FALL SEM – (20-2021)

MAT – 2003

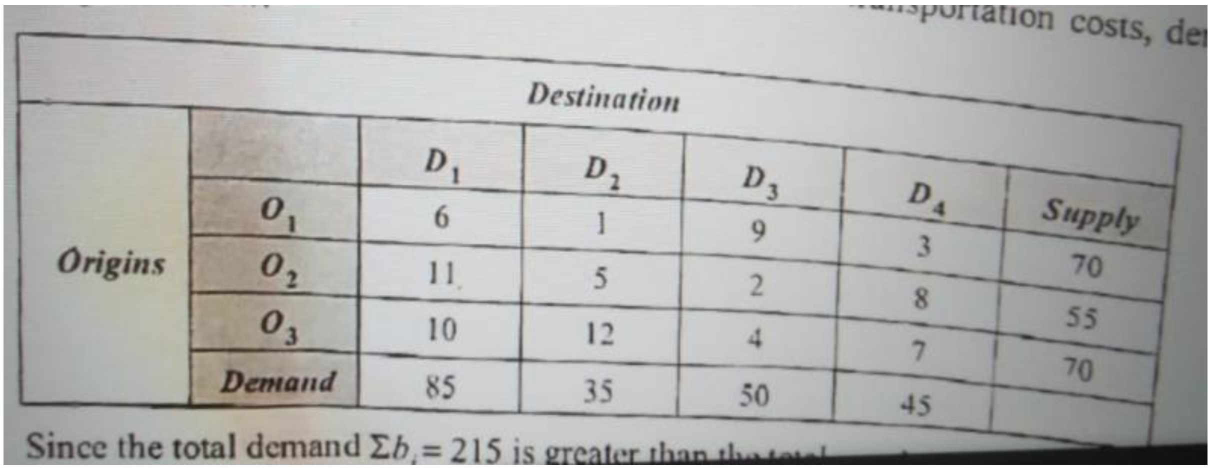
SUBMITTED BY: SRIHARSHITHA DEEPALA

REG NO: 19BCD7246

LAB NO: 8

SLOT: L6

QUESTION – 1:



...nsportation costs, de

		Destination				
		D_1	D_2	D_3	D_4	*Supply*
Origins	O_1	6	1	9	3	70
	O_2	11	5	2	8	55
	O_3	10	12	4	7	70
	Demand	85	35	50	45	

Since the total demand $\Sigma b_j = 215$ is greater than the total

CODE:

OUTPUT:

```
Enter the cost Matrix
[6 1 9 3;11 5 2 8;10 12 4 7;0 0 0 0]
Enter the supplies for column vector
[70;55;70;20]
Enter the demands for row vector
[85,35,50,45]
Elapsed time is 153381.117766 seconds.
Total cost of Non-degeneracy VAM
        1080
```

```
Occupied matrix of VAM
    25     0     0    45
    20    35     0     0
    20     0    50     0
    20     0     0     0

Elapsed time is 0.025881 seconds.
Balance
Balance
Balance
Total cost of MODI
   960

Occupied matrix of MODI
     0    30     0    40
     0     5    50     0
    65     0     0     5
    20     0     0     0

Elapsed time is 0.039319 seconds.
```

```
% Vogel'Approximation Method (VAM) and Modified distribution method (MODI)
clc
clear all
c=input("Enter the cost Matrix");
[m,n]=size(c);
s=input("Enter the supplies for column vector");
d=input("Enter the demands for row vector");
r=0.1;% value of extra element in the corrective degeneracy problem
numbasic=m+n-1;


c1=zeros(m,n);


s1=zeros(m,1);


d1=zeros(1,n);
x=zeros(m,n);


% sum demand and supply
sumd=0;
for j=1:n
sumd=sumd+d(j);
end
```

```
sums=0;


for i=1:m
sums=sums+s(i);
end
% Checking balance of demand and supply
if sumd==sums
else
disp('Review amount of supply and demand');
return
end
% Equivalant matrix
for j=1:n
for i=1:m
c1(i,j)=c(i,j);
end
end
for i=1:m
s1(i)=s(i);
end
for j=1:n
d1(j)=d(j);
end
toc
tic
%% Vogel's approximation method
iteration=0;
for k=1:m+n-1
iteration=iteration+1;
%% Row Difference
minrow1=zeros(m,1);
minrow2=zeros(m,1);
jmin=zeros(1,m);


for i=1:m
min1=inf;
for j=1:n
if c1(i,j)<min1
min1=c1(i,j);
jmin(i)=j;
end
end
minrow1(i)=min1;
end
for i=1:m
min2=inf;
for j=1:n
if j~=jmin(i)
if c1(i,j)<=min2
min2=c1(i,j);
end
end
end
minrow2(i)=min2;
```

```
end
%% Column Difference
mincol1=zeros(1,n);
mincol2=zeros(1,n);
imin=zeros(n,1);
for j=1:n
minR1=inf;
for i=1:m
if c1(i,j)<minR1
minR1=c1(i,j);
imin(j)=i;% position of minR1 each column
end
end
mincol1(j)=minR1;
end
for j=1:n
minR2=inf;
for i=1:m
if i~=imin(j)
if c1(i,j)<=minR2
minR2=c1(i,j);
end
end
end
mincol2(j)=minR2;
end
%% Difference
diffrow=zeros(m,1);
diffcol=zeros(1,n);
for i=1:m
diffrow(i)=minrow2(i)-minrow1(i);
end
for j=1:n
diffcol(j)=mincol2(j)-mincol1(j);
end
%% The greatest difference
R=0;
Row=zeros(m,1);
for i=1:m
if diffrow(i)>=R
R=diffrow(i);
iminrow=i; % the greatest diff. on column
end
end
Row(iminrow)=R;
S=0;
Col=zeros(1,n);
for j=1:n
if diffcol(j)>=S
S=diffcol(j);
jmincol=j;% the greatest diff. on row
end
end
Col(jmincol)=S;
great=zeros(1,n);
for j=1:n
```

```
if S>=R
great(jmincol)=Col(jmincol);
Colline=1;
else
great(iminrow)=Row(iminrow);
Colline=0;
end
end
%% Search the entry cell
if Colline==1 %Colline = 1
j=jmincol;
R1=inf;
for i=1:m
if c1(i,jmincol)<=R1
R1=c1(i,jmincol);
igreat=i; % the lowest cost on the jmincol
end
end
if s1(igreat)>d1(jmincol)
x(igreat,jmincol)=d1(jmincol);
s1(igreat)=s1(igreat)-d1(jmincol);
d1(jmincol)=0;
eliminaterow=0; % If current demand =0 (eliminaterow=0), eliminate a column.
elseif s1(igreat)<d1(jmincol)
x(igreat,jmincol)=s1(igreat);
d1(jmincol)=d1(jmincol)-s1(igreat);
s1(igreat)=0;
eliminaterow=1; % If supply =0 (eliminaterow=1), eliminate a row.
elseif s1(igreat)==d1(jmincol)
x(igreat,jmincol)=s1(igreat);
d1(jmincol)=0;
s1(igreat)=0;
eliminaterow=2;% If supply=demnad (eliminaterow=2),eliminate both a row and a
column
end
% Eliminate a column or a row
if eliminaterow==0% Eliminate a column
for i=1:m
c1(i,jmincol)=inf;
end
elseif eliminaterow==1 % Eliminate a row
for j=1:n
c1(igreat,j)=inf;
end
elseif eliminaterow==2
for i=1:m
c1(i,jmincol)=inf;
end
for j=1:n
c1(igreat,j)=inf;
end
end
else % Colline=0;
i=iminrow;
R2=inf;
for j=1:n
```

```
if c1(iminrow,j)<R2
R2=c1(iminrow,j);
                 jgreat=j; % the lowest cost on the iminrow
             end
         end

         if s1(iminrow)>d1(jgreat)
             x(iminrow,jgreat)=d1(jgreat);
             s1(iminrow)=s1(iminrow)-d1(jgreat);
             d1(jgreat)=0;
             eliminaterow=0; % If current demand=0 (eliminaterow=0), eliminate a
column.
         elseif s1(iminrow)<d1(jgreat)
             x(iminrow,jgreat)=s1(iminrow);

             d1(jgreat)=d1(jgreat)-s1(iminrow);
             s1(iminrow)=0;
             eliminaterow=1; % If current supply =0 (eliminaterow=1),eliminate a
row
         elseif s1(iminrow)==d1(jgreat)
             x(iminrow,jgreat)=s1(iminrow);
             d1(jgreat)=0;
             s1(iminrow)=0;
             eliminaterow=2; % If current supply =demand
(eliminaterow=2),eliminate both a row and a column
         end
            % Eliminate a column or a row
          if eliminaterow==0% Eliminate a column
              for i=1:m
                  c1(i,jgreat)=inf;% jmincol
              end
          elseif eliminaterow==1 % Eliminate a row
              for j=1:n
                  c1(iminrow,j)=inf; % iminrow = the greatest diff. row
              end
          elseif eliminaterow==2 %Eliminate both a row and a column
              for i=1:m
                  c1(i,jgreat)=inf;% Eliminate a column
              end
              for j=1:n
                  c1(iminrow,j)=inf; % Eliminate a row
              end
          end

     end

     %% Calculate the objective function

     ZVAM=0;
     for j=1:n
         for i=1:m
             if x(i,j)>0
                ZVAM=ZVAM+c(i,j)*x(i,j);
             end
         end
     end
```

```
end
       %% The degeneracy
     countx=0;


     for i=1:m
         for j=1:n
             if x(i,j)>0
                 countx=countx+1;
                 x1(i,j)=x(i,j);
                 x2(i,j)=x(i,j);
             end
         end
     end
   if countx>=numbasic
         degen=0;
         disp('Total cost of Non-degeneracy VAM');
         disp(ZVAM);
         disp('Occupied matrix of VAM')
         disp(x)
   else
         degen=1;
         disp('Total cost of Degeneracy VAM');
         disp(ZVAM);
         disp('Occupied matrix of VAM')
         disp(x)
   end
   toc
   %% How to correct degeneracy matrix
   if degen==1
numdegen=numbasic-countx;
iterationDegen=0;
for A=1:numdegen
iterationDegen=iterationDegen+1;
%% Construct the u-v variables
udual=zeros(m,1);
vdual=zeros(1,n);
udual(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0
vdual(j)=c(i,j)-udual(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i
udual(iu)=c(i,j)-vdual(j);
end
```

```
end
end
for k=1:m*n
for i=1:m
if udual(i)>0
iu=i;
for j=1:n
if x(iu,j)>0
vdual(j)=c(iu,j)-udual(iu);
end
end
end
end
for j=1:n
if vdual(j)>0
jv=j;
for i=1:m
if x(i,jv)>0
udual(i)=c(i,jv)-vdual(jv);
end
end
end
end
countu=0;
countv=0;
for i=1:m
if udual(i)<inf
countu=countu+1;
end
end;
for j=1:n;
if vdual(j)<inf;
countv=countv+1;
end
end
if (countu==m) && (countv==n)
return
end
end
%% Find the non-basic cells
unx=zeros(m,n);
for j=1:n
for i=1:m
if x(i,j)==0
unx(i,j)=c(i,j)-udual(i)-vdual(j)
end
end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx=inf;
for j=1:n
for i=1:m
if unx(i,j)<=maxunx
maxunx=unx(i,j);
imax=i;
jmax=j;
```

```
end
end
end
%% Count the number of basic variable on each and row
for j=1:n
sumcol=0;
for i=1:m
if x(i,j)>0
sumcol=sumcol+1;
end
end
x(m+1,j)=sumcol;
end
for i=1:m
sumrow=0;
for j=1:n
if x(i,j)>0
sumrow=sumrow+1;
end
end
x(i,n+1)=sumrow;
end
%% Construct the equipvalent x matrix
for j=1:n+1
for i=1:m+1
x1(i,j)=x(i,j);
end
end
%% Eliminate an entering variable for adding a new one
for j=1:n
for i=1:m
if (x(i,j)==1 || x1(i,j)==1)
x1(i,j)=0;
end
end
end
% Add a new entering variable to x1 matrix
% Assign the small value =1
for j=1:n;
for i=1:m
x1(imax,jmax)=1;
end
end
% Seaching and adding the entering point for corrective action
for i=1:m
if i~=imax
if x1(i,jmax)>0
ienter=i;
for j=1:n
if j~=jmax
if x1(ienter,j)>0 && x1(imax,j)==0
jenter=j;
end
end
end
end
```

```
end
end
x1(imax,jenter)=1;
x(imax,jenter)=1;
end
else
end
tic;
iterationOpt=0;
for q=1:n*m
iterationOpt=iterationOpt+1;
% The Modified distribution method
%% Construct the u-v variables
udual=zeros(m,1);
vdual=zeros(1,n);
for i=1:m
udual(i)=inf;
end
for j=1:n
vdual(j)=inf;
end
udual(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0 && udual(i)<inf
vdual(j)=c(i,j)-udual(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i;
if udual(iu)~=inf
vdual(j)=c(i,j)-udual(iu);
else
if vdual(j)~=inf
udual(iu)=c(i,j)-vdual(j);
end
end
end
end
end
for k=1:m*n
for i=1:m
if udual(i)~=inf
iu=i;
for j=1:n
if x(iu,j)>0 && udual(iu)<inf
vdual(j)=c(iu,j)-udual(iu);
end
end
end
end
for j=1:n
if vdual(j)~=inf
```

```matlab
jv=j;
for i=1:m
if x(i,jv)>0 && vdual(jv)<inf
udual(i)=c(i,jv)-vdual(jv);
end
end
end
end
countu=0;
countv=0;
for i=1:m
if udual(i)<inf
countu=countu+1;
end
end
for j=1:n
if vdual(j)<inf
countv=countv+1;
end
end
if (countu==m) && (countv==n)
break
end
end
%% Find the non-basic cells
unx=zeros(m,n);
for j=1:n
for i=1:m
if x(i,j)==0
unx(i,j)=udual(i)+vdual(j)-c(i,j);
end
end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx=0;
for j=1:n
for i=1:m
if unx(i,j)>=maxunx
maxunx=unx(i,j);
imax=i;
jmax=j;
end
end
end
%% The objective function value
Z=0;
for j=1:n
for i=1:m
if x(i,j)>0
Z=Z+x(i,j)*c(i,j);
end
end
end
iterationOpt=iterationOpt+1;
%% Control loop
if maxunx==0
```

```matlab
break;
else
end
%% Entering a new basic variable add into the basic variable matrix
x1=zeros(m+1,n+1);
x2=zeros(m+1,n+1);
% Construct the equivalent basic variable matrix
for j=1:n
for i=1:m
if x(i,j)>0
x1(i,j)=x(i,j);
x2(i,j)=x(i,j);
end
end
end
% Entering the new variable
x1(imax,jmax)=inf;
x2(imax,jmax)=inf;
for j=1:n
countcol=0;
for i=1:m
if x1(i,j)>0 || x1(i,j)==inf
countcol=countcol+1;
end
end
x1(m+1,j)=countcol;
x2(m+1,j)=countcol;
end
for i=1:m
countrow=0;
for j=1:n
if x1(i,j)>0
countrow=countrow+1;
end
end
x1(i,n+1)=countrow;
x2(i,n+1)=countrow;
end
%% Construct loop


% Eliminate the basic variables that has only one on each row
iterationloop=0;
for i=1:m
iterationloop=iterationloop+1;
for i=1:m
if x2(i,n+1)==1
ieliminate=i;
for j=1:n
if x2(ieliminate,j)<inf && x2(ieliminate,j)>0
jeliminate=j;
x2(ieliminate,jeliminate)=0;% Eliminate the basic variable on row
x2(ieliminate,n+1)=x2(ieliminate,n+1)-1; % decrease the number of the basic
variable on row one unit
x2(m+1,jeliminate)=x2(m+1,jeliminate)-1; % decrease the number of the basic
variable on column one unit
```

```
end
end
end
end
% Eliminate the basic variables that has only one on each column
for j=1:n
if x2(m+1,j)==1
jeliminate1=j;
for i=1:m
if x2(i,jeliminate1)<inf && x2(i,jeliminate1)>0
ieliminate1=i;
x2(ieliminate1,jeliminate1)=0;% Eliminate the basic variable on row
x2(m+1,jeliminate1)=x2(m+1,jeliminate1)-1; % decrease the number of the basic
variable on column one unit
x2(ieliminate1,n+1)=x2(ieliminate1,n+1)-1;% decrease the number of the basic
variable on row one unit
end
end
end
end
% Control the constructing loop path
for j=1:n
for i=1:m
if (x2(i,n+1)==0 || x2(i,n+1)==2) && (x2(m+1,j)==0 || x2(m+1,j)==2)
break;
else
end
end
end
end
%% Make +/-sign on basic variables in the loop path (x2)
%1. Add - sign on basic variable on row(imax) and on basic variable on
%column (jmax)
for j=1:n
if (x2(imax,j)~=0 && x2(imax,j)<inf && x2(imax,n+1)==2)
jneg=j;
x2(imax,jneg)=(-1)*x2(imax,jneg);
x2(m+1,jneg)=1;
x2(imax,n+1)=1;
for i=1:m
if (x2(i,jneg)>0 && x2(m+1,jneg)==1)
ineg=i;
end
end
end
end
for p=1:n
for j=1:n
if (j~=jneg && x2(ineg,j)>0 )&& (x2(ineg,n+1)==2)
jneg1=j;
x2(ineg,jneg1)=(-1)*x2(ineg,jneg1);
x2(ineg,n+1)=1;
x2(m+1,jneg1)=1;
for i=1:m
if (x2(i,jneg1)>0 && x2(m+1,jneg1)==1)
ineg1=i;
```

```
ineg=ineg1;
jneg=jneg1;
end
end
end
end
% Control loop
if jneg1==jmax
break
end
end
%% Search the net smallest negative basic variable in the loop path
small=inf;
for j=1:n
for i=1:m
if x2(i,j)<0
if abs(x2(i,j))<small
small=abs(x2(i,j));
end
end
end
end
% Construct the loop path
x3=zeros(m,n);
for j=1:n
for i=1:m
x3(i,j)=x2(i,j);
end
end
%% Add the smallest value to the positive basis variable and subtract to the
negative basic variable
for i=1:m
for j=1:n
x3(imax,jmax)=small;
if x3(i,j)~=0
if x3(i,j)<0
x3(i,j)=(x3(i,j))+small;
if x3(i,j)==0
x3(i,j)=inf;
end
else
if i~=imax && j~=jmax
x3(i,j)=x3(i,j)+small;
end
end
end
end
end
%% Combine the new absolute loop path to the x matrix
xpath=zeros(m,n);
for j=1:n
for i=1:m
xpath(i,j)=x(i,j);
end
end
for j=1:n
for i=1:m
if x3(i,j)~=0
if x3(i,j)==inf
xpath(i,j)=0;
else
xpath(i,j)=abs(x3(i,j));
end
end
end
end
%% The objective function
Zopt=0;
for j=1:n
for i=1:m
if round(xpath(i,j))>0
Zopt=Zopt+round(xpath(i,j))*c(i,j);
end
end
end
%% Check balance
sumbal=0;
for j=1:n
for i=1:m
if xpath(i,j)>0
sumbal=sumbal+xpath(i,j);
end
end
end
if sums==sumbal
disp('Balance');
else
break;
end
%% Transfer x to xpath
for j=1:n
for i=1:m
x(i,j)=xpath(i,j);
end
end
%% Checking degeneracy
countxMODI1=0;
for j=1:n
for i=1:m
if x(i,j)>0
countxMODI1=countxMODI1+1;
end
end
end
if countxMODI1<numbasic
degen2=1;
disp('Degeneracy within Loop');
disp(q);
else
degen2=0;
end
%% Corrective Degeneracy
```

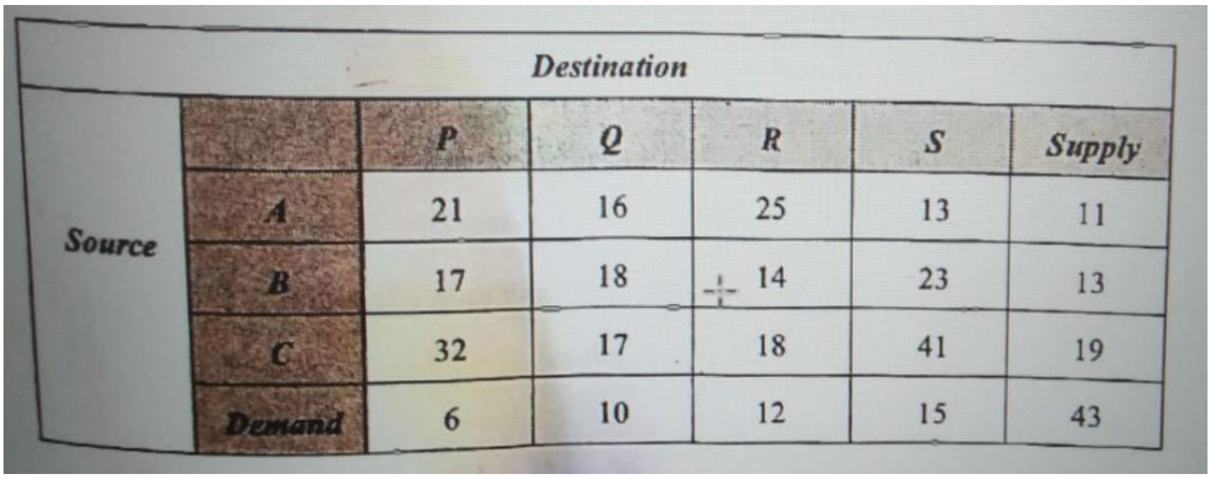
```
if degen2==1
numdegen2=numbasic-countxMODI1;
iterationDegen2=0;
for A=1:numdegen2
iterationDegen2=iterationDegen2+1;
%% Construct the u-v variables
udual2=zeros(m,1);
vdual2=zeros(1,n);
for i=1:m
udual2(i)=inf;
end
for j=1:n
vdual2(j)=inf;
end
udual2(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0
vdual2(j)=c(i,j)-udual2(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i;
if udual2(iu)~=inf
vdual2(j)=c(i,j)-udual2(iu);
else
if vdual2(j)~=inf
udual2(iu)=c(i,j)-vdual2(j);
end
end
end
end
end
for k=1:m*n
for i=1:m
if udual2(i)~=inf
iu=i;
for j=1:n
if x(iu,j)>0
vdual2(j)=c(iu,j)-udual2(iu);
end
end
end
end
for j=1:n
if vdual2(j)~=inf
jv=j;
for i=1:m
if x(i,jv)>0
udual2(i)=c(i,jv)-vdual2(jv);
end
end
end
```

```
end
countu2=0;
countv2=0;
for i=1:m
if udual2(i)<inf
countu2=countu2+1;
end
end
for j=1:n
if vdual2(j)<inf
countv2=countv2+1;
end
end
if (countu2==m) & (countv2==n)
break
end
end
%% Find the non-basic cells
unx2=zeros(m,n);
for j=1:n
for i=1:m
if x(i,j)==0
unx2(i,j)=udual2(i)+vdual2(j)-c(i,j);
end
end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx2=0;
for j=1:n
for i=1:m
if unx2(i,j)>=maxunx2
maxunx2=unx2(i,j);
imax2=i;
jmax2=j;
end
end
end
%% Count the number of basic variable on each and row
for j=1:n
sumcol=0;
for i=1:m
if x(i,j)>0
sumcol=sumcol+1;
end
end
x(m+1,j)=sumcol;
end
for i=1:m
sumrow=0;
for j=1:n
if x(i,j)>0
sumrow=sumrow+1;
end
end
x(i,n+1)=sumrow;
end
```

```
%% Construct the equivalent x matrix
x12=zeros(m+1,n+1);
for j=1:n+1
for i=1:m+1
x12(i,j)=x(i,j);
end
end
%% Eliminate an entering variable for adding a new one
for j=1:n
for i=1:m
if (x(i,j)==r|| x12(i,j)==r)
x12(i,j)=0;
end
end
end
% Add a new entering variable to x1 matrix
for j=1:n
for i=1:m
x12(imax2,jmax2)=r;
end
end
% Seaching and adding the entering point for corrective action
for i=1:m
if i~=imax2
if x1(i,jmax2)>0
ienter2=i;
for j=1:n
if j~=jmax2
if x1(ienter2,j)>0 && x1(imax2,j)==0
jenter2=j;
end
end
end
end
end
end
x1(imax2,jenter2)=r;
x(imax2,jenter2)=r;
end
end% End of degen2==1;
end% End of q=1:n;
%% Checking degeneracy
countxMODI2=0;
for j=1:n
for i=1:m
if x(i,j)>0
countxMODI2=countxMODI2+1;
end
end
end
%% Total cost
ZMODI=0;
for j=1:n
for i=1:m
if x(i,j)>0
ZMODI=ZMODI+x(i,j)*c(i,j);
```

```
end
end
end
if countxMODI2<numbasic
disp('Degeneracy in solution of MODI');
end
if maxunx==0
disp('Total cost of MODI');
disp(ZMODI);
disp('Occupied matrix of MODI');
disp(x);
else
disp('Reviewed Loop');
end
toc
```

QUESTION – 2:



		Destination				
		P	Q	R	S	Supply
Source	A	21	16	25	13	11
	B	17	18	14	23	13
	C	32	17	18	41	19
	Demand	6	10	12	15	43

OUTPUT:

```
Enter the cost Matrix
[21 16 25 13;17 18 14 23;32 17 18 41]
Enter the supplies for column vector
[11;13;19]
Enter the demands for row vector
[6 10 12 15]
Elapsed time is 1032.259773 seconds.
Total cost of Non-degeneracy VAM
   711

Occupied matrix of VAM
     0     0     0    11
     6     0     3     4
     0    10     9     0

Elapsed time is 0.020643 seconds.
Total cost of MODI
   711

Occupied matrix of MODI
     0     0     0    11
     6     0     3     4
     0    10     9     0
Elapsed time is 0.009144 seconds.
```

CODE:

```matlab
% Vogel'Approximation Method (VAM) and Modified distribution method (MODI)
clc
clear all
c=input("Enter the cost Matrix");
[m,n]=size(c);
s=input("Enter the supplies for column vector");
d=input("Enter the demands for row vector");
r=0.1;% value of extra element in the corrective degeneracy problem
numbasic=m+n-1;


c1=zeros(m,n);


s1=zeros(m,1);


d1=zeros(1,n);
x=zeros(m,n);


% sum demand and supply
sumd=0;
for j=1:n
sumd=sumd+d(j);
end


sums=0;


for i=1:m
sums=sums+s(i);
end
% Checking balance of demand and supply
if sumd==sums
else
```

```
disp('Review amount of supply and demand');
return
end
% Equivalant matrix
for j=1:n
for i=1:m
c1(i,j)=c(i,j);
end
end
for i=1:m
s1(i)=s(i);
end
for j=1:n
d1(j)=d(j);
end
toc
tic
%% Vogel's approximation method
iteration=0;
for k=1:m+n-1
iteration=iteration+1;
%% Row Difference
minrow1=zeros(m,1);
minrow2=zeros(m,1);
jmin=zeros(1,m);


for i=1:m
min1=inf;
for j=1:n
if c1(i,j)<min1
min1=c1(i,j);
jmin(i)=j;
end
end
minrow1(i)=min1;
end
for i=1:m
min2=inf;
for j=1:n
if j~=jmin(i)
if c1(i,j)<=min2
min2=c1(i,j);
end
end
end
minrow2(i)=min2;
end
%% Column Difference
mincol1=zeros(1,n);
mincol2=zeros(1,n);
imin=zeros(n,1);
for j=1:n
minR1=inf;
for i=1:m
if c1(i,j)<minR1
```

```
minR1=c1(i,j);
imin(j)=i;% position of minR1 each column
end
end
mincol1(j)=minR1;
end
for j=1:n
minR2=inf;
for i=1:m
if i~=imin(j)
if c1(i,j)<=minR2
minR2=c1(i,j);
end
end
end
mincol2(j)=minR2;
end
%% Difference
diffrow=zeros(m,1);
diffcol=zeros(1,n);
for i=1:m
diffrow(i)=minrow2(i)-minrow1(i);
end
for j=1:n
diffcol(j)=mincol2(j)-mincol1(j);
end
%% The greatest difference
R=0;
Row=zeros(m,1);
for i=1:m
if diffrow(i)>=R
R=diffrow(i);
iminrow=i; % the greatest diff. on column
end
end
Row(iminrow)=R;
S=0;
Col=zeros(1,n);
for j=1:n
if diffcol(j)>=S
S=diffcol(j);
jmincol=j;% the greatest diff. on row
end
end
Col(jmincol)=S;
great=zeros(1,n);
for j=1:n
if S>=R
great(jmincol)=Col(jmincol);
Colline=1;
else
great(iminrow)=Row(iminrow);
Colline=0;
end
end
%% Search the entry cell
```

```
if Colline==1 %Colline = 1
j=jmincol;
R1=inf;
for i=1:m
if c1(i,jmincol)<=R1
R1=c1(i,jmincol);
igreat=i; % the lowest cost on the jmincol
end
end
if s1(igreat)>d1(jmincol)
x(igreat,jmincol)=d1(jmincol);
s1(igreat)=s1(igreat)-d1(jmincol);
d1(jmincol)=0;
eliminaterow=0; % If current demand =0 (eliminaterow=0), eliminate a column.
elseif s1(igreat)<d1(jmincol)
x(igreat,jmincol)=s1(igreat);
d1(jmincol)=d1(jmincol)-s1(igreat);
s1(igreat)=0;
eliminaterow=1; % If supply =0 (eliminaterow=1), eliminate a row.
elseif s1(igreat)==d1(jmincol)
x(igreat,jmincol)=s1(igreat);
d1(jmincol)=0;
s1(igreat)=0;
eliminaterow=2;% If supply=demnad (eliminaterow=2),eliminate both a row and a
column
end
% Eliminate a column or a row
if eliminaterow==0% Eliminate a column
for i=1:m
c1(i,jmincol)=inf;
end
elseif eliminaterow==1 % Eliminate a row
for j=1:n
c1(igreat,j)=inf;
end
elseif eliminaterow==2
for i=1:m
c1(i,jmincol)=inf;
end
for j=1:n
c1(igreat,j)=inf;
end
end
else % Colline=0;
i=iminrow;
R2=inf;
for j=1:n
if c1(iminrow,j)<R2
R2=c1(iminrow,j);
jgreat=j; % the lowest cost on the iminrow
end
end
if s1(iminrow)>d1(jgreat)
x(iminrow,jgreat)=d1(jgreat);
s1(iminrow)=s1(iminrow)-d1(jgreat);
d1(jgreat)=0;
```

```
eliminaterow=0; % If current demand=0 (eliminaterow=0), eliminate a column.
elseif s1(iminrow)<d1(jgreat)
x(iminrow,jgreat)=s1(iminrow);
d1(jgreat)=d1(jgreat)-s1(iminrow);
s1(iminrow)=0;
eliminaterow=1; % If current supply =0 (eliminaterow=1),eliminate a row
elseif s1(iminrow)==d1(jgreat)
x(iminrow,jgreat)=s1(iminrow);
d1(jgreat)=0;
s1(iminrow)=0;
eliminaterow=2; % If current supply =demand (eliminaterow=2),eliminate both a row
and a column
end
% Eliminate a column or a row
if eliminaterow==0% Eliminate a column
for i=1:m
c1(i,jgreat)=inf;% jmincol
end
elseif eliminaterow==1 % Eliminate a row
for j=1:n
c1(iminrow,j)=inf; % iminrow = the greatest diff. row
end
elseif eliminaterow==2 %Eliminate both a row and a column
for i=1:m
c1(i,jgreat)=inf;% Eliminate a column
end
for j=1:n
c1(iminrow,j)=inf; % Eliminate a row
end
end
end
%% Calculate the objective function
ZVAM=0;
for j=1:n
for i=1:m
if x(i,j)>0
ZVAM=ZVAM+c(i,j)*x(i,j);
end
end
end
end
%% The degeneracy
countx=0;
for i=1:m
for j=1:n
if x(i,j)>0
countx=countx+1;
x1(i,j)=x(i,j);
x2(i,j)=x(i,j);
end
end
end
if countx>=numbasic
degen=0;
disp('Total cost of Non-degeneracy VAM');
disp(ZVAM);
disp('Occupied matrix of VAM')
disp(x)
else
degen=1;
disp('Total cost of Degeneracy VAM');
disp(ZVAM);
disp('Occupied matrix of VAM')
disp(x)
end
toc
%% How to correct degeneracy matrix
if degen==1
numdegen=numbasic-countx;
iterationDegen=0;
for A=1:numdegen
iterationDegen=iterationDegen+1;
%% Construct the u-v variables
udual=zeros(m,1);
vdual=zeros(1,n);
udual(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0
vdual(j)=c(i,j)-udual(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i
udual(iu)=c(i,j)-vdual(j);
end
end
end
for k=1:m*n
for i=1:m
if udual(i)>0
iu=i;
for j=1:n
if x(iu,j)>0
vdual(j)=c(iu,j)-udual(iu);
end
end
end
end
for j=1:n
if vdual(j)>0
jv=j;
for i=1:m
if x(i,jv)>0
udual(i)=c(i,jv)-vdual(jv);
end
end
end
end
```

```
countu=0;
countv=0;
for i=1:m
if udual(i)<inf
countu=countu+1;
end
end;
for j=1:n;
if vdual(j)<inf;
countv=countv+1;
end
end
if (countu==m) && (countv==n)
return
end
end
%% Find the non-basic cells
unx=zeros(m,n);
for j=1:n
for i=1:m
if x(i,j)==0
unx(i,j)=c(i,j)-udual(i)-vdual(j)
end
end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx=inf;
for j=1:n
for i=1:m
if unx(i,j)<=maxunx
maxunx=unx(i,j);
imax=i;
jmax=j;
end
end
end
%% Count the number of basic variable on each and row
for j=1:n
sumcol=0;
for i=1:m
if x(i,j)>0
sumcol=sumcol+1;
end
end
x(m+1,j)=sumcol;
end
for i=1:m
sumrow=0;
for j=1:n
if x(i,j)>0
sumrow=sumrow+1;
end
end
x(i,n+1)=sumrow;
end
%% Construct the equipvalent x matrix
```

```
for j=1:n+1
for i=1:m+1
x1(i,j)=x(i,j);
end
end
%% Eliminate an entering variable for adding a new one
for j=1:n
for i=1:m
if (x(i,j)==1 || x1(i,j)==1)
x1(i,j)=0;
end
end
end
% Add a new entering variable to x1 matrix
% Assign the small value =1
for j=1:n;
for i=1:m
x1(imax,jmax)=1;
end
end
% Seaching and adding the entering point for corrective action
for i=1:m
if i~=imax
if x1(i,jmax)>0
ienter=i;
for j=1:n
if j~=jmax
if x1(ienter,j)>0 && x1(imax,j)==0
jenter=j;
end
end
end
end
end
end
x1(imax,jenter)=1;
x(imax,jenter)=1;
end
else
end
tic;
iterationOpt=0;
for q=1:n*m
iterationOpt=iterationOpt+1;
% The Modified distribution method
%% Construct the u-v variables
udual=zeros(m,1);
vdual=zeros(1,n);
for i=1:m
udual(i)=inf;
end
for j=1:n
vdual(j)=inf;
end
udual(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0 && udual(i)<inf
vdual(j)=c(i,j)-udual(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i;
if udual(iu)~=inf
vdual(j)=c(i,j)-udual(iu);
else
if vdual(j)~=inf
udual(iu)=c(i,j)-vdual(j);
end
end
end
end
end
for k=1:m*n
for i=1:m
if udual(i)~=inf
iu=i;
for j=1:n
if x(iu,j)>0 && udual(iu)<inf
vdual(j)=c(iu,j)-udual(iu);
end
end
end
end
for j=1:n
if vdual(j)~=inf
jv=j;
for i=1:m
if x(i,jv)>0 && vdual(jv)<inf
udual(i)=c(i,jv)-vdual(jv);
end
end
end
end
countu=0;
countv=0;
for i=1:m
if udual(i)<inf
countu=countu+1;
end
end
for j=1:n
if vdual(j)<inf
countv=countv+1;
end
end
if (countu==m) && (countv==n)
break
end
```

```
end
%% Find the non-basic cells
unx=zeros(m,n);
for j=1:n
for i=1:m
if x(i,j)==0
unx(i,j)=udual(i)+vdual(j)-c(i,j);
end
end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx=0;
for j=1:n
for i=1:m
if unx(i,j)>=maxunx
maxunx=unx(i,j);
imax=i;
jmax=j;
end
end
end
%% The objective function value
Z=0;
for j=1:n
for i=1:m
if x(i,j)>0
Z=Z+x(i,j)*c(i,j);
end
end
end
iterationOpt=iterationOpt+1;
%% Control loop
if maxunx==0
break;
else
end
%% Entering a new basic variable add into the basic variable matrix
x1=zeros(m+1,n+1);
x2=zeros(m+1,n+1);
% Construct the equivalent basic variable matrix
for j=1:n
for i=1:m
if x(i,j)>0
x1(i,j)=x(i,j);
x2(i,j)=x(i,j);
end
end
end
% Entering the new variable
x1(imax,jmax)=inf;
x2(imax,jmax)=inf;
for j=1:n
countcol=0;
for i=1:m
if x1(i,j)>0 || x1(i,j)==inf
countcol=countcol+1;
```

```
end
end
x1(m+1,j)=countcol;
x2(m+1,j)=countcol;
end
for i=1:m
countrow=0;
for j=1:n
if x1(i,j)>0
countrow=countrow+1;
end
end
x1(i,n+1)=countrow;
x2(i,n+1)=countrow;
end
%% Construct loop


% Eliminate the basic variables that has only one on each row
iterationloop=0;
for i=1:m
iterationloop=iterationloop+1;
for i=1:m
if x2(i,n+1)==1
ieliminate=i;
for j=1:n
if x2(ieliminate,j)<inf && x2(ieliminate,j)>0
jeliminate=j;
x2(ieliminate,jeliminate)=0;% Eliminate the basic variable on row
x2(ieliminate,n+1)=x2(ieliminate,n+1)-1; % decrease the number of the basic
variable on row one unit
x2(m+1,jeliminate)=x2(m+1,jeliminate)-1; % decrease the number of the basic
variable on column one unit
end
end
end
end
% Eliminate the basic variables that has only one on each column
for j=1:n
if x2(m+1,j)==1
jeliminate1=j;
for i=1:m
if x2(i,jeliminate1)<inf && x2(i,jeliminate1)>0
ieliminate1=i;
x2(ieliminate1,jeliminate1)=0;% Eliminate the basic variable on row
x2(m+1,jeliminate1)=x2(m+1,jeliminate1)-1; % decrease the number of the basic
variable on column one unit
x2(ieliminate1,n+1)=x2(ieliminate1,n+1)-1;% decrease the number of the basic
variable on row one unit
end
end
end
end
% Control the constructing loop path
for j=1:n
for i=1:m
```

```matlab
if (x2(i,n+1)==0 || x2(i,n+1)==2) && (x2(m+1,j)==0 || x2(m+1,j)==2)
break;
else
end
end
end
end
%% Make +/-sign on basic variables in the loop path (x2)
%1. Add - sign on basic variable on row(imax) and on basic variable on
%column (jmax)
for j=1:n
if (x2(imax,j)~=0 && x2(imax,j)<inf && x2(imax,n+1)==2)
jneg=j;
x2(imax,jneg)=(-1)*x2(imax,jneg);
x2(m+1,jneg)=1;
x2(imax,n+1)=1;
for i=1:m
if (x2(i,jneg)>0 && x2(m+1,jneg)==1)
ineg=i;
end
end
end
end
for p=1:n
for j=1:n
if (j~=jneg && x2(ineg,j)>0 )&& (x2(ineg,n+1)==2)
jneg1=j;
x2(ineg,jneg1)=(-1)*x2(ineg,jneg1);
x2(ineg,n+1)=1;
x2(m+1,jneg1)=1;
for i=1:m
if (x2(i,jneg1)>0 && x2(m+1,jneg1)==1)
ineg1=i;
ineg=ineg1;
jneg=jneg1;
end
end
end
end
% Control loop
if jneg1==jmax
break
end
end
%% Search the net smallest negative basic variable in the loop path
small=inf;
for j=1:n
for i=1:m
if x2(i,j)<0
if abs(x2(i,j))<small
small=abs(x2(i,j));
end
end
end
end
% Construct the loop path
```

```
x3=zeros(m,n);
for j=1:n
for i=1:m
x3(i,j)=x2(i,j);
end
end
%% Add the smallest value to the positive basis variable and subtract to the
negative basic variable
for i=1:m
for j=1:n
x3(imax,jmax)=small;
if x3(i,j)~=0
if x3(i,j)<0
x3(i,j)=(x3(i,j))+small;
if x3(i,j)==0
x3(i,j)=inf;
end
else
if i~=imax && j~=jmax
x3(i,j)=x3(i,j)+small;
end
end
end
end
end
%% Combine the new absolute loop path to the x matrix
xpath=zeros(m,n);
for j=1:n
for i=1:m
xpath(i,j)=x(i,j);
end
end
for j=1:n
for i=1:m
if x3(i,j)~=0
if x3(i,j)==inf
xpath(i,j)=0;
else
xpath(i,j)=abs(x3(i,j));
end
end
end
end
%% The objective function
Zopt=0;
for j=1:n
for i=1:m
if round(xpath(i,j))>0
Zopt=Zopt+round(xpath(i,j))*c(i,j);
end
end
end
%% Check balance
sumbal=0;
for j=1:n
for i=1:m
```

```matlab
if xpath(i,j)>0
sumbal=sumbal+xpath(i,j);
end
end
end
if sums==sumbal
disp('Balance');
else
break;
end
%% Transfer x to xpath
for j=1:n
for i=1:m
x(i,j)=xpath(i,j);
end
end
%% Checking degeneracy
countxMODI1=0;
for j=1:n
for i=1:m
if x(i,j)>0
countxMODI1=countxMODI1+1;
end
end
end
if countxMODI1<numbasic
degen2=1;
disp('Degeneracy within Loop');
disp(q);
else
degen2=0;
end
%% Corrective Degeneracy
if degen2==1
numdegen2=numbasic-countxMODI1;
iterationDegen2=0;
for A=1:numdegen2
iterationDegen2=iterationDegen2+1;
%% Construct the u-v variables
udual2=zeros(m,1);
vdual2=zeros(1,n);
for i=1:m
udual2(i)=inf;
end
for j=1:n
vdual2(j)=inf;
end
udual2(1)=0;
for i=1:1
for j=1:n
if x(i,j)>0
vdual2(j)=c(i,j)-udual2(i);
end
end
end
for j=1:1
for i=1:m
if x(i,j)>0
iu=i;
if udual2(iu)~=inf
vdual2(j)=c(i,j)-udual2(iu);
else
if vdual2(j)~=inf
udual2(iu)=c(i,j)-vdual2(j);
end
end
end
end
end
for k=1:m*n
for i=1:m
if udual2(i)~=inf
iu=i;
for j=1:n
if x(iu,j)>0
vdual2(j)=c(iu,j)-udual2(iu);
end
end
end
end
for j=1:n
if vdual2(j)~=inf
jv=j;
for i=1:m
if x(i,jv)>0
                 udual2(i)=c(i,jv)-vdual2(jv);
             end
         end
     end
 end
 countu2=0;
 countv2=0;
 for i=1:m
     if udual2(i)<inf
      countu2=countu2+1;
     end
 end
 for j=1:n
     if vdual2(j)<inf
         countv2=countv2+1;
     end
 end
 if (countu2==m) & (countv2==n)
     break
 end
 end
%% Find the non-basic cells
  unx2=zeros(m,n);
  for j=1:n
      for i=1:m
          if x(i,j)==0
              unx2(i,j)=udual2(i)+vdual2(j)-c(i,j);
```

```matlab
        end
    end
end
%% Search maximum positive of udual+vdual-c(i,j) to reach a new basic variable
maxunx2=0;
for j=1:n
    for i=1:m
        if unx2(i,j)>=maxunx2
            maxunx2=unx2(i,j);
            imax2=i;
            jmax2=j;
        end
    end
end
 %% Count the number of  basic variable on each and row

   for j=1:n
       sumcol=0;
       for i=1:m
           if x(i,j)>0
               sumcol=sumcol+1;
           end
       end
           x(m+1,j)=sumcol;
   end
    for i=1:m
       sumrow=0;
       for j=1:n
           if x(i,j)>0
               sumrow=sumrow+1;
           end
       end
           x(i,n+1)=sumrow;
    end
      %% Construct the equivalent x matrix
      x12=zeros(m+1,n+1);
        for j=1:n+1
            for i=1:m+1
                x12(i,j)=x(i,j);
            end
        end
   %% Eliminate an entering variable for adding a new one
       for j=1:n
            for i=1:m
                if (x(i,j)==r|| x12(i,j)==r)

                    x12(i,j)=0;
                end
            end
       end
        % Add a new entering variable to x1 matrix
        for j=1:n
            for i=1:m

                x12(imax2,jmax2)=r;
            end
```

```
          end

 % Seaching and adding the entering point for corrective action
for i=1:m
if i~=imax2
if x1(i,jmax2)>0
ienter2=i;
for j=1:n
if j~=jmax2
if x1(ienter2,j)>0 && x1(imax2,j)==0
jenter2=j;
end
end
end
end
end
end
x1(imax2,jenter2)=r;
x(imax2,jenter2)=r;
end
end% End of degen2==1;
end% End of q=1:n;
%% Checking degeneracy
countxMODI2=0;
for j=1:n
for i=1:m
if x(i,j)>0
countxMODI2=countxMODI2+1;
end
end
end
%% Total cost
ZMODI=0;
for j=1:n
for i=1:m
if x(i,j)>0
ZMODI=ZMODI+x(i,j)*c(i,j);
end
end
end
if countxMODI2<numbasic
disp('Degeneracy in solution of MODI');
end
if maxunx==0
disp('Total cost of MODI');
disp(ZMODI);
disp('Occupied matrix of MODI');
disp(x);
else
disp('Reviewed Loop');
end
toc
```