Assignment 9

Transportation & Warehouse Location Problem

Cinoy Cyriac

Harikrishnan Sabu

Vishakh Cheruparambath

Overview

- Theoretical Background
- Comparison between transportation and assignment problem
- Mathematical model
- Business Application Examples
- Implementation in CPLEX
- Comparison between transportation and warehouse problem
- References

The Transportation Problem

Theoretical Background

- Sending quantities of identical goods from n different origins, such as factories, to m different destinations, such as storage/warehouse/customer.

- Goal: minimize the transportation cost

Assignment problem

Transportation Problem	Assignment Problem
• Minimizing cost of transportation merchandise	• Assigning finite sources to finite destinations where only one destination is allotted for one source with minimum cost
• Number of sources and number of demand need not be equal	• Number of sources and the number of destinations must be equal
• If total demand and total supply are not equal then the problem is said to be unbalanced	• If the number of rows are not equal to the number of columns then problems are unbalanced

Mathematical model(General Solution)

Index set

m number of sources (i = 1…m)

n number of destinations (j = 1…n)

Parameters

s_i supply at source 'i'

d_j demand at destination 'j'

Decision variable

c_{ij} unit cost of shipping from source 'i' to destination 'j'

x_{ij} amount shipped from source 'i' to destination 'j

Mathematical model(General Solution)

Minimize $$\sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij} * x_{ij}$$

Constraints

$$\sum_{j=1}^{n} x_{ij} \leq s_i, \forall\, i = 1 \ldots m$$

$$\sum_{i=1}^{m} x_{ij} = d_j, \forall\, j = 1 \ldots n$$

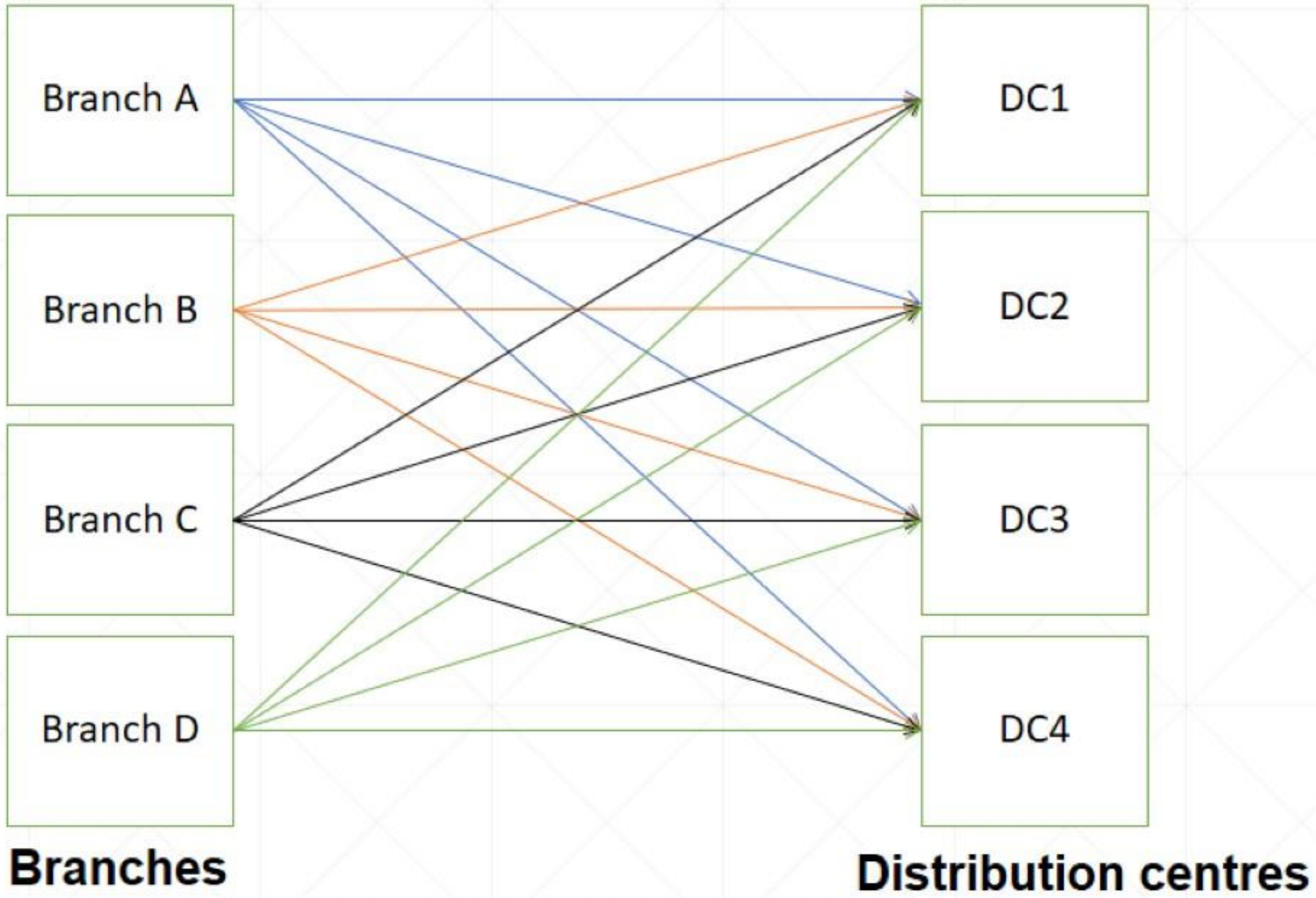
$$x_{ij} \geq 0, \forall\, i \,\&j$$

Business Application Example

- The origins are 4 branches plants with supplies or capacities of **35,50,80,**and **65** units.
- While the distribution are four distributions centres with demand of **70,30,75** and **55** respectively.
- The cost of transporting a unit of product from each branches to each distributions are given.
- Optimize this problem to minimize the transportation cost.

	DC1	DC2	DC3	DC3	**Supply**
Branch A	10	7	6	4	35
Branch B	8	8	5	7	50
Branch C	4	3	6	9	80
Branch D	7	5	4	3	65
Demand	70	30	75	55	

Business Application Example



Implementation in CPLEX

Index set

$i \in I$ Set of branches

$j \in J$ Set of warehouses

model file(.mod)

```
//Index
{int} branches = ...; //set of branches
{int} warehouses= ...; //set of warehouses
```

data file(.dat)

```
//Index
branches={1,2,3,4};
warehouses= {1,2,3,4};
```

Implementation in CPLEX

Parameters

demand	total demand at warehouses 'j'
supply	total supply at branches 'i'
tcost	total transportation cost from branches(i) to warehouses(j)

model file(.mod)

```
//Parameters
int demand[warehouses]= ...; //demand of products
int supply[branches]= ...;   //suppy of products
int tcost[branches][warehouses]=...; //total transportation cost

```

data file(.dat)

```
//Parameters
demand=[70,30,75,55];
supply=[35,50,80,65];
tcost=[[10,7,6,4],[8,8,5,7],[4,3,6,9],[7,5,4,3]];

```

Implementation in CPLEX

Decision variables

amttransp amount shipped from branches 'i' to warehouses 'j'

model file(.mod)

```
//Decision variables
dvar float+ amounttransp[branches][warehouses];
```

Objective function

Minimize

$$\sum_{i=1}^{I}\sum_{j=1}^{J} amounttransp_{ij} * tcost_{ij}$$

```
//Decision expression
dexpr float transpcost = sum( i in branches, j in warehouses)
 amounttransp[i][j]*tcost[i][j];

//Objective fun
minimize transpcost;
```

Implementation in CPLEX

Constraints

$$\sum_{j=1}^{J} amounttransp_{ij} \leq supply_i, \forall\, i = 1 \dots I$$

$$\sum_{i=1}^{I} amounttransp_{ij} = demand_j, \forall\, j = 1 \dots J$$

$$amounttransp_{ij} \geq 0, \forall\, i \in I \;\&\; j \in J$$

model file(.mod)

```
//Constraints
subject to {
  supplycons:
  forall( i in branches)
    sum( j in warehouses) amounttransp[i][j]<=supply[i];
    demandcons:
  forall(j in warehouses)
    sum(i in branches) amounttransp[i][j]==demand[j];
  forall(i in branches, j in warehouses)
    amounttransp[i][j] >= 0;
}
```

Implementation in CPLEX

Whole script - model file(.mod)

```
//Index
{int} branches = ...; //set of branches
{int} warehouses= ...; //set of warehouses

//Parameters
int demand[warehouses]= ...; //demand of products
int supply[branches]= ...;   //suppy of products
int tcost[branches][warehouses]=...; //total transportation cost

//Decision variables
dvar float+ amounttransp[branches][warehouses];

//Decision expression
dexpr float transpcost = sum( i in branches, j in warehouses)
 amounttransp[i][j]*tcost[i][j];

//Objective fun
minimize transpcost;

//Constraints
 subject to {
   supplycons:
   forall( i in branches)
     sum( j in warehouses) amounttransp[i][j]<=supply[i];
     demandcons:
   forall(j in warehouses)
     sum(i in branches) amounttransp[i][j]==demand[j];
   forall(i in branches, j in warehouses)
     amounttransp[i][j] >= 0;
 }
```

Implementation in CPLEX

Whole script - data file(.dat)

```
//Index
branches={1,2,3,4};
warehouses= {1,2,3,4};

//Parameters
demand=[70,30,75,55];
supply=[35,50,80,65];
tcost=[[10,7,6,4],[8,8,5,7],[4,3,6,9],[7,5,4,3]];

```

Implementation in CPLEX

Results

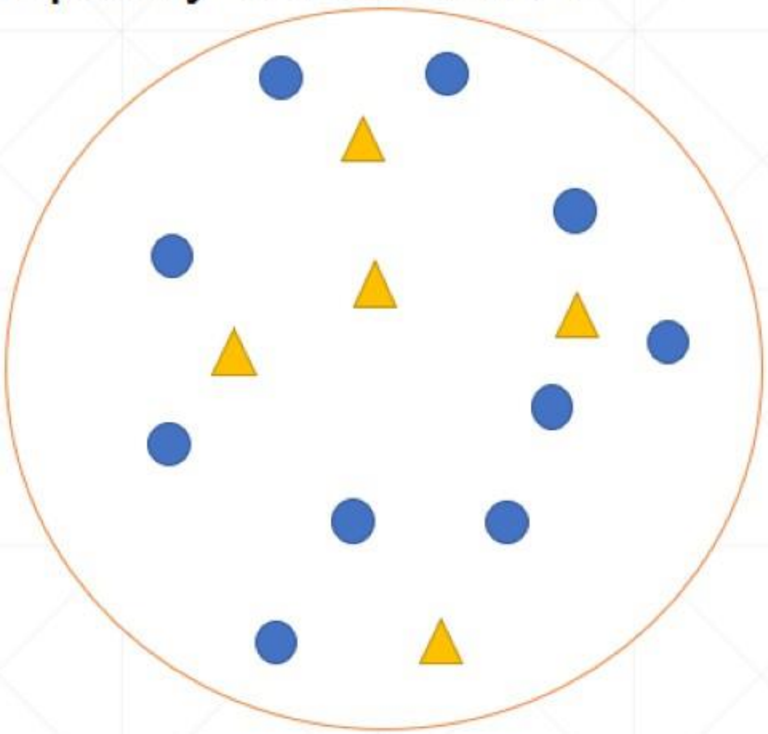
```
// solution (optimal) with objective 960
// Quality There are no bound infeasibilities.
// There are no reduced-cost infeasibilities.
// Maximum Ax-b residual              = 0
// Maximum c-B'pi residual            = 0
// Maximum |x|                        = 70
// Maximum |slack|                    = 70
// Maximum |pi|                       = 7
// Maximum |red-cost|                 = 8
// Condition number of unscaled basis = 1,7e+01
//

amounttransp = [[0
             0 0 35]
             [0 0 50 0]
             [70 10 0 0]
             [0 20 25 20]];
```

The Warehouse Problem

Theoretical Background

- How do we allocate each stores into each warehouses according with capacity constraints ?



Mathematical model(General Solution)

Index set

w ∈ W	set of warehouses
s ∈ S	set of stores

Parameters

F	fixed cost for opening a warehouse
C	maximum number of stores assigned to each warehouses
S	supply cost between each stores and each warehouses

Decision variable

Open	1, if warehouses is open. 0, otherwise
Supply	1, if store supplied by warehouses. 0, otherwise

Mathematical model(General Solution)

Minimize $$\sum_{w=1}^{W} F * O_w + \sum_{w=1}^{W}\sum_{s=1}^{S} C_{sw} * S_{sw}$$ (fixed cost + supply cost)

Constraints

$$\sum_{w=1}^{W} S_{sw} = 1, \ \forall S$$ (Each store has one warehouses)

$$S_{sw} \leq \ O_w \, , \forall \, s, w$$ (If particular warehouses is open, we can supply from that warehouse)

$$\sum_{s=1}^{S} S_{sw} \leq \ C_w$$ Capacity constraints

Business Application Example

Number of stores = 10

Warehouses ="Bonn","Bordeaux","London","Paris","Rome"

Fixed cost = 3000 Euro

Capacity of each warehouses = 1,4,2,1,3

Supply cost from each warehouses 'W' to stores 'S'

```
[20,24,11,225,30],
[28,27,82,83,74],
[74,97,71,96,70],
[2,55,73,69,61],
[46,96,59,83,4],
[42,96,59,83,4],
[1,5,73,59,56],
[10,73,13,43,96],
[93,35,63,85,46],
[47,65,55,71,95]
```

Implementation in CPLEX

Index set

$w \in W$	Set of warehouses
$s \in S$	Set of stores

model file(.mod)

```
//Index
{string} Warehouses =...;
int NbStores =...;
range Stores =0..NbStores-1;
```

data file(.dat)

```
//Index
Warehouses ={"Bonn","Bordeaux","London","Paris","Rome"};
NbStores= 10;
```

Implementation in CPLEX

Parameters

FixedCost fixed cost for opening a warehouse
capacity maximum number of stores assigned to each warehouses
SupplyCost supply cost between each stores and each warehouses

model file(.mod)

```
//Parameters
int FixedCost =...;
int capacity [Warehouses]=...;
int SupplyCost[Stores][Warehouses]=...;
```

data file(.dat)

```
//Parameters
FixedCost = 3000;
capacity=[1,4,2,1,3];
SupplyCost=[[20,24,11,225,30],
            [28,27,82,83,74],
            [74,97,71,96,70],
            [2,55,73,69,61],
            [46,96,59,83,4],
            [42,96,59,83,4],
            [1,5,73,59,56],
            [10,73,13,43,96],
            [93,35,63,85,46],
            [47,65,55,71,95]];
```

Implementation in CPLEX

Decision variables

Open 1, if warehouses is open. 0, otherwise

Supply 1, if store supplied by warehouses. 0, otherwise

model file(.mod)

```
//Decision variables
dvar boolean Open[Warehouses];
dvar boolean Supply[ Stores][Warehouses];

```

Objective function

model file(.mod)

Minimize

$$\sum_{w=1}^{W} FixedCost * Open_w + \sum_{w=1}^{W}\sum_{s=1}^{S} SupplyCost_{sw} * Supply_{sw}$$

```
// Objective function
minimize  sum( w in Warehouses) FixedCost*  Open [w] +
          sum(w  in Warehouses, s  in Stores) SupplyCost[s][w]* Supply[s][w];

```

Implementation in CPLEX

Constraints

$$\sum_{w=1}^{W} Supply_{sw} = 1, \ \forall S$$

$$Supply_{sw} \leq \ Open_w \,, \forall\, s, w$$

$$\sum_{s=1}^{S} Supply_{sw} \leq \ \ capacity_w$$

model file(.mod)

```
//Constraints
subject to {

  forall( s in  Stores)
    cteachStoresHasoneWarehouses:
    sum( w in Warehouses) Supply[s][w]==1;

  forall( w  in Warehouses, s in Stores)
    ctUseopenWarehouses:
    Supply[s][w]<=Open[w];

  forall( w in Warehouses)
    ctMaxUseOfwarehouses:
    sum(s in Stores) Supply[s][w]<=capacity[w];
}
```

Implementation in CPLEX

Whole script - model file(.mod)

```
//Index
{string} Warehouses =...;
int NbStores =...;
range Stores =0..NbStores-1;

//Parameters
int FixedCost =...;
int capacity [Warehouses]=...;
int SupplyCost[Stores][Warehouses]=...;


//Decision variables
dvar boolean Open[Warehouses];
dvar boolean Supply[ Stores][Warehouses];

//Objective function
minimize  sum( w in Warehouses) FixedCost*  Open [w] +
          sum(w  in Warehouses, s  in Stores) SupplyCost[s][w]* Supply[s][w];

//Constraints
subject to {

  forall( s in  Stores)
    cteachStoresHasoneWarehouses:
    sum( w in Warehouses) Supply[s][w]==1;

  forall( w  in Warehouses, s in Stores)
    ctUseopenWarehouses:
    Supply[s][w]<=Open[w];

  forall( w in Warehouses)
    ctMaxUseOfwarehouses:
    sum(s in Stores) Supply[s][w]<=capacity[w];
}
```

Implementation in CPLEX

Whole script - data file(.dat)

```
//Index
Warehouses ={"Bonn","Bordeaux","London","Paris","Rome"};
NbStores= 10;


//Parameters
FixedCost = 3000;
capacity=[1,4,2,1,3];
SupplyCost=[[20,24,11,225,30],
            [28,27,82,83,74],
            [74,97,71,96,70],
            [2,55,73,69,61],
            [46,96,59,83,4],
            [42,96,59,83,4],
            [1,5,73,59,56],
            [10,73,13,43,96],
            [93,35,63,85,46],
            [47,65,55,71,95]];

```

Implementation in CPLEX

Results

```
// solution (optimal) with objective 12236
// Quality Incumbent solution:
// MILP objective                                 1,2236000000e+04
// MILP solution norm |x| (Total, Max)            1,40000e+01  1,00000e+00
// MILP solution error (Ax=b) (Total, Max)        0,00000e+00  0,00000e+00
// MILP x bound error (Total, Max)                0,00000e+00  0,00000e+00
// MILP x integrality error (Total, Max)          0,00000e+00  0,00000e+00
// MILP slack bound error (Total, Max)            0,00000e+00  0,00000e+00
//

Open = [1
         1 1 0 1];
Supply = [[0 0 1 0 0]
             [0 1 0 0 0]
             [0 0 0 0 1]
             [1 0 0 0 0]
             [0 0 0 0 1]
             [0 0 0 0 1]
             [0 1 0 0 0]
             [0 0 1 0 0]
             [0 1 0 0 0]
             [0 1 0 0 0]];
```

Why transportation problem is easier to solve than the warehouse location problem?

- Number of data required
- Complexity

References

- Business optimisation: Using mathematical programming, Macmillan, Basingstoke.: Chapter 4.3.1 & 7.3
- Produktion und Logistik, Springer, Berlin.: Chapter 11.1 & 4.3.3 Kallrath, J. & Wilson, J. M. (1997)
- https://www.sarthaks.com/881661/what-is-the-difference-between-assignment-problem-and-transportation-problem

Thanks for your attention !