Graphical method

clc

clear all

%phase1: input data

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

B=[10;6;2;1;0;0];

%phase 2: DRAW THE LINES/CONSTRAINTS

x1=0:1:max(B)%first variable in constraint

x21=(B(1)-A(1,1)\*x1)/A(1,2)%second variable inn first constraint

x22=(B(2)-A(2,1)\*x1)/A(2,2)%second variable in second constraint

x23=(B(3)-A(3,1)\*x1)/A(3,2)%second variable in third constraint

x24=(B(4)-A(4,1)\*x1)/A(4,2)% second variable in fourth constraint

x21=max(0,x21)

x22=max(0,x22)

x23=max(0,x23)

x24=max(0,x24)

plot(x1,x21,'r',x1,x22,'g',x1,x23,'y',x1,x24,'b')

title('x1 vs x2')

xlabel('value of x1')

ylabel('value of x2')

grid on

hold on

%phase 3

% finding line intersetion with axes

position\_x1=find(x1==0)

position\_x21=find(x21==0)%points with x2 axis

Line1=[x1(:,[position\_x1 position\_x21]);x21(:,[position\_x1 position\_x21])]';

position\_x22=find(x22==0)%points with x2 axis

Line2=[x1(:,[position\_x1 position\_x22]);x22(:,[position\_x1 position\_x22])]';

position\_x23=find(x23==0)%points with x2 axis

Line3=[x1(:,[position\_x1 position\_x23]);x23(:,[position\_x1 position\_x23])]';

position\_x24=find(x24==0)%points with x2 axis

Line4=[x1(:,[position\_x1 position\_x24]);x24(:,[position\_x1 position\_x24])]';

intersection\_pts\_axes=unique([Line1;Line2;Line3;Line4],'rows')

%phase 4: finding intersection of lines with each other

pt=[0;0]

for i=1:size(A,1)

A1=A(i,:)

B1=B(i,:)

for j=i+1:size(A,1)

A2=A(j,:)

B2=B(j,:)

A4=[A1;A2]

B4=[B1;B2]

X=A4\B4 %inverse of matrix

pt=[pt X]

end

end

ptt=pt'

%phase 5: write all corner points

cor\_points=[intersection\_pts\_axes;ptt]

P=unique(cor\_points,'rows')

size(P)

%phase 6: feasible region points

b1=P(:,1);%write first column of matrix

b2=P(:,2);

%write 1st constraint %all constraints are of <=sign

cons1=round(b1+(2.\*b2)-10);

s1=find(cons1>0);

P(s1,:)=[];

%write 2nd constraint %all constraints are of <=sign

b1=P(:,1);%write first column of matrix

b2=P(:,2);

cons2=round(b1+b2-6);

s2=find(cons2>0);

P(s2,:)=[];

%write 3rd constraint %all constraints are of <=sign

b1=P(:,1);%write first column of matrix

b2=P(:,2);

cons3=round(b1-b2-2);

s3=find(cons3>0);

P(s3,:)=[];

%write 4th constraint %all constraints are of <=sign

b1=P(:,1);%write first column of matrix

b2=P(:,2);

cons4=round(b1-(2.\*b2)-1);

s4=find(cons4>0);

P(s4,:)=[];

%write 5th constraint %all constraints are of <=sign

b1=P(:,1);%write first column of matrix

b2=P(:,2);

cons5=round(-b1);

s5=find(cons5>0);

P(s5,:)=[];

%write 3rd constraint %all constraints are of <=sign

b1=P(:,1);%write first column of matrix

b2=P(:,2);

cons6=round(-b2);

s6=find(cons6>0);

P(s6,:)=[];

f\_points=P

%phase 7:objective function value

c=[2,1];

for i=1:size(P,1)

fn(i,:)=(sum(P(i,:).\*c));

optim=max(fn)

end

%standard form

lc

clear all

format short

C=[1,3,7];

A=[2 3 4;1 5 2;2 4 3];

b=[1;8;4]

I=[0 0 1];%give slack var. values <=0 & >=1

s=eye(size(A,1))

index=find(I>0)

s(index,index)=-s(index,index)

mat=[A s b]% augmented matrix %columns sie should be same

obj=array2table(C)%command to write array to table

obj.Properties.VariableNames(1:size(C,2))={'x\_1','x\_2','x\_3'}

cons=array2table(mat)

cons.Properties.VariableNames(1:size(mat,2))={'x\_1','x\_2','x\_3','s1','s2','s3','b'}

% bfsmethod

clc

clear all

format short

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

C=[3 4 0 0]

b=[450;600]

n=size(A,2)%column

m=size(A,1)%rows

if(n>m)

nCm=nchoosek(n,m)% binomial coffecient of n and m

pair=nchoosek(1:n,m)%all possible cases(x1&x2,x1&x3...)

sol=[];

for i=1:nCm

y=zeros(n,1)

x=A(:,pair(i,:))\b %pair(i,:)ith row, including all columns

if all(x>=0 & x~=inf & x~=-inf)

y(pair(i,:))=x

sol=[sol,y]

end

end

else

error('nCm does not exists')

end

Z=C\*sol

[Zmax,Zindex]=max(Z)

basicfs=sol(:,Zindex)

optimal\_value=[basicfs' Zmax]

optimal\_bfs=array2table(optimal\_value)

optimal\_bfs.Properties.VariableNames(1:size(optimal\_bfs,2))={'x\_1','x\_2','x\_3','x\_4','Z'}

%simplex method

% Simplex Method

%max z=2x1+5X2

%x1+4x2<=24

%3x1+1x2<=21

%x1+x2<=9

clc

clear all

format short

Noofvariables=2;

C=[2 5];

a=[1 4; 3 1; 1 1];

b=[24; 21; 9];

s=eye(size(a,1));

A=[a s b]

cost=zeros(1,size(A,2))

cost(1:Noofvariables)=C

bv= Noofvariables+1:1:size(A,2)-1

zjcj=cost(bv)\*A-cost

zcj=[zjcj; A]

simptable=array2table(zcj);

simptable.Properties.VariableNames(1:size(zcj,2))={'x\_1','x\_2','s\_1','s\_2','s\_3','sol'}

RUN=true;

while RUN

if any(zjcj<0); %check for (most) negative value

fprintf(' the current BFS is not optimal \n')

zc=zjcj(1:end-1);

[Enter\_val, pvt\_col]= min(zc)

if all(A(:,pvt\_col)<=0)

error('LPP is Unbounded all enteries are <=0 in column %d',pvt\_col);

else

sol=A(:,end)

column=A(:,pvt\_col)

for i=1:size(A,1)

if column(i)>0

ratio(i)= sol(i)./column(i)

else

ratio(i)=inf

end

end

[leaving\_val, pvt\_row]=min(ratio)

end

bv(pvt\_row)=pvt\_col

pvt\_key=A(pvt\_row, pvt\_col)

A(pvt\_row,:)=A(pvt\_row,:)./pvt\_key

for i=1:size(A,1)

if i~=pvt\_row

A(i,:)=A(i,:)-A(i, pvt\_col).\*A(pvt\_row,:)

end

end

zjcj=zjcj-zjcj(pvt\_col).\*A(pvt\_row,:)

zcj=[zjcj;A]

table=array2table(zcj)

table.Properties.VariableNames(1:size(zcj,2))={'x\_1','x\_2','s\_1','s\_2','s\_3','sol'}

else

RUN=false;

fprintf('The current BFS is optimal \n')

end

end

% big m method

%%q2 bigM METHOD PART 3 SOLVED

% FOR MINIMIZATION WE HAVE TO CHANGE FOLLOWING CONDITIONS IN CODE:

%1.in cost function value of M we be in +ve

%2. in while loop we will change the conditions of if else to greater than

format short

clear all

clc

% Max z = 3x1 +2x2

% x1 + x2 <= 2

% x1 + 3x2 <= 3

% x1 - x2 = 1

% x1, x2 >= 0

Cost = [3 2 0 0 -1000000 0]

A = [1 1 1 0 0 2;

1 3 0 1 0 3 ;

1 2 0 0 1 1]

BV = [3 4 5]

ZjCj = Cost(BV)\*A-Cost

zcj = [Cost;ZjCj;A];

BigMtable = array2table(zcj);

BigMtable.Properties.VariableNames(1:size(zcj,2)) = {'x\_1','x\_2','s\_1','s\_2','a\_1','sol'}

RUN = true;

while RUN

zc = ZjCj(1:end-1)

if any(zc<0)

fprintf('The current BFS is not optimal\n')

[ent\_col,pvt\_col] = min(zc)

fprintf('Entering Col = %d\n',pvt\_col);

sol = A(:,end)

Column = A(:,pvt\_col)

if Column<=0

error('LPP is unbounded');

else

for i=1:size(A,1)

if Column(i)>0

ratio(i) = sol(i)./Column(i)

else

ratio(i) = inf

end

end

[MinRatio,pvt\_row] = min(ratio)

fprintf('Leaving Row = %d\n',pvt\_row);

end

BV(pvt\_row) = pvt\_col;

pvt\_key = A(pvt\_row,pvt\_col);

A(pvt\_row,:) = A(pvt\_row,:)./pvt\_key;

for i = 1:size(A,1)

if i ~= pvt\_row

A(i,:) = A(i,:) - A(i,pvt\_col).\*A(pvt\_row,:);

end

end

ZjCj = ZjCj-ZjCj(pvt\_col).\*A(pvt\_row,:)

zcj = [ZjCj;A]

TABLE = array2table(zcj);

TABLE.Properties.VariableNames(1:size(zcj,2)) = {'x\_1','x\_2','s\_1','s\_2','a\_1','sol'}

else

RUN = false;

fprintf('Current BFS is Optimal \n');

end

end

%2 phase method

%%q1 TWO PHASE METHOD PART 1 SOLVED

% FOR MAXIMIZATION WE HAVE TO CHANGE FOLLOWING CONDITIONS IN CODE:

%1.value of A B Z MATRIX

%2.Tablename.Properties.VariableNames change in names of column assigned

%accordingly.

clc

clear

format rat

% MIN Z=3x1-5x2

% x1+3x2>=3

% x1+x2>=2

%x1,x2>=0

%% PHASE 1: INPUT PARAMETER

A=[1 3; 1 1 ];

B=[3; 2];

Z=[3 5];

%

% A=[5 1; 6 5;1 4 ];

% B=[10; 30;8];

% Z=[12 10];

%% PHASE 2: STANDARD FORM

s=eye(size(A,1));

m=size(A,1);

n=size(A,2);

col=size(A,2);

I=[1 1];

greater\_than=find(I==1);

for i=1:size(greater\_than,2)

mat=zeros(1,m);

mat(greater\_than(i))=-1;

mat=mat';

A=[A mat];

end

artificial\_var=find(I==2 | I==1);

artificial\_var\_in\_table(1:size(artificial\_var,2))=(artificial\_var+size(A,2));

A=[A s B];

Cj=zeros(1,size(A,2));

for i=1:size(artificial\_var\_in\_table,2)

Cj(artificial\_var\_in\_table(i))=-1;

end

%% PHASE I OF TWO-PHASE METHOD

% FIRST TABLE

bv=(n+size(greater\_than,2)+1):size(A,2)-1;

zjcj=Cj(bv)\*A-Cj;

fprintf("----------PHASE I----------\n\n");

fprintf("INITIAL TABLE:\n");

ZjC=[zjcj; A];

simpTable=array2table(ZjC)

simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','a1','a2','Sol'};

disp(simpTable);

% PHASE I- SIMPLEX TABLES

table=1;

RUN=true;

zc=zjcj(1:size(A,2)-1);

optimal=true;

while RUN

if any(zc<0)

[minvalzjcj, minindzjcj]=min(zc);

pivot\_col\_ind=minindzjcj;

pivot\_col=A(:,pivot\_col\_ind);

if all(pivot\_col<=0)

print('LPP is unbounded');

optimal=false;

break;

else

for i=1:size(pivot\_col,1)

if pivot\_col(i)>0

ratio(i)=B(i)./pivot\_col(i);

else

ratio(i)=inf;

end

end

[min\_ratio, ratio\_ind]=min(ratio);

pivot\_row\_ind=ratio\_ind;

bv(ratio\_ind)=pivot\_col\_ind;

pivot\_value=A(pivot\_row\_ind,pivot\_col\_ind);

A(pivot\_row\_ind,:)=A(pivot\_row\_ind,:)./pivot\_value;

for i=1:size(A,1)

if i~=pivot\_row\_ind

A(i,:)=A(i,:)-(pivot\_col(i)\*A(pivot\_row\_ind,:));

end

end

zjcj=zjcj-(zjcj(pivot\_col\_ind)\*A(pivot\_row\_ind,:));

end

ZjC=[zjcj; A];

zc=zjcj(1:size(A,2)-1);

B(1:m)=(A(:,size(A,2)))';

simpTable=array2table(ZjC);

simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','a1','a2','Sol'};

simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','x3','s1','s2','s3','a1','a2','a3','Sol'};

fprintf("TABLE %d:\n",table);

table=table+1;

disp(simpTable);

else

RUN=false;

end

end

%% PHASE II OF TWO-PHASE METHOD

optimal=true;

for i=1:size(bv,2)

index=find(artificial\_var\_in\_table==bv(i));

if index~=0

optimal=false;

end

end

if optimal==false

disp('INFEASIBLE SOLUTION');

else

fprintf("\n\n----------PHASE II----------\n");

artificial\_variables\_number=size(artificial\_var\_in\_table,2);

A=A(1:end,1:end-artificial\_variables\_number-1);

A=[A B];

Cj=zeros(1,size(A,2));

Cj(1:n)=Z;

% FIRST TABLE

zjcj=Cj(bv)\*A-Cj;

fprintf("INITIAL TABLE:\n");

ZjC=[zjcj; A];

simpTable=array2table(ZjC);

simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','Sol'};

disp(simpTable);

% PHASE II SIMPLEX TABLES

table=1;

RUN=true;

zc=zjcj(1:size(A,2)-1);

optimal=true;

while RUN

if any(zc<0)

[minvalzjcj, minindzjcj]=min(zc);

pivot\_col\_ind=minindzjcj;

pivot\_col=A(:,pivot\_col\_ind);

if all(pivot\_col<=0)

disp('LPP is unbounded');

optimal=false;

break;

else

for i=1:size(pivot\_col,1)

if pivot\_col(i)>0

ratio(i)=B(i)./pivot\_col(i);

else

ratio(i)=inf;

end

end

[min\_ratio, ratio\_ind]=min(ratio);

pivot\_row\_ind=ratio\_ind;

bv(ratio\_ind)=pivot\_col\_ind;

pivot\_value=A(pivot\_row\_ind,pivot\_col\_ind);

A(pivot\_row\_ind,:)=A(pivot\_row\_ind,:)./pivot\_value;

for i=1:size(A,1)

if i~=pivot\_row\_ind

A(i,:)=A(i,:)-(pivot\_col(i)\*A(pivot\_row\_ind,:));

end

end

zjcj=zjcj-(zjcj(pivot\_col\_ind)\*A(pivot\_row\_ind,:));

end

ZjC=[zjcj; A];

zc=zjcj(1:size(A,2)-1);

B(1:m)=(A(:,size(A,2)))';

simpTable=array2table(ZjC);

simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','Sol'};

fprintf("TABLE %d:\n",table);

table=table+1;

disp(simpTable);

else

RUN=false;

end

end

% OPTIMAL SOLUTION

if optimal==true

fprintf("FINAL TABLE:\n");

disp(simpTable);

fprintf("OPTIMAL SOLUTION:\n");

for i=1:col

index=find(bv==i);

if(index>0)

fprintf("x%d = %.3f\n",i,A(index,size(A,2)));

else

fprintf("x%d = %d\n",i,0);

end

end

fprintf("Min Z = %f",ZjC(1,size(ZjC,2)));

end

end