genetic Minimal Cut Sets - gMCSs

Authors: Iñigo Apaolaza, University of Navarra, TECNUN School of Engineering, Spain

Luis V. Valcarcel, University of Navarra, TECNUN School of Engineering, Spain

Francisco J. Planes, University of Navarra, TECNUN School of Engineering, Spain

Reviewer(s):

INTRODUCTION

Minimal Cut Sets (MCSs) are minimal sets of reaction knockouts which deprive a network from accomplishing a given metabolic task[1]. On the other hand, genetic Minimal Cut Sets consist of minimal sets of genes whose simultaneous inhibition would render the functioning of a given metabolic task impossible[2]. Therefore, the concepts of MCSs and gMCSs are equivalent but at different levels: at the reaction level for the former and at the gene level for the latter.

MCSs are not directly converted into feasible knockout strategies at a gene level due to the fact that the Gene-Protein-Reaction (GPR) rules, typically provided in genome-scale metabolic reconstructions, are not one-to-one. Certainly, we can find impractical reaction knockouts because they are catalyzed by enzymes involving several genes, which increase the cost of the intervention. On the other hand, the resulting gene knockout interventions from MCSs may provoke undesired metabolic effects, as we may have reactions (included in a particular MCS) catalyzed by an enzyme that additionally participate in other relevant reactions (not included in the MCS under consideration). This fact is further discussed in Apaolaza et al., 2017(b)[3].

Moreover, state-of-the-art methods that search for efficient gene knockout interventions use a combinatorial strategy and, as a consequence, are not viable for seeking for high order solutions in large metabolic networks. This tutorial aims also to demonstrate this is possible with the function provided.

EQUIPMENT SETUP

Initialize The Cobra Toolbox and select the solver (~20 sec)

If necessary, initialise the cobre toolbox:

```
initCobraToolbox
```

```
                                                     |
      _____   _____   _____   _____     _____        |
     /  ___| /  _  \ |  _  \ |  _  \   / ___ \       |   COnstraint-Based Reconstruction and Analysis
     | |     | | | | | |_| | | |_| |  | |___| |      |   The COBRA Toolbox - 2017
     | |     | | | | |  _  { |  _  /  |  ___  |      |
     | |___  | |_| | | |_| | | | \ \  | |   | |      |   Documentation:
     \_____| \_____/ |_____/ |_|  \_\ |_|   |_|      |   http://opencobra.github.io/cobratoolbox
                                                     |

 > Checking if git is installed ...  Done.
fatal: not in a git directory
Warning: Your global git configuration could not be changed.

 > 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 (this may take a while)... Done.
> Adding all the files of The COBRA Toolbox ...  Done.
> Define CB map output... set to svg.
> 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: C:\gurobi750\win64\matlab
  - [----] TOMLAB_PATH: --> set this path manually after installing the solver ( see instructions )
  - [----] 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  active            0      0      0      0      -
dqqMinos      active            0      -      -      -      -
glpk          active            1      1      -      -      -
gurobi        active            0      0      0      0      -
ibm_cplex     active            1      1      1      -      -
matlab        active            1      -      -      -      1
mosek         active            0      0      0      -      -
pdco          active            1      -      1      -      -
quadMinos     active            0      -      -      -      0
tomlab_cplex  active            0      0      0      0      -
qpng          passive           -      -      1      -      -
tomlab_snopt  passive           -      -      -      -      0
gurobi_mex    legacy            0      0      0      0      -
lindo_old     legacy            0      -      -      -      -
lindo_legacy  legacy            0      -      -      -      -
lp_solve      legacy            0      -      -      -      -
opti          legacy            0      0      0      0      0
-------------------------------------------------------------------
Total         -                 4      2      3      0      1

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


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

> Checking for available updates ...
> The COBRA Toolbox is up-to-date.
Warning: Your global git configuration could not be restored.
```

Select the appropriate solver removing the "%" symbol only for the desired solver. Note that the approaches to search for MCS and gMCS problems are based on Mixed Integer Linear Programming (MILP).

```
changeCobraSolver('ibm_cplex', 'all');
```

```
> IBM ILOG CPLEX interface added to MATLAB path.
> The solver compatibility is not tested with MATLAB R2017a.
> Solver for LP problems has been set to ibm_cplex.

> IBM ILOG CPLEX interface added to MATLAB path.
> The solver compatibility is not tested with MATLAB R2017a.
```

```
> Solver for MILP problems has been set to ibm_cplex.

> IBM ILOG CPLEX interface added to MATLAB path.
> The solver compatibility is not tested with MATLAB R2017a.
> Solver for QP problems has been set to ibm_cplex.
> Solver ibm_cplex not supported for problems of type MIQP. No solver set for this problemtype
> Solver ibm_cplex not supported for problems of type NLP. Currently used: matlab
```

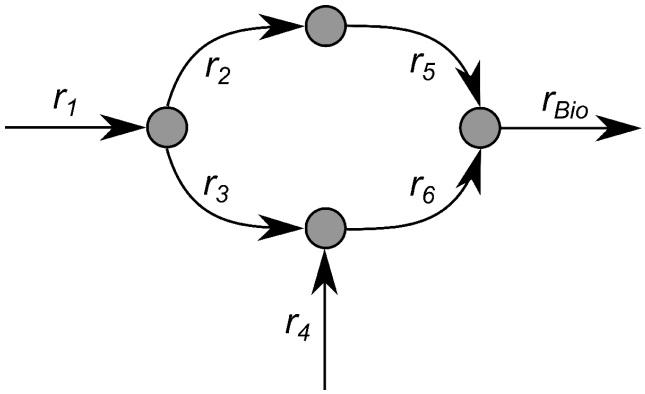
```
% changeCobraSolver('gurobi', 'all');
```

PROCEDURE

This tutorial will be divided into two different parts. First, a toy example will be used to illustrate the difference between MCSs and gMCSs and, second, gMCSs will be calculated for Recon2.v04 metabolic model[4].

1. Toy example (10 sec ~ 1 min)

The toy example under study is presented below:



First, we are going to load the MAT-file which contains the metabolic network in the toy example.

```
load('gMCStoyExample.mat');
```

As different metabolic models in COBRA format, it contains the following fields: *S*, *ub*, *lb*, *rxns*, *mets*, *genes*, *rules*, *grRules*, *rxnGeneMat* and *c*.

```
rxns = model.rxns
```

```
rxns = 7×1 cell array
    'r1'
    'r2'
    'r3'
    'r4'
    'r5'
    'r6'
    'rBio'
```

```
grRules = model.grRules
```

```
grRules = 7×1 cell array
    '(g1)'
    '(g2)'
    '(g2) or (g3)'
    '(g4)'
```

```
    '(g5) and (g6)'
    '(g5)'
    ''
```

We can observe that all reactions and genes have a one-to-one relationship except for *r3* and *r5*, which have rules containing 2 different genes and an "or" and an "and" relationship, respectively. On the other hand, *g2* and *g5* take part in two different reactions each. The existence of non-trivial relationships between genes and reactions are the cause of not obtaining the same solutions with MCSs and gMCSs.

We aim to calculate MCSs which will render the biomass production impossible via the function named *calculateMCS()*. Therefore, the desired target metabolic task (represented in the field *c* of the structure) will be the biomass reaction.

```
metab_task = model.rxns(logical(model.c))
```

```
metab_task = cell
    'rBio'
```

We will calculate 10 MCSs.

```
n_MCS = 10;
```

If needed, we can change the different optional arguments of the functions. In this case, we will calculate MCSs involving reactions 1 to 6. The biomass reaction (*rBio*) is omitted since it is the target metabolic task.

```
optional_inputs.rxn_set = {'r1'; 'r2'; 'r3'; 'r4'; 'r5'; 'r6'};
```

The time limit for the calculation of each MCS will be set to 30 seconds.

```
optional_inputs.timelimit = 30;
```

Although not shown here, other optional arguments may be set, as detailed in the documentation of the function.

We will now proceed with the calculation of the MCSs.

```
[MCSs, MCS_time] = calculateMCS(model, n_MCS, optional_inputs);
```

```
Calculating 10 MCSs...
10%     [....                                   ]20%     [........                                   ]30%
All existing MCSs have been calculated.
```

Despite having tried to calculate 10 MCSs, only 7 will be calculated. Notice that the last MCS is NaN, meaning that there only exist 6 MCSs for this toy example. The results are shown in the following piece of code:

```
MCSs{1}
```

```
ans = 2×1 cell array
    'r1'
    'r4'
```

```
MCSs{2}
```

```
ans = 2×1 cell array
    'r1'
    'r6'
```

```
MCSs{3}
```

```
ans = 2×1 cell array
    'r2'
    'r6'
```

```
MCSs{4}
```

```
ans = 2×1 cell array
    'r5'
    'r6'
```

```
MCSs{5}
```

```
ans = 3×1 cell array
    'r3'
    'r4'
    'r5'
```

```
MCSs{6}
```

```
ans = 3×1 cell array
    'r2'
    'r3'
    'r4'
```

```
MCSs{7}
```

```
ans = NaN
```

We now translate these minimal reaction knockout strategies to the gene level following the *grRules*. The following table shows the 8 possible genetic interventions which must be fulfilled to accomplish the respective reaction knockouts:

Minimal Cut Set	Gene knockout
r_1, r_4	g_1, g_4
r_1, r_6	g_1, g_5
r_2, r_6	g_2, g_5
r_5, r_6	g_5
	g_5, g_6
r_2, r_3, r_4	g_2, g_3, g_4
r_3, r_4, r_5	g_2, g_3, g_4, g_5
	g_2, g_3, g_4, g_6

In order to check if the gene knockouts that have to be carried out to perform the inhibition of the found MCSs are minimal, we will calculate gMCSs for the same toy example using the function *calculateGeneMCS()*.

First, we need to define the name of the model to create the gene knockout constraints matrix, *G*. The binary *G* matrix defines for each row the set of blocked reactions arising from the knockout of an irreducible subset of genes. The subset of genes associated with each row in G is interrelated and their simultaneous knockout is required to delete at least one of the reactions in the metabolic network. This matrix may be needed for other calculations. We will name here 'toy_example_gMCS' and again will calculate 10 gMCSs.

```
model_name = 'toy_example_gMCS';
n_gMCS = 10;
```

Next, we set the optional inputs. The maximum time for the calculation of each gMCS will be 30 seconds.

```
optional_inputs.timelimit = 30;
```

We will now proceed with the calculation of the gMCSs.

```
[gMCSs, gMCS_time] = calculateGeneMCS(model_name, model, n_gMCS, optional_inputs);
```

```
Calculating 10 gMCSs...
10%     [....                                   ]20%     [........                                   ]30%
All existing gMCSs have been calculated.
```

In the same way as with MCSs, 4 gMCSs have been calculated in total. As the last on is NaN, the toy example only has 3 gMCSs.

```
gMCSs{1}
```

```
ans = cell
    'g5'
```

```
gMCSs{2}
```

```
ans = 2×1 cell array
    'g1'
    'g4'
```

```
gMCSs{3}
```

```
ans = 3×1 cell array
    'g2'
    'g3'
    'g4'
```

```
gMCSs{4}
```

```
ans = NaN
```

As shown in the previous table, calculating MCSs would result in 8 different genetic intervention strategies, even when, as we have just demonstrated, only 3 minimal genetic interventions exist. Moving on to gMCSs seems, therefore, a more efficient strategy to calculate optimal gene knockout interventions.

2. gMCSs in large metabolic networks (20 min ~ 1 hour)

The algorithm presented in this tutorial is able to calculate intervention strategies in large metabolic networks. In addition, it is sufficiently flexible to calculate gMCSs involving a particular gene knockout. Interesting results regarding this issue can be found in Apaolaza et al., 2017(a)[2].

We are now going to calculate 6 gMCSs for the human metabolic model Recon2.v04[4] involving gene RRM1 (Entrez ID: 6240). To do so, the folllowing piece of code has to be executed.

First, we are going to load the metabolic model.

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

Next, in the same way as in the toy example, we will proceed to set the name of the model ('Recon2'), the number of gMCSs to be calculated (6) and the optional input variables. Regarding the latter, the maximum time for the calculation of gMCSs will be set to 2 minutes (120 seconds), the KO will be '6240', namely the Entrez ID of RRM1. In this case, we will also have to set the "separate_transcript" field to '.', because, if we do not, the calculation of gMCSs will be done at the transcript level as a consequence of the usage of the aforementioned character in Recon2.v04 to differentiate the transcripts of the same gene.

```
model_name = 'Recon2';
n_gMCS = 6;
optional_inputs.KO = '6240';
optional_inputs.separate_transcript = '.';
optional_inputs.timelimit = 2*60;
```

Finally, we will calculate the gMCSs. Turning the parallel pool on is recommended if the G matrix has not been calculated yet.

```
% parpool;
[gMCSs, gMCS_time] = calculateGeneMCS(model_name, modelR204, n_gMCS, optional_inputs);
```

```
Calculating 6 gMCSs...
16%     [......                                  ]33%     [.............                           ]50%

All existing gMCSs have been calculated.
```

The results obtained are the following:

```
gMCSs{1}
```

```
ans = 2×1 cell array
    '2987'
    '6240'
```

```
gMCSs{2}
```

```
ans = 3×1 cell array
    '1716'
    '6240'
    '7296'
```

```
gMCSs{3}
```

```
ans = 4×1 cell array
    '1633'
    '1635'
    '6240'
    '7296'
```

```
gMCSs{4}
```

```
ans = 4×1 cell array
    '26289'
    '51727'
    '6240'
    '7296'
```

```
gMCSs{5}
```

```
ans = 5×1 cell array
    '2766'
    '3251'
    '51292'
    '6240'
    '8833'
```

```
gMCSs{6}
```

```
ans = NaN
```

5 gMCSs have been calculated involving 2, 3, 4, 4 and 5 genes, respectively. It is important to note that no more gMCSs have been found as a consequence of setting the time limit to 2 minutes. If we increase the time limit, more gMCSs involving RRM1 will be calculated, which will involve a higher number of genes. It is important to note that the calculation of a minimal intervention strategy involving 5 genes with an exhaustive combinatorial strategy is actually inviable.

TIMING

1. Equipment Setup: ~20 sec.
2. Toy Example: 10 sec ~ 1 min.
3. gMCSs in large metabolic networks: ~20 min -1 hour.

Note that both *calculateMCS()* and *calculateGeneMCS()* functions return MCS_time and gMCS_time, respectively, which contain the timing for all the solving steps.

REFERENCES

1. von Kamp, A. & Klamt, S. Enumeration of smallest intervention strategies in genome-scale metabolic networks. *PLoS computational biology*, *10*(1), e1003378 (2014).

2. Apaolaza, I., *et al*. An in-silico approach to predict and exploit synthetic lethality in cancer metabolism. *Nature Communications*, *8*(1), 459 (2017).

3. Apaolaza, I., José-Eneriz, E. S., Agirre, X., Prósper, F. & Planes, F. J. COBRA methods and metabolic drug targets in cancer. *Molecular & Cellular Oncology*, (just-accepted), 00-00 (2017).

4. Thiele, I., *et al*. A community-driven global reconstruction of human metabolism. *Nature biotechnology*, *31*(5), 419-425 (2013).