Department of Industrial Engineering & Operations Research

IEOR 162: Linear Programming and Network Flows (Spring 2022)

Giapetto's Woodcarving Problem

A toy soldier sells for \$27 and uses \$10 worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by \$14. A train sells for \$21 and uses \$9 worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by \$10. The manufacture of wooden soldiers and trains requires two types of skilled labor: carpentry and finishing. A soldier requires 2 hours of finishing labor and 1 hour of carpentry labor. A train requires 1 hour of finishing labor and 1 hour of carpentry labor. Each week, Giapetto can obtain all the needed raw material but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldiers are bought each week. Giapetto wants to maximize weekly profit.

We can formulate the linear programming problem as:

$$
\begin{aligned}
\text{maximize} \quad & 3x_1 + 2x_2 && \text{(Profit function)} \\
\text{subject to} \quad & 2x_1 + x_2 \leq 100 && \text{(Finishing constraint)} \\
& x_1 + x_2 \leq 80 && \text{(Carpentry constraint)} \\
& x_1 \leq 40 && \text{(Demand constraint for soldiers)} \\
& x_1, x_2 \geq 0 && \text{(Non-negativity constraints)}
\end{aligned}
$$

AMPL Code

We can formulate Giapetto's problem with the following AMPL model:

```
# Giapetto's Woodcarving Problem

var x1; # variable for soldiers
var x2; # variable for trains

maximize profit: 3*x1 + 2*x2;

subject to
finishing: 2*x1 + x2 <= 100;
carpentry: x1 + x2 <= 80;
demand:  x1 <= 40;
nonneg1: x1 >= 0;
nonneg2: x2 >= 0;
```

For larger size problem, we can also formulate Giapetto's problem with the following AMPL model:

```
param n;
param profit_item{i in 1..n};
param finishing_hour{i in 1..n};
param carpentry_hour{i in 1..n};
param finishing_cap;
```

```
param carpentry_cap;

var x{i in 1..n};

maximize profit: sum{i in 1..n} profit_item[i]*x[i];

subject to
finishing: sum{i in 1..n} finishing_hour[i]*x[i] <= finishing_cap;
carpentry: sum{i in 1..n} carpentry_hour[i]*x[i] <= carpentry_cap;
demand: x[1] <= 40;
nonneg{i in 1..n}: x[i] >= 0;
```

Also we need the data for the parameter defined above:

```
param n := 2;
param: profit_item, finishing_hour, carpentry_hour :=
1 3 2 1 # The first 1 is the index of parameter
2 2 1 1 # The first 2 is the index of parameter
;
param finishing_cap := 100;
param carpentry_cap := 80;
```

AMPL Output

Depending on the solver we use, we can get AMPL to output information about the solved problem.

```
ampl: reset;
ampl: model Giapetto.mod; # May need entire path
ampl: data Giapetto.dat; # Type this line only if you use the second formulation
ampl: solve;
```

We can output the optimal solution by typing 'display [variable1], [variable2],...':

```
ampl: display x1,x2;
# Output below:
x1 = 20
x2 = 60
```

Or if you use the second formulation:

```
ampl: display x;
# Output below:
x [*] :=
1  20
2  60
;
```