Integer programming models

Topic objectives:

• Present integer programming models for special cases in industrial engineering.
• See chapter 9.1 in the textbook.
• Integer programming models are either direct description of decision problems or as convenience for problems that are not integer by themselves.
• Integer programming models are either pure (all variables are integer) or mixed (some but not all are integer).
• Integer decision variables that are restricted to either 0 or 1 are called binary variables.
Capital budgeting

- The capital budgeting problem deals with the allocation of limited resources (fund, space, time, etc) to selected courses of action.
- An industrial engineer is planning the production of five products that will be produced during the following week.
- All products pass through the same three-stage process, but the stage costs are different for the different products:

<table>
<thead>
<tr>
<th>Product</th>
<th>Stage cost (riyals)</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Allocated fund (riyals)</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
Given the limited funds allocated for each stage, the industrial engineer has to select the products that will maximize the revenue for the company.

Define the binary variable

\[ x_j = \begin{cases} 1 & \text{if product } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \]

The objective function will be

\[ \max z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 \]
• The constraints impose the budget limitation:

\[
\begin{align*}
5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 & \leq 25 \\
x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 & \leq 25 \\
8x_1 + 10x_2 + 2x_3 + x_4 + x_5 & \leq 25 \\
x & = (0,1)
\end{align*}
\]
Set covering

• The set covering problem can be described as allocating service stations that will cover all service demand locations.

• In a plant, there are 11 work stations and 8 possible locations of tool cabinets:
• Each tool cabinet can serve as number of work stations as desired.
• These tool cabinet locations are determined in a way that workers in work stations can find an adjacent cabinet.
• The industrial engineer has to determine the minimum number of tool cabinets that will serve the work stations.
• Define the binary decision variable

\[ x_j = \begin{cases} 
1 & \text{if tool cabinet is installed in position } j \\
0 & \text{otherwise}
\end{cases} \]
• Since the purpose is to install the smallest number of cabinets in the plant, the objective function then becomes

\[ \min z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \]

• Each work station has to be served by one a close-by cabinet, the constraints become

\[
\begin{align*}
    x_1 + x_2 & \geq 1 \\
    x_4 + x_5 & \geq 1 \\
    x_7 + x_8 & \geq 1 \\
    x_1 + x_5 & \geq 1 \\
    x_4 + x_7 & \geq 1 \\
    x_j & = (0,1) \\
    x_2 + x_3 & \geq 1 \\
    x_5 + x_6 & \geq 1 \\
    x_3 + x_8 & \geq 1 \\
    x_6 + x_7 & \geq 1 \\
    x_3 + x_4 & \geq 1 \\
    x_1 + x_3 & \geq 1
\end{align*}
\]
Traveling salesman problem TSP

- TSP is a problem of finding the best route (path) or sequence to visit all locations and stop at the starting location with minimum cost, distance, or time.
- An industrial engineer has to plan for the distribution of products manufactured by the company.
• The all delivery points have to be visited and the truck returns back to the factory with the least distance.
• Define the binary variable

\[ x_{ij} = \begin{cases} 
1 & \text{if point } j \text{ is visited from point } i \\
0 & \text{otherwise} 
\end{cases} \]

• Define the parameter

\[ d_{ij} = \text{distance between point } i \text{ and } j \]
• If point j cannot be visited directly from point i, set \( d_{ij} = \infty \).
• By design, \( d_{ii} = \infty \), point i cannot be visited directly from itself.
The objective function for this problem becomes:

\[
\min z = d_{12}x_{12} + d_{15}x_{15} + d_{16}x_{16} + d_{21}x_{21} + d_{23}x_{23} + d_{25}x_{25} \\
+ d_{26}x_{26} + d_{32}x_{32} + d_{34}x_{34} + d_{35}x_{35} + d_{43}x_{43} + d_{45}x_{45} + d_{46}x_{46} \\
+ d_{51}x_{51} + d_{52}x_{52} + d_{53}x_{53} + d_{54}x_{54} + d_{56}x_{56} + d_{61}x_{61} + d_{62}x_{62} \\
+ d_{64}x_{64} + d_{65}x_{65}
\]

The model has two sets of constraints.

The first set states that every point has to be connected to the following point:

\[
\begin{align*}
x_{12} + x_{15} + x_{16} &= 1 \\
x_{21} + x_{23} + x_{25} + x_{26} &= 1 \\
x_{32} + x_{34} + x_{35} &= 1
\end{align*}
\]
\[
x_{43} + x_{45} + x_{46} = 1 \\
x_{51} + x_{52} + x_{53} + x_{54} + x_{56} = 1 \\
x_{61} + x_{62} + x_{64} + x_{65} = 1
\]

- The second set states that every point is connected to a preceding point:
  \[
x_{21} + x_{51} + x_{61} = 1 \\
x_{12} + x_{32} + x_{52} + x_{62} = 1 \\
x_{23} + x_{43} + x_{53} = 1 \\
x_{34} + x_{54} + x_{64} = 1 \\
x_{15} + x_{25} + x_{35} + x_{45} + x_{65} = 1 \\
x_{16} + x_{26} + x_{46} + x_{56} = 1
\]
• These constraints are not enough to produce a single path or a tour that will visit all points:

• A sub-tour elimination strategy has to be implemented to prevent sub-tours from happening.
• Notice that any sub-tour has links equal to the number of points.
• To prevent sub-tours from forming, we can impose a restriction that:
  – Any group of two points should not have more than one link;
  – Any group of three points should not have more than two links;
  – ….
  – Any group of five point should not have more than 4 links.
• Only the 6 points are allowed to have 6 links.
So, we can write

\[
\begin{align*}
 &x_{12} + x_{21} \leq 1 \\
 &x_{15} + x_{51} \leq 1 \\
 &x_{16} + x_{61} \leq 1 \quad \text{Two points or nodes} \\
 &x_{23} + x_{32} \leq 1 \\
 &\vdots \\
 &x_{12} + x_{26} + x_{16} \leq 2 \\
 &x_{16} + x_{62} + x_{21} \leq 2 \\
 &x_{15} + x_{56} + x_{61} \leq 2 \quad \text{Three nodes} \\
 &x_{16} + x_{65} + x_{51} \leq 2 \\
 &\vdots 
\end{align*}
\]
\[
\begin{align*}
&x_{12} + x_{25} + x_{56} + x_{61} \leq 3 \\
&x_{16} + x_{65} + x_{52} + x_{21} \leq 3 \\
&\vdots \\
&x_{12} + x_{23} + x_{35} + x_{56} + x_{61} \leq 4 \\
&\vdots \\
&x_{ij} = (0,1)
\end{align*}
\]
The traveling salesman problem can be written compactly as

$$\min \; z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

s.t. \hspace{1cm} \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq \mid S \mid - 1, \quad S \subset N, 2 \leq \mid S \mid \leq n - 1$$

$$x_{ij} = (0,1)$$
Fixed charge

• Manufacturing costs usually have two components, fixed cost and variable cost.
• The fixed cost does not change with the volume of production.
• The variable cost increases with the production volume.
• When there is no production, the manufacturing cost is equal to zero.
• Therefore, if \( x \) = the production volume, then

\[
C(x) = \begin{cases} 
F + Vx & \text{for } x > 0 \\
0 & \text{for } x = 0 
\end{cases}
\]
• Dealing with this form of cost function is difficult since it is discontinuous at $x = 0$.
• Integer programming easily and conveniently is going to put the problem into a mathematical model.
• The industrial engineer is planning the purchase of raw materials used for production.
• The raw materials can be purchased from three local manufacturing plants:

<table>
<thead>
<tr>
<th>Company</th>
<th>Fixed cost (riyals)</th>
<th>Unit variable cost (riyals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saudi Fabrication</td>
<td>1600</td>
<td>250</td>
</tr>
<tr>
<td>National Manufacture</td>
<td>2500</td>
<td>210</td>
</tr>
<tr>
<td>Expert Staff</td>
<td>1800</td>
<td>220</td>
</tr>
</tbody>
</table>
• It has been estimated that 200 units are needed for production.
• Define the integer variables:
  \( x_1 = \) number of units ordered from the first company
  \( x_2 = \) number of units ordered from the second company
  \( x_3 = \) number of units ordered from the third company
• Define the binary variable:

\[
y_j = \begin{cases} 
1 & \text{if raw materials ordered from company } j \\
0 & \text{otherwise}
\end{cases}
\]
• To ensure that $y_j = 1$ for $x_j > 0$, we can write
  \[ x_j < M y_j \]
• $M$ is any large number greater than 200 (needed raw materials).
• The integer model is
  \[
  \begin{align*}
  \text{min } z &= 250x_1 + 210x_2 + 220x_3 + 1600y_1 + 2500y_2 + 1800y_3 \\
  \text{s.t. } x_1 + x_2 + x_3 &= 200 \\
  x_1 &\leq 200y_1 \\
  x_2 &\leq 200y_2 \\
  x_3 &\leq 200y_3 \\
  x_j &\geq 0 \\
  y_j &= (0,1)
  \end{align*}
  \]
Either-or and if-then constraints

• In general LP models, all the constraints are enforced.
• Integer programming is used to model cases where logical conditions are encountered, such as either one restriction is enforced or another; or conditions in which a restriction is enforced only when a specific situation happens.
• An industrial engineer is planning the sequence or order of 3 jobs on a machine:

<table>
<thead>
<tr>
<th>Job</th>
<th>Processing time $p_j$ (days)</th>
<th>Due date $d_j$</th>
<th>Late penalty (riyals per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>35</td>
<td>34</td>
</tr>
</tbody>
</table>
• The penalty value signifies the importance of finishing the job on time, and it does not mean that any money will be paid.

• Hence, the best sequence is the one that has the least penalty cost.

• Define the decision variable:

\[ x_j = \text{start day of job } j \]

• In this problem, there are two groups of constraints
  – Processing of a single job on the machine in any given time
  – Job finish time calculation
• Two jobs i and j will not be processed at the same time if either job i starts after job j completes or job j starts after job i completes:

\[ x_i - x_j \geq p_j \quad \text{or} \quad x_j - x_i \geq p_i \]

• Introduce the binary variable

\[ y_{ij} = \begin{cases} 
1 & \text{if job } i \text{ is before job } j \\
0 & \text{if job } j \text{ is before job } i 
\end{cases} \]

• The either-or constraint above can be enforced as

\[ M y_{ij} + (x_i - x_j) \geq p_j \]

\[ M (1 - y_{ij}) + (x_j - x_i) \geq p_i \]

for M very large.
• When $y_{ij} = 0$, then
  $x_i - x_j \geq p_j$ (binding)
  and
  $M + x_j - x_i \geq p_i \rightarrow x_j - x_i \geq p_i - M$ (non binding)
• Hence, job $i$ comes after job $j$.
• When $y_{ij} = 1$, then
  $M + x_i - x_j \geq p_j \rightarrow x_i - x_j \geq p_j - M$ (non binding)
  and
  $x_j - x_i \geq p_i$ (binding)
• Hence, job $j$ comes after job $i$. 

Dr Muhammad Al-Salamah, Industrial Engineering, KFUPM
• The deviation of job $j$ from its due date is
  \[ s_j = d_j - (x_j + p_j) \]
  
• Job $j$ will not be penalized if $s_j \geq 0$.

• Replace $s_j$ by two positive deviational variables:
  \[ s_j = s^-_j - s^+_j \]
  
• Hence, job $j$ is delayed if $s^+_j > 0$.

• The objective function of the job sequence problem is
  \[ \text{Min } z = 19s^+_{1} + 12s^+_{2} + 34s^+_{3} \]
• Assume that there is an additional restriction that if job i comes before job j, then job k should come before job m.
• This restriction can be written as
  \[ \text{If } x_i + p_i \leq x_j \text{ then } x_k + p_k \leq x_m \]
• It can be also written as
  \[ \text{If } x_j - (x_i + p_i) > 0 \text{ then } x_k + p_k - x_m < 0 \]
• Introduce a binary variable \( y = (0,1) \).
• The if-then restriction can be explained by two constraints:
  \[ x_j - (x_i + p_i) \leq M(1 - y) - \varepsilon \]
  \[ x_k + p_k - x_m \leq My \]
• If the optimal solution requires \( x_j - (x_i + p_i) > 0 \), that is job \( i \) is after job \( j \), then \( y \) has to be 0, and hence 
\[ x_k + p_k - x_m \leq 0, \] 
that is job \( m \) has to come after job \( k \).
• If the optimal solution is for \( x_j - (x_i + p_i) < 0 \), then \( y \) can be 0 or 1 and it has no effect on the second constraint.