function [pnuinf,pnuinf1,mz,xxx]=maximize(m,n,A,b,c,xu)

%maximize(m,n,A,b,c,xu) generates random points using a certain probability
%distribution with considering upper bounds which are produced by my_upfun
%and selects the best feasible solution of primal problem which is
%maximization.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization

% Initialization

tot=0; k = 1; kk = 1; pnuinf=0; j=1; pnuinf1=0; mz=-inf; add=0;x=[];
clear functions
% 100 feasible solutions are generated under a certain probability
% distribution.
while k <= 100
    for j = 1 : n
        x(j,k)= xu(j)*gamrnd(1,2);% xu(j)* betarnd(.75,.5);% .5*xu(j)+randn*(.5^.5*xu
(j));% 
        % geornd(.5)*xu(j);% poissrnd(.5*xu(j));% wblrnd(xu(j),3.5);%
        % rand*xu(j);% raylrnd(xu(j));%
        % nbinrnd(2,.5);% xu(j)*chi2rnd(1);% trnd(5)+xu(j);%
        %unifrnd(0,xu(j));% randraw('normaltrunc',[0,xu(j),xu(j),xu(j)],1); %
    end

% Testing the feasibleilty

    if x(:,k)>=0
        AX = A * x(:,k);
        if AX <= b
            z(k) = c' * x(:,k);
            add=add+1;
            k = k + 1;
            tot=tot+1;
        else
            pnuinf=pnuinf+1;
            x(:,k) =[];
            if pnuinf >= 50000 break % The number of infeasible points are
controlled
        end
    else
        pnuinf1=pnuinf1+1;
        x(:,k)= [ ];
        if pnuinf1 >= 50000 break % The number of infeasible (negative) poits
are controlled.
    end
end

[mz,I] = max(z); % the maximum of primal objective function value is selected also
% the associated solution in current iteration and forwarded it
xxx=x(:,I); % to the main program.
function [dnuinf,dnuinf1,ming,yyy]=minimize(m,n,A,b,c,yl)

%minimize(m,n,A,b,yl) generates random points using a certain probability
%distribution with considering lower bounds which are produced by my_lowfun
%and select the best feasible solution of dual problem which is
%minimization.

%==========================================================================
% Initialization
k=1; kk=1; dnuinf=0;dnuinf1=0;add=0; tot=0;y=[];
clear functions
% 100 feasible solutions are generated under a certain probability
% distribution.

while k <= 100
    for j= 1 : m
        y(j,k)= geornd(.5)+yl(j);% yl(j)+ betarnd(.05,.05);%.5*yl(j)+randn;%% wblrnd(yl(j),2.5);% gamrnd(1,1.5)+yl(j);% raylnrd(yl(j));%
        if y(:,k) >= 0
            if A'*y(:,k)>=c
                g(k)=b'*y(:,k);
                add=add+1;
                k=k+1;
                tot=tot+1;
            else
                dnuinf=dnuinf+1;
                y(:,k)=[];
                if dnuinf>=50000  dnuinf
                    % The number of infeasible points are controlled
                    break
                end
            end
        else
            dnuinf1=dnuinf1+1;
            y(:,k)=[];
            if dnuinf1>=50000 break % The number of infeasible (negative) points are controlled.
                end
        end
    end
    [ming(kk),I]=min(g); % the minimum of dual objective function value is selected also
        % the associated solution in current iteration and forwarded it
    yyy=y(:,I);
        % to the main program.
function [yl]=low(m,n,A,c)
% low(m,n,A,c) produces lower bound of decision variables of a dual
% problem in linear programming problem.
%==========================================================================
A=A';
yyyf=[];
for j = 1 : m
    k=1;
    for i=1:n
        if A(i,j) == 0
            else o(i)=c(i)/A(i,j);
        end
    end
    yy =[]; yyf=[];
    % Find intersections of constraints mutually.
    for ii = 1 : n-1
        for jj = ii+1 : n
            qq = [A(ii,:);A(jj,:)]\[c(ii);c(jj)];
            yy(:,k) = qq;
            if qq ~= inf
                C=A*qq;
                %if (c-C) >= 0
                if (C-c) >= 0
                    yyf=[yyf yy(:,k)];
                    yy(:,k)=[];
                k=k-1;
            else
                k=k+1;
            end
        end
    end
    % Find intersections of constraints in triple.
    if m >= 3
        for u = 1 : m-2
            for v = u+1 : m-1
                for w = v+1 : m
                    qqq = [A(u,:);A(v,:);A(w,:)]\[c(u);c(v);c(w)];
                    yy(:,k) = qqq;
                    k = k+1
                end
            end
        end
    end
    % end
    % end
% end
lb = o;

[ro col]=size(yy);
for r=1:col
    lb=lb;
    yy(:,r);
    lb=[lb,yy(j,r)];
end
my = zeros(m,1);
ll=1;

% The infeasible intersections are ignored.
while ll <= length(lb)
    my(j) = lb(ll);
    if A * my >= c
        ll = ll+1;
        yyyf(:,j)=my;
    else
        lb(ll) = inf;
        ll = ll + 1;  % the number of infeasible points are counted.
    end
end

if size(yyf) ~=[0 0]
    llb=[lb,yyf(j)];
else
    llb=lb;
end
yl(j)=min(llb);
if (yl(j)<0) | (yl(j)==inf) yl(j)=0 ;
end

for p=1:m
    if size(yyyf) ~=[0 0]
        ylow =[yl(p) yyyf(p,:)];
        yl(p)=min(ylow);
        if (yl(p)<0) | (yl(p) ==0) yl(p)=0 ;
    end
end
end
end
yl  % lower bounds are printed.
function [xu]=upp(m,n,A,b)
% upp(m,n,A,b) produces upper bound of decision variables of a primal
% problem in linear programming problem.
%==========================================================================
% Initialization
x = (1:n)*0;
x = x';
t = 0;
for i = 1 : m
    for j = 1 : n
        if A(i,j) <= 0
            t = t+1;
        end
    end
end
if t > 0
    for j = 1 : n
        o=zeros(1,m);
        up=[];
        for i = 1 : m
            if (A(i,j) == 0) | (A(i,j) < 0)
                o(i)=A(i,j)(i);
            end
        end
    end
    % find the intersections of constraints mutually.
    k = 1;
    xx=zeros(n,1);
    for ii = 1 : n-1
        for jj = ii+1 : n
            gg = [A(:,ii),A(:,jj)];
            if (gg ~= inf) | (gg ~= -inf)
                xx([ii jj],k)=gg;
            end
            k = k+1;
        end
    end
    % Find the intersections of constraints in triple.
    % if n >= 3;
    %    for u=1 : n-2
    %        for v=u+1 : n-1
    %            for w=v+1 : n
    %                ggg=[A(:,u),A(:,v),A(:,w)];
    %                if ggg ~= inf
    %                    xx([u v w],k) = ggg;
    %                k=k+1;
    %            end
    %        end
    %    end
end
% The infeasible intersection points are ignored.
%        mx=zeros(n,1);
%        ll=1;
%        while ll <= length(up)
%            mx(j)=up(ll);
%            if A*mx <= b    ll=ll+1;
%            else    up(ll)=0;    ll=ll+1;
%        end
%        end
xu(j)=max(up);

end

end

xu

else  %(if t>0)
    for j = 1 : n
        for i = 1 : m
            o(i) = b(i)/A(i,j);
        end
        xu(j) = min(o);
    end
    xu  % upper bounds are printed.
end
function prdu(m,n,A,b,c,nu)
% prdu(m,n,A,b,c,nu) is the program to the stochastic search algorithm
% that invokes the functions which are "my_lowfun", "my_upfun", "minimize"
% and "maximize". This program restarts the minimize and maximize functions.
% nu is the number of restarting. This MATLAB function finds the best
% solutions to the primal and dual linear programming problems which are called
% near-optimal solutions.

% Here, m is the number of constraints, n is the number of variables,
% A is an m by n matrix including the variable coefficients in constraints
% b is the RHS vector and c is an n by 1 vector of coefficient variables
% in the objective function of primal problem.
%==========================================================================
t=cputime;
% Initialization

ttot=1;ggg1=1000;

[yl]=my_lowfun(m,n,A,c);
[xu]=my_upfun(m,n,A,b);

% Restart the minimize and maximize functions while the length of
% gap(difference between primal and dual objective function values) is
% bigger than or equal to 0.001 (the initial value of gap is arbitrary) and
% the number of restart does not reach to its initial value.

while (ggg1 >= 0.001) & (ttot <= nu)
    [pnuinf,pnuinf1,mz,xxx]=maximize(m,n,A,b,c,xu);
    ppnuinf(ttot)=pnuinf;
    ppnuinf1(ttot)=pnuinf1;
    mmax(ttot)=mz;
    xxop(:,ttot)=xxx;
    [dnuinf,dnuinf1,ming,yyy]=minimize(m,n,A,b,c,yl);
    ddnuinf(ttot)=dnuinf;
    ddnuinf1(ttot)=dnuinf1;
    mmin(ttot)=ming;
    yyop(:,ttot)=yyy;
    z1=max(mmax);
    g1=min(mmin);
    ggg1=g1-z1;
    tttot=ttot+1;
end
% output the near-optimal objective function values of primal and dual
% problems with their associated solution.

[maxprimal,P]=max(mmax)
xop=xxop(:,P)
[mindual,D]=min(mmin)
yop=yyop(:,D)
MGAP1=(mindual-maxprimal)
numberrestart=ttot-1
sumprimalinf=sum(ppnuinf);
primalinfmax=ppnuinf(P);
sumprimalinf1=sum(ppnuinf1);
primalinf1max=ppnuinf1(P);
sumdualinf=sum(ddnuinf);
dualinfmin=ddnuinf(D);
sumdualinf1=sum(ddnuinf1);
dualinf1min=ddnuinf1(D);

for pt=2:ttot-1
    PN(pt)=pt*100;
    if mmax(pt)>pf(pt-1)
        pf(pt)=mmax(pt);
    else
        pf(pt)=pf(pt-1);
    end
end

% Output the graph of near-optimal values of primal and dual problems in
% iterations.
df(1)=mmin(1);DN(1)=100;
for dt=2:ttot-1
    DN(dt)=dt*100;
    if mmin(dt)<df(dt-1)
        df(dt)=mmin(dt);
    else
        df(dt)=df(dt-1);
    end
end

plot(PN,pf,DN,df)
grid on

% Compute the number of infeasible points in the all iterations and output
% the number of infeasible points in that iteration which is given the near
% -optimal value for both primal and dual problems.

[maxprimal,P]=max(mmax)
time=cputime-t
ISCprimal=primalinfmax+primalinf1max
TOTprimal=sumprimalinf+sumprimalinf1
ISCdual=dualinfmin+dualinf1min
TOTdual=sumdualinf+sumdualinf1
numbergeneratedpoints=ISCprimal+ISCdual+200
% This script runs the prdu function. Different examples are mentioned.
clc
clear functions

% Beal's Example
prdu(3,4,[.25 -60 -.04 9;.5 -90 -.02 3;0 0 1 0],[0;0;1],[.75;-150;.02;-6],50)

% Kuhn's Example
prdu(3,4,[-2 -9 1 9;.33 1 -.33 -2;2 3 -1 -12],[0;0;2],[2;3;-1;-12],50)

% Klee & Minty Example.
prdu(4,4,[16 8 4 1;8 4 1 0;4 1 0 0;1 0 0 0],[625;125;25;5],[8;4;2;1],50)

% Further Example
prdu(2,2,[5 20;10 15],[400;450],[45;80],50)