Solving Multi-Commodity Transportation Problems

Shervin Iranaghideh

Instructor : Dr. Farhad Hosseinzadeh Lotfi

Mathematical Modeling of Multi-Commodity Transportation Problem

Multi-commodity transportation problems are a type of Multi-commodity Network Flow Problems (MCNF) which all themselves are a type of optimization problems that the goal is to minimize the total cost of transportation of multiple types of good from different sources to different destinations.

These kind of problems have many applications in various real-life fields, here we are describing a simple example of the Multi-commodity transportation problems.

In the first part we are going to formulate the whole problem into a linear programming problem in which we describe the Decision Variables, Objective Function and Constraints in a mathematical model.

The second part is dedicated to solving the problem with computer using Python and one of it's libraries called Scipy, we use linprog() method from scipy.optimize to solve our problem.

To describe the problem lets say there is a firm producing p kind of goods at n different supply centers. We label these $\ell = 1, \ldots, p$ and $i = 1, \ldots, n$ respectively. The amount of each ℓ supply produced at each supply center i is $S_{i,\ell}$. There are also m different demand centers that we label them $j = 1, \ldots, m$. The demand of ℓth good at jth center is $D_{j,\ell}$. The problem is to get goods from supply centers to demand centers at minimum cost. Lets assume that the cost of shipping one unit of product ℓ from supply center i to demand center j is $c_{i,j,\ell}$, and that shipping cost is linear meaning that if we shipped $x_{i,j,\ell}$ units of product ℓ from supply center i to demand center j , then the cost would be $c_{i,j,\ell} \, x_{i,j,\ell}$.

As the formulating the problem goes, lets define $x_{i,j,\ell}$ to be the number of units of the product ℓ shipped from supply center i to demand center j as said above, which in our linear programming problem are called the Decision Variables. The objective is to minimize the total cost of the shipping which we define as our Objective Function :

$$min \sum_{i} \sum_{j} \sum_{\ell} c_{i,j,\ell} \, x_{i,j,\ell}$$

where $c_{i,j,\ell}$ is the cost of shipping one unit of product ℓ from supply center i to demand center j as said above.

Our constraints are that we must meet the demands of each product at each demand center and we cannot surpass our supply available at each supply center.

Now consider the supply center i, the total product ℓ shipped out of the supply center i is $\sum_{i=1}^{n} x_{i,j,\ell}$ $(for\ all\ j = 1,\ \dots,\ m)$. The sum given adds up all the shipments from supply center i which cannot surpass the quantity available at the supply center. Hence we have the constraint

$$\sum_{j} x_{i,j,\ell} \leq S_{i,\ell}\ for\ \forall i, \ell$$

in which $S_{i,\ell}$ represents the amount of product ℓ available in supply center i.

Likewise the constraints that makes sure that we satisfy the demand of each demand centers are:

$$\sum_{i} x_{i,j,\ell} = D_{j,\ell}\ for\ \forall j, \ell$$

in which $D_{j,\ell}$ is the demand of product ℓ at supply center j.

There is another constraint which is the maximum amount that can be shipped in one visit (assuming each supply center only ships products maximum of one time to each demand center), this constraint is a lower bound and a upper bound for our decision variables.

The number of units shipped obviously cannot be negative so we have one constraint for that, and also the amount shipped cannot exceed our upper bound (lets name it u), so to formulate this we have:

$$0 \leq x_{i,j,\ell} \leq u \ \ for \ \ \forall i, j, \ell$$

Wrapping it all up into a linear programming problem:

$$min \sum_i \sum_j \sum_\ell c_{i,j,\ell} \, x_{i,j,\ell}$$

$$S.T. \ \ \sum_j x_{i,j,\ell} \leq S_{i,\ell} \ \ for \ \ \forall i, \ell$$

$$\sum_i x_{i,j,\ell} = D_{j,\ell} \ \ for \ \ \forall j, \ell$$

$$0 \leq x_{i,j,\ell} \leq u \ \ for \ \ \forall i, j, \ell$$

Now lets solve a simple example:

Suppose we have two supply centers, A and B. Center A has 10 units of good X and 15 units of good Y. Center B has 20 units of good X and 5 units of good Y.

We also have two demand centers, C and D. Center C requires 15 units of good X and 10 units of good Y. Center D requires 15 units of good X and 10 units of good Y.

The cost of transporting one unit of good X from center A to center C is $2, from center A to center D is $3, from center B to center C is $4, and from center B to center D is $1.

The cost of transporting one unit of good Y from center A to center C is $3, from center A to center D is $2, from center B to center C is $1, and from center B to center D is $4.

Can you find the optimal transportation plan that minimizes the total transportation cost?

Next page I made a diagram for the problem.
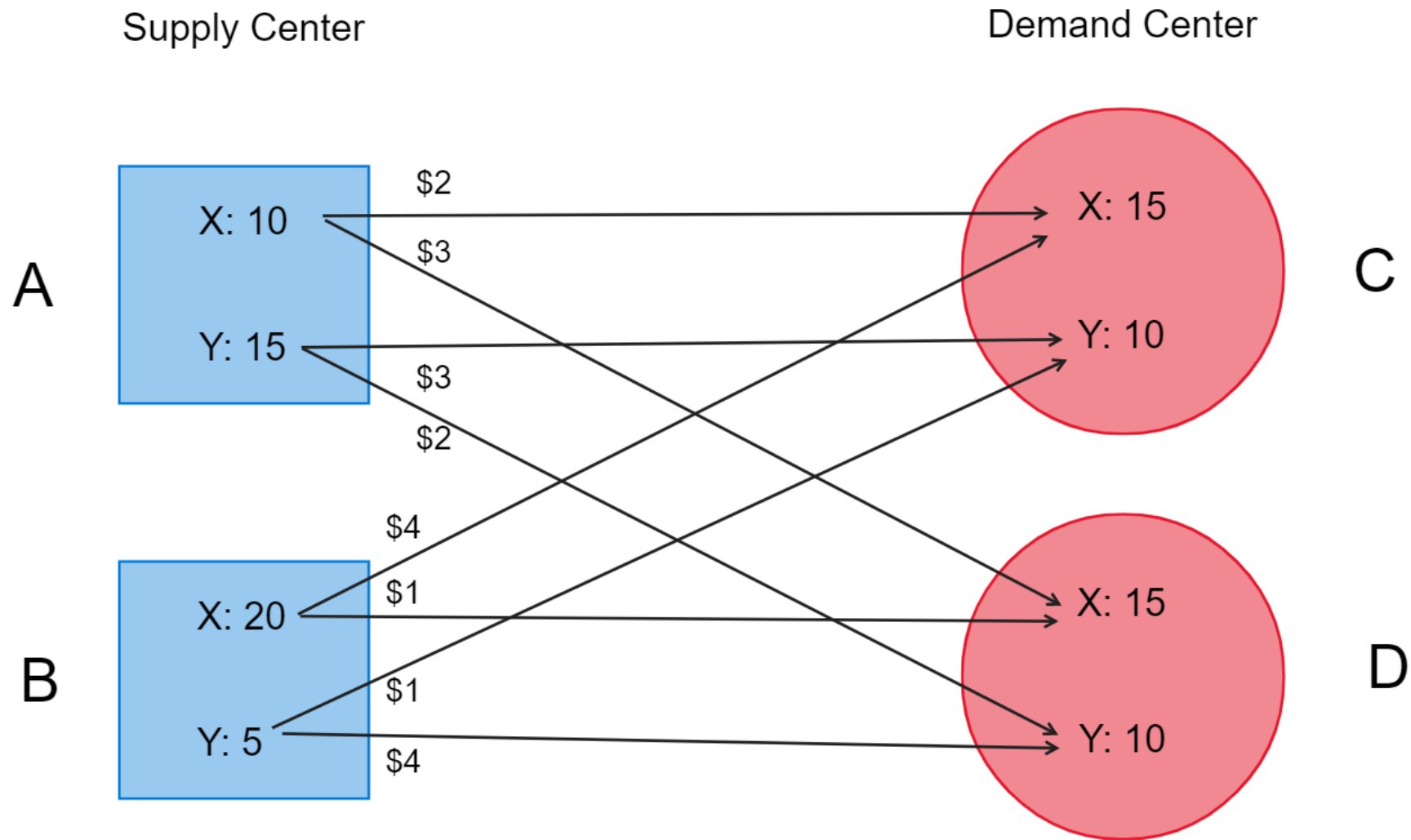
Solving The Problem Using Python Scipy :

Here we are using scipy.optimize to solve the problem , the library has a method called linprog() which takes different parameters, here we are explaining how to organize multi-commodity problem data so we can use them as a parameter for this function.

First we have to import the libraries we want to work with, here I'm using numpy for easier data manipulation, so we need to import numpy in addition to scipy.optimize :

```python
import numpy as np
from scipy.optimize import
linprog
```

Then we are going to store our data into variables so we can manipulate them, our data is the number of supply centers, demand centers and product types, we call them m, n and p respectively. We also dedicate a list for each supply center and demand center in which we include the supply and demand of each product in order and respectively.

So we should have something like this:

```python
# Number of supply centers
m = 2

# Number of demand centers
n = 2

# Number of goods
p = 2

# Supply limit for each good at each supply center (m x p matrix)
supply = [[10, 15], [20, 5]]

# Demand amount for each good at each demand center (n x p matrix)
demand = [[15, 10], [15, 10]]

# Cost of sending each good to each demand center from each supply center (m x n x p matrix)
cost = [[[2, 3], [3, 2]], [[4, 1], [1, 4]]]
```

As you can see [10, 15] in supply means that in supply center 1, the demand for product 1 is 10 and the demand for product 2 is 15.
Likewise [15, 10] in demand means the in the first demand center the demand for product 1 is 15 and the demand for product 2 is 10.
We also added the costs of each transportation, as you can see in the cost list there are 2 nested lists which represent our supply centers, then in each of the two nested lists there are 2 other nested lists representing our demand centers in which the demand for each product is included respectively.

The lingprog function takes the following parameters:

Parameters:

c : 1-D array

The coefficients of the linear objective function to be minimized.

A_ub : 2-D array, optional

The inequality constraint matrix. Each row of A_ub specifies the coefficients of a linear inequality constraint on x.

b_ub : 1-D array, optional

The inequality constraint vector. Each element represents an upper bound on the corresponding value of A_ub @ x.

A_eq : 2-D array, optional

The equality constraint matrix. Each row of A_eq specifies the coefficients of a linear equality constraint on x.

b_eq : 1-D array, optional

The equality constraint vector. Each element of A_eq @ x must equal the corresponding element of b_eq.

As said before c is the list of the coefficients of our objective function which is in order of our decision variables in respect of their index so for making c we simply have:

```python
# Objective function coefficients
c = []
for i in range(m):
    for j in range(n):
        for k in range(p):
            c.append(cost[i][j][k])
```

For A_ub and b_ub which are our inequality constraints coefficients we have:

```python
# Inequality constraints (supply constraints)
A_ub = []
b_ub = []
for i in range(m):
    for k in range(p):
        zero = np.zeros((m, n, p))
        for j in range(n):
            zero[i][j][k] = 1
        zero = zero.reshape(-1).tolist()
        A_ub.append(zero)
        b_ub.append(supply[i][k])
```

For A_eq and b_eq which are our equality constraints coefficients we have:

```python
# Equality constraints:
A_eq = []
b_eq = []
for j in range(n):
    for k in range(p):
        zero = np.zeros((m, n, p))
        for i in range(m):
            zero[i][j][k] = 1
        zero = zero.reshape(-1).tolist()
        A_eq.append(zero)
```

Now that we organized our data for the function we can use them:

```python
solver = linprog(c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds=(0, 100), method='highs')
```

The program outputs the optimal cost of $95.0 and optimal values [10. 5. 0. 10. 5. 5. 15. 0.] for our decision variables.

About this project

This project is done in purpose of better understanding the course I took on Linear Optimization and also getting familiar with writing and getting a hands on experience in solving these kind of problems using Python.

The code for this project which is a python program with UI for solving these kind of transportation problems can be found here.