Operations Research Lab

Chetan Gupta
DTU/2K15/CO/044

INDEX

Sno	**Aim**	**Sign**
1	Solve using graphical method.	
2	Implement simplex algorithm in MATLAB	
3	Implement BIGM method in MATLAB	
4	Sensitivity analysis using MATLAB.	
5	Integer Programming using MATLAB	
6	Mixed Integer Programming using MATLAB	
7	Transportation problem using MATLAB	

PROBLEM 1

MAXIMISE : 3x1 + 8x2

Constraints

- **x1 + x2 >= 8**
- **2x1 – 3x2 <= 0**
- **x1 + 2x2 <= 30**
- **3x1 – x2 >= 0**
- **x1 <= 10**
- **x2 >= 9**
- **x1,x2 >=0**

TORA SOLUTION:

*** OPTIMUM SOLUTION SUMMARY ***

Title: LPP Graphical Solution
Final iteration No: 3
Objective value (min) = 81.0000

Variable	Value	Obj Coeff	Obj Val Contrib
x1	3.0000	3.0000	9.0000
x2	9.0000	8.0000	72.0000

Constraint	RHS	Slack(-)/Surplus(+)
1 (>)	8.0000	4.0000+
2 (<)	0.0000	21.0000-
3 (<)	30.0000	9.0000-
4 (>)	0.0000	21.0000+
5 (<)	10.0000	7.0000-
6 (>)	9.0000	9.0000+

*** SENSITIVITY ANALYSIS ***

Objective coefficients -- Single Changes:

Variable	Current Coeff	Min Coeff	Max Coeff	Reduced Cost
x1	3.0000	0.0000	infinity	0.0000
x2	8.0000	-1.0000	infinity	0.0000

Right-hand Side -- Single Changes:

Constraint	Current RHS	Min RHS	Max RHS	Dual Price
1 (>)	8.0000	-infinity	12.0000	0.0000
2 (<)	0.0000	-21.0000	infinity	0.0000
3 (<)	30.0000	21.0000	infinity	0.0000
4 (>)	0.0000	-9.0000	21.0000	1.0000
5 (<)	10.0000	3.0000	infinity	0.0000
6 (>)	9.0000	6.0000	12.8571	9.0000

Objective Coefficients -- Simultaneous Changes d:

Nonbasic Var	Optimality Condition
Sx6	-1.0000 + -0.3333 d1 <= 0
Sx8	-9.0000 + -0.3333 d1 + -1.0000 d2 <= 0

Right-hand Side Ranging -- Simultaneous Changes D:

Basic Var	Value/Feasibilty Condition
Sx3	4.0000 + -1.0000 D1 + 0.3333 D4 + 1.3333 D6 >= 0
sx4	21.0000 + 1.0000 D2 + -0.6667 D4 + 2.3333 D6 >= 0
sx5	9.0000 + 1.0000 D3 + -0.3333 D4 + -2.3333 D6 >= 0
x1	3.0000 + 0.3333 D4 + 0.3333 D6 >= 0
sx7	7.0000 + -0.3333 D4 + 1.0000 D5 + -0.3333 D6 >= 0
x2	9.0000 + 1.0000 D6 >= 0

End of Solution Summary

MATLAB SOLUTION :

CODE:

```
c = [3 8]';
A = [1 1; 2 -3; 1 2; 3 -1; 1 0; 0 1];
b = [8; 0; 30; 0; 10; 9];
[x_min z_min] = glpk(c, A, b, [0; 0], [], 'LUULUL', 'CC', 1);
printf('x1 = %d, x2 = %d\n', x_min);
printf('Solution = %d\n', z_min);
```

OUTPUT:

```
x1 = 3, x2 = 9
Solution = 81
```

Problem 2

$$Min\ Z\ =\ -3x1 + x2\ + x3$$
$$subject\ to\ :$$
$$x1\ -\ 2x2\ +\ x3\ <=\ 11$$
$$-4x1\ +\ x2\ +\ x3\ >=\ 3$$
$$-2x1\ +\ x3\ =\ 1$$
$$x1,\ x2,\ x3\ >=\ 0$$

Converting to a maximisation problem and keeping constraints the same:

$$Max\ -Z\ =\ 3x1\ -\ x2\ -\ x3$$

Now using the format:

Max Z = CX

Subject to :

AX <= B

Here

C = [3 -1 -1]';

X = [x1 x2 x3];

A = [1 -2 1; -4 1 1; -2 0 1];

B = [11; 3; 1];

```
function [z_max, x_max, status] = simplex_lp_solver(c, A, b, maxiter=100)
      [T, BV] = first_simplex_tableau(c, A, b);
      status = 'unknown';
      iter = 1;
      while(iter < maxiter && !strcmp(status, 'unbounded') &&
!strcmp(status, 'optimal'))
            fprintf('Iteration %d : \n', iter);
            disp(T);
            fprintf('\n');
            [T, BV, status] = new_tableau(T, BV, c);
            iter = iter + 1;
      end;
      if(iter >= maxiter || strcmp(status, 'unbounded'))
            z_max = 0;
            x_max = zeros(length(c), 1);
            return;
      end;
```

```
z_max = T(1:end, 1)' * T(1:end, columns(T));
x_max = zeros(length(c) + length(b), 1);
x_max(BV) = T(2 : (length(b) + 1), columns(T));
x_max = x_max(1 : length(c));
fprintf('Maximum Value = %d \n', z_max);
fprintf('Optimal Position : ');
for i=1:length(x_max)
      fprintf('x%d = %d, ', i, x_max(i));
end

function [T, BV] = first_simplex_tableau(c, A, b)
      [m,n] = size(A);
      T = [1 c' zeros(1, m) 0;
            zeros(m, 1) A eye(m) b];
      BV = ( (n+1):(n+m) )';
end

function [T, BV, status] = new_tableau(T, BV)
      status = 'unknown';
      B = T(2:end, columns(T):columns(T));
      CB = T(2:end, 1:1);
      Aij = T(2:end, 2:columns(T)-1);
      Cj = T(1:1, 2:columns(T)-1);
      Zj = CB'*Aij;
      Cbar = Cj - Zj;

      if( all(Cbar <= 0) )
            status = 'optimal';
            return;
      end

      [_, enteringVariable] = max(Cbar);
      keyColumn = T(2:end, 1+enteringVariable:1+enteringVariable);
      if(all(keyColumn <= 0))
            status = 'unbounded';
            return;
      end
      minRatio = 10000000;
      leavingVariable = 0;
      for i=1:length(keyColumn)
            if(keyColumn(i) > 0)
                  curRatio = B(i)/keyColumn(i);
```

```
                        if(curRatio < minRatio)
                            minRatio = min(curRatio, minRatio);
                            leavingVariable = i;
                        end
                    end
                end
                pivot = T(leavingVariable+1, enteringVariable+1);
                keyRow = T(leavingVariable+1, 2:columns(T))/ pivot;
                BV(leavingVariable) = enteringVariable;
                pivotThing = repmat(keyColumn, 1,length(keyRow)) .*
repmat(keyRow, length(keyColumn), 1);
                newT = T;
                newT(2:end, 2:end) = T(2:end, 2:end) - pivotThing;
                newT(leavingVariable+1, 2:end) = keyRow;
                newT(leavingVariable+1, 1) = Cj(enteringVariable);
                T = newT;
            end
end
```

On solving using BigM method :

```
octave:1> c = [3;-1;-1];
octave:2> a = [1 -2 1; -4 1 1; -2 0 1];
octave:3> b = [11; 3; 1];
octave:4> ind = [-1;1;0];
octave:5> simplexBIGM_lp_solver(c, a, b, ind)
Iteration 1 :
       1       3      -1      -1       0       0  -10000  -10000       0
       0       1      -2       1       1       0       0       0      11
  -10000      -4       1       1       0      -1       1       0       3
  -10000      -2       0       1       0       0       0       1       1

Iteration 2 :
       1       3      -1      -1       0       0  -10000  -10000       0
       0       3      -2       0       1       0       0      -1      10
  -10000      -2       1       0       0      -1       1      -1       2
      -1      -2       0       1       0       0       0       1       1

Iteration 3 :
       1       3      -1      -1       0       0  -10000  -10000       0
       0      -1       0       0       1      -2       2      -3      14
      -1      -2       1       0       0      -1       1      -1       2
      -1      -2       0       1       0       0       0       1       1

Maximum Value = -3
Optimal Position : x1 = 0, x2 = 2, x3 = 1,
```

Problem 3

$Max\ Z\ =\ 4x1\ +\ 3\,x2$

$subject\ to$

$3x1\ +\ 4x2\ <=\ 12$

$4x1\ +\ 2x2\ <=\ 8$

$x1\ +\ x2\ =\ 4$

Now using the format:

Max Z = CX

Subject to :

AX <= B

Here

C = [4 3]';

X = [x1 x2];

A = [3 4; 4 2; 1 1];

B = [12; 8; 4];

```
function [z_max, x_max, status] = simplexBIGM_lp_solver(c, A, b, ind
,maxiter=100)
      [T, BV] = first_simplex_tableau(c, A, b, ind);
      status = 'unknown';
      iter = 1;
      while(iter < maxiter && !strcmp(status, 'unbounded') &&
!strcmp(status, 'optimal') && !strcmp(status, 'infeasable'))
            fprintf('Iteration %d : \n', iter);
            disp(T);
            fprintf('\n');
            [T, BV, status] = new_tableau(T, BV);
            iter = iter + 1;
      end;
      if(strcmp(status, 'infeasable'))
            fprintf('Infeasible\n');
      else
            if(iter >= maxiter || strcmp(status, 'unbounded'))
                  z_max = 0;
                  x_max = zeros(length(c), 1);
                  return;
            end;
```

```
        z_max = T(1:end, 1)' * T(1:end, columns(T));
        x_max = zeros(length(c) + length(b), 1);
        x_max(BV) = T(2 : (length(b) + 1), columns(T));
        x_max = x_max(1 : length(c));
        fprintf('Maximum Value = %d \n', z_max);
        fprintf('Optimal Position : ');
        for i=1:length(x_max)
                fprintf('x%d = %d, ', i, x_max(i));
        end
        fprintf('\n');
endif

function [T, BV] = first_simplex_tableau(c, A, b, ind)
        [m,n] = size(A);
        index = 1;
        addedCost = [];
        BV = [];
        XB = [];
        augA = A;
        M = 1000;
        for i=1:length(ind)
                switch ind(i)
                case -1
                        addedCost(index) = 0;
                        XB(i) = 0;
                        BV(i) = index+m;
                        temp = zeros(m, 1);
                        temp(i) = 1;
                        A(:, n+index:n+index) = temp;
                        index+=1;
                case 0
                        addedCost(index) = -M;
                        XB(i) = -M;
                        BV(i) = index+m;
                        temp = zeros(m, 1);
                        temp(i) = 1;
                        A(:, n+index:n+index) = temp;
                        index+=1;
                case 1
                        addedCost(index) = 0;
                        addedCost(index+1) = -M;
                        XB(i) = -M;
```

```
                BV(i) = index+m+1;
                temp = zeros(m, 1);
                temp(i) = -1;
                A(:, n+index:n+index) = temp;
                temp(i) = 1;
                A(:, n+index+1:n+index+1) = temp;
                index+=2;
            end
        end
        T = [1 c' addedCost 0;
            XB' A b];
end

function [T, BV, status] = new_tableau(T, BV)
        status = 'unknown';
        B = T(2:end, columns(T):columns(T));
        CB = T(2:end, 1:1);
        Aij = T(2:end, 2:columns(T)-1);
        Cj = T(1:1, 2:columns(T)-1);
        Zj = CB'*Aij;
        Cbar = Cj - Zj;

        if( all(Cbar <= 0) )
            if(all(T(:, 1) > -1000))
                status = 'optimal';
            else
                status = 'infeasable';
            endif
            return;
        end

        [_, enteringVariable] = max(Cbar);
        keyColumn = T(2:end, 1+enteringVariable:1+enteringVariable);
        if(all(keyColumn <= 0))
            status = 'unbounded';
            return;
        end
        minRatio = 10000000;
        leavingVariable = 0;
        for i=1:length(keyColumn)
            if(keyColumn(i) > 0)
                curRatio = B(i)/keyColumn(i);
```

```
                    if(curRatio < minRatio)
                        minRatio = min(curRatio, minRatio);
                        leavingVariable = i;
                    end
                end
            end
            pivot = T(leavingVariable+1, enteringVariable+1);
            keyRow = T(leavingVariable+1, 2:columns(T))/ pivot;
            BV(leavingVariable) = enteringVariable;
            pivotThing = repmat(keyColumn, 1,length(keyRow)) .*
repmat(keyRow, length(keyColumn), 1);
            newT = T;
            newT(2:end, 2:end) = T(2:end, 2:end) - pivotThing;
            newT(leavingVariable+1, 2:end) = keyRow;
            newT(leavingVariable+1, 1) = Cj(enteringVariable);
            T = newT;
        end
    end
```

On solving using BigM method :

```
octave:1> c = [4;3];
octave:2> b = [12;8;4];
octave:3> A = [3 4; 4 2; 1 1;];
octave:4> ind = [-1; -1 ; 0];
octave:5> simplexBIGM_lp_solver(c, A, b, ind);
Iteration 1 :
       1       4       3       0       0   -1000       0
       0       3       4       1       0       0      12
       0       4       2       0       1       0       8
   -1000       1       1       0       0       1       4

Iteration 2 :
   1.0000e+00   4.0000e+00   3.0000e+00   0.0000e+00   0.0000e+00  -1.0000e+03   0.0000e+00
   0.0000e+00   0.0000e+00   2.5000e+00   1.0000e+00  -7.5000e-01   0.0000e+00   6.0000e+00
   4.0000e+00   1.0000e+00   5.0000e-01   0.0000e+00   2.5000e-01   0.0000e+00   2.0000e+00
  -1.0000e+03   0.0000e+00   5.0000e-01   0.0000e+00  -2.5000e-01   1.0000e+00   2.0000e+00

Iteration 3 :
   1.0000e+00   4.0000e+00   3.0000e+00   0.0000e+00   0.0000e+00  -1.0000e+03   0.0000e+00
   3.0000e+00   0.0000e+00   1.0000e+00   4.0000e-01  -3.0000e-01   0.0000e+00   2.4000e+00
   4.0000e+00   1.0000e+00   0.0000e+00  -2.0000e-01   4.0000e-01   0.0000e+00   8.0000e-01
  -1.0000e+03   0.0000e+00   0.0000e+00  -2.0000e-01  -1.0000e-01   1.0000e+00   8.0000e-01

Infeasable
```

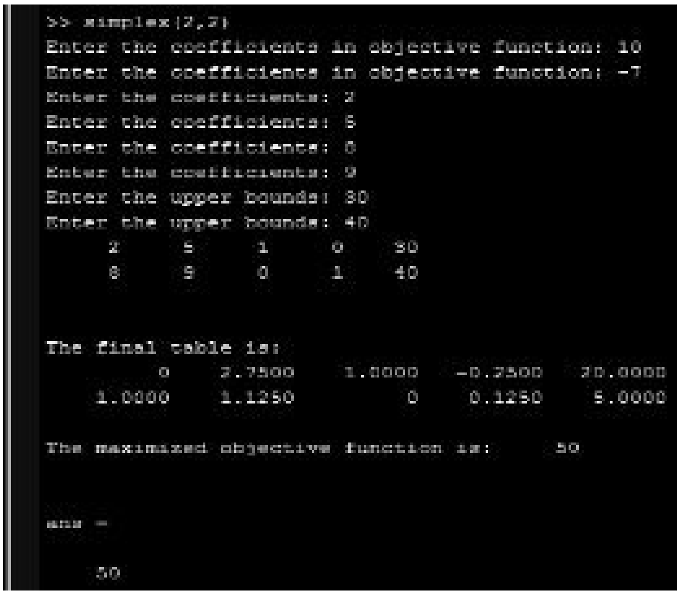
Problem 4

Solve by MATLAB using sensitivity analysis method.

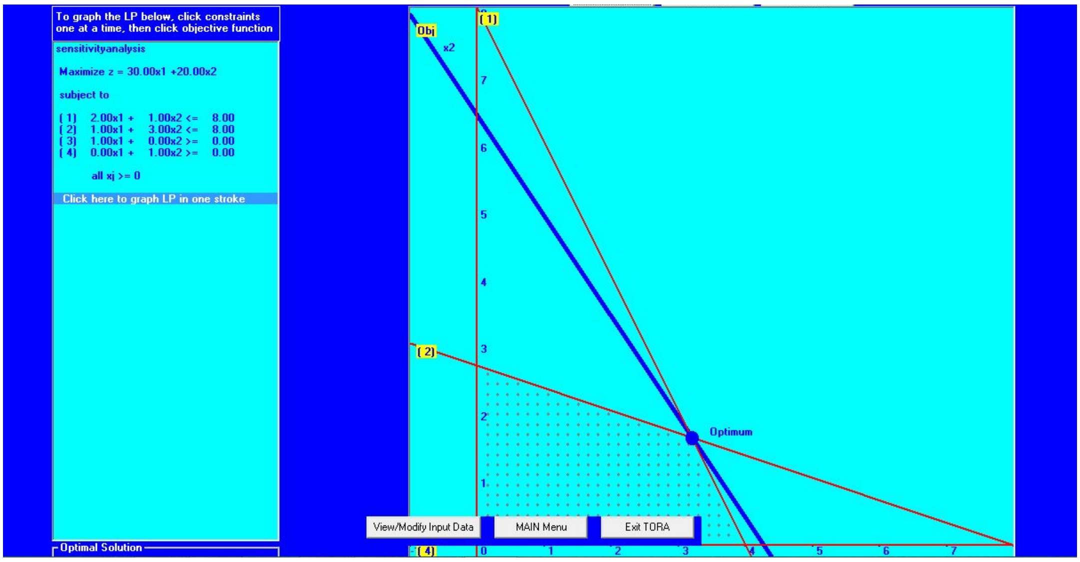
Minimize Z = - 3x + y + z
Subject to x - 2y + z <= 11
-4x + y + z >= 3

```
function objective = simplex(m,n)
% m represents no of constraints and n, the no of variables.
B = zeros(m,1);
for i = 1:n
    C(i) = input('Enter the coefficients in objective function: ');
end
for i = 1:m
    for j = 1:n
        A(i,j) = input('Enter the coefficients: ');
    end
end
for i = 1:m
    B(i) = input('Enter the upper bounds: ');
end
I = eye(m);
A = [A I B];
disp(A);
cb = zeros(1,m);
[z,val,ebv] = zed(A,C,cb,m,n);
for loop = 1:n
if(val < 0)
    lbv = leaving(A,ebv);
end
if(lbv > 0)
    A = modify(A,ebv,lbv);
    cb(lbv) = C(ebv);
    [z,val,ebv] = zed(A,C,cb,m,n);
end
end
fprintf('\nThe final table is: \n'),disp(A);
objective = 0;
for i = 1:m
    if(cb(i)~=0)
        objective = objective + cb(i)*A(i,end);
    end
end
```

```
fprintf('The maximized objective function is: '),disp(objective);
end
 function [z,val,ebv] = zed(A,C,cb,m,n)
C = [C cb];
for i = 1:n
    z(i) = 0;
    for j = 1:m
        z(i) = z(i) + cb(j)*A(j,i);
    end
end
for i = 1:n
    z(i) = z(i) - C(i);
end
[val, ebv] = min(z);
function lbv = leaving(A,i)
[m,n] = size(A);
for k = 1:m
    if(A(k,i)>0)
        row(i) = A(k,n)/A(k,i);
    end
end
[val,lbv] = min(row(row>0));
index = find(row==val,1,'first');
lbv = index;
function final = modify(A,i,j)
[m,~] = size(A);
A(j,:) = A(j,:)/ A(j,i);
for k = 1:m
    if(k~=j)
        A(k,:) = A(k,:) - A(k,i)*A(j,:);
    end
end
final = A;
```



```
>> simplex(2,2)
Enter the coefficients in objective function: 10
Enter the coefficients in objective function: -7
Enter the coefficients: 2
Enter the coefficients: 5
Enter the coefficients: 8
Enter the coefficients: 9
Enter the upper bounds: 30
Enter the upper bounds: 40
      2      5      1      0     30
      8      9      0      1     40


The final table is:
         0    2.7500    1.0000   -0.2500   20.0000
    1.0000    1.1250         0    0.1250    5.0000

The maximized objective function is:      50


ans =

    50
```



Title: sensitivityanalysis
Final Iteration No.: 7
Objective Value (Max) =128.00

Next Iteration | All Iterations | Write to Printer

Variable	Value	Obj Coeff	Obj Val Contrib
x1: x1	3.20	30.00	96.00
x2: x2	1.60	20.00	32.00

Constraint	RHS	Slack-/Surplus+
1 (<)	8.00	0.00
2 (<)	8.00	0.00
3 (>)	0.00	3.20+
4 (>)	0.00	1.60+

Sensitivity Analysis

Variable	Current Obj Coeff	Min Obj Coeff	Max Obj Coeff	Reduced Cost
x1: x1	30.00	6.67	40.00	0.00
x2: x2	20.00	15.00	90.00	0.00

Constraint	Current RHS	Min RHS	Max RHS	Dual Price
1 (<)	8.00	2.67	16.00	14.00
2 (<)	8.00	4.00	24.00	2.00
3 (>)	0.00	-infinity	3.20	0.00
4 (>)	0.00	-infinity	1.60	0.00

Problem 5

Problem: Solve by MATLAB using Integer Programming
Minimize Z = - 3x + y + z
Subject to x - 2y + z <= 11
-4x + y + z >= 3

```
function [x,val,status]=IP1(f,A,b,Aeq,beq,lb,ub,M,e)
options = optimset('display','off');
bound=inf;
[x0,val0]=linprog(f,A,b,Aeq,beq,lb,ub,[],options);
[x,val,status,b]=rec(f,A,b,Aeq,beq,lb,ub,x0,val0,M,e,bound);
function [xx,val,status,bb]=rec(f,A,b,Aeq,beq,lb,ub,x,v,M,e,bound)
options = optimset('display','off');
[x0,val0,status0]=linprog(f,A,b,Aeq,beq,lb,ub,[],options);
if status0<=0 | val0 > bound
    xx=x; val=v; status=status0; bb=bound;
    return;
end
ind=find( abs(x0(M)-round(x0(M)))>e );
if isempty(ind)
    status=1;
    if val0 < bound
        x0(M)=round(x0(M));
        xx=x0;
        val=val0;
        bb=val0;
    else
        xx=x;  % return the input solution
        val=v;
        bb=bound;
    end
    return
End
i=ind(1)
br_var=M(ind(1));
br_value=x(br_var);
if isempty(A)
    [r c]=size(Aeq);
else
    [r c]=size(A);
```

```
end
A1=[A ; zeros(1,c)];
A1(end,br_var)=1;
b1=[b;floor(br_value)];
A2=[A ;zeros(1,c)];
A2(end,br_var)=-1;
b2=[b; -ceil(br_value)];
[x1,val1,status1,bound1]=rec(f,A1,b1,Aeq,beq,lb,ub,x0,val0,M,e,bound);
status=status1;
if status1 >0 & bound1<bound   xx=x1;
   val=val1;
   bound=bound1;
   bb=bound1;
else
    xx=x0;
    val=val0;
    bb=bound;
end
    [x2,val2,status2,bound2]=rec(f,A2,b2,Aeq,beq,lb,ub,x0,val0,M,e,bound);
if status2 >0 & bound2<bound    status=status2;
    xx=x2;
    val=val2;
    bb=bound2;
end
```

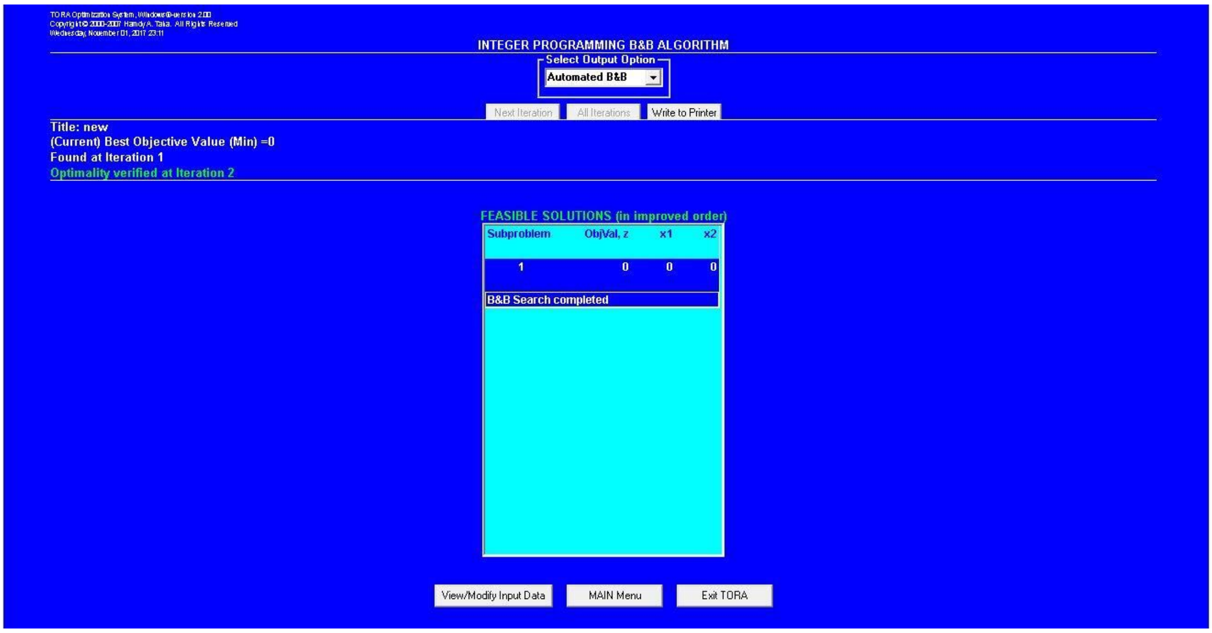
```
octave:25> integer
The optimal solution is 3
At :
   0
   3
   0

octave:26>
```

Problem 6

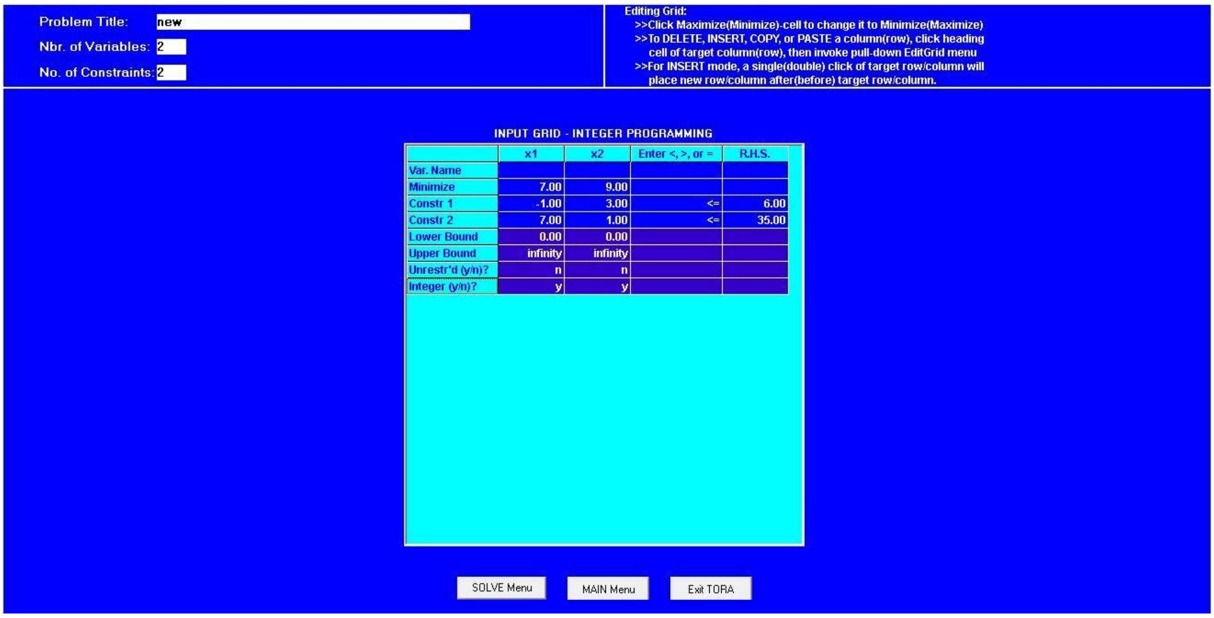
Problem: Solve by MATLAB using Mixed Integer Programming
Minimize Z = 2X+ Y
Subject to X+3Y<=9
X+5Y<=8

```
function [x,val,status]=IP1(f,A,b,Aeq,beq,lb,ub,M,e)
options = optimset('display','off');
bound=inf; % the initial bound is set to +ve infinity
[x0,val0]=linprog(f,A,b,Aeq,beq,lb,ub,[],options);
[x,val,status,b]=rec(f,A,b,Aeq,beq,lb,ub,x0,val0,M,e,bound);
function [xx,val,status,bb]=rec(f,A,b,Aeq,beq,lb,ub,x,v,M,e,bound)
options = optimset('display','off');
[x0,val0,status0]=linprog(f,A,b,Aeq,beq,lb,ub,[],options);
if status0<=0 | val0 > bound
    xx=x; val=v; status=status0; bb=bound;
    return;
end
ind=find( abs(x0(M)-round(x0(M)))>e );
if isempty(ind)
    status=1;
    if val0 < bound
        x0(M)=round(x0(M));
        xx=x0;
        val=val0;
        bb=val0;
    else
        xx=x;
        val=v;
        bb=bound;
    end
    return
End
i=ind(1)
br_var=M(ind(1));
br_value=x(br_var);
if isempty(A)
    [r c]=size(Aeq);
```

```
else
    [r c]=size(A);
end
A1=[A ; zeros(1,c)];
A1(end,br_var)=1;
b1=[b;floor(br_value)];
i=ind(1)
A2=[A ;zeros(1,c)];
A2(end,br_var)=-1;
b2=[b; -ceil(br_value)];
[x1,val1,status1,bound1]=rec(f,A1,b1,Aeq,beq,lb,ub,x0,val0,M,e,bound);
status=status1;
if status1 >0 & bound1<bound
   xx=x1;
   val=val1;
   bound=bound1;
   bb=bound1;
else
    xx=x0;
    val=val0;
    bb=bound;
end
[x2,val2,status2,bound2]=rec(f,A2,b2,Aeq,beq,lb,ub,x0,val0,M,e,bound);
if status2 >0 & bound2<bound
    status=status2;
    xx=x2;
    val=val2;
    bb=bound2;
end
```



TORA Optimization System, Windows®-version 2.00
Copyright© 2000-2007 Hamdy A. Taha. All Rights Reserved
Wednesday, November 01, 2017 23:11
INTEGER PROGRAMMING B&B ALGORITHM
Select Output Option
Automated B&B
Next Iteration
All Iterations
Write to Printer
Title: new
(Current) Best Objective Value (Min) =0
Found at Iteration 1
Optimality verified at Iteration 2
FEASIBLE SOLUTIONS (in improved order)
Subproblem ObjVal, z x1 x2
1 0 0 0
B&B Search completed
View/Modify Input Data
MAIN Menu
Exit TORA



Problem Title: new
Nbr. of Variables: 2
No. of Constraints: 2
Editing Grid:
>>Click Maximize(Minimize)-cell to change it to Minimize(Maximize)
>>To DELETE, INSERT, COPY, or PASTE a column(row), click heading cell of target column(row), then invoke pull-down EditGrid menu
>>For INSERT mode, a single(double) click of target row/column will place new row/column after(before) target row/column.
INPUT GRID - INTEGER PROGRAMMING
x1 x2 Enter <, >, or = R.H.S.
Var. Name
Minimize 7.00 9.00
Constr 1 -1.00 3.00 <= 6.00
Constr 2 7.00 1.00 <= 35.00
Lower Bound 0.00 0.00
Upper Bound infinity infinity
Unrestr'd (y/n)? n n
Integer (y/n)? y y
SOLVE Menu
MAIN Menu
Exit TORA

Problem 7

Problem : Solve the transportation problem using MATLAB

Cost Table :

Source/Dest.	**1**	**2**	**3**
1	-	3	5
2	7	4	9
3	1	8	6

```
% cij = [61 72 45 55 66; 69 78 60 49 56; 59 66 63 61 47;];
% si = [15 20 15]';
% dj = [11 12 9 10 8]';
function output = transportation_problem(cij, si, dj)
      printf('The parameter table is : \n')
      display(cij)
      printf('The supply is :')
      display(si')
      printf('The demand is :')
      display(dj')
      c = cij'(:);
      [m, n] = size(cij);
      A = zeros(m, m*n);
      for i=1:m
            starting = (i-1)*n +1;
            ending = i*n;
            A(i, starting:ending) = ones(1, n);
      end
      for j=1:n
            nows = zeros(m, n);
            for i=1:m
                  nows(i, j) = 1;
            end
            A(m+j, :) = nows'(:);
      end
      b = cat(1, si, dj);
      ind = zeros(m+n, 1);
      ctype = [];
```

```
	vartype = [];
	for i=1:m+n
		ctype(i) = 'S';
	end
	for i=1:m*n
		vartype(i) = 'I';
	end
	[xopt, fmin, errnum, extra] = glpk(c, A, b,[], [], char(ctype), 
char(vartype));
	printf('The optimal solution is %d, using : \n', fmin);
	idx = 1;
	for i=1:m
		for j=1:n
			if xopt(idx) ~= 0
				printf('X%d%d = %d, ', i, j,xopt(idx))
			end
			idx+=1;
		end
	end
end
```

```
octave:1> cij = [1000000 3 5;7 4 9;1 8 6];
octave:2> dj = [5 6 19]';
octave:3> si = [4 7 19]';
octave:4> transportation_problem(cij, si, dj)
The parameter table is :
   1000000         3         5
         7         4         9
         1         8         6

The supply is :    4    7   19

The demand is :    5    6   19

The optimal solution is 142, using :
X13 = 4, X22 = 6, X23 = 1, X31 = 5, X33 = 14, octave:5>
```