Unit.9 Integer Programming

Xiaoxi Li

EMS & IAS, Wuhan University

Dec. 22 - 29, 2016 (revised)
Contents:

- Integer Programming: Modeling Approach
  - Basic Models and Classifications: IP, MIP, BIP
  - A Prototype Example and Some BIP Applications
  - Some Innovative Uses of Binary Variables in Model Formulation
  - Some Formulation Examples

- Solving an IP (1): the Branch-and-Bound Algorithm
  - The Branch-and-Bound Algorithm for BIP
  - The Branch-and-Bound Algorithm for MIP

- Solving an IP (2): the Cutting Plane Approach
The notion of Integer Programming (IP) has the following two senses.

- The *broad* sense. An IP is a Mathematical Programming (LP or Non LP) with the restrictions that some decision variables take integer values.
- The *narrow* sense: An IP is a LP with the restrictions that some decision variables take integer values.

In this course, we restrict ourselves to the *narrow* sense. Thus a LP can be regarded as a *relaxation* of some IP.
Classifications of IP:

- MIP: some variables are asked to take integer values
- Pure IP: all variables are asked to take integer values
- BIP: all variables are asked to take 0-1 values
A Prototype Example with BIP Modeling: The California Manufacturing Company

- The California Manufacturing Company is a company with factories and warehouses throughout California.
- It is currently considering whether to build a new factory in Los Angeles and/or San Francisco.
- Management is also considering building one new warehouse where a new factory has been recently built.
- Should the CMC build factories and/or warehouses in Los Angeles and/or San Francisco?
A Prototype Example with BIP Modeling: The California Manufacturing Company (cont.)

There are (at most) four possible decisions to make:

- To build a factory in LA? \( (x_1) \)
- To build a factory in SF? \( (x_2) \)
- To build a warehouse in LA (if a factory has been built there)? \( (x_3) \)
- To build a warehouse in SF (if a factory has been built there)? \( (x_4) \)

So the decision variables take the *binary* form, to be *Yes* or *No*:

\[
x_j = \begin{cases} 
1 & \text{if decision } j \text{ is yes,} \\
0 & \text{if decision } j \text{ is no,} 
\end{cases} 
(j = 1, 2, 3, 4).
\]
The following restrictions exist amongst $x_j$:

- $x_3$ and $x_4$ are **mutually exclusive alternatives**: $x_3 + x_4 \leq 1$;  
  (since at most ONE warehouse is to be constructed)

- $x_3$ is a **contingent variable** on $x_1$: $x_3 \leq x_1$ (same for (resp. $x_4 \leq x_2$)).  
  (a warehouse is constructed only where a factory is constructed)
A Prototype Example with BIP Modeling: The California Manufacturing Company (cont.)

The BIP for the problem is as follows:

$$\text{Max } Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$$

subject to:

$$\begin{align*}
6x_1 + 3x_2 + 5x_3 + 2x_4 &\leq 10 \\
x_3 + x_4 &\leq 1 \\
x_3 &\leq x_1 \\
x_4 &\leq x_2 \\
x_j &\in \{0, 1\}, \ j = 1, 2, 3, 4.
\end{align*}$$

TABLE 12.1 Data for the California Manufacturing Co. example

<table>
<thead>
<tr>
<th>Decision Number</th>
<th>Yes-or-No Question</th>
<th>Decision Variable</th>
<th>Net Present Value</th>
<th>Capital Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Build factory in Los Angeles?</td>
<td>$x_1$</td>
<td>$9 \text{ million}$</td>
<td>$6 \text{ million}$</td>
</tr>
<tr>
<td>2</td>
<td>Build factory in San Francisco?</td>
<td>$x_2$</td>
<td>$5 \text{ million}$</td>
<td>$3 \text{ million}$</td>
</tr>
<tr>
<td>3</td>
<td>Build warehouse in Los Angeles?</td>
<td>$x_3$</td>
<td>$6 \text{ million}$</td>
<td>$5 \text{ million}$</td>
</tr>
<tr>
<td>4</td>
<td>Build warehouse in San Francisco?</td>
<td>$x_4$</td>
<td>$4 \text{ million}$</td>
<td>$2 \text{ million}$</td>
</tr>
</tbody>
</table>

Capital available: $10 \text{ million}$
Some Other Applications

- Investment Analysis
- Site Selection
- Designing a Production and Distribution Network
- Dispatching Shipments
- Scheduling Interrelated Activities
- Airline Application
- ...

Operations Research (Li, X.)
Unit.9 Integer Programming
Dec. 22 - 29, 2016 (revised)
Some Innovative Uses of Binary Variables in Model Formulation

To name a few:

- Either-Or Constraint
- $K$ out of $N$ Constraints Must Hold
- Functions with $N$ Possible Values
- The Fixed-Charge Problem
Either-Or Constraint

- Suppose you have a situation where you have two potential constraints, but only one of them can hold. For example:
  - either "5x_1 + 9x_2 \leq 24" (Material 1 be used for production) – ①
  - or "8x_1 + 6x_2 \leq 35" (Material 2 be used for production) – ②

- To handle this, we introduce a binary variable \( y_1 \), a big \( M > 0 \), and add to one constraint the term \( My_1 \) to the RHS; add to the other constraint the term \( M(1 - y_1) \) to the RHS.

- To obtain: "5x_1 + 9x_2 \leq 24 + My_1 \) and \( 8x_1 + 6x_2 \leq 35 + M(1 - y_1)".

Why they are equivalent?

- When \( y_1 = 0 \), Constraint ① is active and Constraint ② is automatically satisfied for any feasible solution (always bounded);
- When \( y_1 = 1 \), Constraint ② is active and Constraint ① is automatically satisfied for any feasible solution (always bounded).
There are $N$ constraints among which only $K$ (must) hold ($K < N$).

Introduce binary variables $y_i$, $i = 1, \ldots, N$; add to Constraint (i) the term $My_i$, and another constraint $\sum_{i=1}^{N} y_i = N - K$.

For an example ($M = 3$ and $K = 1$):

- $5x_1 + 12x_2 \leq 34 + My_1$
- $6x_1 + 11x_2 \leq 43 + My_2$
- $7x_1 + 10x_2 \leq 57 + My_3$
- $y_1 + y_2 + y_3 = 1$

Why they are equivalent?

- The same reasoning as the Either-Or Constraint ($M = 2$, $K = 1$);
- "$y_i = 0$" corresponds to the case that "Constraint (i)" is active, so $\sum_{i=1}^{N} y_i = N - K$ is the number of constraints that do not hold.
Functions with $N$ Possible Values

- An example: the constraint is "$5x_1 + 31x_2 = 25$ or $50$ or $75$".
- We introduce binary variables $y_1$, $y_2$ and $y_3$ to obtain the equivalent:

  $$5x_1 + 31x_2 = 25y_1 + 50y_2 + 75y_3 \text{ and } y_1 + y_2 + y_3 = 1.$$  

- Indeed, "the RHS taking the $i$-th value" corresponds to the case

  "$y_i = 1$ and $y_j = 0, \forall j \neq i$."
The Fixed-Charge Problem

- The cost function for producing activity \( j \) is

\[
f_j(x_j) = \begin{cases} 
  k_j + c_j x_j & \text{if } x_j > 0 \\
  0 & \text{if } x_j = 0 
\end{cases}
\]

where \( x_j \) is the level of activity \( j \), \( k_j \) is the fixed (setup) cost and \( c_j \) is the marginal cost.

- The objective function is not linear (mainly at the point 0).

- To handle the issue, we introduce for each \( j \)
  - the binary variable \( y_j \) for the decision "if activity \( j \) is produced";
  - the continuous variable \( x_j \) for the level of activity \( j \).

  Thus \( y_j \) is seen as a contingent variable of \( x_j \), i.e. "\( y_j = 1 \) iff \( x_j > 0 \)".
Here, it is somehow more complicated (than the previous case) because \( x_j \) can take very large value (a priori no given bounded).

- Let \( M > 0 \) be sufficiently large that exceeds the maximum feasible value of any \( x_j, \ j = 1, ..., n \).
- Introduce the constraints "\( x_j \leq My_j, \ j = 1, ..., n \)".
- The objective function is \( Z = \sum_{j=1}^{n}(c_jx_j + k_jy_j) \).

Why they are equivalent to the constraints "\( y_j = 1 \iff x_j > 0, \ j = 1, ..., n \)"?

- First of all, "\( x_j \leq My_j \)" is
- It is straightforward that "\( x_j \leq My_j \)" implies "\( y_j = 1 \) if \( x_j > 0 \);
- On the other hand, if \( x_j = 0 \), \( y_j \) could be 0 or 1 as a feasible solution. However, by the objective function, at optimum, we should have \( y_j = 0 \) whenever \( x_j = 0 \).

This leads us to a MIP.
Example 1: Making Choices When the Decision Variables Are Continuous

- Objective: Maximize Profits
- Decision Variables: Product 1, Product 2, and Product 3
- Constraints:
  - Production time available for Plants 1 and 2
  - At most two out of the three products can be produced
  - Only one of the two plants can produce the new products
Example 1: Making Choices When the Decision Variables Are Continuous (cont.)

The data for the problem:

<table>
<thead>
<tr>
<th>TABLE 12.2 Data for Example 1 (the Good Products Co. problem)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Production Time Used for Each Unit Produced</strong></td>
</tr>
<tr>
<td><strong>Product 1</strong></td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Plant 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Plant 2</td>
</tr>
<tr>
<td>Unit profit</td>
</tr>
<tr>
<td>Sales potential</td>
</tr>
</tbody>
</table>
Example 1: Making Choices When the Decision Variables Are Continuous (cont.)

We construct the following MIP for the problem:

Max \[ Z = 5x_1 + 7x_2 + 3x_3 \]

\[
\begin{align*}
    & x_1 - My_1 \leq 0, \quad x_2 - My_2 \leq 0, \quad x_3 - My_3 \leq 0 \\
    & y_1 + y_2 + y_3 \leq 2 \\
    & x_1 \leq 7, \quad x_2 \leq 5, \quad x_3 \leq 9 \\
    & 3x_1 + 4x_2 + 2x_3 \leq 30 + Mz_1 \\
    & 4x_1 + 6x_2 + 2x_3 \leq 30 + M(1 - z_1) \\
    & x_1, x_2, x_3 \geq 0, \quad y_1, y_2, y_3, z_1 \in \{0, 1\}.
\end{align*}
\]

**Interpretations.** In class.
Example 2: Proportionality Assumption Violated

- **Objective:** Maximize Profits
- **Decision Variables:** Number of TV spots for Product 1, Product 2, and Product 3
- **Constraints:** Number of TV spots allocated to the three products cannot be more than five
- **Major Issue:** Proportionality Assumption is violated
Example 2: Proportionality Assumption Violated (cont.)

The data for the problem:

<table>
<thead>
<tr>
<th>Number of TV Spots</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE 12.3** Data for Example 2 (the Supersuds Corp. problem)

*Note: Proportionality Assumption Violated!*
Example 2: Proportionality Assumption Violated (cont.)

We construct the following MIP for the problem (solution 1):

Max \[ Z = y_{11} + 3y_{12} + 3y_{13} + 2y_{22} + 3y_{23} - y_{31} + 2y_{32} + 4y_{33} \]
\[
\begin{align*}
    y_{11} + y_{12} + y_{13} &\leq 1 \\
y_{21} + y_{22} + y_{23} &\leq 1 \\
y_{31} + y_{32} + y_{33} &\leq 1 \\
y_{11} + 2y_{12} + 3y_{13} + y_{21} + 2y_{22} + 3y_{23} + y_{31} + 2y_{32} + 3y_{33} &= 5 \\
y_{ij} \in \{0, 1\}, i,j = 1,2,3.
\end{align*}
\]

For each \(i\), the auxiliary binary variable \(y_{ij}\) is introduced for each possible value "j" of the variable \(x_i\) i.e. \(y_{ij} = 1\) if \(x_i = j\) and \(y_{ij} = 0\) if \(x_i \neq j\).

**Interpretations.** In class.
Example 2: Proportionality Assumption Violated (cont.)

We construct the following MIP for the problem (solution 2):

Max \[ Z = y_{11} + 2y_{12} + 2y_{22} + y_{23} - y_{31} + 3y_{32} + 2y_{33} \]

\[ \begin{align*}
  y_{12} &\leq y_{11}, \quad y_{13} \leq y_{12} \\
  y_{22} &\leq y_{21}, \quad y_{23} \leq y_{22} \\
  y_{32} &\leq y_{31}, \quad y_{33} \leq y_{32} \\
  y_{11} + y_{12} + y_{13} + y_{21} + y_{22} + y_{23} + y_{31} + y_{32} + y_{33} &= 5 \\
  y_{ij} &\in \{0, 1\}, \ i, j = 1, 2, 3.
\end{align*} \]

For each \( i \), the auxiliary binary variable \( y_{ij} \) is introduced for possible value no smaller than "\( j \)" of the variable \( x_i \), i.e. \( y_{ij} = 1 \) if \( x_i \geq j \) and \( y_{ij} = 0 \) if \( x_i < j \). \( \implies \) \( y_{ij} \) are the increments and \( x_i = y_{i1} + y_{i2} + y_{i3} \).

Interpretations. In class.
Exercise.
Some general comments on solving IP.

- **Solving an IP is NOT easier than solving a LP as was thought.**
  - **Having a finite number of feasible solutions ensures that it is readily solvable.** Wrong!
  - **Removing some feasible solutions (the non-integer ones) from a LP makes an IP easier to solve.** Wrong!

- The fact that an IP is in general more difficult to solve than a LP (relaxation) can be understood as follows.
  - The simplex method (solving LP) visits only the corner-point feasible solutions, which generally have a much smaller number than the integer feasible solutions.
  - Most successful IP algorithms incorporate a LP algorithm by relating an IP to its LP relaxation.
A LP relaxation of an IP is to keep all the original program except for removing the integer-value restrictions on variables.

- Consider a maximization problem. The value of a LP relaxation is always NO SMALLER than that of an IP. Thus solving a corresponding LP relaxation gives a upper bound for an IP.

- It is fortuitous if a LP relaxation’s optimal solution happens to be integer values, in which case the solution is also optimal to the IP. However, this is in general not guaranteed.

- Nevertheless, in some specific applications, it is fortuitous to have this solution structure: minimum cost flow problem and its special cases (transportation problem, assignment problem, shortest-path problem, maximum flow problem).
One temptation for solving an IP is to first solve its LP relaxation, and then *round* the optimal solution to integer values.

- This is an *approximation* method which may not always lead to an optimal solution. The approximation might be good if the variables’ values are quite large.

- But in general, it is not accepted due to the following two pitfalls:
  - The rounded "optimal" solution may not be feasible.
  - Even if feasibility is not a problem, the rounded "optimal" solution may be far from optimal.
Pitfall 1. The rounded "optimal" solution may not be feasible.
Pitfall 2. The rounded "optimal" solution may be far from optimal.

**FIGURE 12.3**
An example where rounding the optimal solution for the LP relaxation is far from optimal for the IP problem.
The Branch-and-Bound Algorithm: a motivating example of *Poker*.

- **Objective.** Within a random bundle of Poker, find out one card with the *highest value* and has a *red-heart* format.
- **Tool.** Among any bundle of cards, a machine that is able to pick out the card(*) with the *highest value* (no ability to read the format).
- **An algorithm.**
  - Put the whole bundle into the machine and obtain the card(*) with the highest value. STOP if the card’s format is red-heart (the whole bundle is *discarded*); otherwise branch the bundle into two.
  - In a general step, for a newly branched bundle, detect it by the machine for a highest-valued card(*). Several possibilities:
    - *Discard* this bundle if this card(*) is red-heart, and call it *incumbent* if its value is higher than the current incumbent one;
    - *Discard* this bundle if this card(*)’s value is *lower* than the current incumbent one;
    - Otherwise, brunch the bundle for further detection by machine.
  - STOP when all cards are discarded. Incumbent = optimal value.
Objective. For a LP problem, find out one feasible solution with the highest value and take binary values.

Tool. The simplex method to solve for a LP (relaxation).

An algorithm.

- Apply the simple method to the LP relaxation to solve for an optimal solution. STOP if the optimal solution take binary values; otherwise, branch the problem into two subproblems by letting $x_1 = 0$ or $x_1 = 1$.
- In a general step, for a newly branched subproblem, solve its LP relaxation by the simplex method. Several possibilities:
  - **Discard (fathom)** this subproblem if this optimal solution take integer values, and call this solution incumbent if its value is higher than the current incumbent one;
  - **Discard** this subproblem if this optimal value is no larger than the current incumbent one or if it is infeasible;
  - Otherwise, branch the subproblem by $x_j = 0$ or $x_j = 1$ ($j$ in order).
- STOP when all subproblems are either branched or fathomed.
- Incumbent = optimal solution.
The Branch-and-Bound Algorithm to solving an BIP: an example.

The (whole) problem is

\[
\begin{align*}
\text{Max} & \quad Z = 9x_1 + 5x_2 + 6x_3 + 4x_4 \\
\text{s.t.} & \quad \begin{cases} 
6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\
x_3 + x_4 \leq 1 \\
-x_1 + x_3 \leq 0 \\
-x_2 + x_4 \leq 0 \\
x_j \in \{0, 1\}, \; j = 1, 2, 3, 4.
\end{cases}
\end{align*}
\]
Branching. Branch the BIP into two by letting $x_1 = 0$ or $x_1 = 0$.

Subproblem 1 ($x_1 = 0$).

Maximize $Z = 5x_2 + 6x_3 + 4x_4$

subject to

\[
\begin{align*}
3x_2 + 5x_3 + 2x_4 & \leq 10 \\
-x_2 + x_4 & \leq 0 \\
x_3 + x_4 & \leq 1 \\
x_3 & \leq 0 \\
x_j & \in \{0, 1\}, j = 2, 3, 4.
\end{align*}
\]
**Branching.** Branch the BIP into two by letting $x_1 = 0$ or $x_1 = 0$.

**Subproblem 2 ($x_1 = 1$).**

Maximize

$$Z = 9 + 5x_2 + 6x_3 + 4x_4$$

subject to

$$\begin{align*}
3x_2 + 5x_3 + 2x_4 &\leq 4 \\
x_3 + x_4 &\leq 1 \\
x_3 &\leq 1 \\
x_2 + x_4 &\leq 0 \\
x_j &\in \{0, 1\}, j = 2, 3, 4.
\end{align*}$$
**Branching.** Branch the BIP into two by letting $x_1 = 0$ or $x_1 = 0$. The **branching tree** created by the branching for the first iteration. 

$x_1$ is called the **branching variable**.
Bounding. This procedure involves solving the LP relaxation of a subproblem so as to obtain an upper bound on the BIP’s optimal value. Consider for example the whole problem. Its LP relaxation is

\[
\text{Max } Z = 9x_1 + 5x_2 + 6x_3 + 4x_4 \\
\left\{
\begin{array}{l}
6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\
x_3 + x_4 \leq 1 \\
-x_1 + x_3 \leq 0 \\
-x_2 + x_4 \leq 0 \\
0 \leq x_j \leq 1, \ j = 1, 2, 3, 4.
\end{array}
\right.
\]

Solving it by simplex method: \((x_1, x_2, x_3, x_4) = (\frac{5}{6}, 1, 0, 1)\) with \(Z = 16\frac{1}{2}\).

- Therefore any feasible solution for BIP has a value \(\leq 16\frac{1}{2}\).
- Coefficients in the objective function are integers, so \(16\frac{1}{2}\) can be rounded to 16.
Bounding. Now we use the similar approach to bound the Subproblem 1 \((x_1 = 0)\) and the Subproblem 2 \((x_1 = 1)\).

**Subproblem 1 relaxed \((x_1 = 0)\).**

\[
\begin{align*}
\text{Max } & \quad Z = 5x_2 + 6x_3 + 4x_4 \\
\text{s.t. } & \quad \begin{cases} 
3x_2 + 5x_3 + 2x_4 \leq 10 \\
x_3 + x_4 \leq 1 \\
x_3 \leq 0 \\
x_2 - x_4 \leq 0 \\
0 \leq x_j \leq 1, \ j = 2, 3, 4.
\end{cases}
\end{align*}
\]

Solving it by simplex method: \((x_1, x_2, x_3, x_4) = (0, 1, 0, 1)\) with \(Z = 9\).
Bounding.

Subproblem 1 relaxed ($x_1 = 1$).

Maximize $Z = 9 + 5x_2 + 6x_3 + 4x_4$

Subject to:

\[
\begin{align*}
3x_2 + 5x_3 + 2x_4 & \leq 4 \\
x_3 + x_4 & \leq 1 \\
x_3 & \leq 1 \\
x_2 - x_4 & \leq 0 \\
0 & \leq x_j \leq 1, \quad j = 2, 3, 4.
\end{align*}
\]

Solving it by simplex method: $(x_1, x_2, x_3, x_4) = (1, \frac{4}{5}, 1, \frac{4}{5})$ with $Z = 16\frac{1}{5}$. 
Fathoming.

- The Whole Problem: not fathomed because its LP relaxation has an optimal (*feasible*) solution which is *not binary valued*, and there is *no previous incumbent value* (think to be $Z^* - \infty$).

- The Subproblem 1 ($x_1 = 0$): fathomed because its LP relaxation has an optimal solution that is *binary valued*. Moreover, the solution $(0, 1, 0, 1)$ and its value $Z^* = 9$ are stored as the incumbent since there is no previous ones.

- The Subproblem 2 ($x_1 = 1$): not fathomed because its LP relaxation has an optimal (*feasible*) solution which is *not binary valued*, and its value $Z = 16$ is *higher than the incumbent value* $Z^* = 9$. 
Fathoming.

General rules for fathoming test. A (sub)problem is fathomed if it satisfies one of the following three.

1. Its bound \( \leq Z^* \);
2. Its LP relaxation has no feasible solution.
3. Its LP relaxation has integer (binary) solution (which is stored as the incumbent if its value is higher than the current incumbent one).
Solving an IP: the Branch-and-Bound Algorithm (cont.)

**FIGURE 12.5**
The results of bounding for the first iteration of the BIP branch-and-bound algorithm for the example in Sec. 12.1.

Variable:

- $x_1 = 0$
  - 9
  - (0, 1, 0, 1)

- $x_1 = 1$
  - 16
  - ($\frac{5}{6}, 1, 0, 1$)

**FIGURE 12.6**
The branching tree after the first iteration of the BIP branch-and-bound algorithm for the example in Sec. 12.1.

Variable:

- $x_1 = 0$
  - $F(3)$
  - 9 = $Z^*$
  - (0, 1, 0, 1) = incumbent

- $x_1 = 1$
  - 16

All
Completing the example (after Iteration 1)

**Iteration 2.** Subproblem 3 \((x_1 = 1, x_2 = 0)\).

\[
\begin{align*}
\text{Max} & \quad Z = 9 + 6x_3 + 4x_4 \\
\text{s.t.} & \quad \begin{cases}
5x_3 + 2x_4 \leq 10 \\
x_3 + x_4 \leq 1 \\
x_3 \leq 1 \\
x_4 \leq 0 \\
x_j \in \{0, 1\}, \; j = 3, 4.
\end{cases}
\end{align*}
\]

Its LP relaxation is to replace \(x_j \in \{0, 1\}\) by \(0 \leq x_j \leq 1\) for \(j = 3, 4\).

- Solving it by simplex method: \((x_1, x_2, x_3, x_4) = (1, 0, \frac{4}{5}, 0)\), which is not integer valued, and with \(Z = 13\frac{4}{5}\).
- Bound for Subproblem 3: \(Z \leq 13\), larger than the incumbent \(Z^* = 9\).
- Thus Subproblem 3 is not fathomed.
Completing the example (after Iteration 1)

Iteration 2. *Subproblem 4* \((x_1 = 1, x_2 = 1).\

Max \( Z = 14 + 6x_3 + 4x_4 \)

\[
\begin{align*}
5x_3 + 2x_4 & \leq 1 \\
x_3 + x_4 & \leq 1 \\
x_3 & \leq 1 \\
x_4 & \leq 1 \\
x_j & \in \{0, 1\}, \ j = 3, 4.
\end{align*}
\]

Its LP relaxation is to replace \( x_j \in \{0, 1\} \) by \( 0 \leq x_j \leq 1 \) for \( j = 3, 4 \).

- Solving it by simplex method: \((x_1, x_2, x_3, x_4) = (1, 1, 0, \frac{1}{2})\), which is not integer valued, and with \( Z = 16 \).
- Bound for *Subproblem 4*: \( Z \leq 16 \), larger than the incumbent \( Z^* = 9 \).
- Thus *Subproblem 4* is not fathomed.
Completing the example (after Iteration 2)

**FIGURE 12.7**
The branching tree after iteration 2 of the BIP branch-and-bound algorithm for the example in Sec. 12.1.

Variable:
- $x_1$
  - $x_1 = 0$
    - $F(3)$
    - $9 = Z^*$
    - $(0, 1, 0, 1) = \text{incumbent}$
  - $x_1 = 1$
    - $13$
    - $(1, 0, \frac{4}{5}, 0)$
    - $16$
    - $(1, 1, 0, \frac{1}{2})$
- $x_2 = 0$
- $x_2 = 1$

$16$
Completing the example (...after Iteration 3)

Variable:

- $x_1 = 0$ with $F(3)$
- $x_2 = 0$
- $x_3 = 0$

9 = $Z^*$

(0, 1, 0, 1) = incumbent

**FIGURE 12.8**
The branching tree after iteration 3 of the BIP branch-and-bound algorithm for the example in Sec. 12.1.
Completing the example (...after Iteration 4)

**FIGURE 12.9**
The branching tree after the final (fourth) iteration of the BIP branch-and-bound algorithm for the example in Sec. 12.1.
Completing the example: the Iteration Procedure

- **Operation 1. Bounding Problem [all]**, to obtain the solution $X_{all} = \left( \frac{5}{6}, 1, 0, 1 \right)$ and the bound $Z_{all} = 16$.

- **Operation 2. Branching Problem [all]** since $X_0$ is not integer-valued, into Subproblem [0] ($x_0 = 0$) and [1] ($x_1 = 1$).
  - **Operation 2.1. Bounding Subproblem [0]**, to obtain the solution $X_0 = (0, 1, 0, 1)$ and the bound $Z_0 = 9$. [0] fathomed and set the incumbent $Z^* = Z_0 = 9$ since $X_0$ is the first integer-valued solution so far.
  - **Operation 2.2. Bounding Subproblem [1]**, to obtain the solution $X_1 = (1, \frac{4}{5}, 0, \frac{4}{5})$ and the bound $Z_1 = 16$. [1] not fathomed because $X_1$ is not integer-valued.
Completing the example: the Iteration Procedure (cont.)


- Operation 3.1. Bounding Subproblem [10], to obtain the solution $X_{10} = (1, 0, \frac{4}{5}, 0)$ and the bound $Z_{10} = 13 > 9 = Z^*$. [10] not fathomed.
- Operation 3.2. Bounding Subproblem [11], to obtain the solution $X_{11} = (1, 1, 0, \frac{1}{2})$ and the bound $Z_{11} = 16 > 9 = Z^*$. [11] not fathomed.

- Operation 4.1. Bounding Subproblem [110], to obtain the solution $X_{110} = (1, 1, 0, \frac{1}{2})$ and the bound $Z_{110} = 16 > 9 = Z^*$. [110] not fathomed.
- Operation 4.2. Bounding Subproblem [111], to obtain no feasible solution so [111] fathomed.
Completing the example: the Iteration Procedure (cont.)

Operation 5. Branching Subproblem [110] rather than Subproblem [10] because [110] is newly created, into Subproblem [1100] ($x_4 = 0$) and Subproblem [1101] ($x_4 = 1$).

- Operation 5.1. Bounding Subproblem [1100], to obtain the solution $X_{1100} = (1, 1, 0, 0)$ and the bound $Z_{1100} = 14$. Subproblem [1100] fathomed and revise the incumbent $Z^* = Z_{1100} = 14$ as $X_{1100}$ is integer-valued and $14 > 9$.

Operation 6. Fathoming Subproblem [10] due to the fact that the revised incumbent value $Z^* = 14$ now is larger than $Z_{10} = 13$.

STOP because all subproblems are either fathomed or branched. The optimal solution is $X^# = X_{1100} = (1, 1, 0, 0)$ and the optimal value is $Z^# = Z^* = 14$. 

Operations Research (Li, X.)
A mixed integer programming problem (MIP) in general form:

Maximize \[ Z = \sum_{j=1}^{n} c_j x_j, \]

subject to \[ \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad \text{for } i = 1, 2, \ldots, m, \]

and \[ x_j \geq 0, \quad \text{for } j = 1, 2, \ldots, n, \]

\[ x_j \text{ is integer}, \quad \text{for } j = 1, 2, \ldots, I; \ I \leq n. \]

(When \( I = n \), this problem becomes the pure IP problem.)
A variation of the Branch-and-Bound algorithm solves MIP. The only differences are:

1. **Selecting the branching variable.** In the optimal solution for the LP relaxation, consider the *integer-restricted* variables having *non-integer* values, and pick the *first* one in natural ordering;

2. **Branching into two subproblems.** Let \( x^*_j \) be the optimal (non-integer) value of the branching variable in the optimal solution of the LP relaxation, then the two subproblems correspond to "\( x_j \leq \lfloor x^*_j \rfloor \)" and "\( x_j \geq \lfloor x^*_j \rfloor + 1 \)."

3. **Bounding without rounding.** This is due to the fact that some variables are not integer-restricted, so the optimal value is in general not required to be integers.

4. **Fathoming conditions.** The subproblem is fathomed if the optimal solution for its LP relaxation have integer values for those integer-restricted variables.
Solving a MIP with the Branch-and-Bound Algorithm: an example.

See textbook Hillier and Lieberman (10e) Section 12.7.
Solving an IP: the Cutting Plane Approach

An example of adding "cuts" to obtain an optimal solution with integer values for an IP.

Below we follow here *Operations Research: Applications and Algorithms* by W.L. Winston, Section 9.8, to present the *Cutting Plane Approach* for IP.

**TABLE 83**

Optimal Tableau for LP Relaxation of Telfa

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>rhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1.25</td>
<td>0.75</td>
<td>41.25</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2.25</td>
<td>-0.25</td>
<td>2.25</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1.25</td>
<td>0.25</td>
<td>3.75</td>
</tr>
</tbody>
</table>

(max $z = 8x_1 + 5x_2$

s.t. $x_1 + x_2 \leq 6$

$9x_1 + 5x_2 \leq 45$

$x_1, x_2 \geq 0$; $x_1, x_2$ integer)
Optimality test. At optimum, \( x_1 = 3.75 \) and \( x_2 = 2.25 \), both are non-integer values. We pick an constraint with the RHS non-integer. For example,

\[
x_1 - 1.25s_1 + 0.24s_2 = 3.75 - - - - (\ast)
\]

Looking for a cut.

Now we write each variable’s coefficient and the constraint’s RHS of Equation (***) in the form \( x = [x] + f \) with \( 0 \leq f < 1 \), to obtain:

\[
x_1 - 2s_1 + 0.5s_1 + 0s_2 + 0.25s_2 = 3 + 0.75.
\]

Group integer parts into the LHS and non-integer parts the RHS:

\[
x_1 - 2s_1 - 3 = 0.75 - 0.5s_1 - 0.25s_2 - - - - (\ast\ast)
\]

The "cut" of this iteration is obtained as "RHS of (***) \( \leq 0 \)", i.e.

\[
0.75 - 0.5s_1 - 0.25s_2 \leq 0 - - - - (\ast\ast\ast)
\]
The above-obtained cut (constraint) has the following two properties:

1. Any feasible point for the IP will satisfy the cut.
2. Current optimal solution to LP relaxation doesn’t satisfy the cut.

Why?

Remark.

- A cut "cut off" the current optimal solution to the LP relaxation, but not any feasible solution to the IP.
- When the cut to the LP relaxation is added, it is hoped that the optimal solution to the LP relaxation with the cut constraint added will be integer-valued. If it is the case, an optimal solution to the IP is obtained; if not, we look for a new cut.
- Gomory (1958) found that this process will yield an optimal solution to an IP after a finite number of iterations (cuts).
Substituting \( s_1 = 6 - x_1 - x_2 \) and \( s_2 = 45 - 9x_1 - 5x_2 \) into the cut 
\( 0.75 - 0.5s_1 - 0.25s_2 \leq 0 \) to obtain the cut constraint 
"\( 3x_1 + 2x_2 \leq 15 \)" thus the cutting plane "\( 3x_1 + 2x_2 = 15 \)."

Illustration of the cut to the LP relaxation.
We put the obtained cut constraint "3\(x_1 + 2x_2 \leq 15\)" into the LP relaxation. Introducing the slack variable \(s_3\) into this constraint, we obtain the following tableau:

Next we use the **dual simplex algorithm** to solve the above LP (optimal but not feasible): select \(s_3\) as the *leaving variable* since \(-0.75 < 0\) and then select \(s_1\) as the *entering variable* since \(-0.75 < 0\) and \(|1.25 \div (-0.75)| < |0.75 \div (-0.25)|\).
The **dual simplex algorithm**, applied to LP satisfying the **optimality** (positive coefficient in Eq.(0)) but not the **feasibility** condition (positive RHS), is essentially a set of operations on constraints so as reaching **feasibility** condition while keeping **optimality** condition (see Section 8.1 in Hillier and Lieberman (10e) for more details).

Using "−0.75" (row-$s_3$, column-$s_1$) as the pivot number, we perform the Gaussian elimination operations on rows to obtain (the same as the simplex method, to have the column-$s_1$ finally read as $(0, 0, 0, 1)^T$):

![Optimal Tableau for Cutting Plane](image)

At optimum, all variables are integers so it is feasible thus optimal to IP.