Integer Programming (IP)

Integer programming or integer linear programming, deals with models that are the same as linear programming with the one additional restriction that the variables must have integer values. If only some of the variables are required to have integer values, this is called a mixed integer programming (MIP) problem. When all the variables are binary variables, this is a binary integer programming (BIP) problem.

Rounding the numbers in the optimal solution may not work:

Example 1.

Maximize \( Z = x_1 + 5x_2 \)
Subject to \( x_1 + 10x_2 \leq 20 \)
\( x_1 \leq 2 \)
\( x_1, x_2 \geq 0 \)
\( x_1, x_2 \) are integers

The optimal solution of the LP will be \( x_1 = 2 \) and \( x_2 = 1.8 \) that gives \( z = 11 \). Rounding up would give us \( x_1 = 2 \) and \( x_2 = 1 \) with \( z = 7 \) while the optimal solution of the IP is at \( x_1 = 0, x_2 = 2 \) that gives \( z = 10 \).

Example 2.

Maximize \( Z = x_2 \)
Subject to \( -x_1 + x_2 \leq 0.5 \)
\( x_1 + x_2 \leq 3.5 \)
\( x_1, x_2 \geq 0 \)
\( x_1, x_2 \) are integers
The optimal solution of the LP is $x_1 = 1.5, x_2 = 2$. If we round off this solution (either up or down), the points $x_1 = 1, x_2 = 2$ or $x_1 = 2, x_2 = 2$ are out of the feasible region.

Even though the integer requirement makes IP problems harder to solve, it also makes IP a better model for a lot problems. Not only for the problems that requires the variables to be integers by its nature, but also for some decision problems that can be modelled by BIP.

**Problems involve mutually exclusive alternatives**

Or problems in that we may have to make contingent decisions, that is, one decision may rely on the outcome of another decision.

Example 3. Prototype example on page 539.

A new factory either in LA or SF, or even both. Also possible a new warehouse in a city with a new factory. The goal is to use the available capital (at most 10 million) to maximize the total net present value.

Four decisions:
1. Build factory in LA?
2. Build factory in SF?
3. Build warehouse in LA?
4. Build Warehouse in SF?

Four decision variables: $x_j = \begin{cases} 1 & \text{if the decision is yes;} \\ 0 & \text{if the decision is no.} \end{cases}$ $(j = 1, 2, 3, 4)$.

Since at most one warehouse will be built, decision 3 and 4 are **mutually exclusive**. This can be modelled by a constraint

$$x_3 + x_4 \leq 1$$

Also, the decision whether to build a warehouse in a city is **contingent** on whether there is a new factory in that city. This can be modelled by

$$x_3 \leq x_1 \quad x_4 \leq x_2$$
So the model of this problem is

\[ \text{Maximize } Z = 9x_1 + 5x_2 + 6x_3 + 4x_4 \]

Subject to

\[
\begin{align*}
6x_1 + 3x_2 + 5x_3 + 2x_4 & \leq 10 \\
x_3 + x_4 & \leq 1 \\
-x_1 + x_3 & \leq 0 \\
-x_2 + x_4 & \leq 0 \\
x_j & = 0 \text{ or } 1.
\end{align*}
\]

**Either-Or Constraints**

Example 4.

Either \(3x_1 + 2x_3 \leq 18\)

or \(x_1 + 4x_3 \leq 16\)

can be changed to

\[
\begin{align*}
3x_1 + 2x_3 & \leq 18 + My \\
x_1 + 4x_3 & \leq 16 + M(1 - y)
\end{align*}
\]

Here \(M\) is a huge number and \(y\) is a binary variable.

This can be generalized to the situation that \(K\) out of \(N\) constraints must hold: The previous example is just a case for \(K = 1\) and \(N = 2\). Suppose that we have these \(N\) constraints:

\[
\begin{align*}
f_1(x_1, x_2, ..., x_n) & \leq d_1 \\
f_2(x_1, x_2, ..., x_n) & \leq d_2 \\
& \vdots \\
f_n(x_1, x_2, ..., x_n) & \leq d_n
\end{align*}
\]

and \(K\) of them must hold. We can reformulate it as

\[
\begin{align*}
f_1(x_1, x_2, ..., x_n) & \leq d_1 + My_1 \\
f_2(x_1, x_2, ..., x_n) & \leq d_2 + My_2 \\
& \vdots \\
f_n(x_1, x_2, ..., x_n) & \leq d_n + My_N
\end{align*}
\]

\[
\sum_{i=1}^{N} y_i = N - K
\]

\(y_i\) is binary for all \(i\).
Functions with \( N \) possible values

BIP deals with binary variable, that is, the variables can have two values: 0 and 1. What if some variable, say, \( x_1 \) can be \( N \) values: 0, 1, 2, \ldots, \( N - 1 \)? In general, we may have a function

\[
f(x_1, x_2, \ldots, x_n) = d_1, \text{ or } d_2 \text{ or } \cdots \text{ or } d_N.
\]

This can be formulated as

\[
f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{N} y_i d_i
\]

\[
\sum_{i=1}^{N} y_i = 1 \text{ and } y_i \text{ is binary.}
\]

Example 5. Problems with Fixed-Charge (setup cost):

Suppose that for activity \( j \), there is a setup cost \( k_j \), that is, it is a fixed cost as long as the level of the activity \( x_j > 0 \) but it will be 0 if \( x_j = 0 \). Therefore the total cost for activity \( j \) will be

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

In this case, we cannot use

\[
Z = \sum (k_j + c_j x_j)
\]

for then it will cause a cost \( k_j \) even \( x_j = 0 \), but we cannot ignore the \( k_j \)'s either. We can use a binary variable \( y_j \) such that

\[
y_j = \begin{cases} 
    1 & \text{if } x_j > 0 \\
    0 & \text{if } x_j = 0
\end{cases}
\]

If we know how to formulate such contingent variables, then we can make the objective function to be

\[
\text{Minimize } Z = \sum (k_j y_j + c_j x_j)
\]

It is easy to force \( y_j = 1 \) when \( x_j > 0 \): We can use a constraint

\[
x_j \leq M y_j
\]

Now how can make sure that \( y_j = 0 \) when \( x_j = 0 \)? As it turns out, in the minimization problem, it will take care itself. If \( x_j = 0 \), then \( y_j \) can be either 0 or 1. But if \( y_j = 1 \), the objective function will be bigger, so in the optimal solution, \( y_j \) must be 0.

Every IP can be changed into a BIP.

Let \( 2^{N+1} \) be an upper bound of the \( x_i \)'s, i.e., \( x_i < 2^{N+1} \) for all \( i \). Then we can write

\[
x_i = \sum_{j=1}^{N} 2^j y_{i,j}
\]

\( y_{i,j} \)'s are binary. So we can substitute every integer with \( N \) binary numbers.
More examples

Example 6. The problem:

1. There are 3 new products. At most 2 of them can be produced.

There are 2 plants, only one should be chosen to produce the new products.

<table>
<thead>
<tr>
<th>Products</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Available hours per week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant 1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>Plant 2</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>40</td>
</tr>
</tbody>
</table>

Unit Profit 5 7 3 in thousands of dollars
Sales Potential 7 5 9 unit per week

Without the two conditions, this can be formulated as a linear programming problem: Let $x_1, x_2, x_3$ be the units of product 1, 2, and 3 to be produced. (That is, if we know which plant will be used to produce them.)

Maximize $Z = 5x_1 + 7x_2 + 3x_3$

subject to

\[
\begin{align*}
3x_1 + 4x_2 + 2x_3 &\leq 30 & (1) \\
4x_1 + 6x_2 + 2x_3 &\leq 40 & (2) \\
x_1 &\leq 7 \\
x_2 &\leq 5 \\
x_3 &\leq 9 \\
\end{align*}
\]

and

$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0.$

So here if we know which plant to use then we will use one of (1) or (2). Also we completely ignored the requirement that only at most 2 of the 3 new products will be produced. We will use some binary variables to be our auxiliary variables that can help us to formulate those requirements.

Let $y_1, y_2, y_3$ be such that

\[
y_j = \begin{cases} 
1 & \text{if } x_j > 0 \\
0 & \text{if } x_j = 0 
\end{cases}
\]

and we add the constraints

\[
\begin{align*}
x_j &\leq My_j \\
y_1 + y_2 + y_3 &\leq 2 
\end{align*}
\]

That takes care of the two out of three part. For the other requirement, we use a fourth binary variable $y_4$:

\[
y_j = \begin{cases} 
1 & \text{if (1) must hold;} \\
0 & \text{if (2) must hold.}
\end{cases}
\]
This can be accomplished by adding these two constraints:

\[3x_1 + 4x_2 + 2x_3 \leq 30 + M(1 - y_4)\]
\[4x_1 + 6x_2 + 2x_3 \leq 40 + My_4\]

The complete model then is as on page 549. Example 3 on page 553. Set-covering problem.

An airline has 11 flights. There are 12 possible sequences of flights for a crew. Exactly three of the sequences need to be chosen (one per crew) in such a way that every flight is covered. (If two crews are on the same flight, both will have to be paid.) The cost of assigning a crew to a sequence of flights is given (in thousands of dollars). The objective is to minimize the total cost of the assignments.

<table>
<thead>
<tr>
<th>Flight</th>
<th>Sequence of flights</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF to LA</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>SF to Denver</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>SF to Seattle</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>LA to Chicago</td>
<td>2 2 3 2 3</td>
</tr>
<tr>
<td>LA to SF</td>
<td>2 3 5 5</td>
</tr>
<tr>
<td>Chicago to Denver</td>
<td>3 3 4</td>
</tr>
<tr>
<td>Chicago to Seattle</td>
<td>3 3 3 3 4</td>
</tr>
<tr>
<td>Denver to SF</td>
<td>2 4 4 5</td>
</tr>
<tr>
<td>Denver to Chicago</td>
<td>2 2 2</td>
</tr>
<tr>
<td>Seattle to SF</td>
<td>2 4 4 5</td>
</tr>
<tr>
<td>Seattle to LA</td>
<td>2 2 4 4 2</td>
</tr>
<tr>
<td>cost</td>
<td>2 3 4 6 7 5 7 8 9 8 9</td>
</tr>
</tbody>
</table>

Decision variables:

\[x_j = \begin{cases} 
1 & \text{if sequence } j \text{ is assigned;} \\
0 & \text{otherwise.} 
\end{cases}\]

\[\text{Min } 2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12}\]

Subject to

\[x_1 + x_4 + x_7 + x_{10} \geq 1\]
\[x_2 + x_5 + x_8 + x_{11} \geq 1\]
\[x_3 + x_6 + x_9 + x_{12} \geq 1\]
\[x_4 + x_7 + x_9 + x_{10} + x_{12} \geq 1\]
\[\vdots\]
\[x_1 + x_2 + \cdots + x_{12} = 3\]

\(x\) is binary.