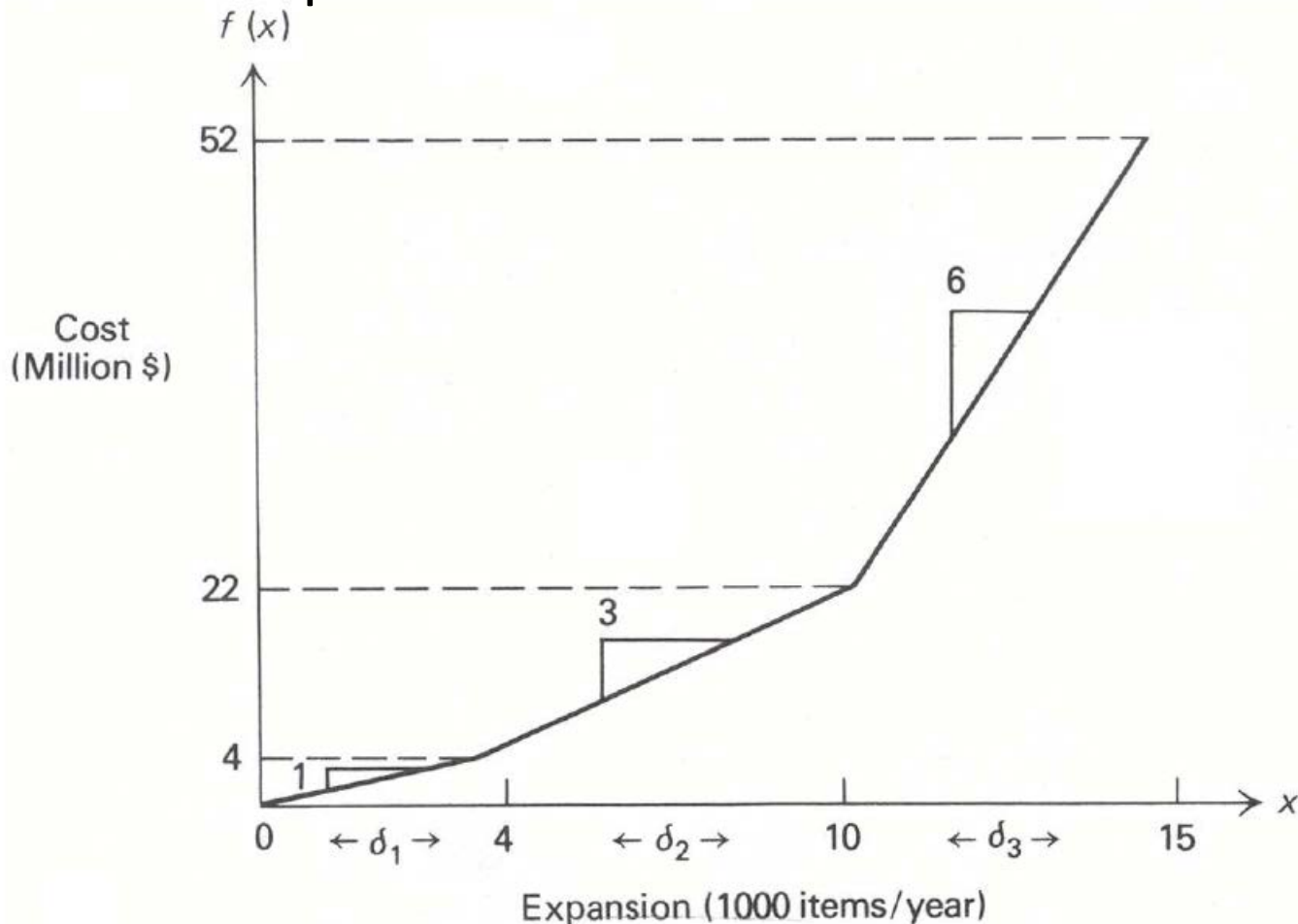
$$6w_2 \leq \delta_2 \leq 6w_1,$$

$$0 \leq \delta_3 \leq 5w_2,$$

$$w_1 \quad \text{and} \quad w_2 \text{ binary,}$$

Diseconomies of Scale

- When marginal costs are increasing for a minimization problem or marginal returns are decreasing for a maximization problem.



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

subject to:

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

- 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 δ_1 , δ_2 , δ_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 $\delta_3 > 0$.
- This representation without integer variables is not valid, however, if **economies of scale** are present.

SOME CHARACTERISTICS OF INTEGER PROGRAMS

A sample problem

Determine:

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

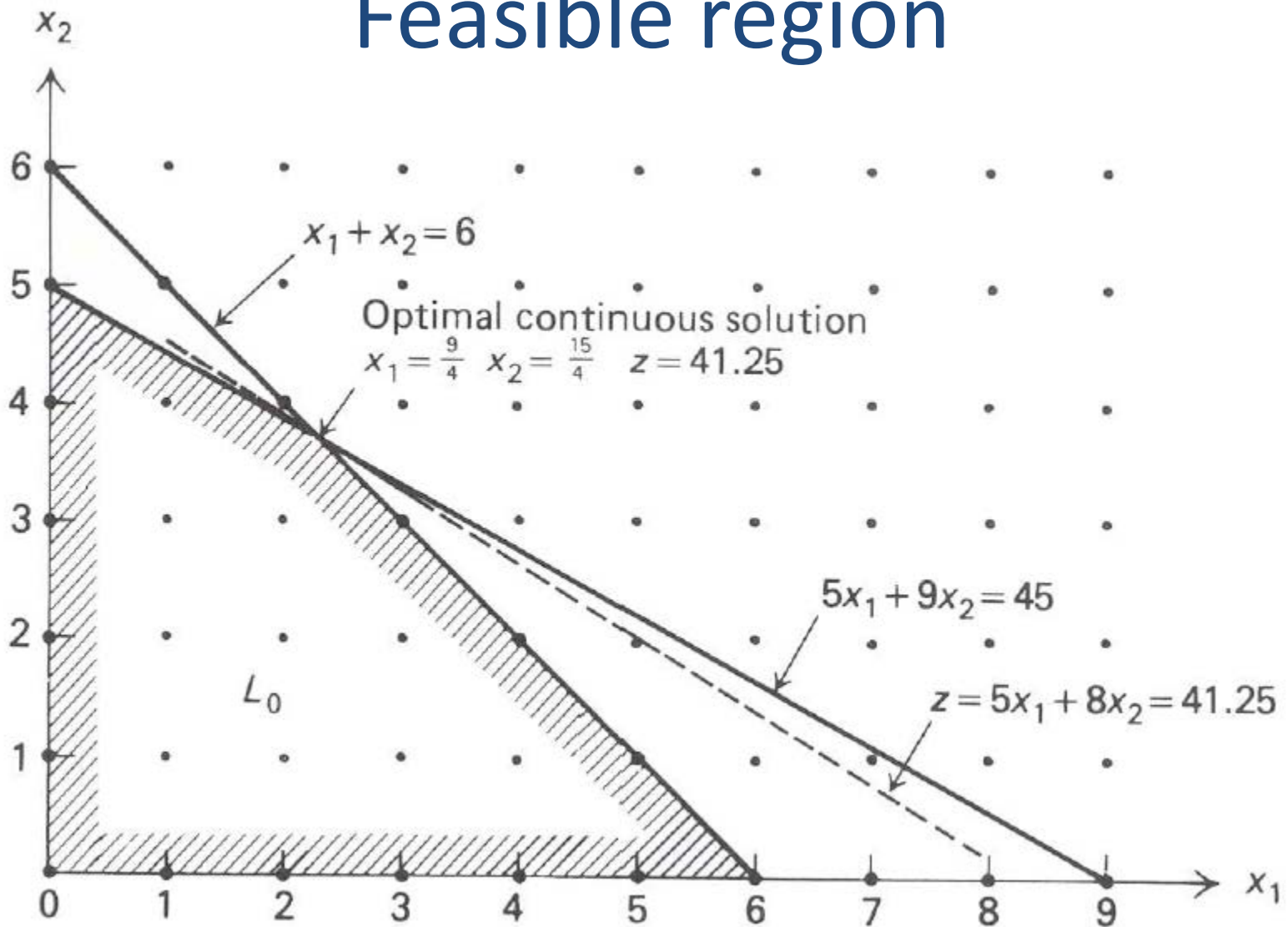
subject to:

$$x_1 + x_2 \leq 6,$$

$$5x_1 + 9x_2 \leq 45,$$

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

Feasible region



Dots in the shaded region are feasible integer points.

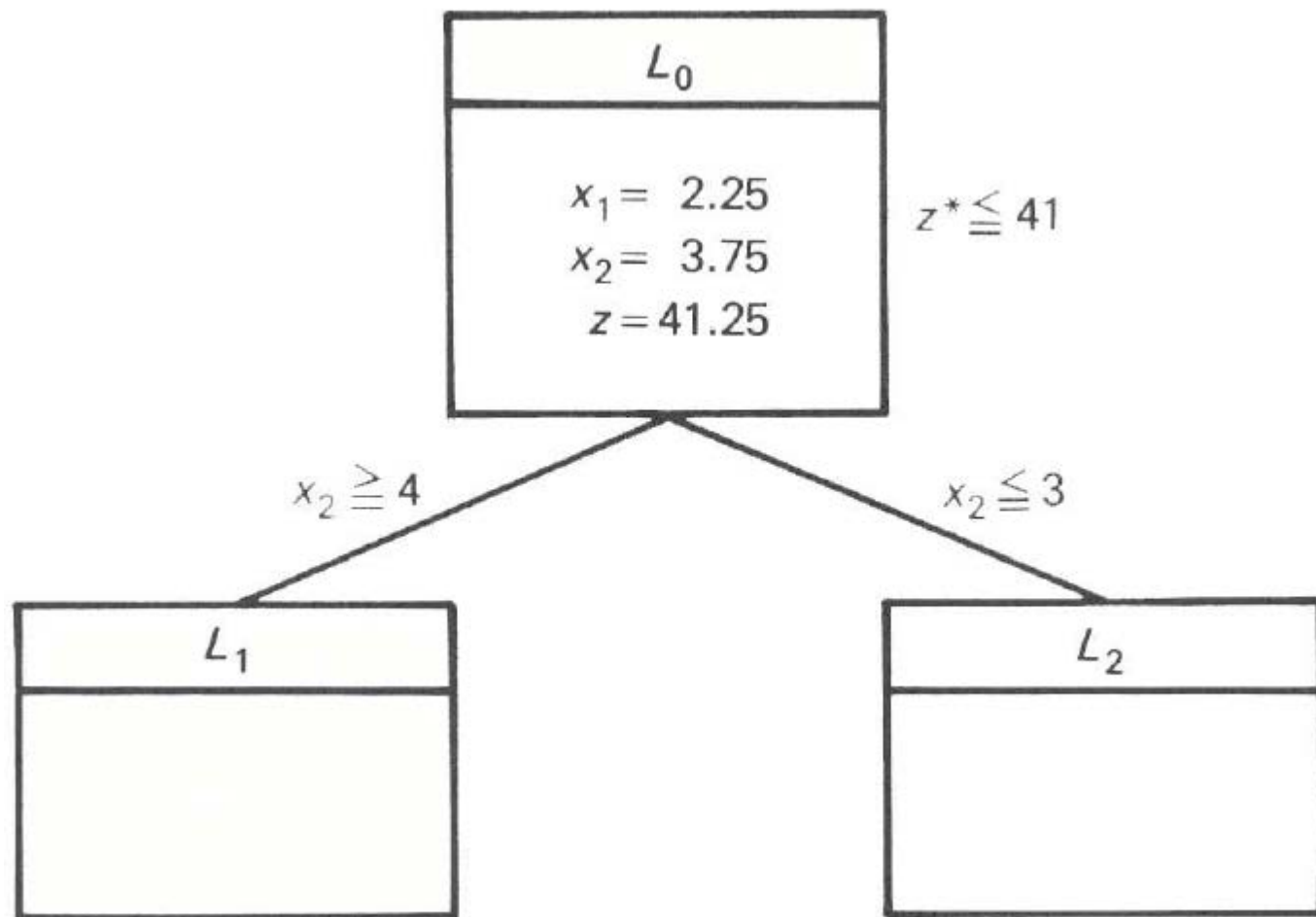
Somes notes

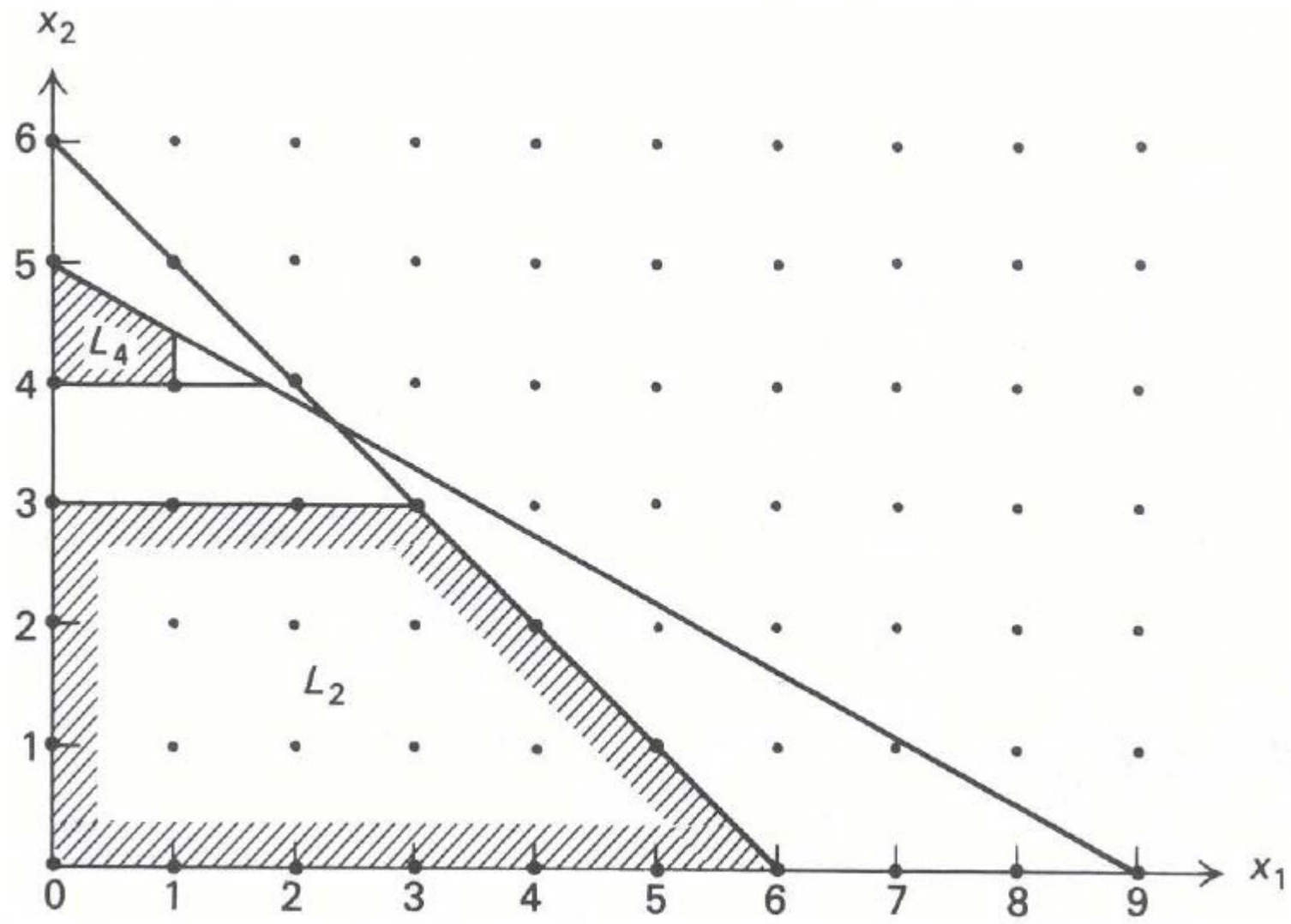
- If the integrality restrictions on variables are dropped, the resulting problem is a linear program. We will call it the **associated linear program**.
- The optimal integer-programming solution is not obtained by *rounding* the linear-programming solution.
- The closest point to the optimal linear-program solution is not even feasible.
- The nearest feasible integer point to the linear-program solution is far removed from the optimal integer point.
- **It is not sufficient simply to round linear-programming solutions.**
- Systematic and explicit enumeration does not work even for small problems.

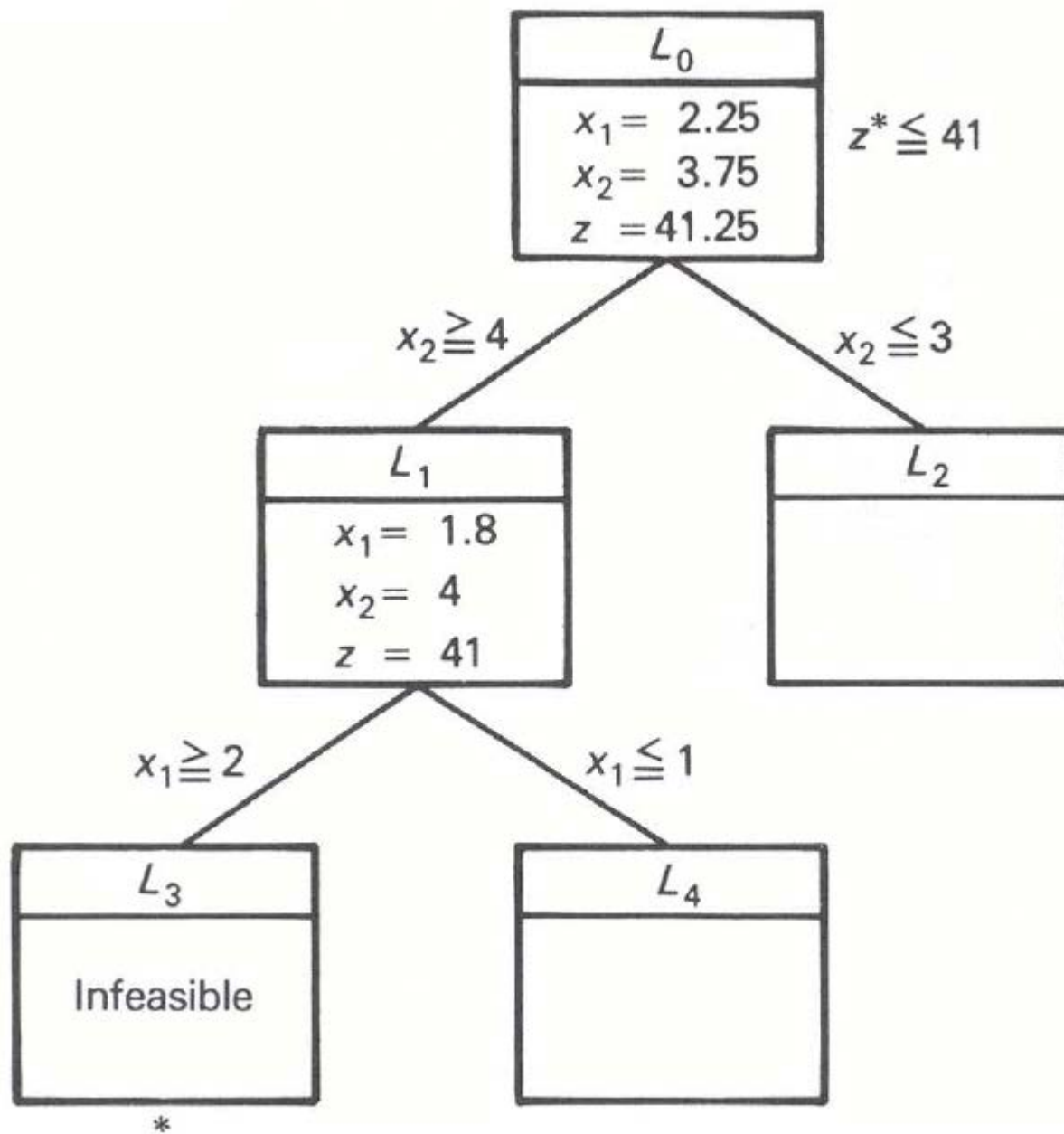
BRANCH-AND-BOUND

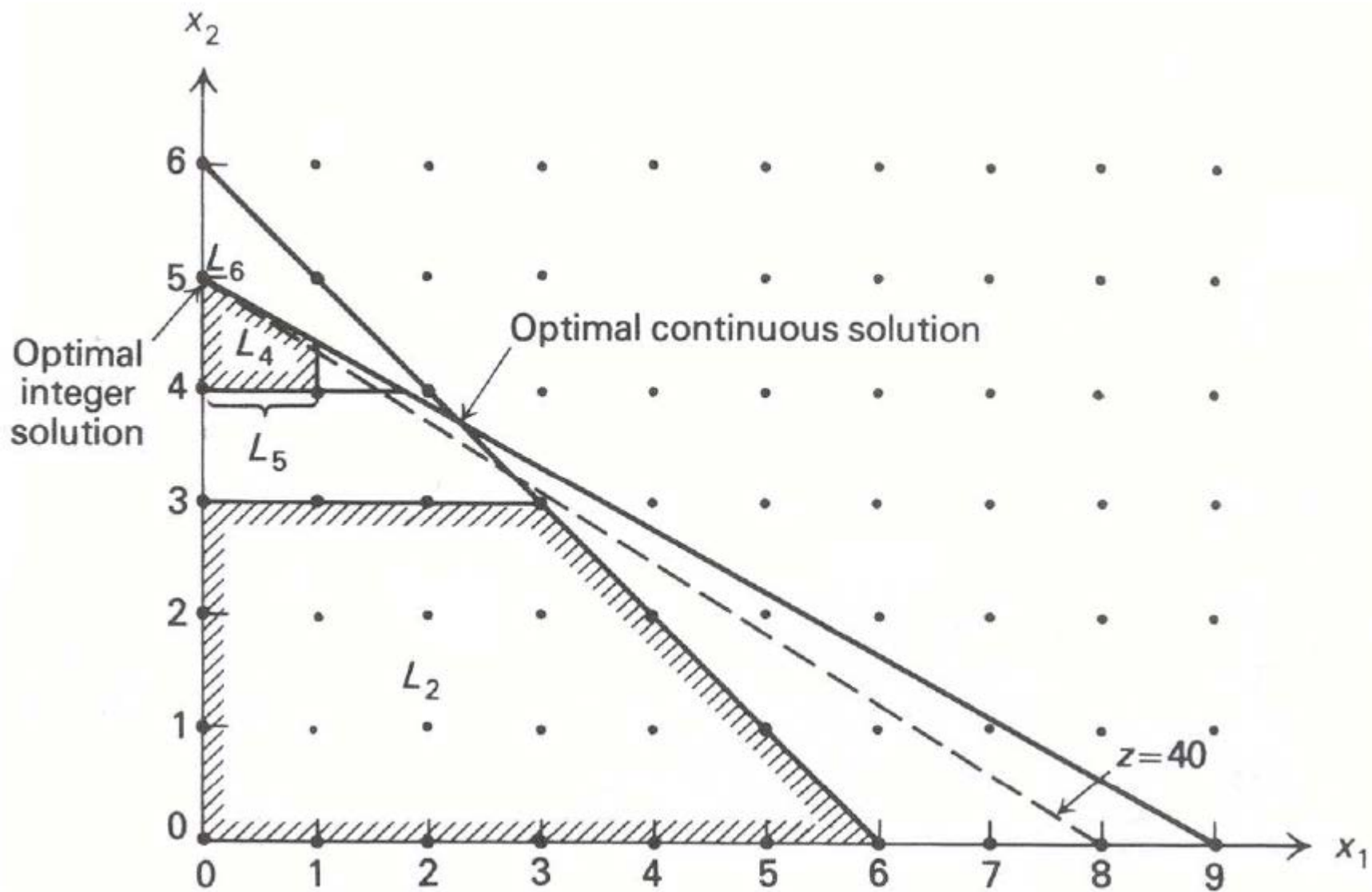
Strategy of “divide and conquer”

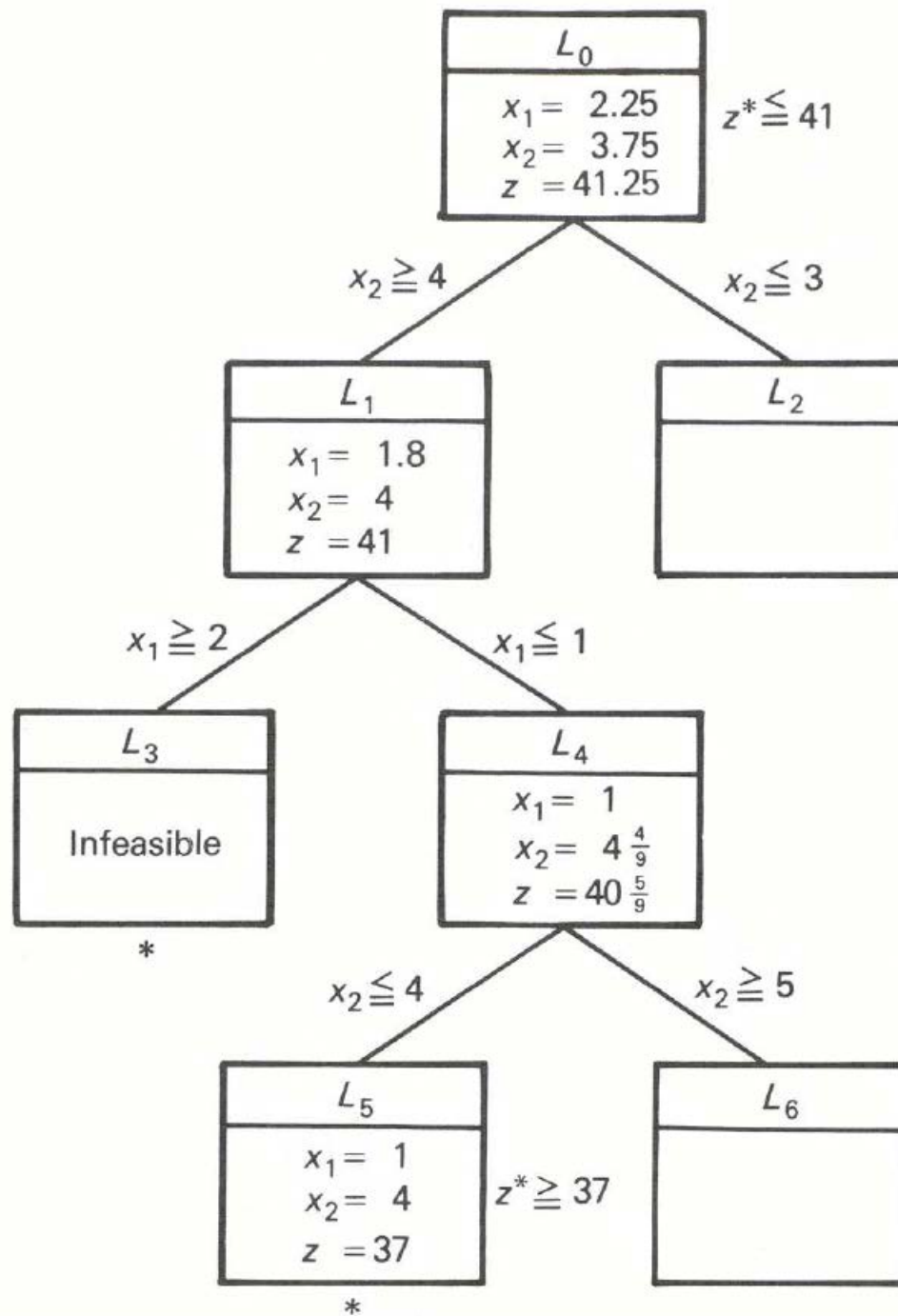
- An integer linear program is a linear program further constrained by the integrality restrictions.
- Thus, in a **maximization problem**, the value of the objective function, at the linear-program optimum, will always be an **upper bound** on the optimal integer-programming objective.
- In addition,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

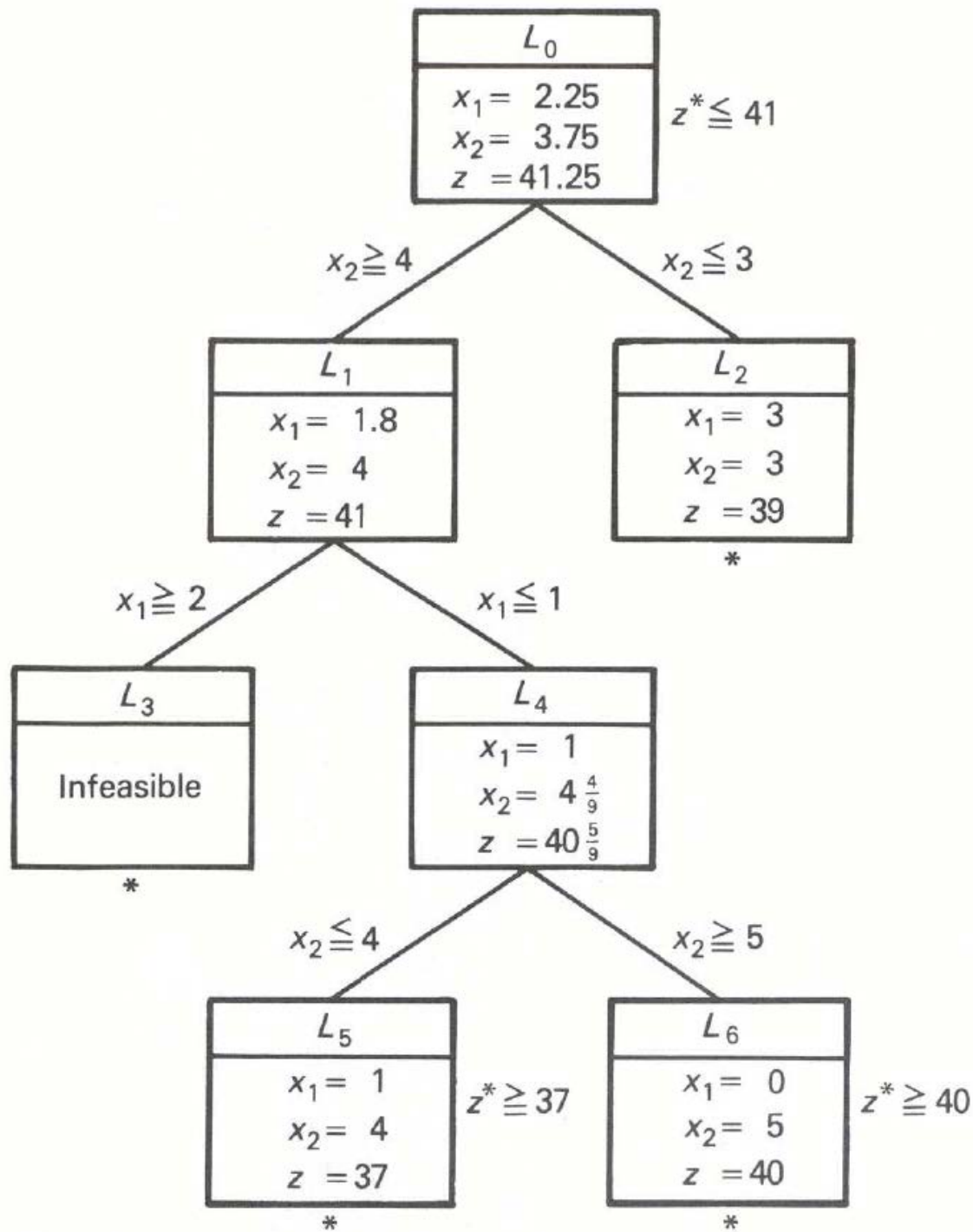












Further Considerations

- 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?

Can the linear programs corresponding to the subdivisions be solved efficiently?

- Dual simplex method!
 - Adding the new constraint:

$$x_2 \geq 4 \quad \text{or} \quad x_2 - s_3 = 4 \quad (s_3 \geq 0)$$

- Results in:

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

$$\begin{array}{rcl} & -x_2 & + s_3 = -4, \quad \text{Added constraint} \\ x_1, x_2, s_1, s_2, s_3 & \geq 0, & \end{array}$$

- We must pivot to isolate x_2 in the second constraint and re-express the system as:

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

and perform a dual simplex method iteration.

What is the best way to subdivide a given region, and which unanalyzed subdivision should be considered next?

- If we can make our choice of subdivisions in such a way as to rapidly obtain a good (with luck, near-optimal) integer solution z' , then we can **eliminate many potential subdivisions** immediately
 - as if any region has its linear programming value $z < z'$, then the objective value of no integer point in that region can exceed 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.
 - on a last-generated–first-analyzed basis (depth-first).
- Rules for determining which fractional variables to use in constructing subdivisions are more subtle...

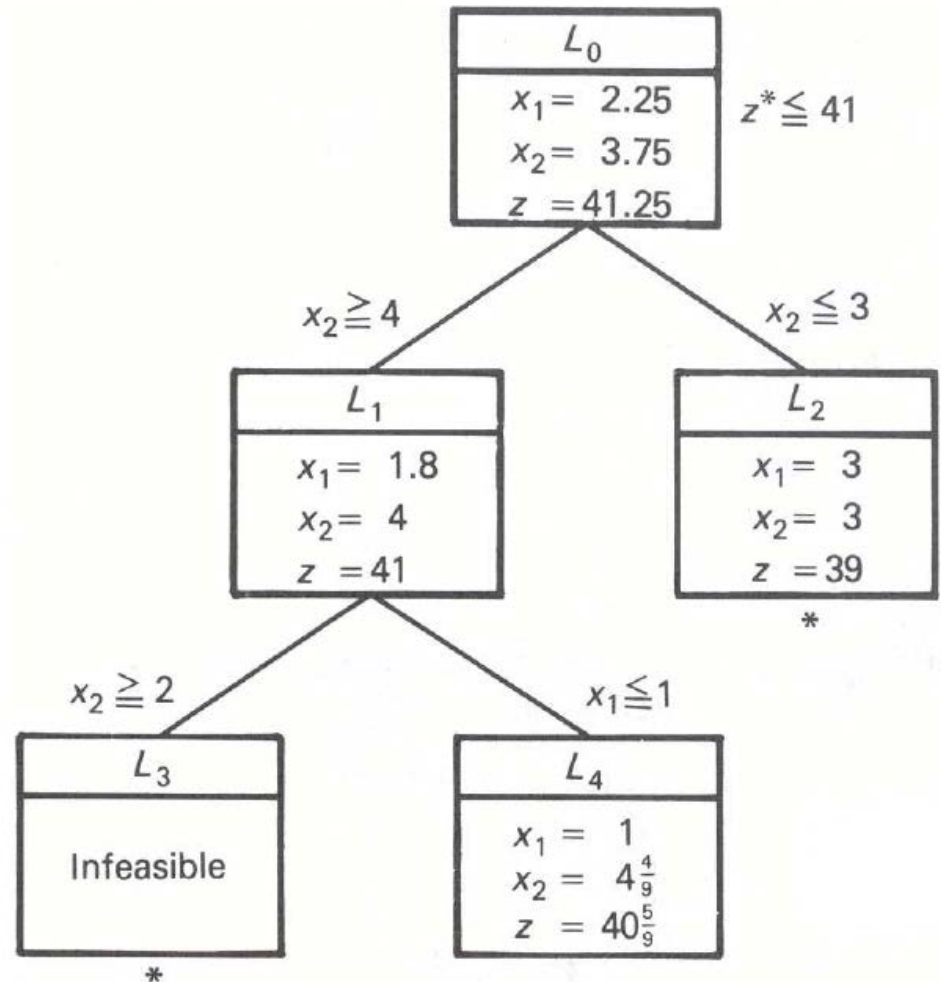
Can the upper bound on the optimal value z^* of the integer program be improved while the problem is being solved?

- Initial upper bound = 41.25

41 if we recall that the objective function coefficients are also integer.

- Current upper bound = $40 \frac{5}{9}$

40, for the same reasons.



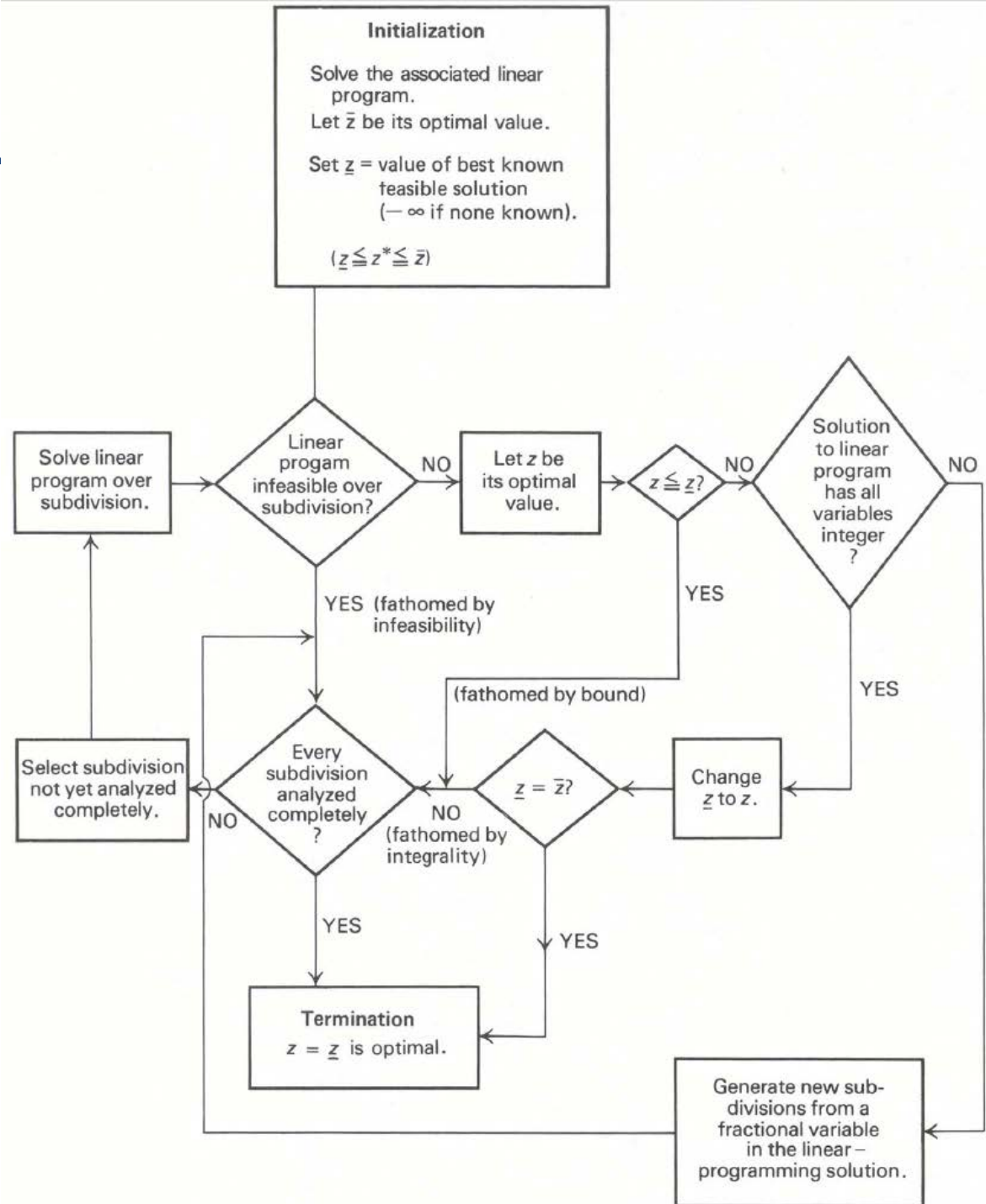
Summary

- The essential idea of branch-and-bound is to subdivide the feasible region to develop **bounds** $z_L < z^* < z^U$ on z^* .
- For a maximization problem, the **lower bound** z_L 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.

Summary

- If *either*
 - 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 $z^j \leq z_L$ (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.

Branch-and-bound flowchart



BRANCH-AND-BOUND FOR MIXED-INTEGER PROGRAMS

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.

Example

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

subject to:

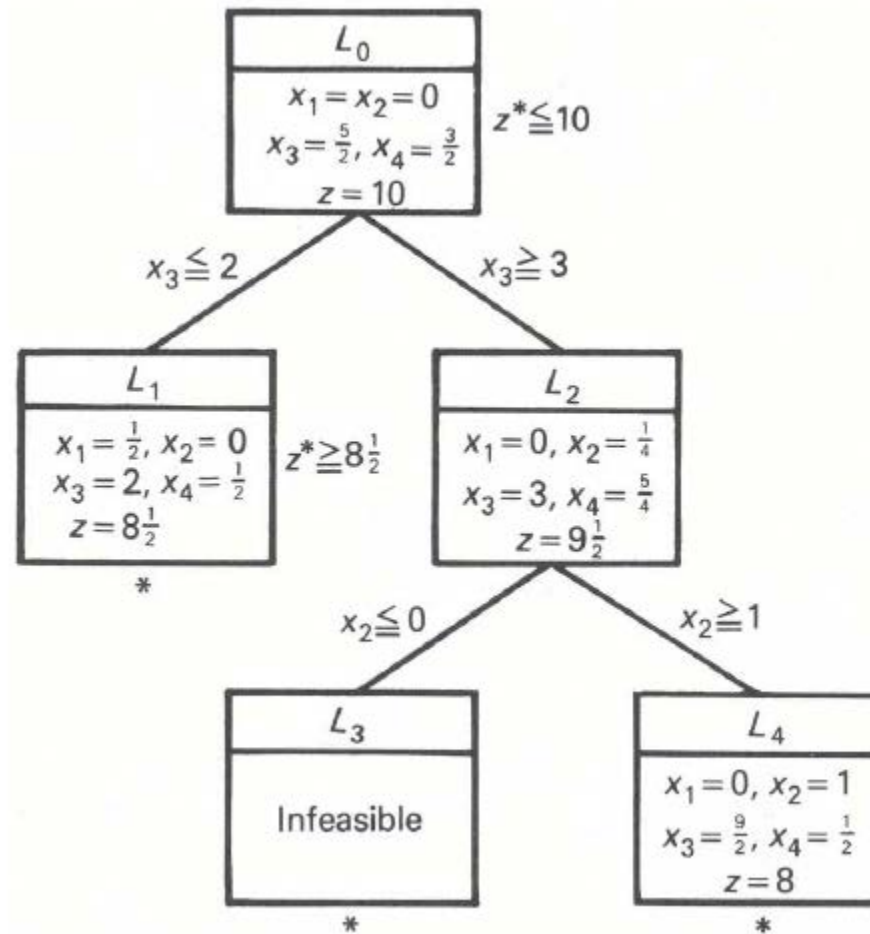
$$x_1 - 2x_2 + x_3 = \frac{5}{2},$$

$$2x_1 + x_2 + x_4 = \frac{3}{2},$$

$$x_j \geq 0 \quad (j = 1, 2, 3, 4),$$

x_2 and x_3 integer.

Search Tree



Simplex tableaus

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	$-\frac{1}{2}$	$\ominus 1$	2			1

↑

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

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

Simplex tableaux

L_2

<i>Basic variables</i>	<i>Current values</i>	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	$\ominus 2$			1

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

↑

<i>Basic variables</i>	<i>Current values</i>	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			$-\frac{1}{2}$

Implicit enumeration

BINARY BRANCH-AND-BOUND

A special branch-and-bound procedure for integer programs with only binary variables

- The algorithm has the advantage that it requires no linear-programming solutions.
- In the ordinary branch-and-bound procedure, subdivisions were analyzed by maintaining the linear constraints and dropping the integrality restrictions – **relax the variables**
- Here, we adopt the opposite tactic of always maintaining the 0–1 restrictions, but ignoring the linear inequalities – **relax the constraints**

General idea

- 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, when the inequality constraints are ignored, the objective function is maximized by setting the free variables to zero, since their **objective function coefficients are negative**.
 - This is mandatory to apply this algorithm.

Example

maximization problem

all coefficients are negative

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

subject to:

$$-3x_1 - 3x_2 + x_3 + 2x_4 + 3x_5 \leq -2,$$

$$-5x_1 - 3x_2 - 2x_3 - x_4 + x_5 \leq -4,$$

$$x_j = 0 \quad \text{or} \quad 1 \quad (j = 1, 2, \dots, 5).$$

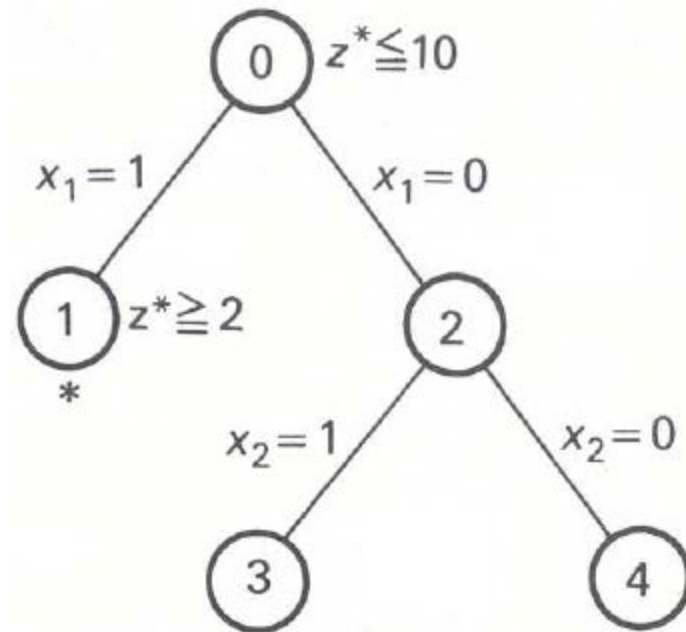
constraints are specified as
“less than or equal to”
inequalities

- 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. 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
$$z = -8(1) - 2(0) - 4(0) - 5(0) + 10 = 2$$
and happens to satisfy the inequalities, so 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 $x_1 = 1$.
 - No other feasible 0–1 combination in subdivision 1 needs to be evaluated explicitly.
 - These combinations have been considered implicitly.

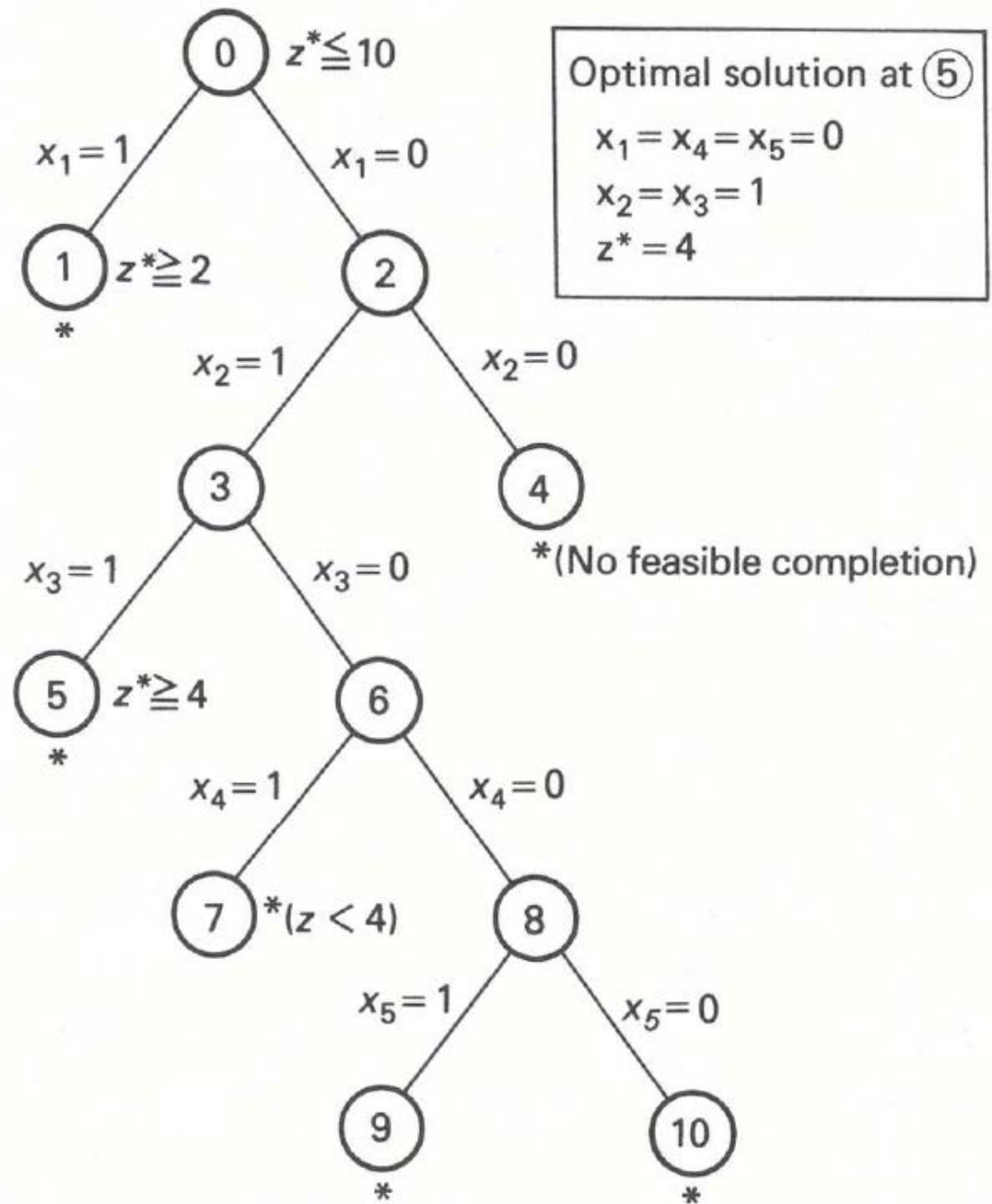
- The solution with $x_2 = x_3 = x_4 = x_5 = 0$ in subdivision 2 is the same as the original solution with every variable at zero, and is infeasible.
- Consequently, the region must be subdivided further, say with $x_2 = 1$ or $x_2 = 0$, giving:
 - For subdivision 3 : $x_1 = 0, x_2 = 1$;
 - For subdivision 4 : $x_1 = 0, x_2 = 0$.

Enumeration tree to this point



- Completing it...

Complete tree



Note that

1. 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 .
2. 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.
3. 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.

Note that

4. 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.

Generalizing

- Subdivisions are fathomed if any of three conditions hold:
 - i. the integer program is known to be infeasible over the subdivision, for example, by the previous 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.

Once again

- This algorithm is designed for problems in which:
 1. the objective is a maximization with all coefficients negative; and
 2. constraints are specified as “less than or equal to” inequalities.
- If not, then...

Problem transformation

- 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.
- “Greater than or equal to” constraints can be multiplied by -1 to become “less than or equal to” constraints.