Minimize_scrap

August 6, 2024

1 Minimize Scrap in Production

Given N number of metal bars with length L. We need to cut full length bar in to smaller M bars with length l_j and demand D_j for $j \in [1, M]$

[ ]:

```
from pulp import *
import pandas as pd
```

[ ]:

```
# Number of available resources (bars)
N = 20
# Length of each original bar (in)
L = 288

# Bar data
bar_data = pd.read_excel("./Data/bar_data.xlsx")

# Number of cut bars
M = len(bar_data)

# length of each bar (in)
l = bar_data.Length

# Demand of each bar (bars)
D = bar_data.Demand

bar_data
```

[ ]:

```
   Length  Demand
0      15      10
1      20      15
2      25       7
3      30       7
4      45       6
5      66       5
6      78       9
```

2 Simple Case

Assumption: There are enough material to meet demand

Variable: X_{ij} be the number of length l_j bar that is cut out off the orginal i^{th} bar

Objective: Minimize scrap

- The unusable material remaining after a bar is cut to size is scrap.

$$\min \sum_{i=1}^{N} \left[L - \sum_{j=1}^{M} l_j X_{ij} \right]$$

Constraints:

- Total length of cut bars within original bar length
  - $\sum_{j=1}^{M} l_j X_{ij} \leq L$ for $i \in [1, N]$
- Meet demand
  - $\sum_{i=1}^{N} X_{ij} \geq D_j$ for $j \in [1, M]$
- Positivity result:
  - $X_{ij} \geq 0$ for $i \in [1, N]$ and $j \in [1, M]$

[ ]:

```
# Define variable
X = LpVariable.dicts("bar", [f"{i}_{j}" for i in range(1, N + 1) for j in range(1, M + 1)], lowBound=0, cat='Integer')

# Initialize model
model = LpProblem("MinimizeScrap", LpMinimize)

# Objective function
model += lpSum([L - lpSum(l[j - 1] * X[f"{i}_{j}"] for j in range(1, M + 1)) for i in range(1, N + 1)]), "Scrap"

# Constraints
# Length within original bar length
for i in range(1, N + 1):
    model += lpSum(l[j - 1] * X[f"{i}_{j}"] for j in range(1, M + 1)) <= L

# Meet demand
for j in range(1, M + 1):
    model += lpSum([X[f"{i}_{j}"] for i in range(1, N + 1)]) >= D[j - 1]

# Positivity
for i in range(1, N + 1):
    for j in range(1, M + 1):
        model += X[f"{i}_{j}"] >= 0
```

[ ]:

```
# Solve model
model.solve()
LpStatus[model.status]
```

```
[ ]: 'Optimal'
```

```
[ ]: df = pd.DataFrame({f"bar_{l[j - 1]}_in": [int(X[f"{i}_{j}"].varValue) for i in␣
      ↪range(1, N + 1)]
                         for j in range(1, M + 1)}).
      ↪sort_values(by=f"bar_{min(l)}_in", ascending=False).reset_index(drop=True)
     df.index.name = 'Bar_ID'
     df
```

```
[ ]:         bar_15_in  bar_20_in  bar_25_in  bar_30_in  bar_45_in  bar_66_in  \
     Bar_ID
     0              14          0          0          0          0          0
     1              14          0          0          0          0          0
     2              14          0          0          0          0          0
     3              14          0          0          0          0          0
     4              14          0          0          0          0          0
     5               1          0          6          0          1          0
     6               0          1          1          0          1          3
     7               0          6          0          0          2          0
     8               0          0          3          0          3          0
     9               0          6          0          0          2          0
     10              0          0          0          3          0          3
     11              0          0          0          1          4          0
     12              0          0          0          4          2          0
     13              0          0          0          1          4          0
     14              0          0          0          3          0          3
     15              0          0          0          0          0          2
     16              0          8          2          0          0          0
     17              0          0          0          0          0          2
     18              0          6          0          0          2          0
     19              0          1          4          0          2          0

             bar_78_in
     Bar_ID
     0               1
     1               1
     2               1
     3               1
     4               1
     5               1
     6               0
     7               1
     8               1
     9               1
     10              0
     11              1
     12              1
```

```
13                1
14                0
15                2
16                1
17                2
18                1
19                1
```

[ ]:

```
# Check total
pd.DataFrame(df.sum(), columns=["Total"]).T
```

[ ]:

```
       bar_15_in  bar_20_in  bar_25_in  bar_30_in  bar_45_in  bar_66_in  \
Total         71         28         16         12         23         13

       bar_78_in
Total         19
```

3 Advanced Case

Assumption:

- We allow to not meet demand with a cost penalty
- CR is material disposal or recycling cost
- CD is penalty cost for shortage

Variable:

- X_{ij} be the number of length l_j bar that is cut out off the orginal i^{th} bar
- y_j be the amount of shortage bar with length l_j

Objective: Minimize scrap

- The unusable material remaining after a bar is cut to size is scrap.

$$\min\left\{\sum_{i=1}^{N}\left[L-\sum_{j=1}^{M}l_jX_{ij}\right]*CR+\sum_{j=1}^{M}y_jl_j*CD\right\}$$

Constraints:

- Total length of cut bars within original bar length
  - $\sum_{j=1}^{M}l_jX_{ij}\leq L$ for $i\in[1,N]$
- Meet demand
  - $\sum_{i=1}^{N}X_{ij}+y_j\geq D_j$ for $j\in[1,M]$
- Positivity result:
  - $X_{ij}\geq 0$ for $i\in[1,N]$ and $j\in[1,M]$
  - $y_j\geq 0$ for $j\in[1,M]$

[ ]:

```
# Define cost. These values need to change based on the market values
CR = 0.5
CD = 2
```

```
# Define variable
X = LpVariable.dicts("bar", [f"{i}_{j}" for i in range(1, N + 1) for j in␣
 ↪range(1, M + 1)], lowBound=0, cat='Integer')
Y = LpVariable.dicts("bar", [f"{j}" for j in range(1, M + 1)], lowBound=0,␣
 ↪cat='Integer')

# Initialize model
model = LpProblem("MinimizeScrap", LpMinimize)

# Objective function
model += lpSum([L - lpSum(l[j - 1] * X[f"{i}_{j}"] for j in range(1, M + 1))␣
 ↪for i in range(1, N + 1)]) * CR + \
         lpSum(Y[f"{j}"] * l[j - 1] for j in range(1, M + 1)) * CD, "Cost"

# Constraints
# Length within original bar length
for i in range(1, N + 1):
    model += lpSum(l[j - 1] * X[f"{i}_{j}"] for j in range(1, M + 1)) <= L

# Meet demand
for j in range(1, M + 1):
    model += lpSum([X[f"{i}_{j}"] for i in range(1, N + 1)]) + Y[f"{j}"] >= D[j␣
 ↪- 1]
```

[ ]:

```
# Solve model
model.solve()
LpStatus[model.status]
```

[ ]:

```
'Optimal'
```

[ ]:

```
df_advanced = pd.DataFrame({f"bar_{l[j - 1]}_in": [int(X[f"{i}_{j}"].varValue)␣
 ↪for i in range(1, N + 1)]
                  for j in range(1, M + 1)}).
 ↪sort_values(by=f"bar_{min(l)}_in", ascending=False).reset_index(drop=True)
df_advanced.index.name = 'Bar_ID'
df_advanced
```

[ ]:

```
        bar_15_in  bar_20_in  bar_25_in  bar_30_in  bar_45_in  bar_66_in  \
Bar_ID
0              10          3          0          0          0          0
1              10          3          0          0          0          0
2               6          0          3          0          1          0
3               4          0          0          1          0          3
4               3          7          1          0          0          0
5               3          6          0          0          1          0
6               3          7          1          0          0          0
7               2          9          0          0          0          0
```

```
8               2          9          0          0          0          0
9               1          7          1          1          0          0
10              0          0          0          0          0          2
11              0          0          0          0          0          2
12              0          0          0          7          0          0
13              0          0          0          3          0          3
14              0          0          3          0          3          0
15              0          3          6          0          0          0
16              0          0          0          7          0          0
17              0          9          0          1          0          0
18              0          0          0          7          0          0
19              0          0          0          0          2          3

        bar_78_in
Bar_ID
0               1
1               1
2               1
3               0
4               1
5               1
6               1
7               1
8               1
9               1
10              2
11              2
12              1
13              0
14              1
15              1
16              1
17              1
18              1
19              0
```

```
[ ]: # Check total
     pd.DataFrame(df_advanced.sum(), columns=["Total"]).T
```

```
[ ]:        bar_15_in  bar_20_in  bar_25_in  bar_30_in  bar_45_in  bar_66_in  \
     Total         44         63         15         27          7         13

            bar_78_in
     Total         19
```

```
[ ]: # Check total number of shortage bar
     df_shortage = pd.DataFrame({f"bar_{l[j - 1]}_in": [int(Y[f"{j}"].varValue)]
                        for j in range(1, M + 1)}).
      ↪sort_values(by=f"bar_{min(l)}_in", ascending=False).reset_index(drop=True)
     pd.DataFrame(df_shortage.sum(), columns=["# shortage"]).T
```

```
[ ]:            bar_15_in  bar_20_in  bar_25_in  bar_30_in  bar_45_in  bar_66_in  \
     # shortage          0          0          0          0          0          0

                bar_78_in
     # shortage          0
```