The Resource Allocation Problem

The Linear Programming Problem
Plants:

- Plant 1: Aluminum frame production.
- Plant 2: Wood frame production.
- Plant 3: Glass production and assembly.

Production is in batches of 200 units.

Data:

<table>
<thead>
<tr>
<th></th>
<th>Hrs/batch</th>
<th>Hrs avail</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Door</td>
<td>Window</td>
</tr>
<tr>
<td>Plant 1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Plant 2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Plant 3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Profit/batch</td>
<td>$3,000</td>
<td>$5,000</td>
</tr>
</tbody>
</table>
\begin{align*}
2x_2 & \leq 12 \\
3x_1 + 2x_2 & \leq 18 \\
x_1, x_2 & \geq 0
\end{align*}
maximize profit: 3 * Batches1 + 5 * Batches2;

subject to P1_Hrs_Avail: Batches1 <= 4;
subject to P2_Hrs_Avail: 2*Batches2 <= 12;
subject to P3_Hrs_Avail: 3*Batches1 + 2*Batches2 <= 18;

solve;

display Batches1, Batches2, profit;

Output:

MINOS 5.4: optimal solution found.
2 iterations, objective 36
Batches1 = 2
Batches2 = 6
profit = 36
param avail {Resources};
param unit_profit {Activities};
param usage {Resources, Activities};

var amt {Activities} >= 0;

maximize profit:
    sum {j in Activities} unit_profit[j] * amt[j];

subject to capacity {i in Resources}:
    sum {j in Activities} usage[i,j] * amt[j]
        <= avail[i];
param avail :=
    Plant1  4
    Plant2 12
    Plant3 18
    ;

param unit_profit :=
    Door    3
    Window  5
    ;

param usage: Prod1 Prod2 :=
    Plant1  1  0
    Plant2  0  2
    Plant3  3  2
    ;
ampl: solve,
MINOS 5.4: optimal solution found.
2 iterations, objective 36

ampl: display amt, profit;
amt [*] :=
Prod1  2
Prod2  6
;

profit = 36

ampl: quit
\[ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \]

\[
\vdots
\]

\[ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \]

\[ x_1, x_2, \ldots, x_n \geq 0. \]
Much of the course will be devoted to study of such algorithms.

Most real-world problems don’t quite fit the LP paradigm.

They have nonlinearities in either the objective function, the constraints, or both.

Later in the course we will extend the algorithms we develop for LP to some of these more general problems.