Gerald Jones

Independent Project 3

ISE 522, Spring 22

Problem Description:

> The attached spreadsheet contains demand data for **ten weeks**. You are to solve the following replenishment problem. Decide how much to order each week to minimize the total cost, which includes fixed charges for ordering and holding cost. If an order is made, there is a fixed charge paid for that order, regardless of the quantity. Inventory held at the end of the week incurs a holding cost per unit. There is a minimum amount of inventory that must be on hand each week. If an order is placed in a period, the order amount must be at least the minimum order quantity (MOQ). **In this problem there is no longer a fixed charge for placing an order. Rather, there is a quantity discount**, as shown in the spreadsheet. For example, if you order between 5 and 9 items, the cost per item is \
> $2. Butifyouorderbetween10and14, thecostperitemis1$.
>
> The attached spreadsheet includes all of the data that you need as well as an example solution. (This example solution is not necessarily optimal. You need to find the optimal solution.

In [1]:
```
from  _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
from _NOTE_BOOK_UTILS import *

# full data
data_df_full = pd.read_excel("single item data - MOQ and qty discount.xlsx")

# parameters
data_df_params = pd.read_excel("single item data - MOQ and qty discount.xlsx", skiprows=[0, ], nrows=3).iloc[:

# discount ranges
discount_ranges = data_df_params.iloc[1, 1:4].values

# get MOQ
MOQ = pd.read_excel("single item data - MOQ and qty discount.xlsx", nrows=1).iloc[:, 0:2].columns.tolist()
MOQ = int(MOQ[1])
# discount prices for each range
discount_prices =  data_df_params.iloc[2, 1:4].values


# data only
data_df = pd.read_excel("single item data - MOQ and qty discount.xlsx", skiprows=[0,1,2,3,4,5, 7], nrows=10)

display(data_df_full)
print("data df")
display(data_df)

uniPrint(data_df_params)

uniPrint(discount_ranges)

uniPrint(discount_prices)

uniPrint(MOQ)


discounts = {
    discount_prices[0]: [discount_ranges[0], discount_ranges[1]-1],
    discount_prices[1]: [discount_ranges[1], discount_ranges[2]-1],
    discount_prices[2]: [discount_ranges[2]],

}

discounts
```

	minimum order quantity (MOQ)	5	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	Quantity discount schedule	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	minimum order amount	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	5	10	15	NaN	NaN	NaN	NaN	NaN
3	price per unit	2	1	0.5	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Week	Demand	Minimum inventory requirement	Order amount	Inventory held at end of week	Holding cost per unit	Cost of goods purchased	Holding cost	Is order feasible?
6	0	NaN	NaN	NaN	20	NaN	NaN	NaN	NaN
7	1	10	1	6	16	1	12	16	NaN
8	2	10	1	0	6	1	0	6	NaN
9	3	10	1	10	6	1	10	6	NaN
10	4	0	0	0	6	1	0	6	NaN
11	5	0	0	0	6	1	0	6	NaN
12	6	15	1.5	11	2	1	11	2	NaN
13	7	20	2	20	2	1	10	2	NaN
14	8	20	2	20	2	1	10	2	NaN
15	9	0	0	0	2	1	0	2	NaN
16	10	10	1	8	0	1	16	0	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	TOTAL COST	117	NaN	NaN	NaN	NaN	NaN	NaN	NaN

data df

	Week	Demand	Minimum inventory requirement	Order amount	Inventory held at end of week	Holding cost per unit	Cost of goods purchased	Holding cost	Is order feasible?
0	1	10	1.0	6	16	1	12	16	NaN
1	2	10	1.0	0	6	1	0	6	NaN
2	3	10	1.0	10	6	1	10	6	NaN
3	4	0	0.0	0	6	1	0	6	NaN
4	5	0	0.0	0	6	1	0	6	NaN
5	6	15	1.5	11	2	1	11	2	NaN
6	7	20	2.0	20	2	1	10	2	NaN
7	8	20	2.0	20	2	1	10	2	NaN
8	9	0	0.0	0	2	1	0	2	NaN
9	10	10	1.0	8	0	1	16	0	NaN

	Quantity discount schedule	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	NaN	minimum order amount	NaN	NaN	NaN
1	NaN	5	10.0	15.0	NaN
2	price per unit	2	1.0	0.5	NaN

```
array([5, 10.0, 15.0], dtype=object)
array([2, 1.0, 0.5], dtype=object)
5
```

Out[1]:
```
{2: [5, 9.0], 1.0: [10.0, 14.0], 0.5: [15.0]}
```

Model Formulation

Parameters:

> **W** weeks of demand to process
>
> **R** set of ranges for different discount prices = {$[5, 10), [10, 15), [15, M]$}, r $\in$ R
>
> F_r Fixed charges for order amount X_w such that r[0] $\leq X_w$ < r[1]
>
> μ minimum order quantity
>
> **M** total demand expected over W weeks
>
> **K** initial amount on hand

Variables:

> X_w amount to order at week w
>
> H_w stock on hand at end of week w
>
> S_w minimum stock on hand for week w
>
> U_w per unit cost of stock held at end of week w
>
> O_w binary variable for X_w, representing the decision to order or not
>
> $P_{w,r}$ binary 1 if r[0] $\leq X_w$ < r[1], 0 otherwise
>
> D_w demand for week w
>
> C_w total cost in week w
>
> $\mathbf{C}_\mathrm{W}$ total cost after W weeks

Constraints:

MOQ constraint: must at least order the set minimum order amount each week

$$X_w \geq \mu \cdot O_w, \forall \mathrm{w}$$

$$X_w \leq M \cdot O_w + \mu, \forall \mathrm{w}$$

Amount on hand in the current week is dependent on what was ordered (x) and what was demanded (D) in the previous

$$X_{\text{w-1}} - D_{\text{w-1}} + H_{\text{w-1}} = H_{\mathrm{w}}$$

Minimum amount on hand constraint: must have at least the minimum amount in the current week

$$H_{\mathrm{w}} = X_{\text{w-1}} \cdot O_{w-1} - D_{\text{w-1}} + K,\ \mathrm{w} = 0$$

$$H_{\mathrm{w}} = X_{\text{w-1}} \cdot O_{w-1} - D_{\text{w-1}} + H_{\text{w-1}}, \forall\ \mathrm{w} > 0$$

$$H_{\mathrm{w}} \geq S_{\mathrm{w}}, \forall \mathrm{w}$$

picker constraints

$$\sum_r^3 (P_{w,r}) == 1$$

$$P_{\mathrm{w},1} == 1 \implies P_{\mathrm{w},2} == 0 \text{ and } P_{\mathrm{w},3} == 0$$

$$P_{\mathrm{w},1} == 1 \implies X_{w,r} \leq r[1]$$

$$P_{\mathrm{w},2} == 0 \implies X_{w,r} \leq r[1]$$

$$P_{\mathrm{w},3} == 0 \implies X_{w,r} \leq r[1]$$

$$P_{\mathrm{w},2} == 1 \implies P_{\mathrm{w},1} == 0 \text{ and } P_{\mathrm{w},3} == 0$$

$$P_{\mathrm{w},2} == 1 \implies X_{w,2} \geq r[2]$$

$$P_{\mathrm{w},1} == 0 \implies X_{w,2} \geq r[2]$$

$$P_{\mathrm{w},3} == 0 \implies X_{w,2} \geq r[2]$$

$$P_{\mathrm{w},3} == 1 \implies P_{\mathrm{w},1} == 0 \text{ and } P_{\mathrm{w},2} == 0$$

$$P_{\mathrm{w},3} == 1 \implies X_{w,3} \geq r[3]$$

$$P_{\mathrm{w},1} == 0 \implies X_{w,3} \geq r[3]$$

$$P_{\mathrm{w},2} == 0 \implies X_{w,3} \geq r[3]$$

Objective:

> - amount of units left over H_w at the end of the week times the per unit cost (U_w) plus the fixed cost of ordering if one was placed F
> - the summation of these terms for each week represents the overall cost over the W weeks.
> - This is what needs to be minimized

$$\text{minimize } (C_\mathrm{W} = \sum_{w=1}^{\mathrm{W}} (H_w \cdot U_w) + (\sum_{w=1}^{\mathrm{W}} \sum_r^3 (F_r \cdot P_{w,r}) \cdot O_w))$$

In [2]:
```
print("Full Data File:")
uniPrint(data_df_full)

print("Parameters Data File:")
uniPrint(data_df_params)
```

Full Data File:

	minimum order quantity (MOQ)	5	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8
0	Quantity discount schedule	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	minimum order amount	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	5	10	15	NaN	NaN	NaN	NaN	NaN
3	price per unit	2	1	0.5	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Week	Demand	Minimum inventory requirement	Order amount	Inventory held at end of week	Holding cost per unit	Cost of goods purchased	Holding cost	Is order feasible?
6	0	NaN	NaN	NaN	20	NaN	NaN	NaN	NaN
7	1	10	1	6	16	1	12	16	NaN
8	2	10	1	0	6	1	0	6	NaN
9	3	10	1	10	6	1	10	6	NaN
10	4	0	0	0	6	1	0	6	NaN
11	5	0	0	0	6	1	0	6	NaN
12	6	15	1.5	11	2	1	11	2	NaN
13	7	20	2	20	2	1	10	2	NaN
14	8	20	2	20	2	1	10	2	NaN
15	9	0	0	0	2	1	0	2	NaN
16	10	10	1	8	0	1	16	0	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	TOTAL COST	117	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Parameters Data File:

	Quantity discount schedule	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	NaN	minimum order amount	NaN	NaN	NaN
1	NaN	5	10.0	15.0	NaN
2	price per unit	2	1.0	0.5	NaN

In [3]:
```
def generate_obj(df, unit_cost_col, fr, Xws, Ows, Dws, Hws, Pw):
    expression = None
    for i in range(len(Xws)):
        if i == 0:
            order_cost = (fr[i,0] *Pw[i,0] + fr[i,1] *Pw[i,1] + fr[i,2] *Pw[i,2])
            expression =  (Hws[i])*df.loc[i, unit_cost_col] + order_cost
        else:
            order_cost = (fr[i,0] *Pw[i,0] + fr[i,1] *Pw[i,1] + fr[i,2] *Pw[i,2])
            expression +=  (Hws[i])*df.loc[i, unit_cost_col] + order_cost

    return expression
```

In [4]:
```
# generate the model
try:

    # Create a new model
    m = gp.Model("Individual_Project_3")


    ####################################################################################
    #########################          Parameter Set up           ######################
    ####################################################################################
    M = data_df.loc[:, "Demand"].sum()
    W = len(data_df)           # number of weeks
    params_str = "W:{}\nR:{}\n:mu:{}\nM:{}".format(W, discounts, MOQ, M)    # for displaying the parameters
    print(params_str)


    ####################################################################################
    #########################          Variables Set up           ######################
    ####################################################################################
    Xw = m.addVars(W, vtype=GRB.CONTINUOUS, name="X")     # amount to order at week w

    Hw = m.addVars(W, vtype=GRB.CONTINUOUS, name="H")             # stock on hand at end of week w
    Sw = m.addVars(W, vtype=GRB.CONTINUOUS, name="S")     # min stock on hand at end of week w

    Uw = m.addVars(W, vtype=GRB.CONTINUOUS, name="U")     # per unit cost for week w

    Dw = m.addVars(W, vtype=GRB.CONTINUOUS, name="D")     # demand for week w

#     Cs = m.addVars(W, 3, vtype=GRB.BINARY, name="C")

    Ow = m.addVars(W, vtype=GRB.BINARY, name="O", lb=0, ub=1)
    Pw = m.addVars(W, 3, vtype=GRB.BINARY, name="P", lb=0, ub=1)
    fixed_rates = [2.0, 1.0, 0.5]
    Fwr = m.addVars(W, 3, vtype=GRB.CONTINUOUS, name="F", lb=.001)

#     Cs = m.addVars(W, 3, vtype=GRB.BINARY, name="C")

# #     for w in range(W):
# #         for r in range(3):
# #             m.addConstr(Cs[w, r] ==  gp.all_(Ow[w], Pw[w, r]))
# # #             m.addConstr( (Cs[w, r] ==   gp.all_(Ow[w], Pw[w, r]))



    ####################################################################################
    #########################          Objective Set up           ######################
    ####################################################################################
    unit_cost_col="Holding cost per unit"
    obj_expr = generate_obj(data_df, unit_cost_col, Fwr, Xw, Ow, Dw, Hw, Pw)
    m.setObjective(obj_expr, GRB.MINIMIZE)


    ####################################################################################
    #########################          Constraints Set up         ######################
    ####################################################################################
    #  set constraints on the fixed costs for the different ranges
    for w in range(W):
        for r in range(3):
            m.addConstr(Fwr[w,r] == fixed_rates[r])



    #                                  add constraint for minimum and maximum amount ordered
    m.addConstrs(Xw[w] <= Ow[w] * M for w in range(W))
    m.addConstrs(Xw[w] >= Ow[w] *MOQ for w in range(W))

    #                                  add constrints on the minimum amount on hand
    m.addConstrs(Sw[w] == data_df.loc[w, "Minimum inventory requirement"]  for w in range(W))

    #                                  add constraints per unit holding costs
    m.addConstrs(Uw[w] == data_df.loc[w, "Holding cost per unit"]  for w in range(W))

    #                                   minimum on hand requirement by week
    m.addConstrs(Hw[w] >= Sw[w]  for w in range(W))

    #                                  Add on hand constraint/equation
    m.addConstr(Hw[0] == Xw[0]*Ow[0] - Dw[0] + 20)

    #                                  on hand constraint
    m.addConstrs(Hw[w] == Xw[w]*Ow[w] - Dw[w] + Hw[w-1]   for w in range(1, W))

    #                                   set weekly demands
    m.addConstrs(Dw[w] == data_df.loc[w, "Demand"] for w in range(W))


    expression = 0

    # constraint to make sure only one of the fixed cost binary selector variables in true at once
    for w in range(W):
        expression = 0
        for r in range(3):
            expression += Pw[w, r]
        m.addConstr(expression <= 1, name="singular_discount_price")

    # add constraints controling when the binary discount range variables are set
    for w in range(w):
        m.addConstr( (Pw[w, 0] == 1) >> (Xw[w] <=9.99))
        m.addConstr( (Pw[w, 1] == 1) >> (Xw[w] >=10) )
        m.addConstr( (Pw[w, 2] == 1) >> (Xw[w] >=15) )

        m.addConstr( (Pw[w, 0] == 0) >> (Xw[w] >= 10))
        m.addConstr( (Pw[w, 1] == 0) >> (Xw[w] <= 9.99) )
        m.addConstr( (Pw[w, 2] == 0) >> (Xw[w] <= 14.99) )


    ####################################################################################
    #########################          Solve Set up               ######################
    ####################################################################################
    m.optimize()

    displayDecisionVars(m, end_sentinel="9")

    print("\n-------------Does it make sense?----------------------")
    print('Obj-cost for the  10 weeks: ${:,.2f}'.format(m.ObjVal))
    #     m.addConstr((Pw[w,r] == 1) >> (Xw[t] < discount_ranges[w+1]) for w in range(W) for r in range(3))


# catch some math errors
except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

```
Restricted license - for non-production use only - expires 2023-10-25
W:10
R:{2: [5, 9.0], 1.0: [10.0, 14.0], 0.5: [15.0]}
:mu:5
M:95
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads
Optimize a model with 100 rows, 120 columns and 150 nonzeros
Model fingerprint: 0x03cb7145
Model has 30 quadratic objective terms
Model has 10 quadratic constraints
Model has 54 general constraints
Variable types: 80 continuous, 40 integer (40 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  QMatrix range    [1e+00, 1e+00]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  QObjective range [2e+00, 2e+00]
  Bounds range     [1e-03, 1e+00]
  RHS range        [5e-01, 2e+01]
  QRHS range       [2e+01, 2e+01]
  GenCon rhs range [1e+01, 2e+01]
  GenCon coe range [1e+00, 1e+00]
Presolve removed 62 rows and 81 columns
Presolve time: 0.00s
Presolved: 78 rows, 49 columns, 193 nonzeros
Variable types: 30 continuous, 19 integer (19 binary)
Found heuristic solution: objective 142.1500000

Root relaxation: objective 5.901200e+01, 28 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   62.81200    0    9  142.15000   62.81200  55.8%     -    0s
H    0     0                      95.0600000   62.81200  33.9%     -    0s
H    0     0                      90.0600000   62.81200  30.3%     -    0s
H    0     0                      83.0300000   67.06000  19.2%     -    0s
H    0     0                      67.0600000   67.06000  0.00%     -    0s
     0     0          -    0        67.06000   67.06000  0.00%     -    0s

Cutting planes:
  Learned: 5
  Gomory: 2
  Implied bound: 7
  MIR: 4
  Flow cover: 6
  RLT: 9
  PSD: 1

Explored 1 nodes (43 simplex iterations) in 0.03 seconds (0.00 work units)
Thread count was 12 (of 12 available processors)

Solution count 5: 67.06 83.03 90.06 ... 142.15

Optimal solution found (tolerance 1.00e-04)
Best objective 6.706000000000e+01, best bound 6.706000000000e+01, gap 0.0000%
X[0] 0
X[1] 5
X[2] 6
X[3] 0
X[4] 11
X[5] 15
X[6] 15
X[7] 15
X[8] 0
X[9] 9

H[0] 10
H[1] 5
H[2] 1
H[3] 1
H[4] 12
H[5] 12
H[6] 7
H[7] 2
H[8] 2
H[9] 1

S[0] 1
S[1] 1
S[2] 1
S[3] 0
S[4] 0
S[5] 2
S[6] 2
S[7] 2
S[8] 0
S[9] 1

U[0] 1
U[1] 1
U[2] 1
U[3] 1
U[4] 1
U[5] 1
U[6] 1
U[7] 1
U[8] 1
U[9] 1

D[0] 10
D[1] 10
D[2] 10
D[3] 0
D[4] 0
D[5] 15
D[6] 20
D[7] 20
D[8] 0
D[9] 10

O[0] 0
O[1] 1
O[2] 1
O[3] 0
O[4] 1
O[5] 1
O[6] 1
O[7] 1
O[8] 0
O[9] 1

P[0,0] 1
P[0,1] 0
P[0,2] 0
P[1,0] 1
P[1,1] -0
P[1,2] 0
P[2,0] 1
P[2,1] 0
P[2,2] 0
P[3,0] 1
P[3,1] 0
P[3,2] 0
P[4,0] 0
P[4,1] 1
P[4,2] 0
P[5,0] 0
P[5,1] 1
P[5,2] 0
P[6,0] 0
P[6,1] 1
P[6,2] 0
P[7,0] 0
P[7,1] 1
P[7,2] 0
P[8,0] 1
P[8,1] 0
P[8,2] 0
P[9,0] 0

P[9,1] 0

P[9,2] 0

F[0,0] 2
F[0,1] 1
F[0,2] 0
F[1,0] 2
F[1,1] 1
F[1,2] 0
F[2,0] 2
F[2,1] 1
F[2,2] 0
F[3,0] 2
F[3,1] 1
F[3,2] 0
F[4,0] 2
F[4,1] 1
F[4,2] 0
F[5,0] 2
F[5,1] 1
F[5,2] 0
F[6,0] 2
F[6,1] 1
F[6,2] 0
F[7,0] 2
F[7,1] 1
F[7,2] 0
F[8,0] 2
F[8,1] 1
F[8,2] 0
F[9,0] 2

F[9,1] 1

F[9,2] 0

-------------------------------------



-------------Does it make sense?----------------------
Obj-cost for the  10 weeks: $67.06
```

sanity check to make sure at least the math makes sense

In [5]:
```
val = 0
for w in range(W):
    val += Hw[w].X * 1
    for r in range(3):
        val += Pw[w, r].X*Fwr[w, r].X

print(val)
```

```
67.05999999999992
```

Solution Discussion:

> From the implemented model the **optimal order amounts are**:
>
> - X[0] = 0
> - X[1] = 5
> - X[2] = 6
> - X[3] = 0
> - X[4] = 11
> - X[5] = 15
> - X[6] = 15
> - X[7] = 15
> - X[8] = 0
> - X[9] = 9
>
> The above details the order amounts for each of the given time steps that lead to an objective value of $67.06K

Save Notebook:

Run the cel below to save the current notebook as a pdf in the same directory

In [ ]:
```
to_PDF("_Independent_Project_3.ipynb")
```

```
filename: _Independent_Project_3.ipynb
```

In [ ]: