A study of Transportation Problem for an Essential Item of Southern Part of North Eastern Region of India as an OR Model and Use of Object Oriented Programming

Nabendu Sen[1], Tanmoy Som[2], Banashri Sinha[1]

[1]Department of Mathematics, Assam University, Silchar; [2]Department of Applied Mathematics, BHU, Varanasi, India

Summary
In this paper we formulate an OR model from the collected data concerning with the transportation of an essential item, rice, from different suppliers of Silchar to different destinations in Mizoram. In this study an attempt has been made to analyze the optimal solution with basic feasible solutions obtained using different methods [10]. Also programs have been developed using object oriented programming, C++.

Key words:
O.R model, feasible solution, VAM, object oriented programming..

1. Introduction

Mizoram, a part of southern region of North East India, not being well connected from the other parts of the nation as well as the North East region, depends on the market of the adjacent district Silchar of Assam for its essential goods like rice, flour, salt etc. Different suppliers of Silchar regularly supply rice to the different markets of Mizoram. [6-9]. As such the related data has been collected from the concerned suppliers for the purpose of the mathematical formulation.

2. Tables, Figures and Equations

2.1 Tables and Figures

Table 1 shows the distance of different destinations in Mizoram from Silchar district.

Table1

Place	**Distance (Km)**
Kolashib	90
Serchip	300
Aizwal	180
Saiha	450

We use the following code for the destination Kolashib, Serchip, Aizwal, Saiha.

X* for Kolashib, Y* for Serchip, Z* for Aizwal and U* for Saiha.

Transportation cost per quintal of rice effective from 2007 from different suppliers to the different destinations (as mentioned above) is displayed in the table 2.

2.2 Table 2

	X*	Y*	Z*	U*
A*	60	120	75	180
B*	58	100	60	165
C*	62	110	65	170
D*	65	115	80	175
E*	70	135	85	195

N.B. i) A*, B*, C*, D*, E* are the code names of the suppliers. ii) The transportation cost of different suppliers to the same destination varies to some extent due to their own policies.

The next table 3 shows the quantity available with these suppliers for a particular year.

Table 3

Supplier	**Quantity Available (Quintal)**
A*	8000
B*	9200
C*	6250
D*	4900
E*	6100

The next table 4 shows the total demand of the destination X*, Y*, Z*, U* from these suppliers during the year

Table 4

Destination	Demand (Quintals)
X*	5000
Y*	2000
Z*	10000
U*	6000

3. Methods of Obtaining Initial Basic Feasible Solutions:

We apply here the following methods to get the initial basic feasible solutions:

(a) Northwest Corner Method (see [10]).
(b) Vogel Approximation Method (see[10]).
(c) Least Cost Method(see[10])
(d) Row Minima Method(see[10])
(e) Column Minima Method(see[10])

4. Formulation of Model:

In this problem we make a transportation schedule for rice, as being the essential commodity (main food of the people) for the state of Mizoram. Combining the data of the tables 2, 3 and 4, we get the following transportation model to determine an optimal schedule so as to minimize the transportation cost for rice to different markets of Mizoram.

	X*	Y*	Z*	U*	Availability
A*	60	120	75	180	8000
B*	58	100	60	165	9200
C*	62	110	65	170	6250
D*	65	115	80	175	4900
E*	70	135	85	195	6100
Demand	5000	2000	10000	6000	

Here $\Sigma\, a_i = 34450$, $\Sigma\, b_j = 23000$. Since $\Sigma\, a_i \neq \Sigma\, b_j$, we introduce a dummy destination V* with requirement of 11450 units and zero (0) transportation cost, as shown in the next table in the form of balanced transportation problem.

	X*	Y*	Z*	U*	V*	Availability
A*	60	120	75	180	0	8000
B*	58	100	60	165	0	9200
C*	62	110	65	170	0	6250
D*	65	115	80	175	0	4900
E*	70	135	85	195	0	6100
Demand	5000	2000	10000	6000	11450	

Now apply the three different methods for initial basic feasible solution.

Initial BASIC FEASIBLE SOLUTION BY NORTHWEST CORNER METHOD:

After applying this method which leads to the following final table as:

Source/Destination	X*	Y*	Z*	U*	V*
A*	5000 60	2000 120	1000 75	180	0
B*	58	100	60	9000 165	200 0
C*	62	110	65	5800 170	450 0
D*	65	115	80	175	4900 0
E*	70	135	85	195	6100 0

From NWCM Method, we find number of occupied cell is (5+5-1=9) which is exactly same as m+n-1. Therefore we get the initial feasible solution as

x_{11} =5000, x_{12} =2000, x_{13} =1000,
x_{23} =9000, x_{24} =200, x_{34} =5800,
x_{35} =450, x_{45} =4900, x_{55} =6100.

Total T.C is Rs. 21, 74,000/

INITIAL BASIC FEASIBLE SOLUTION BY VOGEL APPROXIMATION METHOD:

After applying this method which leads to the following final table as**:**

Source/Destination	X*	Y*	Z*	U*	V*
A*	$\boxed{5000}$ 60	120	$\boxed{3000}$ 75	180	0
B*	58	$\boxed{2000}$ 100	$\boxed{1200}$ 60	$\boxed{6000}$ 165	0
C*	62	110	$\boxed{5800}$ 65	170	$\boxed{450}$ 0
D*	65	115	80	175	$\boxed{4900}$ 0
E*	70	135	85	195	$\boxed{6100}$ 0

From Vogel Approximation Method, we find number of occupied cell is (5+5-1=9) which is exactly same as m+n-1. Therefore we get the initial feasible solution as

x_{11} =5000, x_{13} =3000, x_{22} =2000,
x_{23} =1200, x_{24} =6000, x_{33} =5800,
x_{35} =450, x_{45} =4900, x_{55} =6100.
Total T.C is Rs 12,73,000/

INITIAL BASIC FEASIBLE SOLUTION BY LEAST COST METHOD:

After applying this method which leads to the following final table as:

Source/Destination	X*	Y*	Z*	U*	V*
A*	$\boxed{5000}$ 60	120	$\boxed{750}$ 75	180	$\boxed{2250}$ 0
B*	58	100	60	165	$\boxed{9200}$ 0
C*	62	110	$\boxed{6250}$ 65	170	0
D*	65	$\boxed{1900}$ 115	$\boxed{3000}$ 80	175	0
E*	70	$\boxed{100}$ 135	85	$\boxed{6000}$ 195	0

From Least cost Method, we find number of occupied cell is (5+5-1=9) which is exactly same as m+n-1. Therefore we get the initial feasible solution as

x_{11} =5000, x_{13} =750, x_{15} =2250,
x_{25} =9200, x_{33} =6250, x_{42} =1900,
x_{43} =3000. X_{52} =100, x_{54} =6000
Total T.C is Rs 24,04,500/

INITIAL BASIC FEASIBLE SOLUTION BY ROW MINIMA METHOD:

After applying this method which leads to the following final table as:

Source/Destination	X*	Y*	Z*	U*	V*
A*	60	120	75	180	$\boxed{8000}$ 0
B*	$\boxed{5000}$ 58	100	$\boxed{750}$ 60	165	$\boxed{3450}$ 0
C*	62	110	$\boxed{6250}$ 65	170	0
D*	65	$\boxed{1900}$ 115	$\boxed{3000}$ 80	175	0
E*	70	$\boxed{100}$ 135	85	$\boxed{6000}$ 195	0

From Row minima methods, we find number of occupied cell is (5+5-1=9) which is exactly same as m+n-1. Therefore we get the initial feasible solution as

x_{15} =5000, x_{21} =5000 x_{23} =750,
x_{25} =3450, x_{33} =6250. X_{42} =1900,
x_{43} =3000 ,X_{52} =100, x_{54} =6000
Total T.C is Rs 23,83,250/

INITIAL BASIC FEASIBLE SOLUTION BY COLUMN MINIMA METHOD:

After applying this method which leads to the following final table as:

Source/Destination	X*	Y*	Z*	U*	V*
A*	60	120	$\boxed{1550}$ 75	$\boxed{1100}$ 180	$\boxed{5350}$ 0
B*	$\boxed{5000}$ 58	100	$\boxed{4200}$ 60	165	0
C*	62	$\boxed{2000}$ 110	$\boxed{4250}$ 65	170	0
D*	65	115	80	$\boxed{4900}$ 175	0
E*	70	$\boxed{100}$ 135	85	195	$\boxed{6100}$ 0

From Row minima methods, we find number of occupied cell is (5+5-1=9) which is exactly same as m+n-1. Therefore we get the initial feasible solution as
x_{13} =1550, x_{14} =1100 x_{15} =750,
x_{21} =5000, x_{23} =4200. X_{32} =2000,
x_{33} =4250 ,X_{44} =4900, x_{55} =6000
Total T.C is Rs22,10,000/

5. OPTIMALITY:

Taking initial basic feasible solution due to Vogel's approximation method ,we now proceed for optimality using MODI method . Here we determine a set of u_i and v_j starting with u_1=0 and using the relation $c_{ij} = u_i + v_j$ for occupied basic cells as shown below.

$c_{11}= u_1 + v_1 \Rightarrow 60 = 0 + v_1 \Rightarrow v_1 = 60$, $c_{13} = u_1 + v_3 \Rightarrow 75 = 0 + v_3 \Rightarrow v_3 = 75$,
$c_{22} = u_2 + v_2 \Rightarrow 100 = u_2 + v_2 \Rightarrow v_2 = 75$, $c_{23} = u_2 + v_3 \Rightarrow 60 = u_2 + 75 \Rightarrow u_2 = -15$,
$c_{24} = u_2 + v_4 \Rightarrow 165 = -15 + v_4 \Rightarrow v_4 =180$, $c_{33} = u_3 + v_3 \Rightarrow 65 = u_3 + 75 \Rightarrow u_3 = -10$,
$c_{35} = u_3 + v_5 \Rightarrow 0 = -10 + v_5 \Rightarrow v_5 = 10$, $c_{45} = u_4 + v_5 \Rightarrow 0 = u_4 +10 \Rightarrow u_4 = -10$ and
$c_{55} = u_5 + v_5 \Rightarrow 0 = u_5 +10 \Rightarrow u_5 = -10$

We now find net evaluations for unoccupied cells by using the relation $d_{ij} = z_{ij} - c_{ij}$

$d_{12} = z_{12} - c_{12} = u_1 + v_2 - 120 = -5, d_{14} = z_{14} - c_{14} = u_1 + v_4 - 180 = 0$,
$d_{15} = z_{15} - c_{15} = u_1 + v_5 - 0 = 10, d_{21} = z_{21} - c_{21} = u_2 + v_1 - 58 = -13$,
$d_{25} = z_{25} - c_{25} = u_2 + v_5 - 0 = -5, d_{31} = z_{31} - c_{31} = u_3 + v_1 - 62 = -12$,
$d_{32} = z_{32} - c_{32} = u_3 + v_2 - 110 = -5, d_{34} = z_{34} - c_{34} = u_3 + v_4 - 170 = 0$,
$d_{41} = z_{41} - c_{41} = u_4 + v_1 - 65 = -15, d_{42} = z_{42} - c_{42} = u_4 + v_2 - 115 = -10$,
$d_{43} = z_{43} - c_{43} = u_4 + v_3 - 80 = -15, d_{44} = z_{44} - c_{44} = u_4 + v_4 - 175 = -5$,
$d_{51} = z_{51} - c_{51} = u_5 + v_1 - 70 = -20, d_{52} = z_{52} - c_{52} = u_5 + v_2 - 135 = -30$,
$d_{53} = z_{53} - c_{53} = u_5 + v_3 - 85 = -20, d_{54} = z_{54} - c_{54} = u_5 + v_4 - 195 = -25$

Then the initial iteration is given by

Initial Iteration:

	X*	Y*	Z*	U*	V*	u_i
A*	5000 60	(-5) 20	3000 – θ 75	(0) 180	+θ (10) 0	0
B*	(-13) 58	2000 100	1200 60	6000 165	(-5) 0	-15
C*	(-12) 62	(-5) 110	5800+θ 65	(0) 170	450- θ 0	-10
D*	(-15) 65	(-10) 115	(-15) 80	(-5) 175	4900 0	-10
E*	(-20) 70	(-30) 135	(-20) 85	(-25) 195	6100 0	-10
v_j	60	115	75	180	10	

Since d_{15} is most positive, therefore cell (1, 5) enters the basis. We allocate an unknown quantity θ to this cell and identify a closed loop involving basic cells around this entering cell. Now θ = min {450, 3000} = 450, so we drop cell (3, 5).

Solving this we get the final optimal table after one iteration as given below:

Final optimal table:

	X*	Y*	Z*	U*	V*	u_i
A*	5000 60	(-5) 20	2550 75	(0) 180	450 0	0
B*	(-13) 58	2000 100	1200 60	6000 165	(-5) 0	-15
C*	(-12) 62	(-5) 110	6250 65	(0) 170	(0) 0	-10
D*	(-15) 65	(-10) 115	(-15) 80	(-5) 175	4900 0	-10
E*	(-20) 70	(-30) 135	(-20) 85	(-25) 195	6100 0	-10
v_j	60	115	75	180	10	

Here all $d_{ij} \leq 0$, so an optimal solution has reached as given below:

x_{11} = 5000, x_{13} = 2550, x_{15} = 450,
x_{22} = 2000, x_{23} =1200, x_{24} = 6000,
x_{33} = 6250, x_{45} = 4900, x_{55} = 6100.

Thus the optimal transportation cost is Z = Rs. 12,46,000.00/

6. Pseudo Code for different methods for initial basic feasible solution:

6.1 North West Corner Method-

```
define row_max = 5;
define col_max=5;
//create supply_array and require_array
float supply_array[row_max];
float require_array[col_max];
// creating the cost matrix and unit matrix
float cost_matrix[row_max][col_max];
float unit_matrix[row_max][col_max];
// initialize cost_matrix
for i:0 to row_max-1
    for j:0 to col_max-1
    cin>>cost_matrix[i][j];
    end loop
```

```
end loop
// initialize unit_matrix
for i:0 to row_max-1
   for j:0 to col_max-1
   unit_matrix[i[j]=0;
   end loop
end loop
float cost_minimal = 0.0;
float *supply_ptr;
float *require_ptr;
supply_ptr = &supply_array[0];
require_ptr = &require_array[0];
// initialize supply_array
for i:0 to row_max-1
cin>>supply_array;
end loop
//initialize require_array
for i:0 to col_max-1;
cin>>require_array;
float *matrix_ptr;
matrix_ptr=&cost_matrix[0][0];
int r = 0,c=0,x=0,y=0;
while(x<=row_max-1 && y=col_max-1)
{
if(*require_ptr>*supply_ptr)
{
unit_matrix[x][y]=supply_array[x];
require_array[y]=require_array[y]-unit_matrix[x][y];
supply_array[x]= supply_array[x]-unit_matrix[x][y];
cost_minimal = cost_minimal+unit_matrix[x][y];
x=x+1;
supply_ptr=supply_ptr+1;
matrix_ptr=matrix_ptr+col_max;
continue;
}
if(*require_ptr<*supply_ptr)
{
unit_matrix[x][y]=require_array[y];
require_array[y]=require_array[y]-unit_matrix[x][y];
supply_array[x] = supply_array[x]-unit_matrix[x][y];
cost_minimal                                    =
cost_minimal+unit_matrix[x][y]*cost_matrix[x][y];
y=y+1;
require_ptr=require_ptr+1;
matrix_ptr=matrix_ptr+1;
continue;
}
if ( *require_ptr==*supply_ptr)
{
unit_matrix[x][y]=require_array[y];
require_array[y]=require_array[y]-unit_matrix[x][y];
supply_array[x]=supply_array[x]-unit_matrix[x][y];
cost_minimal=cost_minimal+unit_matrix[x][y]*cost_matrix[x][y
];
y=y+1;
x=x+1;
require_ptr=require_ptr+1;
supply_ptr=supply_ptr+1;
matrix_ptr=matrix_ptr+col_max;
continue;
}
}
// displaying the unit matrix
for i:0 to row_max-1
     for j:0 ti col_max-1
     cout<<unit_matrix[i][j];
     end loop
end loop
//displaying the minimal cost
cout<<"the minimal cost obtained is :"<<cost_minimal;
end
```

6. 2 <u>Vogel Approximation Method-</u>

```
#define TRUE 1
#define FALSE 0
#define INFINITY 1111
#define N 3
#define M 4
void input(void);
void display(void);
void displayfinal(void);
void diffmin(void);
void table(void);
int max(int *,int *,int);
int min(int,int);
int mini(int *,int *,int);
int condition(void);
int arr[N][M];
int arrcopy[N][M];
int value[N][M];
int u[N];
int v[M];
int rowdiffmin[N];
int coldiffmin[M];
int decide[M+N];
int x[N],y[M];  /* x is u y is v */
//main point of execution starts from here
{    int i,j;
 table();    x[0]=0;


 for(i=0;i<3;i++)
  {        for(j=0;j<4;j++)
    {           if(value[i][j]!=0)
        cout<<"U[i+1]+V[j+1]= arr[i][j])";
  }
  }
 getch();}
void table(void)
{    int rowdiffminmaxpos;
   int coldiffminmaxpos;
  int decidemaxpos;
   int temp;
 int temparr[M];
   int i;
  clrscr();
   input();
  diffmin();
    display();
 while(condition())
 {        max(decide,&decidemaxpos,M+N);
     if(decidemaxpos>=0 && decidemaxpos<N)
    {           rowdiffminmaxpos=decidemaxpos;
       for(i=0;i<M;i++)
        temparr[i]=arr[decidemaxpos][i];
         mini(temparr,&coldiffminmaxpos,M);
     }
  else if(decidemaxpos>=N && decidemaxpos<M+N)
      {   coldiffminmaxpos=decidemaxpos-N;
            for(i=0;i<N;i++)
        temparr[i]=arr[decidemaxpos][i];
       temparr[i]=INFINITY;
     mini(temparr,&rowdiffminmaxpos,M);
    }
```

```
 temp=min(u[rowdiffminmaxpos],v[coldiffminmaxpos]);
   value[rowdiffminmaxpos][coldiffminmaxpos]=temp;
  if(temp==u[rowdiffminmaxpos])
   {          for(i=0;i<M;i++)
        arr[rowdiffminmaxpos][i]=INFINITY;
     u[rowdiffminmaxpos]-=temp;
      v[coldiffminmaxpos]-=temp;
     }
     else if(temp==v[coldiffminmaxpos])
     {          for(i=0;i<N;i++)
       arr[i][coldiffminmaxpos]=INFINITY;
      u[rowdiffminmaxpos]-=temp;
       v[coldiffminmaxpos]-=temp;
     }
  diffmin();
 getch();
   display();    }
  getch();
  displayfinal();
 getch();
}
void input(void)
{    int i,j;
 for(i=0;i<N;i++)
       for(j=0;j<M;j++)
      arr[i][j]=arrcopy[i][j]=-1;    /* Demand supply matrix */
 arr[0][0]=arrcopy[0][0]=5;
  arr[0][1]=arrcopy[0][1]=3;
 arr[0][2]=arrcopy[0][2]=6;
  arr[0][3]=arrcopy[0][3]=2;
  arr[1][0]=arrcopy[1][0]=4;
 arr[1][1]=arrcopy[1][1]=7;
  arr[1][2]=arrcopy[1][2]=9;
  arr[1][3]=arrcopy[1][3]=1;
  arr[2][0]=arrcopy[2][0]=3;
  arr[2][1]=arrcopy[2][1]=4;
 arr[2][2]=arrcopy[2][2]=7;
  arr[2][3]=arrcopy[2][3]=5;
 /* Supply */
 u[0]=19;    u[1]=37;    u[2]=34;
 /* Demand */      v[0]=16;      v[1]=18;      v[2]=31;      v[3]=25;
/***************************************************\       5
3    6    2        19    (1)   4    7    9    1        37
(3)   3    4    7    5        34    (1)   16    18    31
25   (1)   (1)   (1)   (1)    --------------------------------------------
--------   5    3    6    1111        19    (2)   4    7    9
1111        12    (3)   3    4    7    1111        34    (1)
16    18    31    0   (1)    (1)    (1)    (0)    -----------------------
-----------------------------   5    3    6    1111        19    (2)
1111   1111   1111   1111        0    (0)   3    4    7
1111        34    (1)   4    18    31    0   (2)    (1)    (1)
(0)    ----------------------------------------------------   5    1111
6    1111        1    (1)   1111   1111   1111   1111
0    (0)   3    1111   7    1111        34    (4)   4    0
31   0   (2)    (0)    (1)    (0)    --------------------------------------
-------------   1111   1111   6    1111        1    (1105)
1111   1111   1111   1111        0    (0)   1111   1111   7
1111        30    (1104)   0    0    31    0   (0)    (0)    (1)
(0)    --------------------------------------------------   1111   1111
1111   1111        0    (0)   1111   1111   1111   1111
0    (0)   1111   1111   7    1111        30    (1104)   0
0    30    0   (0)    (0)    (1104)  (0)    -----------------------------
----------------------   1111   1111   1111   1111        0    (0)
1111   1111   1111   1111        0    (0)   1111   1111
1111   1111        0    (0)   0    0    0    0   (0)    (0)
(0)    (0)    --------------------------------------------------   -----------
---------------------------------------   5    3|18   6|1    2    4|12
7    9    1|25   3|4    4    7|30   5   1111   1111   7
1111        30    (1104)
 X[1][2] = 18
  X[1][3] = 1
 X[2][1] = 12
   X[2][4] = 25
  X[3][1] = 4
 X[3][3]=30
/***************************************************/}
void displayfinal(void)
{    int i,j;
 cout<<endl;
   for(i=0;i<N;i++)
     for(j=0;j<M;j++)
    arr[i][j]=arrcopy[i][j];
    for(i=0;i<N;i++)
  {       for(j=0;j<M;j++)
      if(value[i][j]==0)
         cout<< arr[i][j];
      else
     cout<< arr[i][j]<<"|"<< value[i][j];
     cout<<endl;
 }
 cout<<endl;
   for(i=0;i<N;i++)
     for(j=0;j<M;j++)
     if(value[i][j]!=0)
          cout<<X[i+1][j+1]= value[i][j];}
int condition(void)
{    int i;    int flag;    int temp[M+N];    flag=1;
  for(i=0;i<N;i++)
     temp[i]=u[i];
   for(;i<M+N;i++)
    temp[i]=v[i];
    for(i=0;i<M+N;i++)
   {
    if(temp[i]!=0)
        flag=0;
   }
   if(flag==0)
     return(TRUE);
  else
   return(FALSE);
}
int min(int a,int b)
{    if(a>b)
     return(b);
   else
   return(a);}
int mini(int *a,int *aminpos,int n)
{    int i;    int amin;
  amin=a[0];    *aminpos=0;
    for(i=0;i<n;i++)
  {       if(a[i]<amin)
    {          amin=a[i];          *aminpos=i;       }
   }
 return(amin);}
int max(int *a,int *amaxpos,int n)
{    int i;    int amax;    amax=a[0];    *amaxpos=0;
   for(i=0;i<n;i++)
   {       if(a[i]>amax)
    { amax=a[i];          *amaxpos=i;       }
   }
  return(amax);}
void diffmin(void)
{    int min1,min2;    int arrmin1pos,arrmin2pos;    int i,j;
 for(i=0;i<N;i++)
```

```
  {      min1=arr[i][0];       arrmin1pos=0;
     for(j=0;j<M;j++)
  {          if(arr[i][j]<min1)
   {  min1=arr[i][j];              arrmin1pos=j;           }
   }       if(arrmin1pos==1)
    {          min2=arr[i][0];           arrmin2pos=0;        }
     else
    {          min2=arr[i][1];           arrmin2pos=1;        }
    for(j=0;j<M;j++)
      {          if(arr[i][j]<min2 && j!=arrmin1pos)
       {             min2=arr[i][j];
arrmin2pos=j;           }       }
rowdiffmin[i]=min2-min1;
      decide[i]=rowdiffmin[i];
    }
for(i=0;i<M;i++)
  { min1=arr[0][i];       arrmin1pos=0;       for(j=0;j<N;j++)
      { if(arr[j][i]<min1)
      { min1=arr[j][i];               arrmin1pos=j;
    }
    }
 if(arrmin1pos==1)
     { min2=arr[0][i];           arrmin2pos=0;
    }
   else
    { min2=arr[1][i];  arrmin2pos=1;
     }
    for(j=0;j<N;j++)
   {
    if(arr[j][i]<min2 && j!=arrmin1pos)
      { min2=arr[j][i];               arrmin2pos=j;
         }
     }
    coldiffmin[i]=min2-min1;
      decide[i+N]=coldiffmin[i];
  }
}
void display(void)
{    int i,j;    cout<<endl;    for(i=0;i<N;i++)
  {
    for(j=0;j<M;j++)
        cout<< arr[i][j];
    cout<< u[i]<<"”<< rowdiffmin[i];       printf("\n");
 }
 out<<endl;
for(i=0;i<M;i++)
  cout<< v[i];
    cout<<endl;
 for(i=0;i<M;i++)
       cout<< coldiffmin[i];

}
end;
```

6.3 **Least cost method –**

```
struct matrix
{
int r;
int c;
};
define row_max R
define col_max C;
matrix find_minloc(float*);
/* create initial matrix */
float cost_matrix[R][C];
for i:0 to R-1
for j:0 to C-1
cin>>cost_matrix[i][j];
end loop
end loop
i=0;
j=0;
/* create unit_matrix & initialize it to zero
float unit_matrix[R][C;
for i=0 to R-1;
float j:0 to C-1
unit_matrix[i][j]=0;
end loop;
end loop;
/* create supply_array and demand_array
float supply_array[R];
float require_array[C];
float *cost_matrix_ptr;
cost_matrix_ptr = & cost_matrix[0][0];
while(count<R-C+4)
{
float  minr_array[C]=0;
float minc_array[R]=0;
find_min_cost_matrix(cost_matrix[i][j]);
struct matrix min_loc=find-min_loc(cost_matrix[i][j]);
int a = i;
int b = min_loc;
int c=j;
int x = 0;
j=a;
if(require_array[min_loc.c]>supply_array[min_loc.r])
{
unit_matrix[min_loc.c][min_loc.c]= supply_matrix[min_loc.r];
require_array[min_loc.c]=require_array[min_loc.c]-
unit_matrix[min_loc.r][min_loc.c];
supply_array[min_loc.c]=supply_array[min_loc.r]-
unit_matrix[min_loc.r][min_loc.c];
delete_row_costmatrix(cost_matrix[min_loc.r][min_loc.c]);
// the row is deleted
if(min_loc.r==0)
i=i+1;
//construct new cost matrix(cost_matrix[i][j])
count++;
float              cost              =              cost             +
unit_matrix[min_loc.r][min_loc.c]*cost_matrix[min_loc.r][min_l
oc.c];
delete_row_cost_matrix(cost_matrix[min_loc.r][min_loc.c]);
construct new cost_matrix(cost_matrix[i[j]);
}
if (require_array[min_loc.c]<=supply_array[min_loc.r])
{
unit_matrix[min_loc.r][min_loc.c]-require_array[min_loc.c];
require_array[min_loc.c]=require_array[min_loc.c]-
unit_matrix[min_loc.r][min_loc.c];
supply_array[min_loc.c]=supply_array[min_loc.r]-
unit_matrix[min_loc.r][min_loc.c];
count++;
float              cost              =              cost             +
unit_matrix[min_loc.r][min_loc.c]*cost_matrix[min_loc.r][min_l
oc.c];
if(min_loc.c==0)
j=j+1;
delete_cost_matrix(cost_matrix[min_loc.r][min_loc.c]);
construct_new_cost_matrix(cost_matrix[i][j]);
}
// display the final cost
cout<<”the final cost is :”<<cost;
//display the final unit matrix
int l,m;
```

```
for l:0 to R-1
for m:0 to C-1
cout<<unit_matrix[l][m];
End loop;
End loop;
```

6.4 Row Minima Method-

```
define row_max  R;
define col_max  C;
/* create initial matrix
float cost_matrix[R][C];
for i:0 to R-1
        for  y:0 to C-1
        cin>>cost_matrix[i[j]
        end loop
end loop
i=0;
j=0;
/* Create unit-matrix and initialize it to zero
float unit_matrix[R][C];
for i:0 to R-1
        for j:0 to C-1
        unit_martix[i][j]=0;
        end loop
end loop
/* create supply array and demand array
float supply_matrix{R];
float require_matrix[C];
float*cost_matrix_ptr;
cost_matrix_ptr = &cost_matrix[0][0];
int count=0;
while(count<R-C+4)]
{
float minr_array[C]=0;
float minc_array[R]=0;
create_minr_array(cost_matrix[i][j]);
find_minr_array(cost_matrix[i][j]);
int min_loc = find_min_loc(cost_matrix[i][j]);
int a = i;
int b = min_loc;
int c = j;
int r = 0;
if (require_array[min_loc].c>supply_array[min_loc].r)
{
int  x = min_loc;
int y = 0;
unit_matrix[a[b] = supply_array[y];
require_array[min_loc]=require_array[min_loc]-
unit_matrix[a][b];
supply_array[y]=supply_array[y]-unit_matrix[a][b];
i=i+1;
y=I;
count++;
cost=cost+unit_matrix[a][b]*cost_matrix[a][b];
continue;
}
If(require_array[min_loc]<supply_array[y])
{
unit_matrix[i][b]=require_array[min_loc];
require_array[min_loc]=require_array[min_loc]-
unit_matrix[i][b];
supply_array[y]=supply_array[y]-unit_matrix[a][b];
cost=cost+unit_matrix[a][b]*cost_matrix[a][b];
y=y+1;
continue;
}
if (supply_array[0]==require_array[min_loc]&==0)
{
find 2_min_array(cost_matrix[i][j]);
int loc2= find loc2min_array(cost_matrix[i][j]);
unit_matrix[a][loc2]=0;
i=i+1;
j=1;
cost=cost+unit_matrix[a][b]*cost_matrix[a][b];
x=x+1;
continue;
}
}
/* display unit_matrix
int l,m;
for l:0 to R-1
for m:0 to C-1
cout<<unit_matrix[l][m];
/* display the final min_cost
cout<<"the final minimal cost is" <<cost;
```

6.5 Column minima method –

```
Pseudo-code –
define row_max R;
define col_max C;
/* create initial cost-matrix
float cost_matrix[R][C];
for i:0 to R-1
        for j:0 to C-1
        cin>>cost_matrix[i][j];
        end loop
end loop
i=0,j=0;
/* create unit matrix & initialize it to zero
float unit_matrix[R][C];
for i:0 to R-1
        for j:0  to C-1
        unit_matrix[i][j]=0;
        end loop
end loop
/* create supply_array & demand_array
float supply_array[R];
float demand_array[C];
float *cost_matrix_ptr;
cost_matrix_ptr = &cost_matrix[0][0];
int count=0;
int x=0;
while (count<R-C+4)
{
float  minr_array[C] = 0;
float minc_array[R]=0;
create_minc_array(cost_matrix[i][j]);
find_minr_array(cost_matrix[i][j]);
int min_loc = find min_loc(cost_matrix[i][j]);
a= min_loc;
b=j;
int c = j;
int x=0;
if ( require_array[y]>supply_array[min_loc])
{
unit_matrix[a][b] = supply_array[min_loc];
require_array[y] = require_array[y] – unit_matrix[a][b];
supply_array[min_loc]    =    supply_array[min_loc]    –
unit_matrix[a][b];
j=j+1;
count++;
cost = cost +unit _matrix[a][b]*cost_matrix[a][b];
continue;
}
```

```
if (require_array[y]<supply_array[min_loc])
{
unit_matrix[a][b] = require_array[j];
require_array[j] = require_array[j] – unit_matrix[a][b];
supply_array[min_loc] = supply_array[min_loc] – unit_matrix[a][b];
cost = cost +unit _matrix[a][b]*cost_matrix[a][b];
j=j+1;
continue;
}
If ( supply_array[min_loc]==require_array[0])
{
find loc2min_array(cost_matrix[i][j]);
int loc2 = find loc2min_array(cost_matrix[i][j]);
unit_matrix[loc2][b]= 0;
j=j+1;
i=j;
x=x+1;
cost = cost +unit _matrix[a][b]*cost_matrix[a][b];
}
/* display unit_matrix
int l,m;
for l:0 to R-1
for m:0 to C-1
cout<<unit_matrix;
/* display the final min_cost
cout<<"the final minimal cost is"<<cost;
```

7. Conclusion.

If this above optimal schedule is adopted by the suppliers of rice to Mizoram it would not only involve minimization of the transportation cost but it would also minimize the consumption of fuel in transporting the goods by the different carriers on the other hand. The optimal solution obtained in this present investigation shows much more closeness with initial basic feasible solution obtained by Vogel approximation methods. The comparison of optimal solution have been made with other methods of finding initial solutions and observe that Vogel's method give the better initial feasible solutions which are closer to optimal solution. The objected oriented programs using c++ have been developed and the compared with computed results for initial basic feasible solutions. The comparison shows that the computed results tally with the results obtained c++ programming. Pseudo code for said programs is given for better understanding.

8. Acknowledgments

Authors are thankful to the suppliers for providing necessary data in this regards. Authors are also thankful to Mr. Panthadeep Bhattacharjee, Assam University, silchar for providing technical support during this work.

References

[1] **Appa, G.M. (1973)** The Transportation Problem and Its Variants. Operational Research Quarterly, vol, 24, nr.1, pp.79-99.

[2] **Gass, S.I. (1990)** On Solving the Transportation Problem. Journal of Operational Research, Vol.41,nr.4, pp. 291-297.

[3] **Intrator ,J(1989)** A note on the Optimal Solution of a Transportation Problems. Asia –Pacific Journal of Operation Research ,Vol.6.nr.2,pp.207-208.

[4] **Patel, S.M (1992)** Resolution of Closed Loop in Transportation Problem. International Journal of Management and Systems,Vol. 81.nr.1,pp35-46.

[5] **Rao, S.S (1987)** Optimization Theory and Applications. Wiley Eastern Limited.

[6] **Sen, N. and Som,T. (2008a)** Mathematical Modeling of Transportation Related Problem of Southern Assam and Its Optimal Solution. AUJSc, Vol. 31, nr.1, pp. 22-27.

[7] **Sen, N. and Som, T. (2008b)** Mathematical Modeling of Transportation Related Fare Minimization Problem of South-Assam and an Approach to Its Optimal Solution. IJTM, Vol. 32, nr. 3, pp. 201-208.

[8] **Sen, N. and Som, T. (2008c)** Mathematical Modeling of Transportation Related Problem of Southern Assam with an Approach to Its Optimal Solution. ASR, Vol. 22, nr. 1, pp. 59-67.

[9] **Sen, N. (2008d)** Modeling of Transportation Related Problems of Northeastern Region of India and Their Solutions: A Case Study of Southern Part, PhD Thesis submitted to Assam University, Silchar, India.

[10] **Taha, H.A. (2006)**Operation Research :An Introduction ",Prentice Hall ,7th Editions5,USA.

[11] **Wilson, A.G; Coelho,J. D; Macgill,S.M and Williams, H.C.W.L(1981)** Optimization in Locational and Transport Analysis. John Willey and Sons.

Dr.Nabendu Sen is working as an Assistant Professor, Department of Mathematics, Assam University, Silchar (India). He has received Ph.D (Mathematics) in 2008 from Assam University, Silchar. His area of Research includes Optimization and Mathematical Modeling. He has published several papers journals of good repute. He has reviewed many research papers in reputed Journals.

Ms. Banashri Sinha is currently pursuing her research work under the supervision of Dr. Nabendu Sen in Assam University, silchar (India) on Optimization. She has completed her M.Sc (Mathematics) from Guwahati University, Assam (India) in 2008.

Dr.Tanmoy Som is serving as a Professor in the Department of Applied Mathematics, Institute of Technology, Banaras Hindu University, Varanasi, India. He was awarded Ph.D. in Mathematics by Banaras Hindu University, Varanasi in 1986. He has published more than 55 research papers in the areas of 'Functional Analysis, Optimization Modeling, Fuzzy Set Theory and Image Processing in the Journals of National and International reputes. He is also a reviewer of many National and International Journals and member of many academic bodies. He has delivered several invited talks at National as well as International level Seminars & Conferences. He has produced a good number of M. Phil. and Ph.D. s.