OptGene Tutorial

Author: Sebastián N. Mendoza, Center for Mathematical Modeling, University of Chile. snmendoz@uc.cl

Reviewer(s): Sylvian Arreckx

INTRODUCTION:

In this tutorial we will run optGene For a detailed description of the procedure, please see [1]. Briefly, the problem is to find a set of reactions of size "K" such that when these reactions are deleted from the model, the mutant created will produce a particular target of interest in a higher rate than the wild-type strain.

For example, imagine that we would like to increase the production of succinate in Escherichia coli. Which are the knockouts needed to increase the production of succinate? We will approach this problem in this tutorial.

MATERIALS

EQUIPMENT

1. MATLAB
2. A solver for QP problems. For example, Gurobi. I encourage the users to use Gurobi since I've not obtained good results using glpk.

EQUIPMENT SETUP

Use changeCobraSolver to choose the solver for QP problems.

PROCEDURE

The proceduce consists on the following steps

1) Define contraints (manual task)

2) Select a list of reactions or genes (manual task). Reactions or genes in this list could be deleted. Elements that are not in the list will no be deleted.

3) Define some (manual task)

4) Run optGene. **TIMING**: This task should take from a few minutes to a few days, depending on the size of your reconstruction and the criterion for stoping optGene

```
global TUTORIAL_INIT_CB;
if ~isempty(TUTORIAL_INIT_CB) && TUTORIAL_INIT_CB  ==  1
    initCobraToolbox
    changeCobraSolver('gurobi', 'all');
end

fullPath = which ('tutorial_optGene.mlx');
folder = fileparts(fullPath);
cd(folder);

threshold = 3;

model = readCbModel('iJO1366.mat');
biomass = 'BIOMASS_Ec_iJO1366_core_53p95M';

%SETTING SPECIFIC CONSTRAINTS
% prespecified amount of glucose uptake 10 mmol/grDW*hr
model = changeRxnBounds(model, 'EX_glc__D_e', -10, 'b');

% Unconstrained uptake routes for inorganic phosphate, sulfate and
% ammonia
model = changeRxnBounds(model, 'EX_o2_e', 0, 'l');
model = changeRxnBounds(model, 'EX_pi_e', -1000, 'l');
model = changeRxnBounds(model, 'EX_so4_e', -1000, 'l');
model = changeRxnBounds(model, 'EX_nh4_e', -1000, 'l');

% The optimization step could opt for or against the phosphotransferase
% system, glucokinase, or both mechanisms for the uptake of glucose
model = changeRxnBounds(model, 'GLCabcpp', -1000, 'l');
model = changeRxnBounds(model, 'GLCptspp', -1000, 'l');
model = changeRxnBounds(model, 'GLCabcpp', 1000, 'u');
model = changeRxnBounds(model, 'GLCptspp', 1000, 'u');
model = changeRxnBounds(model, 'GLCt2pp', 0, 'b');

% Secretion routes  for acetate, carbon dioxide, ethanol, formate, lactate
% and succinate are enabled
model = changeRxnBounds(model, 'EX_ac_e', 1000, 'u');
model = changeRxnBounds(model, 'EX_co2_e', 1000, 'u');
model = changeRxnBounds(model, 'EX_etoh_e', 1000, 'u');
model = changeRxnBounds(model, 'EX_for_e', 1000, 'u');
model = changeRxnBounds(model, 'EX_lac__D_e', 1000, 'u');
model = changeRxnBounds(model, 'EX_succ_e', 1000, 'u');

% FINDING RATES IN WILD-TYPE
% The follling rates are those calculated in the wild-type without any
% mutation.

% determine succinate production and growth rate before optimizacion
fbaWT = optimizeCbModel(model);
growthRateWT = fbaWT.f;

model = changeObjective(model, 'EX_succ_e');
fbaWTMin = optimizeCbModel(model, 'min');
```

```
fbaWTMax = optimizeCbModel(model, 'max');
minSuccFluxWT = fbaWTMin.f;
maxSuccFluxWT = fbaWTMax.f;

model = changeObjective(model, biomass);

fprintf('The minimum and maximum production of succinate before optimization is %.1f and %.1f respectively\n', minSuccFluxWT, maxSuccFluxWT
fprintf('The growth rate before optimization is %.2f \n', growthRateWT);

% OPTGENE SETTING
selectedGeneList = {};
% use prespecified reactions. Faster option
selectedRxnList = {'GLCabcpp'; 'GLCptspp'; 'HEX1'; 'PGI'; 'PFK'; 'FBA'; 'TPI'; 'GAPD'; 'PGK'; 'PGM'; 'ENO'; 'PYK'; 'LDH_D'; 'PFL'; 'ALCD2x'
genesByReaction = regexp(regexprep(model.grRules(ismember(model.rxns, selectedRxnList)), '\or|and|\(|\)', ''), '\  ', 'split');
for i = 1:length(genesByReaction)
    selectedGeneList = union(selectedGeneList, genesByReaction{i});
end
```

I) SUCCINATE OVERPRODUCTION

EXAMPLE 1: finding reaction knockouts sets of large 2 or less, using a limit of time to stop optGene

```
EXAMPLE 1: finding reaction knockouts sets of large 2 or less

fprintf('\n...EXAMPLE 1: Finding optGene sets\n\n')
previousSolutions = cell(10, 1);
contPreviousSolutions = 1;
nIter = 0;
while nIter < threshold
    fprintf('...Performing optGene analysis...\n')
    %optGene algorithm is run with the following options: target: 'EX_lac__D_e'
    [~, ~, ~, optGeneSol] = optGene(model, 'EX_succ_e', 'EX_glc__D_e', selectedGeneList, 'MaxKOs', 2, 'TimeLimit', 120);

    SET_M1 = optGeneSol.geneList;

    if ~isempty(SET_M1)
        previousSolutions{contPreviousSolutions} = SET_M1;
        contPreviousSolutions = contPreviousSolutions + 1;
        %printing results
        fprintf('optGene found a knockout set of large %d composed by ', length(SET_M1));
        for j = 1:length(SET_M1)
            if j == 1
                fprintf('%s ',SET_M1{j});
            elseif j == length(SET_M1)
                fprintf('and %s',SET_M1{j});
            else
                fprintf(', %s ',SET_M1{j});
            end
        end
        fprintf('\n');
        fprintf('...Performing coupling analysis...\n');
        [type, maxGrowth, maxProd, minProd] = analyzeOptKnock(model, optGeneSol.geneList, 'EX_succ_e', biomass, 1);
        fprintf('The solution is of type: %s\n',type);
        fprintf('The maximun growth rate after optimizacion is %.2f\n', maxGrowth);
        fprintf('The maximun and minimun production of succinate after optimization is %.2f and %.2f, respectively \n\n', minProd, maxProd)

    else
        if nIter  ==  1
            fprintf('optGene was not able to found an optGene set\n');
        else
            fprintf('optGene was not able to found additional optGene sets\n');
        end
        break;
    end
    nIter = nIter + 1;

end
```

EXAMPLE 2: finding reaction knockouts sets of large 2 or less, using the number of generations to stop optGene

```
fprintf('\n...EXAMPLE 2: Finding optGene sets\n\n')
previousSolutions = cell(10, 1);
contPreviousSolutions = 1;
nIter = 0;
while nIter < threshold
    fprintf('...Performing optGene analysis...\n')
    %optGene algorithm is run with the following options: target: 'EX_lac__D_e'
    [~, ~, ~, optGeneSol] = optGene(model, 'EX_succ_e', 'EX_glc__D_e', selectedGeneList, 'MaxKOs', 2, 'Generations', 20);

    SET_M1 = optGeneSol.geneList;

    if ~isempty(SET_M1)
        previousSolutions{contPreviousSolutions} = SET_M1;
        contPreviousSolutions = contPreviousSolutions + 1;
        %printing results
        fprintf('optGene found a knockout set of large %d composed by ', length(SET_M1));
        for j = 1:length(SET_M1)
            if j == 1
                fprintf('%s ',SET_M1{j});
            elseif j == length(SET_M1)
                fprintf('and %s',SET_M1{j});
            else
                fprintf(', %s ',SET_M1{j});
            end
        end
        fprintf('\n');
        fprintf('...Performing coupling analysis...\n');
        [type, maxGrowth, maxProd, minProd] = analyzeOptKnock(model, optGeneSol.geneList, 'EX_succ_e', biomass, 1);
        fprintf('The solution is of type: %s\n',type);
        fprintf('The maximun growth rate after optimizacion is %.2f\n', maxGrowth);
        fprintf('The maximun and minimun production of succinate after optimization is %.2f and %.2f, respectively \n\n', minProd, maxProd);

    else
        if nIter  ==  1
            fprintf('optGene was not able to found an optGene set\n');
        else
            fprintf('optGene was not able to found additional optGene sets\n');
        end
        break;
    end
    nIter = nIter + 1;

end
```

TIMING

1. EXAMPLE 1: ~ 6 minutes (2 minutes per iteration)
2. EXAMPLE 2: ~ 7 minutes (2-3 minutes per iteration)

TROUBLESHOOTING

1) problem: "optGene didn't find any set"

possible reason: probably, the limit of time or the number of generations has not been enough. Another explination is that the solver is not suited for solving optGene

solution: Try with a higher number for inputs "TimeLimit" of "Generations" or using another solver.

2) problem: "I got an error when running optGene"

possible reason: the solver is not suited for solving optGene

solution: Try with another solver

ANTICIPATED RESULTS

The optGene algorithm will find sets of reactions that should increase the production of your target when they are deleted from the network. Since optGene is based on a genetic algorithm, the solutions found could vary between different runnings, even though the algorithm has been executed with the same input parameters. It is possible that optGene don't find a set of knockouts because the runtime is too short or because the number of generations is too small. In those cases try to increases those input variables.

References

[1] Patil, K. R., Rocha, I., Förster, J., & Nielsen, J. (2005). Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics, 6*(1), 308.

[2] Orth, J. D., Conrad, T. M., Na, J., Lerman, J. A., Nam, H., Feist, A. M., & Palsson, B. Ø. (2011). A comprehensive genome-scale reconstruction of Escherichia coli metabolism—2011. *Molecular systems biology, 7*(1), 535.