Gerald Jones

Individual Project 2

ISE522 Spg 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. Currently the **MOQ**, the minimum amount that you must order if you place an order, is **zero**. Later variations of this problem will have a non-zero order quantity. The attached spreadsheet includes all of the data that you need as well as an example solution.
>
> - fixed cost charges for ordering any amount i.e. regardless of amount
> - inventory held at the end of week has a fixed cost, i.e. any amount above zero costs some set amount
> - minimum amount of stock held at the end of the week must not be crossed, i.e. there is some set cost that will incur based on this minimum held value due to the cost per unit penalty of stock held across weeks.

Model Formulation

Parameters:

> ***M*** minimum order qauntity
>
> F fixed charges for ordering
>
> W number of weeks to forecast

Variables:

> ***w*** week w $\in$ 1={1, ..., W}
>
> H_w stock on hand at end of week w
>
> S_w minimum stock on hand for week w
>
> X_w amount to order in 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
>
> D_w demand for week w
>
> C_W cost after W weeks

Constraints:

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

$$X_w \geq M, \forall \text{ w}$$

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

$$(X_{\text{w-1}} \cdot O_{\text{w-1}}) - D_{\text{w-1}} + H_{\text{w-1}} = H_{\text{w}}$$

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

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

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

$$\text{minimize}(C_{\text{W}} = \sum_{w=1}^{W}(H_{\text{w}} \cdot U_{\text{w}}) + (F \cdot O_{\text{w}}))$$

In [1]:
```python
# there are some methods defined here used to
# * add variables to the model
# * print the results of an model
# * add constraintes to the model
#
from _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
import os

# grab parameter data
params_df = pd.read_excel("single item data.xlsx", skiprows=None, nrows=2, header=None, index_col=0, usecols=[
Fixed_Charge = params_df.loc["Fixed charge for ordering", 1]
MOQ = params_df.loc[params_df.index.to_list()[1], 1]
print("\t\t\tParameters:\n\tFixed order charge: {: 7d}\n\tMinimum order quantity: {: 3.0f}".format(Fixed_Charg

# grab monthly data
data_df = pd.read_excel("single item data.xlsx", skiprows=3, nrows=11)

# get the total cost from the data file
total_cost = pd.read_excel("single item data.xlsx", skiprows=15).iloc[0, 1]

uniPrint(params_df)
uniPrint(data_df)
uniPrint(total_cost)
```

```
                Parameters:
        Fixed order charge:      100
        Minimum order quantity:    0
```

	1
0	
Fixed charge for ordering	100
min order qty	0

	Week	Demand	Minimum inventory requirement	Order amount	Inventory held at end of week	Holding cost per unit	Fixed charge	Holding cost
0	0	NaN	NaN	NaN	20	NaN	NaN	NaN
1	1	10.0	1.0	2.0	12	2.0	100.0	24.0
2	2	10.0	1.0	2.0	4	2.0	100.0	8.0
3	3	10.0	1.0	7.0	1	2.0	100.0	2.0
4	4	0.0	0.0	2.0	3	2.0	100.0	6.0
5	5	0.0	0.0	2.0	5	2.0	100.0	10.0
6	6	15.0	1.5	11.0	1	2.0	100.0	2.0
7	7	20.0	2.0	20.0	1	2.0	100.0	2.0
8	8	20.0	2.0	20.0	1	2.0	100.0	2.0
9	9	0.0	0.0	2.0	3	2.0	100.0	6.0
10	10	10.0	1.0	8.0	1	2.0	100.0	2.0

```
1064
```

In [15]:
```python
def generate_obj(df, unit_cost_col, foc, Xws, Ows, Dws, Hws,):
    expression = None
    print(foc)
    for i in range(len(Xws)):
        print('{}: thing: '.format(i) + str(df.loc[i+1, unit_cost_col]))
        if i == 0:
            expression =  (Hws[i])*df.loc[i+1, unit_cost_col] + (foc*Ows[i])
        else:
            expression += (Hws[i])*df.loc[i+1, unit_cost_col] + (foc*Ows[i])
    return expression

def add_sequential_operation(model, Xws, Ows, Dws, Hws, initval):
    for i in range(0, len(Xws)):
        # use the given initial value for the amount at the end of the first week
        if i == 0:
            print("initval ", initval)
            model.addConstr(Xws[i]*Ows[i] + initval - Dws[i] == Hws[i])
        else:
            model.addConstr(Xws[i]*Ows[i] + Hws[i-1] - Dws[i] == Hws[i])


def add_product_GEQ(model, Xws, Ows, minval):
    for i in range(0, len(Xws)):
            model.addConstr(Xws[i]*Ows[i] == Xws[i])


def add_value_EQconstraints(model, value,  Vl, idx):
    model.addConstr(Vl[idx] == value)

def add_value_GEQconstraints(model, value,  Vl, idx):
    model.addConstr(Vl[idx] >= value)

def add_value_GEQconstraintS(model, value,  Vl, start=0):
    for idx in range(start, len(Vl)):
        model.addConstr(Vl[idx] >= value)

def add_value_LEQconstraints(model, value,  Vl, idx):
    model.addConstr(Vl[idx] <= value)



def add_value_EQconstraintsDF(model, df, Vl, col="Minimum inventory requirement"):
    for i in range(len(Vl)):
        add_value_EQconstraints(model, df.loc[i+1, col],  Vl, i)


def add_value_GEQconstraintsDF(model, df, Vl, col="Minimum inventory requirement"):
    for i in range(len(Vl)):
        add_value_GEQconstraints(model, df.loc[i+1, col],  Vl, i)

def add_value_LEQconstraintsDF(model, df, Vl, col="Minimum inventory requirement"):
    for i in range(len(Vl)):
        add_value_LEQconstraints(model, df.loc[i+1, col],  Vl, i)
```

Model construction and optimization

Assumptions:

- orders are done in integer units
- demands are given in integer units
- the stock on hand at the end of the week is given in integer units

In [16]:
```python
# generate the model
try:
    # Create a new model
    m = gp.Model("Individual_Project_2")
    MOQ = params_df.loc[params_df.index.to_list()[1], 1]


    # add the amount to order variables for each week
    Xw = generate_n_vars(m, count=10, vtype=GRB.INTEGER, base_name="Xw", lb=MOQ)

    # add binary order decision variables for each week
    Ow = generate_n_vars(m, count=10, vtype=GRB.BINARY, base_name="Ow")

    # add demand variables for each week
    Dw = generate_n_vars(m, count=10, vtype=GRB.INTEGER, base_name="Dw")

    # add on hand restrictions variables
    Hw = generate_n_vars(m, count=10, vtype=GRB.INTEGER, base_name="Hw",)

    # add objective function of the form given above
    print('making objective')
    objective_expression =generate_obj(data_df, unit_cost_col="Holding cost per unit",
                                      foc=Fixed_Charge,
                                      Xws=Xw,
                                      Ows=Ow,
                                      Dws=Dw,
                                      Hws=Hw,)
    m.setObjective(objective_expression, GRB.MINIMIZE)

    # set up demand values
    add_value_EQconstraintsDF(m, data_df, Dw, col="Demand")

    # add constraints for amount on hand based on last week
    add_sequential_operation(m, Xw, Ow, Dw, Hw, data_df.loc[0, "Inventory held at end of week"])

    # add constraint on amount on minimum amount on hand
    add_value_GEQconstraintsDF(m, data_df, Hw, col="Minimum inventory requirement")


    m.optimize()

    displayDecisionVars(m, end_sentinel="10")


    print("\n-------------Does it make sense?----------------------")
    print('Obj-cost for the  10 weeks: ${:,.2f}'.format(m.ObjVal))
# catch some math errors
except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

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

```
making objective
100
0: thing: 2.0
1: thing: 2.0
2: thing: 2.0
3: thing: 2.0
4: thing: 2.0
5: thing: 2.0
6: thing: 2.0
7: thing: 2.0
8: thing: 2.0
9: thing: 2.0
initval  20
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 20 rows, 40 columns and 20 nonzeros
Model fingerprint: 0x671d581b
Model has 10 quadratic constraints
Variable types: 0 continuous, 40 integer (10 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  QMatrix range    [1e+00, 1e+00]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [2e+00, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+01]
  QRHS range       [2e+01, 2e+01]
Presolve removed 20 rows and 10 columns
Presolve time: 0.00s
Presolved: 30 rows, 60 columns, 79 nonzeros
Presolved model has 20 SOS constraint(s)
Variable types: 0 continuous, 60 integer (20 binary)
Found heuristic solution: objective 680.0000000

Root relaxation: objective 8.200000e+01, 3 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   82.00000    0    5  680.00000   82.00000  87.9%     -    0s
H    0     0                     582.0000000   82.00000  85.9%     -    0s
H    0     0                     578.0000000   82.00000  85.8%     -    0s
H    0     0                     514.0000000   82.00000  84.0%     -    0s
     0     0  182.00000    0    5  514.00000  182.00000  64.6%     -    0s
H    0     0                     492.0000000  182.00000  63.0%     -    0s
     0     2  182.00000    0    5  492.00000  182.00000  63.0%     -    0s
H    3     8                     464.0000000  182.00000  60.8%   0.3    0s
H   24     8                     442.0000000  304.00000  31.2%   0.3    0s

Explored 35 nodes (11 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 12 (of 12 available processors)

Solution count 7: 442 464 492 ... 680

Optimal solution found (tolerance 1.00e-04)
Best objective 4.420000000000e+02, best bound 4.420000000000e+02, gap 0.0000%
Xw1 0.0
Xw2 11.0
Xw3 0.0
Xw4 0.0
Xw5 0.0
Xw6 36.0
Xw7 0.0
Xw8 29.0
Xw9 0.0
Xw10 0.0

Ow1 0.0
Ow2 1.0
Ow3 0.0
Ow4 0.0
Ow5 0.0
Ow6 1.0
Ow7 0.0
Ow8 1.0
Ow9 0.0
Ow10 0.0

Dw1 10.0
Dw2 10.0
Dw3 10.0
Dw4 0.0
Dw5 0.0
Dw6 15.0
Dw7 20.0
Dw8 20.0
Dw9 0.0
Dw10 10.0

Hw1 10.0
Hw2 11.0
Hw3 1.0
Hw4 1.0
Hw5 1.0
Hw6 22.0
Hw7 2.0
Hw8 11.0
Hw9 11.0
Hw10 1.0

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



-------------Does it make sense?----------------------
Obj-cost for the  10 weeks: $442.00
```

Results Discussion:

> From the results the company should make **orders** in the **second, sixth, and eight weeks** with **amounts of 11, 36, and 29 units** respectively. This will lead to an overall **minimized cost of $442.00** at the end of the ten week period.

References:

- gurobi constraints