```
function [single_section_result] = FOCuS_Single_Section(model_in,biom,rxn_max_in, ↙
bound_glu,bound_O2,bound_ATPM,bound_biom)

%%%%%%%%%%%%% inputs to indexing program %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% model_in          COBRA stoichiometric model in standard structure format
% biom              Biomass reaction
% rxn_max_in        Target reaction
% bound_glu         input flux value for glucose
% bound_O2          input flux value for oxygen
% bound_ATPM        input flux value for maintenance ATP

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
tic

global model
global rxn_max

model = model_in;
rxn_max = rxn_max_in;



para=[500 0.8];
n=para(1);
p=para(2);
m=14;
pop=n;

gen=0;
max_gen=20;

global d
d=5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ↙
%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% RXN Name ↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rxns_size = size(model.rxns);
rxns_count = rxns_size(1);

for i=1:1:rxns_count

    rxn_ind = strcmp(model.rxns(i),rxn_max);

    if rxn_ind > 0

        rxn_max_no = i;

    end
```

```
end
clear i
disp('Target reaction number =')
disp(rxn_max_no);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ↙
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% READ INDEX ↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ind=xlsread('C:\Users\...\MATLAB\cobra\FOCUS\final_search_index.xlsx' )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ↙
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ↙
%%%

global ind_cont
global pind
ind_cont = 1:length(ind);
pind = ind(ind_cont);

Lb = min(ind_cont)*ones(1,d);
Ub = max(ind_cont)*ones(1,d);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ↙
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Inputs to model ↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


model = changeRxnBounds(model,'EX_glc(e)',bound_glu,'l');
model = changeRxnBounds(model,'EX_o2(e)',bound_O2,'l');
model = changeRxnBounds(model,biom,bound_biom, 'l');
model = changeRxnBounds(model,'ATPM',bound_ATPM,'l');

Sol = zeros(pop,d);

for i=1:1:pop
    Sol(i,:)=Lb+(Ub-Lb).*rand(1,d);
    Sol(i,:)=round(Sol(i,:));
    s_1 = unique(Sol(i,:));
    len = length(s_1);

    if len == (d-1)
        s_2 = setdiff(ind_cont, s_1);
        s_3 = randsample(s_2,1);
        Sol(i,:) = [s_1 s_3];
    end

    if len == (d-2)
        s_2 = setdiff(ind_cont, s_1);
        s_3 = randsample(s_2,2);
        Sol(i,:) = [s_1 s_3];
    end
```

```
    if len == (d-3)
        s_2 = setdiff(ind_cont, s_1);
        s_3 = randsample(s_2,3);
        Sol(i,:) = [s_1 s_3];
    end

    clear len s_1 s_2 s_3
end


for i=1:1:pop
    Sol(i,:);
    Fitness(i)=Fun(Sol(i,:));
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%
%%Initialize the population/solutions

[Fitness, indf] = sort(Fitness,'descend');
best_pop = Sol(indf(1:end),:);

fmax=Fitness(1);
best=Sol(indf(1),:);

clear indf

S1=zeros(n,d);
S2=zeros(n,d);

bestu=[];


while(gen<max_gen)

    for i=1:1:n
        S1(i,:) = randsample(ind_cont, d);
        S1(i,:)=simplebounds(S1(i,:),Lb,Ub);
    end


    Fitness1 = zeros(1,n);

    for i=1:1:n
        S2(i,:) = randsample(ind_cont, d);
        S2(i,:)=simplebounds(S2(i,:),Lb,Ub);
    end

    Fitness2 = zeros(1,n);
    Fitness3 = zeros(1,n);
```

```
    if rand>p


        for i=1:1:n
            L=Levy(d);
            dS=L.*(Sol(i,:)-best);
            S1(i,:)=Sol(i,:)+dS;
            S1(i,:)=simplebounds(S1(i,:),Lb,Ub);
            Fitness1(i)=Fun(S1(i,:));
        end


    else
        %%%%%%%%%%%%%%%% for pop %%%%%%%%%%%%%%%%%%%%%%%%%%

        for i=1:1:pop
            Sol(i,:);
            Fitness(i)=Fun(Sol(i,:));
        end

        [Fitness, indf] = sort(Fitness,'descend');

        clone = Sol(indf(1:m),:);
        clear indf

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        clone_pop = zeros(n,d);
        for i=1:1:18
            clone_pop(i,:)=clone(1,:);
        end
        for i=19:1:34
            clone_pop(i,:)=clone(2,:);
        end
        for i=35:1:48
            clone_pop(i,:)=clone(3,:);
        end
        for i=49:1:60
            clone_pop(i,:)=clone(4,:);
        end
        for i=61:1:70
            clone_pop(i,:)=clone(5,:);
        end
        for i=71:1:78
            clone_pop(i,:)=clone(6,:);
        end
        for i=79:1:84
            clone_pop(i,:)=clone(7,:);
        end
        for i=85:1:88
            clone_pop(i,:)=clone(8,:);
        end
        for i=89:1:90
            clone_pop(i,:)=clone(9,:);
        end
        for i=91:1:92
```

```
            clone_pop(i,:)=clone(10,:);
        end
        for i=93:1:94
            clone_pop(i,:)=clone(11,:);
        end
        for i=95:1:96
            clone_pop(i,:)=clone(12,:);
        end
        for i=97:1:98
            clone_pop(i,:)=clone(13,:);
        end
        for i=99:1:100
            clone_pop(i,:)=clone(14,:);
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        for j=1:1:n
            epsilon=rand;
            JK=randperm(n);
            S2(j,:)=clone_pop(j,:)+ (1*epsilon*(Sol(JK(1),:)-Sol(JK(2),:)));
            S2(j,:)=simplebounds(S2(j,:),Lb,Ub);
            Fitness2(j)=Fun(S2(j,:));
        end

    end

    S3 = [Sol; S1; S2];
    size(S3);


    for k=1:1:length(S3)
        Fitness3(k)=Fun(S3(k,:));
    end

    [Fitness3, ind3] = sort(Fitness3,'descend');
    best_S3 = S3(ind3(1:end),:);
    clear ind3

    size(best_S3);
    Sol_lev_clon = [best_S3 Fitness3'];
    size(Sol_lev_clon);
    Sol=best_S3(1:n,:);

    best_1=Sol(1,:);

    if Sol_lev_clon(1,end)>fmax
        best=best_1;
    end

    bestSol=pind(best);
    model.rxns(bestSol);

    fmax_gen(gen+1)=Sol_lev_clon(1,end);

    %%%%%%%%%%%% META-DYNAMICS STEP BEGINS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    if round(gen)==gen
        temp=best;
        %fmin_gen(end);
        bestu=[bestu; temp fmax_gen(end)];
        clear temp
        if gen>0
            sz_bestu=size(bestu);
            k_1=bestu((sz_bestu(1)-1),end);
            k_2=bestu(sz_bestu(1),end);
            Toler = k_1-k_2;
            if Toler >= 0
                for i=2:1:pop
                    Sol(1,:)=best;
                    Sol(i,:)=randsample(ind_cont,d);
                    Sol(i,:)=simplebounds(Sol(i,:),Lb,Ub);
                end

            end
        end

    end
    %%%%%%%%%%%% META-DYNAMICS STEP ENDS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    gen= gen+1;
end


fmax_gen=fmax_gen';
fmax = fmax_gen(end,1);
bestu;
bestSol=pind(best);
model.rxns(bestSol);

single_section_result = bestSol;

toc

end


function s=simplebounds(s,Lb,Ub)

global ind_cont
global d

ns_tmp=s;
I=ns_tmp<Lb;
ns_tmp(I)=Lb(I);
J=ns_tmp>Ub;
ns_tmp(J)=Ub(J);

s=ns_tmp;
s=round(s);
```

```
s_1=unique(s);
len = length(s_1);

if len == (d-1)
    s_2 = setdiff(ind_cont, s_1);
    s_3 = randsample(s_2,1);
    s = [s_1 s_3];
end

if len == (d-2)
    s_2 = setdiff(ind_cont, s_1);
    s_3 = randsample(s_2,2);
    s = [s_1 s_3];
end

if len == (d-3)
    s_2 = setdiff(ind_cont, s_1);
    s_3 = randsample(s_2,3);
    s = [s_1 s_3];
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function L =Levy(d)
beta=3/2;
tmpdiv=(gamma((1+beta)/2)*beta*2^((beta-1)/2))^(1/beta);
sigma = (gamma(1+beta)*sin(pi*beta/2))/tmpdiv;
u=randn(1,d)*sigma;
v=randn(1,d);
step=u./abs(v).^(1/beta);
L=0.01*step;
end

function fun_eval = Fun(p1)

global model
global pind
global rxn_max

p2=pind(p1);
deletions = model.rxns(p2);

nDel = length(deletions);
model_KO = model;
targetRxn=rxn_max;
toler = 1e-7;

for i = 1:nDel
    model_KO = changeRxnBounds(model_KO,deletions{i},0, 'b');
end

AfterKO = optimizeCbModel(model_KO);
```

```
growthRate = AfterKO.f;

if (AfterKO.stat == 1)
    round_off = floor(AfterKO.f/toler)*toler;
    model_KO = changeRxnBounds(model_KO,model_KO.rxns(model_KO.c==1),round_off, 'l');
    model_KO = changeObjective(model_KO,targetRxn);
    Max_Sol = optimizeCbModel(model_KO, 'max');
    Min_Sol = optimizeCbModel(model_KO, 'min');
    Prod_Max = Max_Sol.f;
    Prod_Min = Min_Sol.f;

else
    Prod_Max = 0;
    Prod_Min = 0;
end

fun_eval=Prod_Max;

clear model_KO Max_Sol Min_Sol AfterKO p1 p2 growthRate targxnind
end
```