Sparse Linear Optimisation

Author: Ronan Fleming, Hoai Minh Le, Systems Biochemistry Group, University of Luxembourg.

Reviewer: Stefania Magnusdottir, Molecular Systems Physiology Group, University of Luxembourg.

INTRODUCTION

In this tutorial, we will show how to use the sparse LP solver. This solver aims to solve the following optimisation problem

$$\begin{aligned} \min \quad & \|x\|_0 \\ s.t \quad & A_{eq}x = b_{eq} \\ & A_{ieq}x \leq b_{ieq} \\ & l \leq x \leq u \end{aligned}$$

It has been proved that zero-norm is a non-convex function and the minimisation of zero-norm is a NP-hard problem. Non-convex approximations of zero-norm extensively developed. For a complete study of non-convex approximations of zero-norm, the reader is referred to [1].

The method is described in [1]. The sparse LP solver contains one convex (one-norm) and 6 non-convex approximation of zero-norms

- Capped-L1 norm
- Exponential function
- Logarithmic function
- SCAD (Smoothly Clipped Absolute Deviation) function
- p norm with p<0
- p norm with 0<p<1

The tutorial consist of two parts. Part 1 shows a basic usage of the solver. In part 2 provides an application of the code for finding the minimal set of reactions subject to a LP objective. Ready-made scripts are provided for both parts.

EQUIPMENT SETUP

Initialize the COBRA Toolbox.

If necessary, initialize The Cobra Toolbox using the `initCobraToolbox` function.

```
initCobraToolbox
```

```
 _____   _____   _____   _____      _____
/  ___| /  _  \ |  _  \ |  _  \    / ___ \
| |     | | | | | |_| | | |_| |   | |___| |
| |     | | | | |  _  { |  _  /   |  ___  |
| |___  | |_| | | |_| | | | \ \   | |   | |
\_____| \_____/ |_____/ |_|  \_\  |_|   |_|

COnstraint-Based Reconstruction and Analysis
The COBRA Toolbox - 2017

Documentation:
http://opencobra.github.io/cobratoolbox
```

```
> Checking if git is installed ...  Done.
> Checking if the repository is tracked using git ...  Done.
> Checking if curl is installed ...  Done.
> Checking if remote can be reached ...  Done.
> Initializing and updating submodules ... Done.
> Adding all the files of The COBRA Toolbox ...  Done.
> Define CB map output... set to svg.
> Retrieving models ...   Done.
> TranslateSBML is installed and working properly.
> Configuring solver environment variables ...
  - [---*] ILOG_CPLEX_PATH: C:\Program Files\IBM\ILOG\CPLEX_Studio1271\cplex\matlab\x64_win64
  - [----] GUROBI_PATH :  --> set this path manually after installing the solver ( see instructions )
  - [---*] TOMLAB_PATH: C:\Program Files\tomlab\
  - [----] MOSEK_PATH :  --> set this path manually after installing the solver ( see instructions )
  Done.
> Checking available solvers and solver interfaces ... Done.
> Setting default solvers ... Done.
> Saving the MATLAB path ... Done.
  - The MATLAB path was saved in the default location.

> Summary of available solvers and solver interfaces

    Support           LP   MILP     QP   MIQP    NLP
-------------------------------------------------------------------
cplex_direct  full              0      0      0      0      -
dqqMinos      full              0      -      -      -      -
glpk          full              1      1      -      -      -
gurobi        full              1      1      1      1      -
ibm_cplex     full              1      1      1      -      -
matlab        full              1      -      -      -      1
mosek         full              0      0      0      -      -
pdco          full              1      -      1      -      -
quadMinos     full              0      -      -      -      0
tomlab_cplex  full              1      1      1      1      -
qpng          experimental      -      -      1      -      -
tomlab_snopt  experimental      -      -      -      -      1
gurobi_mex    legacy            0      0      0      0      -
lindo_old     legacy            0      -      -      -      -
lindo_legacy  legacy            0      -      -      -      -
lp_solve      legacy            1      -      -      -      -
opti          legacy            0      0      0      0      0
-------------------------------------------------------------------
Total         -                 7      4      5      2      2

+ Legend: - = not applicable, 0 = solver not compatible or not installed, 1 = solver installed.


> You can solve LP problems using: 'glpk' - 'gurobi' - 'ibm_cplex' - 'matlab' - 'pdco' - 'tomlab_cplex'
> You can solve MILP problems using: 'glpk' - 'gurobi' - 'ibm_cplex' - 'tomlab_cplex'
> You can solve QP problems using: 'gurobi' - 'ibm_cplex' - 'pdco' - 'tomlab_cplex' - 'qpng'
> You can solve MIQP problems using: 'gurobi' - 'tomlab_cplex'
> You can solve NLP problems using: 'matlab' - 'tomlab_snopt'

> Checking for available updates ...
--> You cannot update your fork using updateCobraToolbox(). [3d2698 @ Tutorial-sparseFBA].
    Please use the MATLAB.devTools (https://github.com/opencobra/MATLAB.devTools).
```

COBRA model.

In this tutorial, the model used is the generic reconstruction of human metabolism, the Recon 2.04 [2], which is provided in the COBRA Toolbox. The Recon 2.04 model can also be downloaded from the Virtual Metabolic Human webpage. Before proceeding with the simulations, the path for the model needs to be set up:

```
global CBTDIR
load([CBTDIR filesep 'test' filesep 'models' filesep 'Recon2.v04.mat']);
model = modelR204;
clear modelR204;
```

Recon 2.04 is written in the "old style" COBRA format, and we thus use the function `convertOldStyleModel` to convert it to the new COBRA Toolbox format.

```
model = convertOldStyleModel(model);
```

PROCEDURE

Example of using sparseLP solver on randomly generated data

One randomly generates a matrix $A \in \mathcal{R}^{m \times n}$ and a vector $x_0 \in \mathcal{R}^n$. The right hand side vector $b = A \cdot x_0$. There are three optional inputs for the method.

```
n = 100;
m = 50;
x0 = rand(n,1);
constraint.A = rand(m,n);
constraint.b = constraint.A*x0;
constraint.lb = -1000*ones(n,1);
constraint.ub = 1000*ones(n,1);
constraint.csense = repmat('E', m, 1);
```

The two first: maximum number of iterations (*nbMaxIteration*) and threshold (*epsilon*) are stopping criterion conditions. *theta* is the parameter of zero-norm approximation. The greater the value of *theta*, the better the approximation of the zero-norm. However, the greater the value of *theta*, the more local solutions the problem has. If the value of *theta* is not given then the algorithm will use a default value and update it gradually.

```
params.nbMaxIteration = 100;    % stopping criteria
params.epsilon = 1e-6;   % stopping criteria
params.theta   = 2;       % parameter of l0 approximation
```

Call the solver with a chosen approximation

```
solution = sparseLP('cappedL1',constraint,params);
```

or with default parameter

```
%solution = sparseLP('cappedL1',constraint);
```

Finding the minimal set of reactions subject to a LP objective

Set the tolerance to distinguish between zero and non-zero flux, based on the numerical tolerance of the currently installed optimisation solver.

```
feasTol = getCobraSolverParams('LP', 'feasTol');
```

Select the biomass reaction to optimise

```
model.biomassBool=strcmp(model.rxns,'biomass_reaction');
model.c(model.biomassBool)=1;
```

We will firstly find the optimal value subject to a LP objective

```
%% Solve FBA
% max c'v
% s.t   Sv = b
%       l <= v <= u
% Define the LP structure
[c,S,b,lb,ub,csense] = deal(model.c,model.S,model.b,model.lb,model.ub,model.csense);
[m,n] = size(S);
LPproblem = struct('c',-c,'osense',1,'A',S,'csense',csense,'b',b,'lb',lb,'ub',ub);
% Call solveCobraLP to solve the LP
LPsolution = solveCobraLP(LPproblem);
vFBA = LPsolution.full;
```

We will now find the minimum number of reactions needed to achieve the same max objective found previously. Then one will add one more constraint: $c^T v = c^T v_{FBA} =: f_{FBA}$.

```
constraint.A = [S ; c'];
constraint.b = [b ; c'*vFBA];
constraint.csense = [csense;'E'];
constraint.lb = lb;
constraint.ub = ub;
```

Call the sparseLP solver to solve the problem

$$\begin{aligned} \min \quad & \|v\|_0 \\ s.t \quad & Sv = b \\ & c^T v = f_{FBA} \\ & l \le v \le u \end{aligned}$$

```
% Try all non-convex approximations of zero norm and take the best result
approximations = {'cappedL1','exp','log','SCAD','lp-','lp+'};
bestResult = n;
bestAprox = '';
for i=1:length(approximations)
 solution = sparseLP(char(approximations(i)),constraint);
 if solution.stat == 1
  if bestResult > length(find(abs(solution.x)>eps))
   bestResult=length(find(abs(solution.x)>eps));
   bestAprox = char(approximations(i));
   solutionL0 = solution;
  end
 end
end
```

Now we call the sparse linear step function approximations

```
bestResult = n;
bestAprox = '';
```

```
for i=1:length(approximations)
    solution = sparseLP(char(approximations(i)),constraint);
    if solution.stat == 1
        nnzSol=nnz(abs(solution.x)>feasTol);
        fprintf('%u%s%s',nnzSol,' active reactions in the sparseFBA solution with ', char(appr
        if bestResult > nnzSol
            bestResult=nnzSol;
            bestAprox = char(approximations(i));
            solutionL0 = solution;
        end
    end
end
```

```
247 active reactions in the sparseFBA solution with cappedL1
247 active reactions in the sparseFBA solution with exp
247 active reactions in the sparseFBA solution with log
247 active reactions in the sparseFBA solution with SCAD
247 active reactions in the sparseFBA solution with lp-
247 active reactions in the sparseFBA solution with lp+
```

Select the most sparse flux vector, unless there is a numerical problem.

```
if ~isequal(bestAprox,'')
    vBest = solutionL0.x;
else
    vBest = [];
    error('Min L0 problem error !!!!')
end
```

Report the best approximation

```
display(strcat('Best step function approximation: ',bestAprox))
```

```
Best step function approximation:cappedL1
```

Report the number of active reactions in the most sparse flux vector

```
fprintf('%u%s',nnz(abs(vBest)>feasTol),' active reactions in the best sparse flux balance anal
```

```
247 active reactions in the best sparse flux balance analysis solution.
```

Warn if there might be a numerical issue with the solution

```
feasError=norm(constraint.A * solutionL0.x - constraint.b,2);
if feasError>feasTol
    fprintf('%g\t%s\n',feasError, ' feasibily error.')
    warning('Numerical issue with the sparseLP solution')
end
```

REFERENCES

[1] Le Thi, H.A., Pham Dinh, T., Le, H.M., and Vo, X.T. (2015). DC approximation approaches for sparse optimization. European Journal of Operational Research 244, 26–46.

[2] Thiele, I., Swainston, N., Fleming, R.M., Hoppe, A., Sahoo, S., Aurich, M.K., Haraldsdottir, H., Mo, M.L., Rolfsson, O., Stobbe, M.D., et al. (2013). A community-driven global reconstruction of human metabolism. Nat Biotechnol 31, 419-425.