International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Solution of Multi – Objective Transportation Problem by New Row Maxima Method and Parallel Method Using C – Programme

Malati Yeola[1], Vinayak Jadhav[2]

[1]Assistant Professor, Department of Statistics, Ahmednagar College, Ahmednagar, M.S., India

[2]Head, Department of Statistics, Computer Science & IT, N.E.S. Science College, Nanded, M.S., India

Abstract: ***In this paper, we have developed a C – Programme to get the solution of Multi-objective Transportation Problem using New Row Matrix Maxima Method and Parallel method. This programme is developed using simple C – statements. At the end by running programme for an example output is obtained and checked whether it matches with solution obtained manually.***

Keywords: Multi-objective transportation problem (MOTP), New Row Maxima Method, Parallel Method, Fuzzy linear membership function.

1. Introduction

The basic transportation problem was originally developed by F. L. Hitchcock (1941) in his study entitled "The Distribution of a Product from Several Sources to Numerous Locations ".In 1947 , T.C. Koopmans independently published a study on "Optimum utilization of the transportationsystem."Transportation models deals with the transportation of a product manufactured at different sources to a number destinations. The objective is to satisfy the destination requirements within the sources capacity constraints at the minimum transportation cost. Solution of the transportation models requires the determination of how many units should be transported from each supply source to each demand destination in order to satisfy all destination demands and source availabilities while minimising the total transportation cost associated with it.

As there are so many methods to obtain initial basic feasible solution of single objective transportation problem like north-west corner rule, least cost method, Vogel's approximation method. Also to obtain optimal solution there are different methods like stepping stone method, modified distribution method.Always we are not having a single objective transportation problem. So we have to consider a multi – objective transportation problem(MOTP) because many factors are affecting on the cost of transportation.

Different softwares like TORA, Lpsolve , LINGO are developed to get the initial basic feasible solution and optimum solution of single objective transportation problem,but for MOTP no such software is available to give solution by using different methods. So we developed a C-programme for two methods (New row matrix maxima method and parallel method) which gives the solution of MOTP by entering the cost matrix.

Definition: Here linear membership function is used and defined as:

$$\mu_k(x_{ij}^k) = \begin{cases} 1 & x_{ij}^k \le L_k \\ \dfrac{U_k - x_{ij}^k}{U_k - L_k} & L_k \le x_{ij}^k \le U_k \\ 0 & x_{ij}^k \ge U_k \end{cases} \quad --(1)$$

Where$L_k \neq U_k$, k= 1,2,.....p. If $L_k = U_k$ then membership value is 1 for any value of k.

Stepwise procedure to run the C – Programme

Step 1: i) Enter the number of matrices (number of cost matrix).

ii) Enter the matrix one by one (while entering, enter the matrix row wise).

iii) Enter the supply units.

iv) Enter the demand units.

Step 2 : i) It checks the problem is balanced one or not, if not it makes the problem balanced for every cost matrix.

ii) It prints the given cost matrix along with the demand and supply units one by one.

Step 3: i) Finds maximum cost and minimum cost for each cost matrix and then calculates membership values for all cost matrices.

ii) Prints the membership matrix for each cost matrix.

Step 4: i) Calculates the averaged matrix of all the membership cost matrices.

ii) Prints the average matrix.

Step 5: i) The rows of averaged membership matrix are arranged in descending order.

ii) Prints the descending order matrix.

Step 6: i) Making allocations by New Row Maxima method.

ii) Prints the allocations with remaining supply units and required demand units for that particular source and destination respectively.

Step 7: i) Calculates the total cost for all the cost matrices for new row maxima method.

ii) Prints the total cost for all matrices.

Step 8:i)Calculates the matrix of minimum from all the membership matrices.

ii) Prints the minimum matrix obtained above.

Step 9: i) Calculates ties in the minimum matrix.

ii) Prints the ties matrix obtained above.

Step 10: i) The rows of minimum membership matrix are arranged in descending order.
ii) Prints the descending order matrix.
Step 11: i)) Making allocations by Parallel method.
ii) Prints the allocations with remaining supply units and required demand units for that particular source and destination respectively.
Step 12: i) Calculates the total cost for all the cost matrices using Parallel method.
ii) Prints the total cost for all matrices.
The programme for obtaining solution of MOTP by both methods.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int
i,j,k,a[10][10][10],r,c,n,s[10],d[10],s1=0,d1=0,t[10],max[10]
,min[10];
int
x[10][10],coavg[10],comin[10],s2[10],d2[10],tiem[10][10],ti
ea[10][10];
int m,p,q,p1,q1,l,x1[10][10];
float b[10][10][10],a1,a2,avg[10][10],mini[10][10];
floatavgd[10][10],minid[10][10],t1[10][10];
printf("Enter the number of rows and columns for the
matrix\n");
scanf("%d%d",&r,&c);
printf("Enter the no.of matrices");
scanf("%d",&n);
for(k=0;k<n;k++)
 {
printf("\nEnter the %d th matrix\n",k);
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
scanf("%d", &a[i][j][k]);
   }
 }
printf("Enter the supply units");
for(i=0;i<r;i++)
 {scanf("%d",&s[i]);s2[i]=s[i];s1+=s[i];}
printf("Enter the demand units");
for(j=0;j<c;j++)
{ scanf("%d",&d[j]);d2[j]=d[j];d1+=d[j];}
printf("\nSupply units=%d\tDemand units=%d\n",s1,d1);
/* Checking the balanced condition */
if(s1==d1)
printf("Problem is balanced one\n");
else
printf("Problem is unbalanced\n");
if(s1>d1)
  {
for(k=0;k<n;k++)
   {
for(i=0;i<r;i++)
a[i][c][k]=0;
    }
d[c]=s1-d1;c=c+1;
printf("The entered matrix is by adding dummy
destination\n");
  }
else
if(s1<d1)
 {
for(k=0;k<n;k++)
   {
for(j=0;j<c;j++)
a[r][j][k]=0;
    }
s[r]=d1-s1;r=r+1;s1=d1;
printf("The entered matrix is by adding dummy
source\n");
 }
else
printf("The entered matrix is\n");
for(k=0;k<n;k++)
 {
for(j=0;j<c;j++)
printf("\tD%d",j+1);printf("\tSupply");
for(i=0;i<r;i++)
  {
printf("\nS%d\t",i+1);
for(j=0;j<=c;j++)
{ if(j<c)
printf("%d\t", a[i][j][k]);else printf("%d\n",s[i]);
   }
}printf("\nDemand");
for(i=0;i<c;i++)
printf("\t%d",d[i]);printf("\t%d",s1);
printf("\n");getch();
 }
/* CALCULATION OF MEMBERSHIP VALUES FOR
ALL MATRICES*/
for(k=0;k<n;k++)
 {
max[k]=a[0][0][k];
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
if(a[i][j][k]>max[k])
   {
t[k]=a[i][j][k];
max[k]=t[k];
    }
   }
  }
for(k=0;k<n;k++)
 {
min[k]=a[0][0][k];
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
if(a[i][j][k]<min[k])
   {
t[k]=a[i][j][k];
min[k]=t[k];
    }
   }
  }
for(k=0;k<n;k++)
  {
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
  {
```

```
if(a[i][j][k]==min[k])
b[i][j][k]=1;
else
if(a[i][j][k]==max[k])
b[i][j][k]=0;
else

{
     a1=(float)(max[k]-a[i][j][k]);
     a2=max[k]-min[k];
b[i][j][k]=a1/a2;
    }
   }
  }
 }
```

/*Printing the matrices of membership values*/

```
for(k=0;k<n;k++)
 {
printf("\nmax[%d]=%d\tmin[%d]=%d",k,max[k],k,min[k]);
printf("\n %d th matrix by using membership value is\n",k);
for(j=0;j<c;j++)
printf("\tD%d",j);printf("\tSupply");
for(i=0;i<r;i++)
  {
printf("\nS%d\t",i);
for(j=0;j<=c;j++)
{ if(j<c)
printf("%.2f\t", b[i][j][k]);else printf("%d\n",s[i]);
   }
}printf("\nDemand");
for(i=0;i<c;i++)
printf("\t%d",d[i]);printf("\t%d",s1);
printf("\n");getch();
 }
```

/*CALCULATION OF AVERAGE MATRIX*/

```
for(i=0;i<r;i++)
 {
for(j=0;j<c;j++)
avg[i][j]=0.0;
 }
for(i=0;i<r;i++)
 {
for(j=0;j<c;j++)
   {
for(k=0;k<n;k++)
avg[i][j]+=b[i][j][k];
avg[i][j]=avg[i][j]/n;
avgd[i][j]=avg[i][j];
   }
 }
printf("\nThe averaged matrix is\n");
for(j=0;j<c;j++)
printf("\tD%d",j);printf("\tSupply");
for(i=0;i<r;i++)
 {
printf("\nS%d\t",i);
for(j=0;j<=c;j++)
 {
if(j<c)
printf("%.2f\t",avg[i][j]);else printf("%d\n",s[i]);
  }
  }
printf("\nDemand");
for(j=0;j<c;j++)
printf("\t%d",d[j]);printf("\t%d",s1);
getch();
```

/* Arranging the average matrix in descending order*/

```
for(i=0;i<r;i++)
  {
for(k=0;k<c-1;k++)
for(j=k+1;j<c;j++)
   {
if(avg[i][k]<avg[i][j])
    {
t1[i][j]=avg[i][k];
avg[i][k]=avg[i][j];
avg[i][j]=t1[i][j];
     }
    }
   }
printf("\nThe matrix in the descending orderis\n");
for(i=0;i<r;i++)
   {
for(j=0;j<c;j++)
printf("%.2f\t",avg[i][j]);
printf("\n");
    } getch();
```

/*MAKING ALLOCATIONS BY ROW MAXIMA METHOD FOR AVERAGED MATRIX[2]*/

```
  i=0;
while(i<r)
  {
   k=0;
while(k<c)
   {
for(j=0;j<c;j++)
if(avgd[i][j]==avg[i][k])
     {
          if(s[i]==d[j])
          {
          x[i][j]=s[i];s[i]=0;d[j]=0;

printf("\tx[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,j,x[i][j]
,i,s[i],j,d[j]);
          goto a;
          }
          else
          if(s[i]>d[j])
           {
          x[i][j]=d[j];s[i]=s[i]-d[j];d[j]=0;

printf("\tx[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,j,x[i][j]
,i,s[i],j,d[j]);
          k++;
            }
          else
            {
          x[i][j]=s[i];d[j]=d[j]-s[i];s[i]=0;

printf("\tx[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,j,x[i][j]
,i,s[i],j,d[j]);
          goto a;
             }}  }
a:++i;
  }
```

```
printf("\n");getch();
/*Calculation of total cost for all the matrices*/
for(k=0;k<n;k++)
coavg[k]=0.0;
for(i=0;i<r;i++)
   {
for(j=0;j<c;j++)
    {
for(k=0;k<n;k++)
coavg[k]+=a[i][j][k]*x[i][j];
     }
   }
for(k=0;k<n;k++)
printf("\n cost of %d th matrix is =%d",k,coavg[k]);
getch();

/*CALCULATION OF MINIMUM AMONGST ALL
THE COST MATRIX*/
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
   {
mini[i][j]=b[i][j][0];
for(k=0;k<n-1;k++)
      {
            if(mini[i][j]>b[i][j][k+1])
            mini[i][j]=b[i][j][k+1];
            }
            minid[i][j]=mini[i][j];
    }  }
printf("\nThe minimised matrix is\n");
for(j=0;j<c;j++)
printf("\tD%d",j);printf("\tSupply");
for(i=0;i<r;i++)
  {
printf("\nS%d\t",i);
for(j=0;j<=c;j++)
  {
if(j<c)
printf("%.2f\t",mini[i][j]);else printf("%d\n",s2[i]);
   }   }
printf("\nDemand");
for(j=0;j<c;j++)
printf("\t%d",d2[j]);printf("\t%d",s1);
getch();

/* Calculation of ties in the minimum matrix*/
for(i=0;i<r;i++)
   {for(j=0;j<c;j++)
   {tiem[i][j]=0;
if(j<c-1)
    {for(k=0;k<c;k++)
     {
            if(j!=k)
            if(mini[i][j]==mini[i][k])
            ++tiem[i][j];
     }    }
else
    {
for(j=c-1,k=j-1;k>=0;k--)
if(mini[i][j]==mini[i][k])
    ++tiem[i][j];
    }    }    }
printf("\n");
   /*printing the tied matrix*/
for(i=0;i<r;i++)
   {
for(j=0;j<c;j++)
printf("Tiem[%d][%d]=%d\t",i,j,tiem[i][j]);
printf("\n");
    }
getch();
/* Arranging the minimum matrix in descending order*/
for(i=0;i<r;i++)
  {
for(k=0;k<c-1;k++)
for(j=k+1;j<c;j++)
    {
if(mini[i][k]<mini[i][j])
     {
t1[i][j]=mini[i][k];
mini[i][k]=mini[i][j];
mini[i][j]=t1[i][j];
      }
     }
    }
printf("\nThe matrix in the descending orderis\n");
for(i=0;i<r;i++)
   {
for(j=0;j<c;j++)
printf("%.2f\t",mini[i][j]);
printf("\n");
    } getch();
/*MAKING ALLOCATIONS BY PARALLEL
METHOD FOR MINIMISED MATRIX [6]*/
  i=0;
while(i<r)
  {
   k=0;
while(k<c)
    {
for(j=0;j<c;j++)
if(minid[i][j]==mini[i][k])
    {
if(tiem[i][j]==0)
     {
          d:if(s2[i]==d2[j])
          {
          x1[i][j]=s2[i];s2[i]=0;d2[j]=0;
          printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t
",i,j,x1[i][j],i,s2[i],j,d2[j]);
          goto b;
          }
          else
          if(s2[i]>d2[j])
           {
           x1[i][j]=d2[j];s2[i]=s2[i]-d2[j];d2[j]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,j,x1[i]
[j],i,s2[i],j,d2[j]);
          k++;
            }
          else
            {
             x1[i][j]=s2[i];d2[j]=d2[j]-s2[i];s2[i]=0;
```

```
printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,j,x1[i][j],i,s2[i],j,d2[j]);
            goto b;
               }
             }
            else
    {
             m=i+1;
            if(m<r)
     {
            if(tiem[i][j]==1)
             {
              p=j;
            for(j=0;j<c;j++)
              {
            if(j!=p)
            if(minid[i][p]==minid[i][j])
              q=j;
               }
             }
            for(l=m;l<r;l++)
            { p1=0;q1=0;
            if(minid[i][p]>=minid[i][q])
            p1++;else q1++;
               }
            if(p1>=q1)
               {
                /*j=q;*/
            if(s2[i]==d2[q])
            {
            x1[i][q]=s2[i];s2[i]=0;d2[q]=0;
            printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            goto b;
            }
            else
            if(s2[i]>d2[q])
             {
             x1[i][q]=d2[q];s2[i]=s2[i]-d2[q];d2[q]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            if(s2[i]>d2[p])
             {
             x1[i][p]=d2[p];s2[i]=s2[i]-d2[p];d2[p]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i][p],i,s2[i],p,d2[p]);
            k++;
              }
            else
            if(s2[i]<d2[p])
              {
              x1[i][p]=s2[i];d2[p]=d2[p]-s2[i];s2[i]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i][p],i,s2[i],p,d2[p]);
            goto b;
                }
            else
              {
               x1[i][p]=s2[i];s2[i]=0;d2[p]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i][p],i,s2[i],p,d2[p]);
            goto b;
               }
            k++;
              }
            else
              {
               x1[i][q]=s2[i];d2[q]=d2[q]-s2[i];s2[i]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            goto b;
               }
               }
            else
               {
               /*j=p; */
            if(s2[i]==d2[p])
            {
            x1[i][p]=s2[i];s2[i]=0;d2[p]=0;
            printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i][p],i,s2[i],p,d2[p]);
            goto b;
            }
            else
            if(s2[i]>d2[p])
             {
             x1[i][p]=d2[p];s2[i]=s2[i]-d2[p];d2[p]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i][p],i,s2[i],p,d2[p]);
            if(s2[i]>d2[q])
             {
             x1[i][q]=d2[q];s2[i]=s2[i]-d2[q];d2[q]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            k++;
             }
            else
            if(s2[i]<d2[q])
             {
              x1[i][q]=s2[i];d2[q]=d2[q]-s2[i];s2[i]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            goto b;
              }
            else
              {
              x1[i][q]=s2[i];s2[i]=0;d2[q]=0;

printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,q,x1[i][q],i,s2[i],q,d2[q]);
            goto b;
               }
            k++;
              }
            else
              {
               x1[i][p]=s2[i];d2[p]=d2[p]-s2[i];s2[i]=0;
```

```
printf("\tx1[%d][%d]=%d\ts[%d]=%d\td[%d]=%d\t",i,p,x1[i
][p],i,s2[i],p,d2[p]);
            goto b;
               }
              }
            }
            elsegoto d;
     }
    }
    }
b:++i;
   }
printf("\n");getch();
/*Calculation of total cost for all the matrices*/
for(k=0;k<n;k++)
comin[k]=0.0;
for(i=0;i<r;i++)
     {
for(j=0;j<c;j++)
printf("x[%d][%d]=%d\t",i,j,x1[i][j]);printf("\n");
      }
for(i=0;i<r;i++)
     {
            for(j=0;j<c;j++)
{  for(k=0;k<n;k++)
            comin[k]+=a[i][j][k]*x1[i][j];
     }
    }
for(k=0;k<n;k++)
printf("\n cost of %d th matrix is =%d",k,comin[k]);
getch();
```

} /* **OUTPUT Ex.1,[4]**
Problem is unbalanced
The entered matrix is by adding dummy source

	D1	D2	D3	D4	Supply
S1	3	4	9	9	8
S2	3	11	5	6	10
S3	10	11	6	8	18
S4	0	0	0	0	4
Demand	11	5	14	10	40

	D1	D2	D3	D4	Supply
S1	6	6	5	6	8
S2	7	10	11	12	10
S3	8	4	7	9	18
S4	0	0	0	0	4
Demand	11	5	14	10	40

max[0]=11 min[0]=0
0 th matrix by using membership value is

	D0	D1	D2	D3	Supply
S0	0.73	0.64	0.18	0.18	8
S1	0.73	0.00	0.55	0.45	10
S2	0.09	0.00	0.45	0.27	18
S3	1.00	1.00	1.00	1.00	4
Demand	11	5	14	10	40

max[1]=12 min[1]=0
1 th matrix by using membership value is

	D0	D1	D2	D3	Supply
S0	0.50	0.50	0.58	0.50	8
S1	0.42	0.17	0.08	0.00	10
S2	0.33	0.67	0.42	0.25	18
S3	1.00	1.00	1.00	1.00	4
Demand	11	5	14	10	40

The averaged matrix is

	D0	D1	D2	D3	Supply
S0	0.61	0.57	0.38	0.34	8
S1	0.57	0.08	0.31	0.23	10
S2	0.21	0.33	0.44	0.26	18
S3	1.00	1.00	1.00	1.00	4
Demand	11	5	14	10	40

The matrix in the descending orderis
0.61 0.57 0.38 0.34
0.57 0.31 0.23 0.08
0.44 0.33 0.26 0.21
1.00 1.00 1.00 1.00
x[0][0]=8 s[0]=0 d[0]=3
x[1][0]=3 s[1]=7 d[0]=0
x[1][2]=7 s[1]=0 d[2]=7
x[2][2]=7 s[2]=11 d[2]=0
x[2][1]=5 s[2]=6 d[1]=0
x[2][3]=6 s[2]=0 d[3]=4
x[3][0]=0 s[3]=4 d[0]=0
x[3][1]=0 s[3]=4 d[1]=0
x[3][2]=0 s[3]=4 d[2]=0
x[3][3]=4 s[3]=0 d[3]=0

cost of 0 th matrix is =213
cost of 1 th matrix is =269

The minimised matrix is

	D0	D1	D2	D3	Supply
S0	0.50	0.50	0.18	0.18	8
S1	0.42	0.00	0.08	0.00	10
S2	0.09	0.00	0.42	0.25	18
S3	1.00	1.00	1.00	1.00	3
Demand	11	5	14	10	40

Tiem[0][0]=1 Tiem[0][1]=1 Tiem[0][2]=1
Tiem[0][3]=1
Tiem[1][0]=0 Tiem[1][1]=1 Tiem[1][2]=0
Tiem[1][3]=1
Tiem[2][0]=0 Tiem[2][1]=0 Tiem[2][2]=0
Tiem[2][3]=0
Tiem[3][0]=3 Tiem[3][1]=3 Tiem[3][2]=3
Tiem[3][3]=3

The matrix in the descending orderis
0.50 0.50 0.18 0.18
0.42 0.08 0.00 0.00
0.42 0.25 0.09 0.00
1.00 1.00 1.00 1.00
x1[0][1]=5 s[0]=3 d[1]=0 x1[0][0]=3 s[0]=0
d[0]=8
x1[1][0]=8 s[1]=2 d[0]=0 x1[1][2]=2 s[1]=0
d[2]=12
x1[2][2]=12 s[2]=6 d[2]=0 x1[2][3]=6 s[2]=0
d[3]=4
x1[3][0]=0 s[3]=0 d[0]=0

x[0][0]=3 x[0][1]=5 x[0][2]=0 x[0][3]=0

x[1][0]=8 x[1][1]=0 x[1][2]=2 x[1][3]=0
x[2][0]=0 x[2][1]=0 x[2][2]=12 x[2][3]=6
x[3][0]=0 x[3][1]=0 x[3][2]=0 x[3][3]=0
cost of 0 th matrix is =183
cost of 1 th matrix is =264*/
/* OUTPUT Ex.2,{[1],[3],[5],[7]}.
Problem is balanced one
The entered matrix is

	D1	D2	D3	D4	D5	Supply
S1	9	12	9	6	9	5
S2	7	3	7	7	5	4
S3	6	5	9	11	3	2
S4	6	8	11	2	2	9
Demand	4	4	6	2	4	20

	D1	D2	D3	D4	D5	Supply
S1	2	9	8	1	4	5
S2	1	9	9	5	2	4
S3	8	1	8	4	5	2
S4	2	8	6	9	8	9
Demand	4	4	6	2	4	20

	D1	D2	D3	D4	D5	Supply
S1	2	4	6	3	6	5
S2	4	8	4	9	2	4
S3	5	3	5	3	6	2
S4	6	9	6	3	1	9
Demand	4	4	6	2	4	20

0 th matrix by using membership value is

	D0	D1	D2	D3	D4	Supply
S0	0.30	0.00	0.30	0.60	0.30	5
S1	0.50	0.90	0.50	0.50	0.70	4
S2	0.60	0.70	0.30	0.10	0.90	2
S3	0.60	0.40	0.10	1.00	1.00	9
Demand	4	4	6	2	4	20

max[1]=9 min[1]=1
1 th matrix by using membership value is

	D0	D1	D2	D3	D4	Supply
S0	0.88	0.00	0.12	1.00	0.62	5
S1	1.00	0.00	0.00	0.50	0.88	4
S2	0.12	1.00	0.12	0.62	0.50	2
S3	0.88	0.12	0.38	0.00	0.12	9
Demand	4	4	6	2	4	20

max[2]=9 min[2]=1
2 th matrix by using membership value is

	D0	D1	D2	D3	D4	Supply
S0	0.88	0.62	0.38	0.75	0.38	5
S1	0.62	0.12	0.62	0.00	0.88	4
S2	0.50	0.75	0.50	0.75	0.38	2
S3	0.38	0.00	0.38	0.75	1.00	9
Demand	4	4	6	2	4	20

The averaged matrix is

	D0	D1	D2	D3	D4	Supply
S0	0.68	0.21	0.27	0.78	0.43	5
S1	0.71	0.34	0.38	0.33	0.82	4
S2	0.41	0.82	0.31	0.49	0.59	2
S3	0.62	0.17	0.28	0.58	0.71	9
Demand	4	4	6	2	4	20

The matrix in the descending orderis
0.78 0.68 0.43 0.27 0.21
0.82 0.71 0.38 0.34 0.33
0.82 0.59 0.49 0.41 0.31
0.71 0.62 0.58 0.28 0.17

x[0][3]=2 s[0]=3 d[3]=0
x[0][0]=3 s[0]=0 d[0]=1
x[1][4]=4 s[1]=0 d[4]=0
x[2][1]=2 s[2]=0 d[1]=2
x[3][4]=0 s[3]=9 d[4]=0
x[3][0]=1 s[3]=8 d[0]=0
x[3][3]=0 s[3]=8 d[3]=0
x[3][2]=6 s[3]=2 d[2]=0
x[3][1]=2 s[3]=0 d[1]=0

cost of 0 th matrix is =157
cost of 1 th matrix is =72
cost of 2 th matrix is =86
The minimised matrix is

	D0	D1	D2	D3	D4	Supply
S0	0.30	0.00	0.12	0.60	0.30	5
S1	0.50	0.00	0.00	0.00	0.70	4
S2	0.12	0.70	0.12	0.10	0.38	2
S3	0.38	0.00	0.10	0.00	0.12	9
Demand	4	4	6	2	4	20

Tiem[0][0]=1 Tiem[0][1]=0 Tiem[0][2]=0
Tiem[0][3]=0 Tiem[0][4]=1
Tiem[1][0]=0 Tiem[1][1]=2 Tiem[1][2]=2
Tiem[1][3]=2 Tiem[1][4]=0
Tiem[2][0]=1 Tiem[2][1]=0 Tiem[2][2]=1
Tiem[2][3]=0 Tiem[2][4]=0
Tiem[3][0]=0 Tiem[3][1]=1 Tiem[3][2]=0
Tiem[3][3]=1 Tiem[3][4]=0
The matrix in the descending orderis
0.60 0.30 0.30 0.12 0.00
0.70 0.50 0.00 0.00 0.00
0.70 0.38 0.12 0.12 0.10
0.38 0.12 0.10 0.00 0.00
x1[0][3]=2 s[0]=3 d[3]=0 x1[0][0]=3 s[0]=0 d[0]=1
x1[1][4]=4 s[1]=0 d[4]=0 x1[2][1]=2 s[2]=0 d[1]=2
x1[3][0]=1 s[3]=8 d[0]=0 x1[3][4]=0 s[3]=8 d[4]=0
x1[3][2]=6 s[3]=2 d[2]=0 x1[3][3]=0 s[3]=2 d[3]=0
x1[3][1]=2 s[3]=0 d[1]=0

x[0][0]=3 x[0][1]=0 x[0][2]=0 x[0][3]=2 x[0][4]=0

x[1][0]=0 x[1][1]=0 x[1][2]=0 x[1][3]=0 x[1][4]=4

x[2][0]=0 x[2][1]=2 x[2][2]=0 x[2][3]=0 x[2][4]=0

x[3][0]=1 x[3][1]=2 x[3][2]=6 x[3][3]=0 x[3][4]=0

cost of 0 th matrix is =157

cost of 1 th matrix is =72
cost of 2 th matrix is =86*/

2. Conclusion

This paper gives the solution of MOTP by New row maxima method as well as Parallel method by running C – programme. If we run this programme once it gives solution of MOTP for both the methods. Statements used are so simple which will guide in every step of the programme. In every step the message will be printed by using that message any one can run this programme and get the required answer by these two methods simultaneously. It saves time of calculating the membership value for all matrices, average membership matrix, and minimum membership matrix. Also makes allocations of number units by not doing any mistake and finding the total optimum cost for all matrices by both the methods.

References

[1] A.A. Mousa, Hamdy M. Geneedy and Adel Y Elmekawy, Efficient Evolutionary Algorithm for solving Multiobjective Transportation Problem, Journal of Natural Sciences and Mathematics, Qassim University,Vol.4 No. 1,PP 77 – 102 (june 2010)
[2] A.J. Khan, D.K. Das, New Row Maxima Method to Solve Multi-objective Transportation Problem under Fuzzy Conditions, (IJCMST), 1(1): 42-46,2012.
[3] K Venkatasubbaiah ,S.G. Acharyulu, K V V Chandra Mouli, Fuzzy Goal Programming Methods for Solving Multi-objective Transportation Problems, Global Journal of Engineering, Vol. 11, issue 3 version 1.0,April 2011.
[4] LohgaonkarM.H.,BajajV.H.andJadhav V.A., Additive fuzzy multiple goal programming model for unbalanced multi-objective transportation problem, International Journal of Machine intelligence, ISSN : 0975 – 2927,Issue 1,2010,pp 29-34.
[5] Sayed A Zaki,Abd Allah A.Mousa,Hamdy M Geneedi,Adel Y. Elmekawy, Efficient Multiobjective Genetic Algorithm for Solving Transportation, Assignment, and Transshipment Problems, Applied Mathematics(Scientific Research), 2012,3,92-99.
[6] Yeola M.C. and Jadhav V.A., Solving Multi-objective Transportation Problem Using Fuzzy Programming Technique – Parallel Method, International Journal of Recent Scientific Research, Vol. 7, Issue 1, pp. 8455-8457, January,2016.
[7] Yousria Abo-Elnaga, Bothina El-Sbky, HanadiZahed, Trust Region Algorithm for Multi-Objective Transportation, Assignment, and Transshipment Problems, Life Science Journal 2012;9(3).