OptFill

```
function [TICs,Direction] = OptFillTFP(model,database,mTFP)
%%
% Enumerates all the Thermodynamically infeasible cycles in a given model
% using OptFill's algorithm. This code is adapted based on codes written in
% GAMS and python in the paper "OptFill: A Tool for Infeasible Cycle-Free
% Gapfilling of Stoichiometric Metabolic Models"
% USAGE:
%    [TICs,Direction] = OptFillTFP(model,database,mTFP)
%
% INPUTS:
%      model:      COBRA model structure defining the actual model
%      database:   COBRA model structure defining the database
%      mTFP:       boolean value indicating whether to work on modified-TIC
%                  finding problem (1) or TIC finding problem (0). In TIC
%                  finding problem, atleast one reaction in a TIC will be
%                  from the database.
%
% OUTPUTS:
%      TICs:        List of all the Thermodynamically infeasible cycles in
%                   the given input model-database pair
%      Direction:   Relative flux coefficients for reactions
%                   in the corresponding TICs
%
% .. Author:
%        - Pavan Kumar S, BioSystems Engineering and control (BiSECt) lab,
IIT Madras
%        - Based on: Schroeder, W. L., & Saha, R. (2020). OptFill: a tool
%                    for infeasible cycle-free gapfilling of stoichiometric
%                    metabolic models. IScience, 23(1).

m_Db = mergeTwoModels(model,database); % the combined model
m_Db.origin = ismember(m_Db.rxns,model.rxns); % boolean vector indicating if
a reaction is from model (1) or the database (0)

% parameters used were taken from the OptFill paper
epsilon=0.001;
M=1000;
tolerance=1e-6;

changeCobraSolverParams('LP', 'feasTol', tolerance);

[~,max_phi] = size(m_Db.S); % number of reactions in the model-database pair
IrR = m_Db.ub<=0; % reactions irreversible in reverse direction
temp = m_Db.ub(IrR);
m_Db.S(:,IrR) = -m_Db.S(:,IrR);
m_Db.ub(IrR) = -1*m_Db.lb(IrR);
m_Db.lb(IrR) = -1*temp;
rev = ones(n,1);
```

```
rev(m_Db.lb>=0) = 0;
m_Db.rev=rev;
% converting all the positive lower bounds to zero lower bounds
m_Db.lb(m_Db.lb>0)=0;

% normalising the bounds to max of M
temp1 = max(abs(m_Db.lb));
temp2 = max(abs(m_Db.ub));
m_Db.lb(abs(m_Db.lb)==temp1) = sign(m_Db.lb(abs(m_Db.lb)==temp1))*M;
m_Db.ub(abs(m_Db.ub)==temp2) = sign(m_Db.ub(abs(m_Db.ub)==temp2))*M;

ind = findExcRxns(m_Db); % indices of the exchange reactions

% blocking all the exchange reactions
m_Db.lb(ind) = 0;
m_Db.ub(ind) = 0;

TICs={}; Direction = {}; % empty cell to store the TICs and their respective
direction

for phi = 2:max_phi % minimum size of a TIC has to be two
    found_all = false;
    while ~found_all
        [flux,stat] = findTIC(m_Db,phi,mTFP);



    end
end
end
```

```
function [flux,stat] = findTIC(model,phi,mTFP)

end
```