A study of North east corner method and use of Object Oriented Programming model

R. Palaniyappa¹ and V. Vinoba²

¹Department of Mathematics, IRT Polytechnic College, Chromepet, Chennai – 44, India.
²Department of Mathematics, K. N. G. C. (W), Thanjavur – 7, India.

ABSTRACT

In this paper, the North east corner [NEM] procedure is successfully coded and tested via many randomly generated problem instances. Based on the results we can conclude that the correctness of the newly coded NEM is promising as compared with the previously coded one.

We assumed that the total amount shipped is equal to the total amount received, that is,

\[ \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \]

B. Transportation problem

\[ \text{Min } \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \]

Subject to \( \sum_{j=1}^{n} x_{ij} \leq a_i , i = 1, 2, \ldots, m \)

\( \sum_{i=1}^{m} x_{ij} \leq b_j , j = 1, 2, \ldots, n \), where \( x_{ij} \geq 0 \forall i,j \).

Feasible solution: A set of non negative values \( x_{ij} \), \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \) that satisfies the constraints is called a feasible solution to the transportation problem.

Optimal solution: A feasible solution is said to be optimal if it minimizes the total transportation cost.

Non degenerate basic feasible solution: A basic feasible solution to a \( (m \times n) \) transportation problem that contains exactly \( m + n - 1 \) allocations in independent positions.

Degenerate basic feasible solution: A basic feasible solution that contains less that \( m + n - 1 \) non negative allocations.

Balanced and Unbalanced Transportation problem: A transportation problem is said to be balanced if the total supply from all sources equals the total demand in the destinations and is called unbalanced otherwise.

North East Corner Rule

Procedure:

North East Corner Method:

1. The method starts at the North – East corner cell (route) of the tableau (Variable X1n).
2. Allocate as much as possible to the selected cell and adjust the associated amounts of supply and demand by subtracting the allocated amount.
3. If both a row and a column net to zero simultaneously cross out one only and leave a zero supply (demand in the uncrossed out row or column).
If exactly one row or column is left uncrossed out or below if exactly one row or column is left uncrossed out, stop. Otherwise, move to the cell to the right if a column has just been crossed out or below if a row has been crossed out. Go to step (i).

Start with X1n and end must be Xm1.

Object oriented program code for North east corner method for initial basic feasible solution:

```
#include<stdio.h>
#include<conio.h>

void main()
{
    int sn,dn,i,j,ss=0,ds=0,sum=0;
    int sup[10],dem[10];
    int a[10][10],c[10][10];
    clrscr();
    //getting no.of supply & demand.
    printf("Enter the num of supply:");
    scanf("%d",&sn);
    printf("Enter the num of demand:");
    scanf("%d",&dn);
    //clearing values in array.
    for(i=0,j=0;i<10,j<10;i++,j++)
    {
        sup[i]=0;
        dem[j]=0;
    }
    //input supply values & calculate sum of supply.
    for(i=0;i<sn;i++)
    {
        printf(" Enter supply value sup[%d]:",i);
        scanf("%d",&sup[i]);
        ss=ss+sup[i];
    }
    //input demand values & calculate sum of demand.
    for(j=0;j<dn;j++)
    {
        printf(" Enter demand value dem[%d]:",j);
        scanf("%d",&dem[j]);
        ds=ds+dem[j];
    }
    if(ss!=ds)
    {
        printf(" unbalanced problem..");
        getch();
        exit(0);
    }
    //input transportation cost.
    printf(" Enter array values:");
    for(i=0;i<sn;i++)
    {
        for(j=0;j<dn;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    //clearing cost array.
    for(i=0;i<10;i++)
    {
        for(j=0;j<10;j++)
        {
            c[i][j]=0;
        }
    }
    //calculation.
    for(i=0,j=dn-1;i<sn,j>=0;)
    {
        if(i==sn||j==-1)
            goto L1;
        if(dem[j]==sup[i])  //Checking demand=supply
            {
                c[i][j]=dem[j]*a[i][j];
                printf("c[%d][%d]=%d and sup=%d",i,j,c[i][j],sup[i]);
                j--;
                i++;
            }
        else if(dem[j]<sup[i])  //Checking demand< supply
            {
                c[i][j]=dem[j]*a[i][j];
                sup[i]=sup[i]-dem[j];
                printf("c[%d][%d]=%d and sup=%d",i,j,c[i][j],sup[i]);
                dem[j]=0;
                j--;
            }
        else   //demand >= supply
            {
                c[i][j]=sup[i]*a[i][j];
                dem[j]=dem[j]-sup[i];
                printf("c[%d][%d]=%d and dem=%d",i,j,c[i][j],dem[j]);
                i++;
            }
    }
    L1:
    for(i=0;i<sn;i++)
    {
        for(j=0;j<dn;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    //clearing cost array.
    for(i=0;i<10;i++)
    {
        for(j=0;j<10;j++)
        {
            c[i][j]=0;
        }
    }
    //calculation.
    for(i=0,j=dn-1;i<sn,j>=0;)
    {
        if(i==sn||j==-1)
            goto L1;
        if(dem[j]==sup[i])  //Checking demand=supply
            {
                c[i][j]=dem[j]*a[i][j];
                printf("c[%d][%d]=%d and sup=%d",i,j,c[i][j],sup[i]);
                j--;
                i++;
            }
        else if(dem[j]<sup[i])  //Checking demand< supply
            {
                c[i][j]=dem[j]*a[i][j];
                sup[i]=sup[i]-dem[j];
                printf("c[%d][%d]=%d and sup=%d",i,j,c[i][j],sup[i]);
                dem[j]=0;
                j--;
            }
        else   //demand >= supply
            {
                c[i][j]=sup[i]*a[i][j];
                dem[j]=dem[j]-sup[i];
                printf("c[%d][%d]=%d and dem=%d",i,j,c[i][j],dem[j]);
                i++;
            }
    }
    printf(" sum of transportation cost = %d",sum);
    getch();
}
```

Example:

Solve the transportation problem

<table>
<thead>
<tr>
<th>5</th>
<th>8</th>
<th>6</th>
<th>6</th>
<th>3</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>900</td>
</tr>
</tbody>
</table>

Demand 400 400 50000 400 800

Proof:

The given Problem unbalanced transportation problem.
Because demand # supply i.e. 2,200 # 2,500 (2500-2200=300) so we add a dummy row all the entries are zero and put the supply is 300.

<table>
<thead>
<tr>
<th>5</th>
<th>8</th>
<th>6</th>
<th>6</th>
<th>3</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>100</td>
<td>7</td>
<td>5</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>900</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>300</td>
</tr>
</tbody>
</table>

| 400 | 400 | 500 | 400 | 800 |

By using north East Method

<table>
<thead>
<tr>
<th>5</th>
<th>8</th>
<th>6</th>
<th>6</th>
<th>3</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>100</td>
<td>7</td>
<td>5</td>
<td>500</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>900</td>
</tr>
<tr>
<td>300</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>300</td>
</tr>
</tbody>
</table>

| 400/300| 400 | 500/400 | 400 | 800 |
The initial basic feasible solution is given as:
\[X_{15} = 800, \ X_{23} = 100, \ X_{24} = 400, \ X_{31} = 100, \ X_{32} = 400, \ X_{33} = 400, \ X_{41} = 300\]

The objective function is:
\[\text{Min } Z = 800x_3 + 100x_7 + 400x_6 + 100x_8 + 400x_4 + 400x_6 + 300x_0\]
\[= \$ 10,300\]

Output:
Enter the num of supply: 4
Enter the num of demand: 5
Enter supply value sup[0]: 800
Enter supply value sup[1]: 500
Enter supply value sup[2]: 900
Enter supply value sup[3]: 300
Enter demand value dem[0]: 400
Enter demand value dem[1]: 400
Enter demand value dem[2]: 500
Enter demand value dem[3]: 400
Enter demand value dem[4]: 800
Enter array values: 5 8 6 6 3
4 7 7 6 5
8 4 6 6 4
0 0 0 0 0
c[0][4]=2400 and sup=800
c[1][3]=2400 and sup=100
c[1][2]=700 and dem=400
c[2][2]=2400 and sup=500
c[2][1]=1600 and sup=100
c[2][0]=800 and dem=300
c[3][0]=0 and sup=300
sum of transportation cost = 10300

Conclusion
The optimal solution obtained in this present investigation shows much more closeness with initial basic feasible solution obtained by North east corner rule. The comparison of optimal solution have been made with other methods of finding initial solutions and observe that North east corner method give the better initial feasible solutions which are closer to optimal solution. The object oriented programs using C++ have been developed. This shows that the computed results tally with the results obtained C++ programming. Object oriented program code for said programs is given for better understanding.

References