Gerald Jones, Kimon Swanson, Alex Hines

Home Work 4, Problem 1: Balanced Transportion Problem

ISE522 Spg 22

Notebook Links/Sections:

1. Data Display section: Display of Data for warehouses and customers
2. Model Formulation: Mathematical formulation of problem
3. Method Definitions: Python code for various tasks
4. Gurobi Implementation: Definition and omptimization with python and Gurobi
5. Solution Discussion: A discussion and explanation of the solution

Problem Description:

> A company supplies goods to **three customers**, who **each demand 30 units**. The company has **two warehouses**. **Warehouse 1 has 40 units** on hand and **Warehouse 2 has 30**. The **costs of shipping 1 unit from warehouse to a given customer are shown in the table displayed below in the [Data Display section](#Data-Display)** . There is a **penalty for unmet demand** that is **specific to each customer** and these are also displayed in the Data Display section. The task is to formulate this blanced transportation problem to **minimize the sum of the shortage and shipping costs**.

Notes and Observations

- total supply across all locataion is less than total demand
- confilcting objectives:
  - increasing supply increases cost
  - not supplying what is demanded increases cost
- means need just the right amounts supplied from each warehouse to customer to get optimal solution

Assumptions:

- units are integer values i.e. the smallest value of units above zero is 1
- Customers can be supplied by multiple warehouses to meet demands

Module imports and data loading

In [9]:
```
from _GUROBI_TOOLS_.GUROBI_MODEL_BUILDING_TOOLS import *
from _NOTE_BOOK_UTILS import *

notebook_title = "_HW4_Problem1.ipynb"

try:
    data_file1 = "WarehouseData.xlsx"
    data_file2 = "CustomerPenaltyData.xlsx"
    warehouse_df = pd.read_excel(data_file1)
    customer_df = pd.read_excel(data_file2)
except Exception as ex:
    print("error loading file")
    print("Exception: {}".format(ex))
```

Data Display

In [46]:
```
print("\t\t\t\tWarehouse Data")
display(warehouse_df)
print("\t\t\t\tCustomer Data")
display(customer_df)
```

Warehouse Data

	Warehouse	Customer-1	Customer-2	Customer-3
0	1	15	35	25
1	2	10	50	40

Customer Data

	Customer	Penalty
0	1	90
1	2	80
2	3	110

Model Formulation

Model Formulation Links/Sections:

- Paremeters and Sets
- Variables
- Equations and Constraints
- Objective

Parameters and Sets:

$\mathbf{W}$ set of warehouses, $w \in \mathbf{W}$

$\mathbf{C}$ set of customers, $c \in \mathbf{C}$

D_c demand for customer c

P_c unmet penalty for customer c

H_w amount on hand for warehouse w

$S_{w,c}$ shipping costs for warehouse w to customer c

Variables:

$X_{w,c}$ amount from warehouse w supplied to customer c

Y_c unmet demand for customer c

M total supply cost for warehouse

N total unmet demand cost

Equations and Constraints:

> **Total units supplied by warehouse w constraint**

$$0 \leq \sum_{c=1}^{|C|} X_{w,c} \leq H_w, \forall w$$

> **Total supply cost**

$$M = \sum_{w=1}^{|W|}\sum_{c=1}^{|C|}(X_{w,c} \cdot S_{w,c})$$

> **Total unmet demand for customer c**

$$Y_c = D_c - \sum_{w=1}^{|W|}(X_{w,c}) \quad ,\forall c$$

$$Y_c \geq 0, \forall c$$

> **Total unmet demand cost**

$$N = \sum_{c=1}^{|C|}(Y_c \cdot P_c) \quad ,\forall c$$

Objective: Minimize total costs for shipping and unmet demand

$$\min(N + M)$$

Method Definitions

In [67]:
```
# generate constraints for the amount a given warehouse
# can supply
def Xwc_supply_constraints(model, X, onHands, W, C):

    for w in range(W):
        expression = 0
        for c in range(C):
            expression += X[w, c]
        model.addConstr(expression <= onHands[w])
        # it has to supply something
        model.addConstr(expression >= 1)
    return

# set expression for total supply costs
def total_supply_cost(model, M, X, S, W, C):
    expression = 0
    for w in range(W):
        for c in range(C):
            expression += X[w, c] * S[w, c]
    model.addConstr(M == expression)
    model.addConstr(M >= 1)



# set expression for each customers unmet demand
def set_unmet_demand(model, D, X, Y, C, W):
    # for each customer
    for c in range(C):
        expression = 0
        # sum up the contribution to its demand from each warehouse
        for w in range(W):
            expression += X[w, c]
        # the current customers unmet demand is its demand minus what it was supplied
        # by the warehouses
        model.addConstr(Y[c] == D[c] - expression )

        # the unmet demand can at least be zero
        # this ensures that the sum of the supplied demand can not exceed the demand itself
        model.addConstr(Y[c] >= 1)
    return

# set expression for total unmet demand costs
def total_unmet_demand_cost(model, N, Y, D, P, C):
    expression = 0
    for c in range(C):
        expression += Y[c] * P[c]
    model.addConstr(N == expression)
    model.addConstr(N >= 0)
    return
```

Gurobi Implementation and Solution

In [68]:
```
try:
    # instantiate model object
    m = gp.Model("G_MOD")



    ###########################################################################################
    ################################## Parameters set up ####################################
    ###########################################################################################
    W = 2                                                               # number of warehouses
    C = customer_df.shape[0]                                            # number of customers
    Pc = customer_df["Penalty"].tolist()                                # unmet demand penalties
    Hw = [40, 30]                                                       # on hand for each warehouse

    customer_labels = ["Customer-1", "Customer-2", "Customer-3"]
    ShippingCosts = warehouse_df.loc[:,customer_labels ].values         # shipping costs
    print(ShippingCosts)

    Dc = [30, 30, 30]                                                   # customer demands

    print("Parameters:\nW={}\nC={}\nPc={}\nHw={}\n".format(W, C, Pc, Hw))

    ###########################################################################################
    ################################## Variables set up #####################################
    ###########################################################################################
    Xwc = m.addVars(W, C, vtype=GRB.INTEGER, name="X", lb=0,)
    Yc = m.addVars(C, vtype=GRB.INTEGER, name="Y", lb=0,)
    M = m.addVar(1, vtype=GRB.CONTINUOUS, name="M",)
    N = m.addVar(1, vtype=GRB.CONTINUOUS, name="N")


    ###########################################################################################
    ################################## Objective set up #####################################
    ###########################################################################################
    m.setObjective(N+M, GRB.MINIMIZE)


    ###########################################################################################
    ################################## Constraint set up ####################################
    ###########################################################################################
    Xwc_supply_constraints(m, Xwc, Hw, W, C)


    # set the total shipping cost
    total_supply_cost(m, M, Xwc, ShippingCosts, W, C)

    # set the unmet demand expression
    set_unmet_demand(m, Dc, Xwc, Yc, C, W)

    # set total unmet demand cost expression
    total_unmet_demand_cost(m, N, Yc, Dc, Pc, C)

    ###########################################################################################
    ################################## SOLVE:OPTIMIZE #######################################
    ###########################################################################################


    m.optimize()

    ###########################################################################################
    ################################## Display Results ######################################
    ###########################################################################################
    displayDecisionVars(m, end_sentinel="2")

    print("\n-------------Does it make sense?----------------------")
    print('Obj: {:.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')
```

```
[[15 35 25]
 [10 50 40]]
Parameters:
W=2
C=3
Pc=[90, 80, 110]
Hw=[40, 30]

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 14 rows, 11 columns and 37 nonzeros
Model fingerprint: 0xbf04315a
Variable types: 2 continuous, 9 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+01]
Presolve removed 7 rows and 5 columns
Presolve time: 0.00s
Presolved: 7 rows, 6 columns, 18 nonzeros
Variable types: 0 continuous, 6 integer (0 binary)
Found heuristic solution: objective 3730.0000000

Root relaxation: objective 3.090000e+03, 5 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               0    3090.0000000 3090.00000  0.00%     -    0s

Explored 1 nodes (5 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 12 (of 12 available processors)

Solution count 2: 3090 3730

Optimal solution found (tolerance 1.00e-04)
Best objective 3.090000000000e+03, best bound 3.090000000000e+03, gap 0.0000%
X[0,0] 0.00000
X[0,1] 12.00000
X[0,2] 28.00000

X[1,0] 29.00000
X[1,1] -0.00000
X[1,2] 1.00000

Y[0] 1.00000
Y[1] 18.00000
Y[2] 1.00000

M 1450.00000
N 1640.00000
------------------------------------



-------------Does it make sense?----------------------
Obj: 3090.00
```

Solution Discussion

> ### The solution....
>
> The problem requires that a decision be made about how much each warehouse will supply to which customer. Looking at the unmet demand penalties the general idea should be to supply the customers with the higher penalties more than those with lower penalties to minimize unmet demand cost. From the supply costs from each warehouse to each customer the warehouse 1 has lower supply costs overall compared to warehouse 2 for any given customer. Since warehouse 1 has more supply, I would expect that it supplies most of the demand with warehouse 2 picking up the slack. For both warehouses the magnitude of demand in ascending order is customers 1, 2, and 3. Since customer 1 has a lower cost for warehouse 2 I would try to send most of the supply for that customer from warehouse 2. This would mean that warehouse 1 would need to cover most of the demands for customers 2, and 3. Customers 1 and 3 have higher unmet demand penalties than customer 2 so I would allow for more unmet demand to this customer. I would supply what I had left over in warehouse 1 to customer 2 to met what demand I could. I would supply as close to what customer 3 demanded as possible since it has the highest unmet demand cost. This ad hoc solution is like what is seen in the solution produced by Gurobi.
>
> ### The optimal solution generated by the implemented model suggests to:
>
> - **Have Warehouse 1:**
>   - supplies most of **customers 3's** demand **(28/30 units)**
>   - as well as a portion of the demand for **customer 2 (12/30 units)**
> - **Have Warehouse 2:**
>   - covers almost all the demand for **customer 1 (29/30 units)**
>   - as well as **(1/30) units for customer 3**.
>
> The leads to customer 1 supplied with 28 units, customer 2 with 12 units, and customer 3 with 29 units. This configuration of the model leads to a **total supply cost of 1450, and a total unmet demand cost of 1640 for a total overall cost of 3090.00**.

In [71]:
```
# save the notebook as a pdf
to_PDF(notebook_title)
```

```
filename: _HW4_Problem1.ipynb
```