Transportation Problem

IMPORTS

Untitled.ipynb

Code Python 3

[35]:

```
import shutil
import sys
import os.path

if not shutil.which("pyomo"):
    !pip install -q pyomo
    assert(shutil.which("pyomo"))

if not (shutil.which("cbc") or os.path.isfile("cbc")):
    if "google.colab" in sys.modules:
        !apt-get install -y -qq coinor-cbc
    else:
        try:
            !conda install -c conda-forge coincbc
        except:
            pass

assert(shutil.which("cbc") or os.path.isfile("cbc"))

from pyomo.environ import *
```

The problem objective is to minimize the total shipping cost to all customers from all sources.

$$\text{minimize:Cost}=\sum_{c\in\text{Customers}}\sum_{s\in\text{Sources}}T[c,s]x[c,s]$$

Shipments from all sources can not exceed the manufacturing capacity of the source.

$$\sum_{c\in\text{Customers}}x[c,s]\leq\text{Supply}[s]\forall s\in\text{Sources}$$

Shipments to each customer must satisfy their demand.

$$\sum_{s\in\text{Sources}}x[c,s]=\text{Demand}[c]\forall c\in\text{Customers}$$

DATA FILE (PYOMO MODEL)

[33]:
```
Demand = {
   'Lon':   125,        # London
   'Ber':   175,        # Berlin
   'Maa':   225,        # Maastricht
   'Ams':   250,        # Amsterdam
   'Utr':   225,        # Utrecht
   'Hag':   200         # The Hague
}

Supply = {
   'Arn':   600,        # Arnhem
   'Gou':   650         # Gouda
}

T = {
    ('Lon','Arn'): 1000,
    ('Lon','Gou'): 2.5,
    ('Ber','Arn'): 2.5,
    ('Ber','Gou'): 1000,
    ('Maa','Arn'): 1.6,
    ('Maa','Gou'): 2.0,
    ('Ams','Arn'): 1.4,
    ('Ams','Gou'): 1.0,
    ('Utr','Arn'): 0.8,
    ('Utr','Gou'): 1.0,
    ('Hag','Arn'): 1.4,
    ('Hag','Gou'): 0.8
}
```

CONCRETE MODEL: -

[36]:
```
model = ConcreteModel()
model.dual = Suffix(direction=Suffix.IMPORT)

# Step 1: Define index sets
CUS = list(Demand.keys())
SRC = list(Supply.keys())

# Step 2: Define the decision
model.x = Var(CUS, SRC, domain = NonNegativeReals)

# Step 3: Define Objective
model.Cost = Objective(
    expr = sum([T[c,s]*model.x[c,s] for c in CUS for s in SRC]),
    sense = minimize)

# Step 4: Constraints
model.src = ConstraintList()
for s in SRC:
    model.src.add(sum([model.x[c,s] for c in CUS]) <= Supply[s])

model.dmd = ConstraintList()
for c in CUS:
    model.dmd.add(sum([model.x[c,s] for s in SRC]) == Demand[c])

results = SolverFactory('cbc').solve(model)
results.write()
```

SOLUTION PRINTING: -

[37]:
```
for c in CUS:
    for s in SRC:
        print(c, s, model.x[c,s]())
```

```
Lon Arn None
Lon Gou None
Ber Arn None
Ber Gou None
Maa Arn None
Maa Gou None
Ams Arn None
Ams Gou None
Utr Arn None
Utr Gou None
Hag Arn None
Hag Gou None
```

[38]:
```
if 'ok' == str(results.Solver.status):
    print("Total Shipping Costs = ",model.Cost())
    print("\nShipping Table:")
    for s in SRC:
        for c in CUS:
            if model.x[c,s]() > 0:
                print("Ship from ", s," to ", c, ":", model.x[c,s]())
else:
    print("No Valid Solution Found")
```

```
Total Shipping Costs =  1705.0

Shipping Table:
Ship from  Arn  to  Ber : 175.0
Ship from  Arn  to  Maa : 225.0
Ship from  Arn  to  Utr : 200.0
Ship from  Gou  to  Lon : 125.0
Ship from  Gou  to  Ams : 250.0
Ship from  Gou  to  Utr : 25.0
Ship from  Gou  to  Hag : 200.0
```