EE5327 Optimization

Presentation 2

Sachin Goyal (EE18MTECH11015),
Subhra Shankha Bhattacherjee (EE19MTECH01008)

February 28, 2019

Question:

38) The following table gives the cost matrix of a transportation problem

4	5	6
3	2	2
1	1	2

The basic feasible solution given by $x_{11} = 3$, $x_{13} = 1$, $x_{23} = 6$, $x_{31} = 2$, $x_{32} = 5$ is

1. degenerate and optimal
2. optimal but not degenerate
3. degenerate but not optimal
4. neither degenerate nor optimal

Theoretical Solution:

Check for degeneracy:

To check whether the solution is non-degenerate,

m+n-1 = number of allocated cells,

where m and n are the dimension of the cost matrix.

Given cost matrix is of dimension 3X3.

We calculate $m + n - 1 = 3 + 3 - 1 = 5$, which is equal to the number of allocated cells i.e. 5.

So, we conclude the problem is non degenerate.

Check for optimality:

We use UV or MODI method to check for optimality.
Arrange the cost matrix as follows:

	v_1	v_2	v_3
u_1	4	5	6
u_2	3	2	2
u_3	1	1	2

Table: 1

and $x_{11}=3$, $x_{13}=1$, $x_{23}=6$, $x_{31}=2$, $x_{32}=5$, x_{12}, x_{21}, x_{22}, $x_{33}=0$.

To find the u_i and v_j

For allocated cells, we find the u_i and v_j using the formula

$$u_i + v_j = C_{ij} \tag{1}$$

Where C_{ij} are the coefficient of the cost matrix.
Assuming $u_1 = 0$, we find all the values of u and v, the cost matrix table becomes:

	v_1=4	v_2=4	v_3=6
u_1=0	4	5	6
u_2=-4	3	2	2
u_3=-3	1	1	2

Table: 2

Computing penalties

Computing the penalties for the unallocated cells using the following equation.

$$P_{ij} = u_i + v_j - C_{ij} \tag{2}$$

Penalties are $P_{12} = -1, P_{13} = -3, P_{22} = -1, P_{33} = 1,$

For the feasable solution to be optimal the penalties should be zero or negative. But here we see P_{33} is positive. So the solution is not optimal.

Optimal Solution

We select the unallocated cell with positive penalty and draw a loop in the table starting from that cell through allocated cells.

The loop path: $C_{33} \to C_{13} \to C_{11} \to C_{31}$

We alternately assign (+) and (-) signs to the cells in the loop starting from the first loop being assigned (+).

We select the minimum value of the two negatively assigned cells (in our case $C_{31} = 1$) and subtract this value from the (-) assigned cells and add to the (+) assigned cells.

Optimal Solution

Thus, the new table with new allocated values is:

	v_1	v_2	v_3
u_1	4	5	6
u_2	3	2	2
u_3	1	1	2

Table: 1

with $x_{11} = 4$, $x_{23} = 6$, $x_{31} = 1$, $x_{32} = 5$ and $x_{33} = 1$.

We repeat the above prcoess for finding new u_i and v_j

Optimal Solution

Again assuming $u_1 = 0$, we find all the values using equation (1).

	v_1=4	v_2=4	v_3=5
u_1=0	4	5	6
u_2=-3	3	2	2
u_3=-3	1	1	2

Table: 2

Optimal Solution Found

We find the penalty using equation (2).

$P_{12} = -1, P_{13} = -1, P_{21} = -2, P_{22} = -1,$

Thus, we see all the penalties are negative, which satisfies the condition for optimality.

Thus, the optimal values are

$x_{11} = 4$, $x_{23} = 6$, $x_{31} = 1$, $x_{32} = 5$ and $x_{33} = 1$.

Putting these values in the objective funtion
$4x_{11} + 5x_{12} + 6x_{13} + 3x_{21} + 2x_{22} + 2x_{23} + x_{31} + x_{32} + 2x_{33}$
yields 36.

Conversion to LP

The cost matrix with demand and supply is:

	D_1	D_2	D_3	
S_1	4	5	6	4
S_2	3	2	2	6
S_3	1	1	2	7
	5	5	7	

Table: 3

LP Form

We convert the given problem (Table 3) to LP.

Minimize

$4x_{11} + 5x_{12} + 6x_{13} + 3x_{21} + 2x_{22} + 2x_{23} + x_{31} + x_{32} + 2x_{33}$
s.t.

$$x_{11} + x_{12} + x_{13} \leq 4 \tag{3}$$
$$x_{21} + x_{22} + x_{23} \leq 6 \tag{4}$$
$$x_{31} + x_{32} + x_{33} \leq 7 \tag{5}$$
$$-x_{11} - x_{21} - x_{31} \leq -5 \tag{6}$$
$$-x_{12} - x_{22} - x_{32} \leq -5 \tag{7}$$
$$-x_{31} - x_{23} - x_{33} \leq -7 \tag{8}$$
$$-x_{11}, \ldots, -x_{33} \leq 0 \tag{9}$$

Code using CVXPY

```
from cvxpy import *
from numpy import matrix

A = matrix([

[1,1,1,0,0,0,0,0,0],
[0,0,0,1,1,1,0,0,0],
[0,0,0,0,0,0,1,1,1],
[-1,0,0,-1,0,0,-1,0,0],
[0,-1,0,0,-1,0,0,-1,0],
[0,0,-1,0,0,-1,0,0,-1],
[-1,0,0,0,0,0,0,0,0],
[0,-1,0,0,0,0,0,0,0],
[0,0,-1,0,0,0,0,0,0],
[0,0,0,-1,0,0,0,0,0],
[0,0,0,0,-1,0,0,0,0],
[0,0,0,0,0,-1,0,0,0],
[0,0,0,0,0,0,-1,0,0],
[0,0,0,0,0,0,0,-1,0],
[0,0,0,0,0,0,0,0,-1]])

b = matrix([ 4,6,7,-5,-5,-7,0,0,0,0,0,0,0,0,0])

c=matrix([4.0 , 5.0 , 6.0 , 3.0 ,
2.0 , 2.0 , 1.0 , 1.0 , 2.0])

x = Variable(9,1)
#Cost function
f = c*x
obj = Minimize(f)
#Constraints
constraints = [A*x <= b.transpose()]

#solution
Problem(obj, constraints).solve()
```

Result

```
f = 36.00

x1 = 4.00
x2 = 0.00
x3 = 0.00
x4 = 0.00
x5 = 0.00
x6 = 6.00
x7 = 1.00
x8 = 5.00
x9 = 1.00
```

Thus, we see these values matches our theoretical solution.