Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 44 (2023), No. 2, pp. 255–270
DOI : 10.47974/JIOS-1273

An innovative way for solving transportation problem using modular arithmetic

Dhanashri A. Munot *
Department of Mathematics
SAJVPM'S Gandhi College
Kada. 414001 (MS)
India

Kirtiwant P. Ghadle †
Department of Mathematics
Dr. Babasaheb Ambedkar Marathawada University
Aurangabad, 431004 (MS)
India

Abstract

An innovative algorithm to solve TP using modular arithmetic which minimizes cost of transportation is described. To exhibit the potential use of this algorithm, examples are illustrated which shows that suggested algorithm is simpler and computationally more efficient than some existing methods frequently used in the literature. MATLAB coding of suggested algorithm is also provided.

Subject Classification: *[2010] 90B06.*

Keywords: *Congruence modulo, Cost minimization, MATLAB, Transportation problem.*

1. Introduction

Transportation Problem (TP) is an application of Linear Programming Problem (LPP) widely studied by the researchers of Operations Research [1]. In the literature, several methods are proposed to solve Transportation Problem which aims to minimize time or cost of transportation i.e. time required or cost required for transporting goods from resources to

* *E-mail:* dhnshrmunot50@gmail.com (Corresponding Author)
† *E-mail:* drkp.ghadle@gmail.com

destinations. Transporting goods from ware houses to different endpoints so that transportation cost should be minimum along with time is TP; first studied by Hitchcock F. [2] in 1941. Row minima, Row maxima, Matrix minima, North-West corner, Vogel's Approximation method etc. are some basic methods used to get Initial Basic Feasible Solution of TP. Many researchers have provided improved or different algorithm to solve TP. Charnes A. and Cooper W. [3] gave alternate way to simplex method through stepping stone method. Pandian P. *et al* [4] gave More-for-Less solution of TP with mixed constraints. Pandian P. *et al* [5] solved Transportation Problem through blocking point method. Mhlanga A. *et al* [6] gave an innovative approach to North-West Corner method. Sadeghi J. [7] proposed a new method which doesn't require to find path while solving TP and makes it easy to get solution compared to other methods found in literature.

Here, Author have suggested an innovative way for solving TP with Modular Arithmetic [8, 9, 10].

Definition

Congruence modulo:

"If two numbers x & y have the property that their difference is integrally divisible by a number m i.e. $m|(x-y)$ is an integer then we say that x is congruent to $y \bmod m$."

It means y is the remainder when x is divided by m.

This concept is used in proposed algorithm.

Proposed Ghadle-Munot algorithm to solve Transportation Problem:

I. Set up given problem in a matrix form.

II. For each row and column Calculate penalties by taking the difference of the least and succeeding highest cost available in that row/column. The penalty is zero, for a tie.

III. Calculate congruence *mod P* of all table entries, where P is maximum (minimum) entry among all penalties and represent it in the form $(r_s)^{q}{}_{s}$ where r_s is remainder and q_s is quotient *mod P*

IV. Select the row/column, corresponding to maximum (minimum) penalty (In particular, P for first time). In the selected row/column assign value to the cell having the least (high) cost by comparing quotients. Compare remainder if quotients are equal and select

minimum (maximum) among them. For equal penalties, select one where a row/column with minimum (maximum) unit cost. If there is again a tie, select one where maximum allocation can be made.

V. Row/column, satisfied with the supply and demand should be deleted.

VI. Repeat step (IV) until the entire supply and demands are satisfied.

Numerical Example:

Ex. 1 : Solve given TP to minimize the cost of transportation.

	M_1	M_2	M_3	M_4	Supply
W_1	19	30	50	10	7
W_2	70	30	40	60	9
W_3	40	8	70	20	18
Demand	5	8	7	14	

Step I : Set up given problem in a matrix form.

Step II: For each row and column Calculate penalties by taking the difference of the least and succeeding highest cost available in that row/column. The penalty is zero, for a tie.

Penalty		21	22	10	10	
		M_1	M_2	M_3	M_4	Supply
9	W_1	19	30	50	10	7
10	W_2	70	30	40	60	9
12	W_3	40	8	70	20	18
Demand		5	8	7	14	

Here maximum penalty is 22.

Step III: Calculate congruence *mod P* of all table entries, where *P* is maximum (minimum) entry among all penalties and represent it in the form $(r_s)^{q}{}_s$ where r_s is remainder and q_s is quotient *mod P*

Penalty		21	22	10	10	
		M_1	M_2	M_3	M_4	Supply
9	W_1	19^0	8^1	6^2	10^0	7
10	W_2	4^3	8^1	18^1	16^2	9
12	W_3	18^1	8^0	4^3	20^0	18
Demand		5	8	7	14	

Table Congruence Modulo 22

Step IV : Select the row/column, corresponding to maximum (minimum) penalty (In particular, *P* for first time). In the selected row/column assign value to the cell having the least (high) cost by comparing quotients. Compare remainder if quotients are equal and select minimum (maximum) among them. For equal penalties, select one where a row/column with minimum (maximum) unit cost. If there is again a tie, select one where maximum allocation can be made.

Penalty		21	22	10	10		
		M_1	M_2	M_3	M_4	Supply	Allocation Process
9	W_1	19^0	8^1	6^2	10^0	7	
10	W_2	4^3	8^1	18^1	16^2	9	
12	W_3	18^1	$[8^0]^{(8)}$	4^3	20^0	~~18~~ 10	
Demand		5	~~8~~	7	14		

Step V : Row/column, satisfied with the supply and demand should be deleted. Repeat step IV until the entire supply and demands are satisfied.

Penalty		21	22	10	10			
		M_1	M_2	M_3	M_4	Supply		
9	W_1	$[19^0]^{(5)}$	8^1	6^2	$[10^0]^{(2)}$	~~7~~	~~2~~	00
10	W_2	4^3	8^1	$[18^1]^{(7)}$	$[16^2]^{(2)}$	~~9~~	~~2~~	00
12	W_3	18^1	$[8^0]^{(8)}$	4^3	$[20^0]^{(10)}$	~~18~~	~~10~~	00
Demand		~~5~~	~~8~~	~~7~~	~~14~~			
		00	00	00	~~04~~			

Answer:

19*5 + 10*2 + 40*7 + 60*2 + 8*8 + 20*10 = 779

Ex. 2 : Solve given TP to minimize the cost of transportation

	M_1	M_2	M_3	M_4	Supply
W_1	11	13	17	14	250
W_2	16	18	14	10	300
W_3	21	24	13	10	400
Demand	200	225	275	250	

Given problem is set up in tabular form. Now following the algorithm from step II to step V, we have final allocation as below.

Penalty		5	5	1	0			
		M_1	M_2	M_3	M_4	Supply		
2	W_1	$^{(200)}[1^2]$	$^{(50)}[3^2]$	2^3	4^2	~~250~~	~~50~~	00
4	W_2	1^3	$^{(175)}[3^3]$	4^2	$^{(125)}[0^2]$	~~300~~	~~125~~	00
3	W_3	1^4	4^4	$^{(275)}[3^2]$	$^{(125)}[0^2]$	~~400~~	~~275~~	00
Demand		~~200~~	~~225~~	~~275~~	~~250~~			
		00	~~175~~	00	~~125~~			
		00		00				

Answer:
11*200+13*50+18*175+10*125+13*275+10*125=12075

Comparative Study:

Same examples are solved with North-West Corner method, Least cost method, Vogel's Approximation Method and their answers are compared.

Table 1

Comparison of proposed method with some well-known methods

Method	Example 1	Example 2
By N-W corner method	1015	12200
By Least Cost Method	814	12200
By Maxima – Minima method	814	12200
By Vogel's Approximation Method	779	12075
By Proposed Method	779	12075

Result:

Comparative study (Table 1) shows that proposed method gives better solution than North-West Corner method, Least Cost method, Maxima-Minima method and answer is same as Vogel's Approximation Method but proposed method requires less iterations compared to VAM which saves time of calculation.

Conclusion

In Proposed method congruence mod to penalty is calculated for whole matrix which is very new concept, it requires less iterations.

Numerical examples have been presented for demonstrating the algorithm. MATLAB coding of given algorithm has been done which makes it user friendly. Further this concept can be extended to Assignment problem, Transshipment problem etc.

Appendix:

MATLAB coding of Proposed Algorithm

```
% % Example :
inpMat = [19 30 50 10;70 30 40 60;40 8 70 20] - input matrix \\
demand = [5 8 7 14]\\
supply = [7 9 18]\\
sizeMat = size(inpMat);\\
noOfRows = sizeMat(1);\\
noOfCol = sizeMat(2);\\
countRow = 0;\\
countCol = 0;\\
countRem = 0;\\
diffRowVal = zeros(noOfRows,1);\\
diffColVal = zeros(noOfCol,1);\\
remVal = zeros(sizeMat);\\
demSupply = 0;\\
orgDemand = demand;\\
orgSupply = supply;\\
tempStrRowVal = 0;\\
tempStrColVal = 0;\\
checkDemSup = 0;\\
counter = 0;\\
dummyRow = 1:noOfRows;\\
dummyCol = 1:noOfCol;\\
effCost = 0;\\
for countRow = 1:noOfRows\\
getUniRowVal = find(min(inpMat(countRow,:)) == inpMat(countRow,:));\\
   if length(getUniRowVal)~= 1\\
      diffRowVal(countRow)= 0;\\
   else\\
      temp = unique(inpMat(countRow,:));\\
      diffRowVal(countRow) = temp(2) - temp(1);\\
   end\\
end\\
for countCol = 1:noOfCol\\
   getUniColVal = find(min(inpMat(:,countCol)) == inpMat(:,countCol));\\
   if length(getUniColVal)~= 1\\
      diffColVal(countCol)= 0;\\
```

```
    else \\
        temp = unique(inpMat(:,countCol));\\
        diffColVal(countCol) = temp(2) - temp(1);\\
    end\\
end\\
convergVal = max([diffRowVal;diffColVal]);\\
quoval = fix(inpMat./convergVal);\\
for countRem = 1:noOfRows\\
    for countCol = 1:noOfCol\\
        remVal(countRem,countCol) = rem(inpMat(countRem,countCol),convergVal);\\
    end\\
end\\

minCountVal= min(noOfRows,noOfCol);\\
if sum(demand) == sum(supply)\\
    stopCount = abs(length(demand) - length(supply)) + 1;\\
else\\
    stopCount = abs(length(demand) - length(supply)) + 2;\\
end\\
for counter = 1:minCountVal\\
    convergVal = max([max(diffRowVal);max(diffColVal)]);\\
    convergValRow = find(convergVal == diffRowVal);\\
    convergValCol = find(convergVal == diffColVal);\\
        if ~isempty(convergValRow) && ~isempty(convergValCol)\\
            minSupply = min(supply(convergValRow));\\
            minDemand = min(demand(convergValCol));\\
            minCost = min(minSupply,minDemand);\\
            if minCost == minDemand\\
                checkDemSup =1; \\
                idxAssetVal = find(minCost == demand);\\
            else\\
                checkDemSup = 2;\\
                idxAssetVal = find(minCost == supply);\\
            end\\
        elseif ~isempty(convergValCol)\\
                checkDemSup =1;\\
            if length(convergValCol) > 1\\
                idxAssetVal = find(min(demand(convergValCol)) == demand);\\
            else\\
                idxAssetVal = find(demand(convergValCol) == demand);\\
            end\\
        else\\
            checkDemSup = 2;\\
            if length(convergValRow) > 1\\
```

```
            idxAssetVal = find(min(supply(convergValRow)) == supply);\\
        else\\
            idxAssetVal = find(supply(convergValRow) == supply);\\
        end\\
    end \\
switch checkDemSup\\
case 1\\
checkflag = 1;\\
idxLeastQuoVal = find(min(quoval(:,idxAssetVal))== quoval(:,idxAssetVal));\\
idxDmdDone = idxLeastQuoVal;\\
  if length(idxLeastQuoVal)== 1\\
    if demand(idxAssetVal)==supply(idxLeastQuoVal)\\
  idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
    effCost = effCost + supply(idxLeastQuoVal)*inpMat(idxLeastQuoVal,idxAssetVal);\\
    idxDmdDone = idxLeastQuoVal;\\
    idxSupplyDone = idxDmdDone;\\
    idxDemandDone = idxAssetVal;\\
    demandLeft = 0;\\
    penality = 0;\\
  elseif demand(idxAssetVal)>supply(idxLeastQuoVal)\\
    idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
    effCost = effCost + supply(idxLeastQuoVal)*inpMat(idxLeastQuoVal,idxAssetVal);\\
    idxDmdDone = idxLeastQuoVal;\\
    idxSupplyDone = idxDmdDone;\\
    idxDemandDone = idxAssetVal;\\
    demandLeft = 1;\\
    penality = demand(idxAssetVal) - supply(idxLeastQuoVal);\\
  else
    idxLeastRemVal= find(min(quoval(:,idxAssetVal))==quoval(:,idxAssetVal));\\
    effCost = effCost + demand(idxAssetVal)*inpMat(idxLeastQuoVal,idxAssetVal);\\
    demandLeft = 2;\\
    penality = supply(idxLeastQuoVal) - demand(idxAssetVal);\\
    idxDmdDone = idxAssetVal;\\
    idxSupplyDone = idxLeastQuoVal;\\
    idxDemandDone = idxAssetVal;\\
  end\\
else \\
  idxLeastRemVal= find(min(remVal(:,idxAssetVal))==remVal(:,idxAssetVal));\\
                idxDmdDone = idxLeastRemVal;\\
  if length(idxLeastRemVal) == 1\\
    if demand(idxAssetVal)==supply(idxLeastRemVal)\\
  idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
    effCost = effCost + supply(idxLeastRemVal)*inpMat(idxLeastRemVal,idxAssetVal);\\
    idxDmdDone = idxLeastRemVal;\\
```

```
        demandLeft = 0;\\
        penality = 0;\\
        idxSupplyDone = idxDmdDone;\\
        idxDemandDone = idxAssetVal;\\
    elseif demand(idxAssetVal) > supply(idxLeastRemVal)\\
idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
    effCost = effCost + supply(idxLeastRemVal)*inpMat(idxLeastRemVal,idxAssetVal);\\
idxDmdDone = idxLeastRemVal;\\
    demandLeft = 1;\\
    penality = demand(idxAssetVal) - supply(idxLeastRemVal);\\
    idxSupplyDone = idxDmdDone;\\
    idxDemandDone = idxAssetVal;\\
    else
        idxLeastRemVal= find(min(quoval(:,idxAssetVal))==quoval(:,idxAssetVal));\\
        effCost = effCost + demand(idxAssetVal)*inpMat(idxLeastRemVal,idxAssetVal);\\
        demandLeft = 2;\\
        penality = supply(idxLeastRemVal) - demand(idxAssetVal);\\
        idxSupplyDone = idxLeastRemVal;\\
        idxDemandDone = idxAssetVal;\\
                idxDmdDone = idxLeastRemVal;\\
        end\\
    else
        effCost = effCost + supply(idxLeastRemVal)*inpMat(idxLeastRemVal,idxAssetVal);\\
        idxDmdDone = idxLeastRemVal;\\
        demandLeft = 0;\\
        penality = demand(idxAssetVal) - supply(idxLeastRemVal);\\
        idxSupplyDone = idxDmdDone;\\
        idxDemandDone = idxAssetVal;\\
        end\\
end\\
switch demandLeft \\
case 1\\
if counter == 1\\
idxDmdNotDone = find(idxDmdDone ~= dummyRow);\\
else
idxDmdNotDone = find(idxDmdDone ~= 1:length(getNewMatRowIdx));\\
end\\
newidxLeastQuoVal = find(min(quoval(idxDmdNotDone,idxAssetVal))==quoval(idxDmdNotDone,i
dxAssetVal));\\
if length(newidxLeastQuoVal)== 1\\
while penality ~=0\\
effCost = effCost + penality *inpMat(idxDmdNotDone(newidxLeastQuoVal),idxAssetVal);\\
supply(idxDmdNotDone(newidxLeastQuoVal)) = abs(supply(idxDmdNotDone(newidxLeastQuoV
al)) - penality);\\
penality = demand(idxAssetVal) - (supply(idxDmdDone) + penality);\\
```

```
if penality <= 0\\
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newidxLeastQuoVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newidxLeastQuoVal = min(find(newidxLeastQuoVal~= idxDmdNotDone));\\
idxDmdNotDone(newidxLeastQuoVal) = find(idxDmdNotDone(newidxLeastQuoVal) ~=
idxsuppNotDone);\\
end\\
else\\
newIdxLeastRemVal=find(min(remVal(idxDmdNotDone,idxAssetVal))== remVal(idxDmdNotDone,i
dxAssetVal));\\
while penality ~=0
effCost = effCost + penality *inpMat(idxDmdNotDone(newIdxLeastRemVal),idxAssetVal);\\
supply(idxDmdNotDone(newIdxLeastRemVal)) = abs(supply(idxDmdNotDone(newIdxLeastRemV
al)) - penality);\\
penality = demand(idxAssetVal) - (supply(idxDmdDone) + penality);\\
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newIdxLeastRemVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newIdxLeastRemVal = min(find(newIdxLeastRemVal~= idxDmdNotDone));\\
end\\
end\\
case 2\\
if counter == 1\\
idxDmdNotDone = find(idxDmdDone ~= dummyCol);\\
else\\
idxDmdNotDone = find(idxDmdDone ~= 1:length(getNewMatColIdx));\\
end\\
newidxLeastQuoVal = find(min(quoval(idxLeastQuoVal,idxDmdNotDone))==quoval(idxLeastQuoVa
l,idxDmdNotDone));\\
if length(newidxLeastQuoVal)== 1\\
while penality ~=0\\
effCost = effCost + penality *inpMat(idxLeastQuoVal,idxDmdNotDone(newidxLeastQuoVal));\\
demand(idxDmdNotDone(newidxLeastQuoVal)) = abs(demand(idxDmdNotDone(newidxLeastQuo
Val)) - penality);\\
penality = supply(idxLeastQuoVal) - (demand(idxDmdDone) + penality);\\
if penality <= 0\\
break\\
end\\
```

```
idxDmdNotDone = find(idxDmdNotDone(newidxLeastQuoVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newidxLeastQuoVal = min(find(newidxLeastQuoVal~= idxDmdNotDone));\\
idxDmdNotDone(newidxLeastQuoVal) = find(idxDmdNotDone(newidxLeastQuoVal) ~=
idxsuppNotDone);\\
end\\
else\\
newIdxLeastRemVal = find(min(remVal(idxLeastQuoVal,idxDmdNotDone(newidxLeastQuoVal)))==
remVal(idxLeastQuoVal,idxDmdNotDone(newidxLeastQuoVal)));\\
while penality ~=0\\

effCost = effCost + penality *inpMat(idxLeastQuoVal,idxDmdNotDone(newIdxLeastRemVal));\\
demand(idxDmdNotDone(newIdxLeastRemVal)) = abs(demand(idxDmdNotDone(newIdxLeastRem
Val)) - penality);\\
penality = supply(idxLeastQuoVal) - (demand(idxDmdDone) + penality);\\
if penality <= 0\\
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newIdxLeastRemVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newIdxLeastRemVal = min(find(newIdxLeastRemVal~= idxDmdNotDone));\\
end\\
end\\
end \\
case 2 \\
checkflag = 2;\\
idxLeastQuoVal = find(min(quoval(idxAssetVal,:))== quoval(idxAssetVal,:));\\
        idxDmdDone = idxLeastQuoVal;\\
        if length(idxLeastQuoVal)== 1 \\
if supply(idxAssetVal)==demand(idxLeastQuoVal)\\
            idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
effCost = effCost + demand(idxLeastQuoVal)*inpMat(idxAssetVal,idxLeastQuoVal);\\
idxDmdDone = idxLeastQuoVal;\\
supplyLeft = 0;\\
penality = 0;\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxAssetVal;\\

elseif supply(idxAssetVal) > demand(idxLeastQuoVal)\\
            idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
effCost = effCost + demand(idxLeastQuoVal)*inpMat(idxLeastQuoVal,idxAssetVal);\\
```

```
idxDmdDone = idxLeastQuoVal;\\
supplyLeft = 1;\\
penality = supply(idxAssetVal) - demand(idxLeastQuoVal);\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxAssetVal;\\
supply(idxAssetVal) = penality;\\
else\\
        idxLeastRemVal= find(min(quoval(:,idxAssetVal))==quoval(:,idxAssetVal));\\
effCost = effCost + supply(idxAssetVal)*inpMat(idxAssetVal,idxLeastQuoVal);\\ supplyLeft = 2;\\
penality = demand(idxLeastQuoVal) - supply(idxAssetVal);\\
idxDmdDone = idxAssetVal;\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxLeastQuoVal;\\
demand(idxLeastQuoVal) = penality;\\
end\\
else \\
idxLeastRemVal= find(min(remVal(idxAssetVal,:))==remVal(idxAssetVal,:));\\
idxDmdDone = idxLeastRemVal;\\
if length(idxLeastRemVal) == 1 \\
if supply(idxAssetVal)==demand(idxLeastRemVal)\\
        idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
effCost = effCost + demand(idxLeastRemVal)*inpMat(idxAssetVal,idxLeastRemVal);\\
                idxDmdDone = idxLeastRemVal;\\
supplyLeft = 0;\\
penality = 0;\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxAssetVal;\\
elseif supply(idxAssetVal) > demand(idxLeastRemVal)
      idxLeastQuoVal = find(max(remVal(:,idxAssetVal))== remVal(:,idxAssetVal));\\
effCost = effCost + demand(idxLeastRemVal)*inpMat(idxAssetVal,idxLeastRemVal);\\
          idxDmdDone = idxLeastRemVal;\\
supplyLeft = 1;\\
penality = supply(idxAssetVal) - demand(idxLeastRemVal);\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxAssetVal;\\
else\\
      idxLeastRemVal= find(min(quoval(:,idxAssetVal))==quoval(:,idxAssetVal));\\
effCost = effCost + supply(idxAssetVal)*inpMat(idxAssetVal,idxLeastRemVal);\\ supplyLeft = 2;\\
penality = demand(idxLeastRemVal) - supply(idxAssetVal);\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxLeastRemVal;\\
                idxDmdDone = idxLeastRemVal;\\
end\\
else\\
```

```
effCost = effCost + demand(idxLeastRemVal)*inpMat(idxAssetVal,idxLeastRemVal);\\
idxDmdDone = idxLeastRemVal;\\
supplyLeft = 0;\\
penality = supply(idxAssetVal) - demand(idxLeastRemVal);\\
idxSupplyDone = idxDmdDone;\\
idxDemandDone = idxAssetVal;\\
end\\
end\\
switch supplyLeft\\
case 1\\
if counter == 1\\
idxDmdNotDone = find(idxDmdDone ~= dummyCol);\\
else\\
idxDmdNotDone = find(idxDmdDone ~= 1:length(getNewMatColIdx));\\
end\\
newidxLeastQuoVal = find(min(quoval(idxAssetVal,idxDmdNotDone))==quoval(idxAssetVal,idxDm
dNotDone));\\
if length(newidxLeastQuoVal)== 1\\
while penality ~=0\\
if penality > demand(idxDmdNotDone(newidxLeastQuoVal))\\
effCost = effCost + demand(idxDmdNotDone(newidxLeastQuoVal))\\ *inpMat(idxAssetVal,idxDmd
NotDone(newidxLeastQuoVal));\\
penality = penality - demand(idxDmdNotDone(newidxLeastQuoVal));\\
demand(idxDmdNotDone(newidxLeastQuoVal)) = 0;\\
else\\
effCost = effCost + penality *inpMat(idxAssetVal,idxDmdNotDone(newidxLeastQuoVal));\\
demand(idxDmdNotDone(newidxLeastQuoVal)) = demand(idxDmdNotDone(newidxLeastQuoV
al)) - penality;\\
penality = penality - supply(idxAssetVal);\\
end\\
supply(idxAssetVal) = penality;\\
if penality <= 0\\
break\\
end\\
newidxLeastQuoVal = find(idxDmdNotDone(newidxLeastQuoVal+1) == idxDmdNotDone);\\
end\\
else\\
newIdxLeastRemVal = find(min(remVal(idxAssetVal,idxDmdNotDone))== remVal(idxAssetVal,idxD
mdNotDone));\\
while penality ~=0\\

effCost = effCost + penality *inpMat(idxAssetVal,idxDmdNotDone(newIdxLeastRemVal));\\
demand(idxDmdNotDone(newIdxLeastRemVal)) = abs(demand(idxDmdNotDone(newIdxLeastRem
Val)) - penality);\\
penality = supply(idxAssetVal) - penality;\\
```

```
if penality <= 0\\
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newIdxLeastRemVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newIdxLeastRemVal = min(find(newIdxLeastRemVal~= idxDmdNotDone));\\
end\\
end\\
case 2\\
if counter == 1\\
idxDmdNotDone = find(idxDmdDone ~= dummyRow);\\
else\\
idxDmdNotDone = find(idxDmdDone ~= 1:length(getNewMatRowIdx));\\
end\\
newidxLeastQuoVal = find(min(quoval(idxDmdNotDone,idxLeastQuoVal))==quoval(idxDmdNotDo
ne,idxLeastQuoVal));\\
if length(newidxLeastQuoVal)== 1\\
while penality ~=0\\
effCost = effCost + penality *inpMat(idxDmdNotDone(newidxLeastQuoVal),idxLeastQuoVal);\\
supply(idxDmdNotDone(newidxLeastQuoVal)) = abs(supply(idxDmdNotDone(newidxLeastQuoV
al)) - penality);\\
penality = demand(idxLeastQuoVal) - penality;\\
if penality <= 0\\
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newidxLeastQuoVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newidxLeastQuoVal = min(find(newidxLeastQuoVal~= idxDmdNotDone));\\
idxDmdNotDone(newidxLeastQuoVal) = find(idxDmdNotDone(newidxLeastQuoVal) ~=
idxsuppNotDone);\\
end\\
else\\
newIdxLeastRemVal = find(min(remVal(idxAssetVal,newidxLeastQuoVal))== remVal(idxAssetVal,ne
widxLeastQuoVal));\\
while penality ~=0\\

effCost = effCost + penality *inpMat(idxAssetVal,idxDmdNotDone(newIdxLeastRemVal));\\
supply(idxDmdNotDone(newIdxLeastRemVal)) = abs(supply(idxDmdNotDone(newIdxLeastRemV
al)) - penality);\\
penality = demand(idxLeastQuoVal) - penality;\\
if penality <= 0\\
```

```
break\\
end\\
idxDmdNotDone = find(idxDmdNotDone(newIdxLeastRemVal) ~= idxDmdNotDone);\\
if isempty(idxDmdNotDone)\\
break\\
end\\
newIdxLeastRemVal = min(find(newIdxLeastRemVal~= idxDmdNotDone));\\
end\\
end\\
end\\
end\\
if counter == 1\\
getNewMatRowIdx = find(idxSupplyDone ~= dummyRow);\\
getNewMatColIdx = find(idxDemandDone ~= dummyCol);\\
else\\
if length(getNewMatRowIdx) ~= length(idxSupplyDone) && length(getNewMatColIdx) ~=
length(idxDemandDone)\\
getNewMatRowIdx = find(getNewMatRowIdx(idxSupplyDone) ~= getNewMatRowIdx);\\
getNewMatColIdx = find(getNewMatColIdx(idxDemandDone) ~= getNewMatColIdx);\\
else\\
break\\
end\\
end\\
if isempty(getNewMatRowIdx)\\
getNewMatRowIdx = idxSupplyDone;\\
break\\
end\\
if isempty(getNewMatColIdx)\\
getNewMatRowIdx = idxDemandDone;\\
break\\
end\\
if counter <= stopCount\\
supply = supply(getNewMatRowIdx);\\
demand = demand(getNewMatColIdx);\\
inpMat = inpMat(getNewMatRowIdx,getNewMatColIdx);\\
quoval = quoval(getNewMatRowIdx,getNewMatColIdx);\\
remVal = remVal(getNewMatRowIdx,getNewMatColIdx);\\
diffRowVal = diffRowVal(getNewMatRowIdx);\\
diffColVal = diffColVal(getNewMatColIdx);\\
end\\
end\\
display([‘Allocation is = ‘,num2str(effCost)]);\\

\end{document}
```

References

[1] Gupta P, Hira D, Operations Research. S. Chand and Co., Ltd. New Delhi, 2009.

[2] Hitchcock F. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Phyics*. 1941; 20: 224–230.

[3] Charnes A, Cooper W. The stepping-stone method for explaining linear programming calculation in transportation problem. Management Sci. 1.19542 49–69.

[4] Pandian P. Natarajan G. A new method for finding an optimal solution for transportation problems. *International Journal of Mathematical Sciences & Engineering Applications*. (IJMSEA), 2010; 4: 59-65.

[5] Pandian P. Natarajan G. A New Method for Solving Bottleneck-Cost Transportation Problems. *International Mathematical Forum*, 2011; 6(10):451 – 460.

[6] Mhlanga A, Nduna I, Matarise F, Machisvo A Innovative application of Dantzig's North-West Corner rule to solve a transportation problem. *International Journal of Education and Research.* 2014;2 (2):1–12.

[7] Sadeghi J. A method for solving the transportation problem, *Journal of Statistics and Management Systems*, 2018; 21(5):817-837, DOI: 10.1080/09720510.2018.1453682

[8] Ghadle K, Munot D. A New approach to solve Assignment Problem and its coding in MATLAB. *Advances In Mathematics: Scientific Journal.* 2020; 9: 9551-9557.

[9] Ghadle K, Munot D. A freezing method for solving Bottleneck-Cost transportation problem. *Turkish Journal Of Computer and Mathematics Education.* 2021; 12: 3551-3556

[10] Gallian J. Contempary Abstract Algebra. The mathematical gazette, 1991.

Received February, 2022

Revised July, 2022