

CNApy Guide

This manuscript ([permalink](#)) was automatically generated from [cnapy-org/CNApy-guide@4f4eccc](#) on August 20, 2021.

Authors

- **Sven Thiele**

•  [0000-0002-5812-6963](#) •  [sthiele](#)

Analysis and Redesign of Biological Networks, Max Planck Institute for Dynamics of Complex Technical Systems
Magdeburg

- **Axel von Kamp**

•  [axelvonkamp](#)

Analysis and Redesign of Biological Networks, Max Planck Institute for Dynamics of Complex Technical Systems
Magdeburg

Introduction

This is the user manual of CNApy. CNApy is a graphical interface for the modelling and analysis of metabolic networks on the basis of constraint-based (stoichiometric) modeling approaches. It allows

- Importing/exporting SBML models
- Creating COBRApy/CNA models
- Linking a graphical representation to model data
- Convenient exploration and editing of the model
- Model analysis with standard and advanced constraint-based methods, and
- Saving everything as a CNApy *.cna project

The methods provided for model analysis (some of them are part of CNApy but most are functions from CNA and COBRApy) include:

- Flux balance analysis
- Parsimonious flux balance analysis
- Flux variability analysis
- Minimal cut sets
- Elementary modes
- Phase plane analysis
- Yield optimization

CNApy is available at <https://github.com/cnapy-org/CNApy> under the [Apache-2.0 License](#).

We appreciate any comments or suggestions for improvements and we are greatly interested in your feedback which you can give at our [User Forum](#).

For feature requests and bug reports please use our [Issue tracker](#)

Thank you for using CNApy!

Installation

The easiest way to install CNAPy is using the conda package manager. First install [miniconda](#) and then:

- Create a conda environment with all dependencies

```
conda create -n cnapy-1.0.0 -c conda-forge -c cnapy cnapy=1.0.0
```

- Activate the cnapy conda environment

```
conda activate cnapy-1.0.0
```

- Run CNAPy

```
cnapy
```

On windows you may also use the [graphical installer](#).

Configuration

When we start CNAPy for the first time we are greeted with a dialog like this.

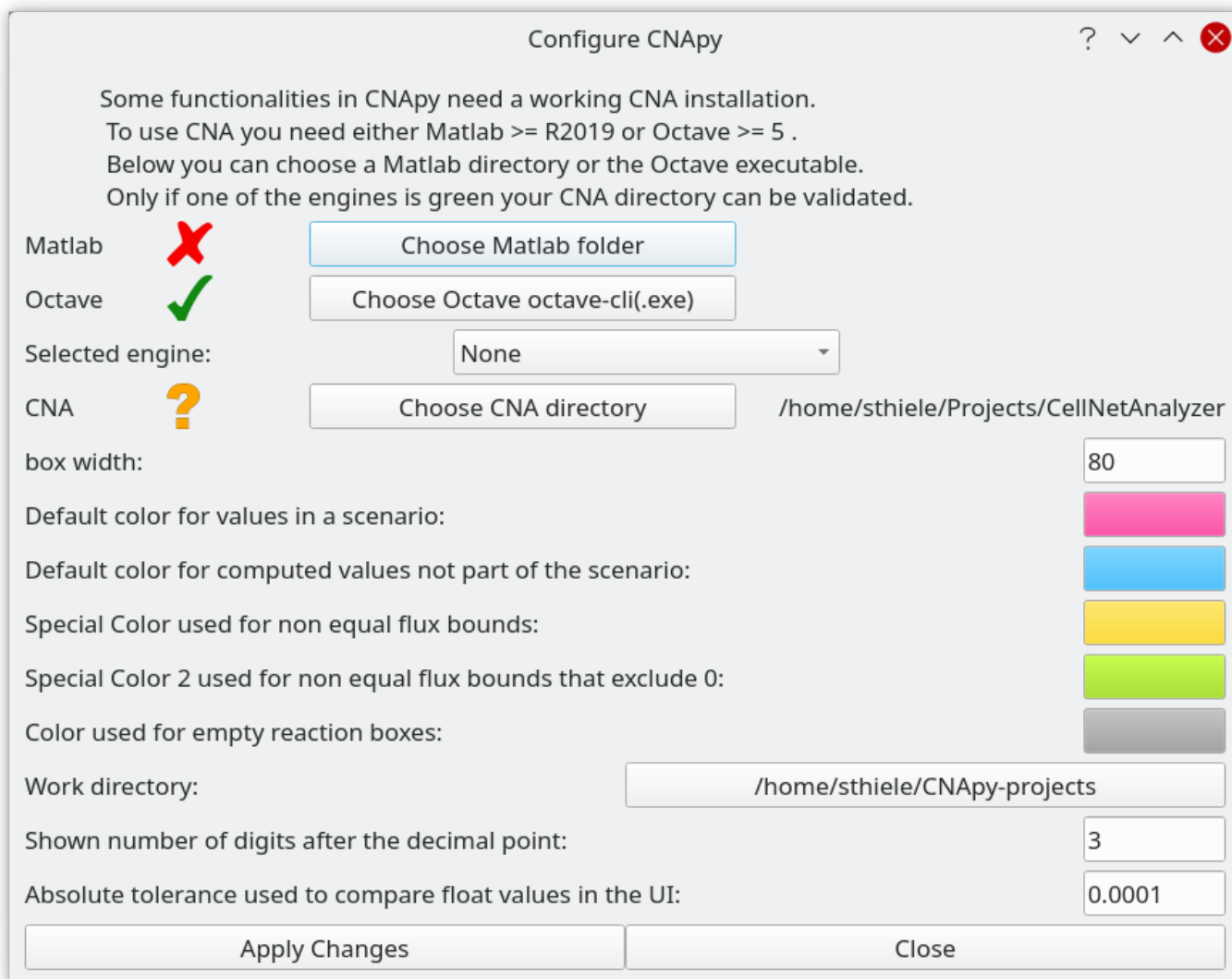


Figure 1: CNAPy Configuration dialog.

With this dialog you can configure which Matlab/Octave installation CNAPy should use.

The minimal cut set computation in CNAPy uses functions that are provided by CellNetAnalyzer. To be able to use these functions you have to provide CNAPy with either a path to a Matlab installation >R2019 or the path to an Octave executable >4, and of course with a path to a recent CellNetAnalyzer installation. You can do this by clicking these buttons (1). If you change the settings CNAPy performs some basic tests to make sure everything is working. If all checks are successful green check marks are shown, if a check fails a red cross is shown.

If no check can be performed, a yellow question mark is shown. For example, the check of the CNA directory needs a working Matlab or Octave installation.

You can also configure which colors CNAPy should use to highlight reactions on the map or in the reaction list.

User interface overview

This section gives an introduction to the CNAPy UI and its components and functions.

When we have finished our first configuration we are greeted with a window like this.

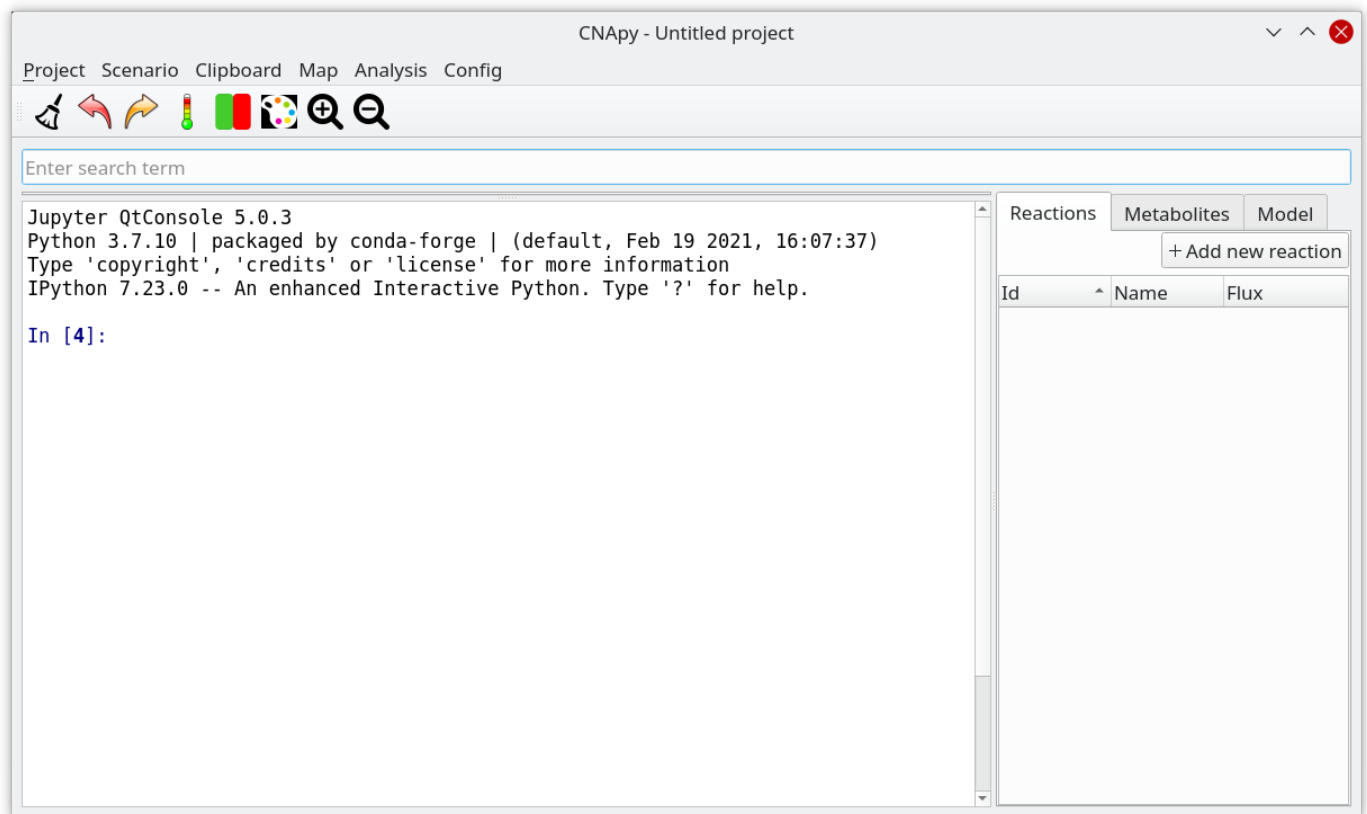


Figure 2: CNAPy with an empty project.

This is an empty project on the right hand side we see the empty lists of reactions and metabolites. On the left we see the embedded Jupyter Console where one can interact programmatically with the model and UI. We can create a new project by importing SBML models, editing the reactions and adding graphical maps. We can also load one of the projects included in our projects repository.

Let's go to Project in our menubar and open the `ECC2comp.cna` project.

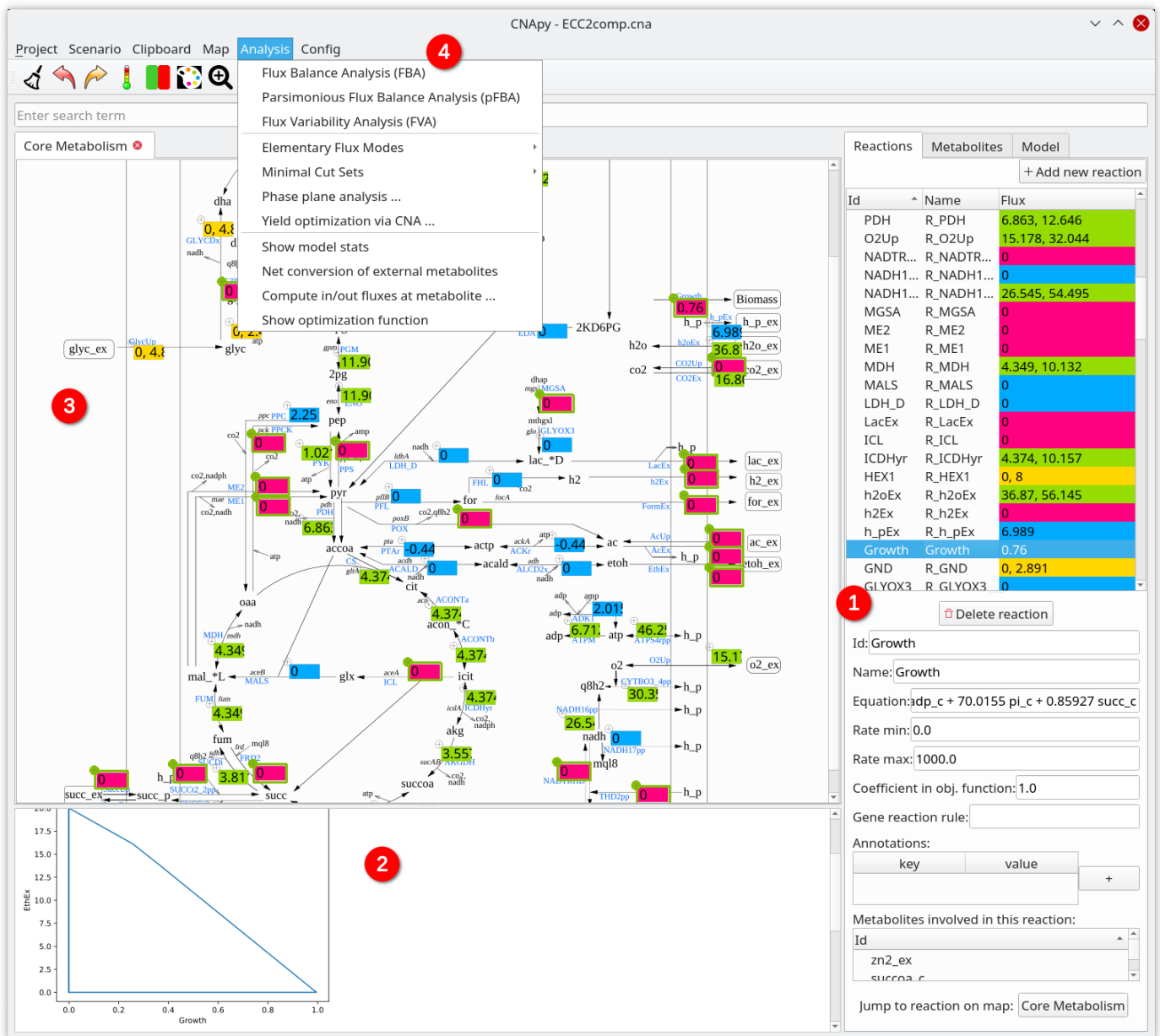


Figure 3: CNApy with the ECC2comp project.

In this picture we see CNApy with the open ECC2comp project. On the right hand side (1) we see the populated reactions and metabolites lists with the color coded current values and a form with corresponding details for the selected reaction/metabolite. The reactions list contains buttons that lets us add/delete reactions to/from the model. Changing the metabolite identifier or other details in the metabolite/reaction form has an instant effect on the corresponding reactions/metabolites in the model.

The console (2) now shows the output of some computations, and above the console we have a map view (3) with a graphical representation of our network. On top (4) we have the menu bar which gives us access to the various functionalities of CNApy and a Toolbar for quick access to often used functions.

We can add new maps via the Map menu, and drag reactions from the reaction list onto the desired position on the map.

Create a new project

In this Section we explain how to create a new network project. We also have a video on that topic <https://youtu.be/bsNXZBmtYWw>.

CNApy starts with an empty Project. You can directly add new reactions via the reaction mask on the right. By pressing the *Add new reaction* button.

Then simply type the reaction formula into the Equation field. For example $A + 2 B \rightarrow C$.

The associated metabolites A , B and C are then automatically added to the model.

Changing metabolite identifiers in the metabolite form instantly rewrites the reaction equation of the associated reactions.

Creating big models by adding single reactions is very cumbersome. Therefore CNApy allows you to create a new project from an SBML file. You find the point *New project from SBML* in the *Project* menu.

After importing an SBML model the reaction list is populated with all the reactions from the model.

You can add a map to our project, by going to the *Map* menu and clicking *add new map*.

From the *Map* menu we can also *change the background image* of our map. Any SVG image can be used as a background.

You can add multiple maps that highlight different aspects of your model.

Reaction boxes can be put on the map by dragging over them from the reactions list.

You can adjust the box size with the keyboard shortcuts $\text{Ctrl} + +$ and $\text{Ctrl} + -$. You can move the reaction box to the desired position by dragging them on their handles. If you want to move all reaction boxes together, press Ctrl while dragging.

Positioning a big number of reaction boxes can be tiresome. CNApy allows you to load and save reaction box positions. This facilitates reuse among similar projects.

The size of the background image can be adjusted with the shortcuts $\text{Ctrl} + \text{Shift} + +$ and $\text{Ctrl} + \text{Shift} + -$.

To save a project click *Save Project as* in the *Project* menu. It is important to add the `.cna` extension to the file name. This way your files can be found by CNApy.

Scenarios

In CNApy you can define a scenario under which metabolic analysis like the FBA will be performed. In essence a scenario is a set of flux constraints for a set of reactions. These constraints can fix the flux of a reaction or constraint lower and upper bounds for the flux.

The scenario can be edited via the map, by entering values into the corresponding reaction boxes. Accepted values are either a single float like `1.5` or a pair of floats like `(-10, 1.2)`. A single float fixes the flux of the reaction to this value (`1.5`). While a pair sets the lower flux bound of the reaction to the first value (`-10`) and the upper flux bound to the second value (`1.2`).


 Figure 4: Reaction box with scenario value.

Figure 4: Reaction box with scenario value.

Reactions boxes with scenario constraints are marked by the scenario color and a green frame. The frame turns yellow, if the scenario constraint contradicts the reaction constraints of the model.

To remove the constraints on a reaction simply delete the scenario value in the reaction box.

You can save and load scenarios as `*.scen` files. One scenario can be set as the default scenario of your project. This can be done via *set current scenario as default scenario* in the *Scenario* menu. The project must then be saved, and the next time you open the project the scenario is already set.

CNApy implements an edit history for the scenario with the tool buttons you can undo and redo your changes to the scenario. You can also import all the values that are currently in the reaction boxes into the scenario, and you can change the model's reaction bounds to the current scenario values.

Analysis functions

Flux balance analysis (FBA)

Optimizes the value of the current objective function and shows a flux distribution that realizes this optimum. The objective function is a linear function over the reaction rates in the network, its coefficients can be modified in the reaction dialogs. To display the current objective function you can choose *Show optimization function* from the *Analysis* menu. The current objective function is shown in the console. Note that the flux distribution calculated by FBA is usually not unique.

Parsimonious FBA (pFBA)

In parsimonious FBA first the objective function is optimized and then with this optimum as additional constraint the sum of the fluxes in the network is minimized. This has the advantage that redundant internal cyclic fluxes are suppressed and that a lot of reactions will have zero flux which makes the solution easier to understand.

Flux variability analysis (FVA)

In FVA the maximal/minimal rate of each reaction is calculated. This gives the possible flux range for each reaction.

Elementary modes (EFM)/elementary flux vectors (EFV)

EFM can be calculated via CNA or directly via `efmtool` while for EFV CNA is required.

 Figure 5: Elementary Flux Mode Computation Dialog.

Figure 5: Elementary Flux Mode Computation Dialog.

Options for CNA and `efmtool`

When reactions are marked with 0 flux in a scenario and the option "consider 0 in current scenario as off" is activated then only the subset of EFM/EFV is calculated in which these reactions do not participate

Additional options for CNA

- use flux bounds to calculate elementary flux vectors To calculate EFV instead of EFM, activate this option. Flux bounds defined in the reactions and the current scenario are then taken into account as inhomogeneous constraints. A flux bound is only included when its absolute value is greater than the value given as "Threshold for bounds to be unconstrained".
- check reversibility

Unchecking this option will lead to all reactions being considered reversible even if they have a rate minimum ≥ 0 .

- only convex basis

Currently has no effect (need to be able to switch computation method first).

- consider isozymes only once

If a group of reactions has the same stoichiometry then only the modes with one representative of that group are calculated.

- use rational numbers

Use rational numbers of arbitrary length for EFM/EFV computation (efmtool part only, results will still appear as floating point numbers) To calculate EFV instead of EFM, activate this option. Flux bounds defined in the reactions and the current scenario are then taken into account as inhomogeneous constraints. A flux bound is only included when its absolute value is greater than the value given as "Threshold for bounds to be unconstrained".

Minimal cut sets

Minimal cut sets (MCS) can be calculated in CNApy using the dual method.

Minimal Cut Sets Computation
?
^
x

Target Region(s)

region no	T	\geq/\leq	t
1 1		\leq	0

+
-

Desired Region(s)

region no	D	\geq/\leq	d
1 1		\leq	0

+
-

☐ Gene KOs
☐ Exclude boundary reactions as cuts

Max. Solutions inf
Max. Size 7
Time Limit [sec] inf

☐ Consider constraint given by scenario
☐ Advanced: Define knockout/addition costs for genes/reactions

Solver

☐ glpk (optlang)
☐ CPLEX (MATLAB)
☐ CPLEX (Octave)
☐ intlinprog (MATLAB)
☒ GLPK (Octave/MATLAB)

MCS search

☒ any MCS (fast)
☐ by cardinality
☐ smallest MCS first

Compute MCS
Close

Figure 6: Minimal cut sets computation dialog.

- Target and Desired region(s)

For the dual method one or more target regions must be defined which describe the network behaviour that is to be suppressed. In addition, one or more desired regions may be defined which describe the network behaviour that is to be preserved. Each target/desired region is defined by a set

of linear equality constraints over the reaction rates in the network. All target and desired regions must be initially feasible or an error will be raised. Also make sure that 0 is not included in any target or desired region (0 as target cannot be suppressed and 0 as desired is always fulfilled).

Note that all non-default reaction bounds (as defined by `cobra.Configuration().bounds`) are automatically added to all target and desired regions before MCS computation.

- Solver

MCS can either be calculated via CNA (using either Octave or Matlab) or via optlang using different MILP solvers. For CNA the solver must be chosen while optlang uses the solver for the current model (cf. Solver for current model). Not all variants and solvers allow for the same options so these will partly be disabled depending on the chosen solver.

- Max. solutions

Maximal number of MCS to calculate.

- Max. size

Only calculate MCS which consists of up to this number of cuts.

- Time limit

Do not run the search for longer than this limit. It is strongly recommended to use this parameter because MCS computation can take very long especially when searching for smallest MCS.

- MCS search

MCS can be enumerated iteratively, either the smallest first (least number of cuts) or any MCS (up to the number of cuts given by max. size). With the latter option new MCS are found more quickly but may not be the smallest.

With CPLEX or Gurobi as solver, MCS can also be enumerated by cardinality, i.e. all MCS of successively increasing size (up to max. size) are computed. In addition, with these solvers a continuous search mode is possible which is similar to the any MCS method but often faster.

- Gene KOs (CNA only)

If the model contains the gene-reaction association, search for gene knock-outs instead of reaction knock-outs.

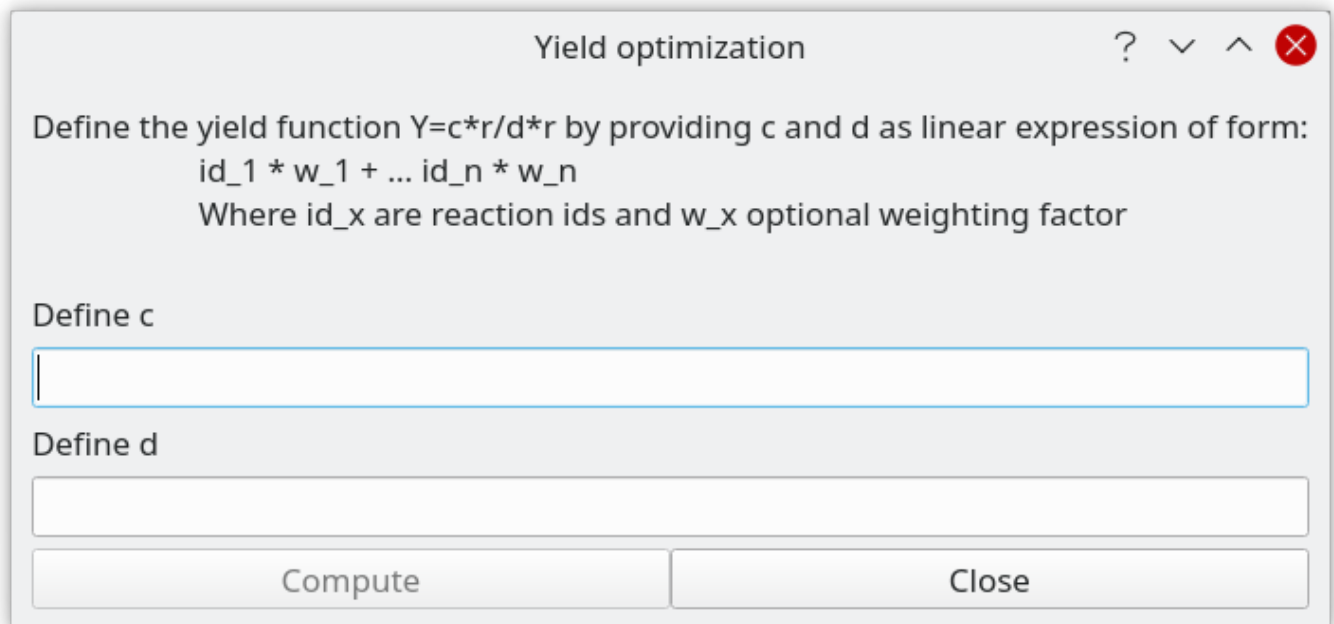
- Exclude boundary reactions as cuts (optlang only)

Do not allow knock-out of boundary reactions (reactions that cross the system boundary, e.g. exchange reactions).

- Consider constraints given by scenario

Take reaction bounds set by the current scenario into account for all target and desired regions.

Yield optimization



Yield optimization

Define the yield function $Y=c*r/d*r$ by providing c and d as linear expression of form:
 $id_1 * w_1 + \dots id_n * w_n$
 Where id_x are reaction ids and w_x optional weighting factor

Define c

Define d

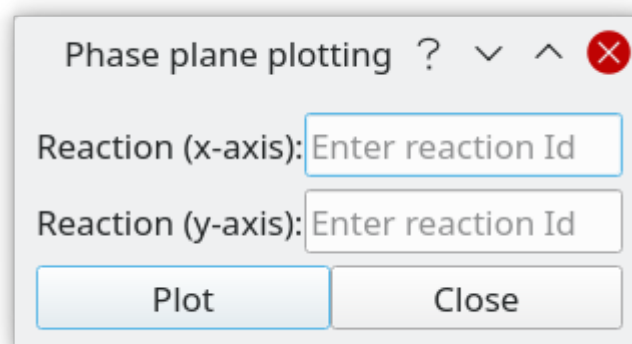
Compute
Close

Figure 7: Yield optimization dialog.

Maximizes the value of a yield function c/d where c and d are linear expressions over the fluxes in the network. For this to work correctly, there must be no flux vector in the network where the denominator becomes zero and the nominator is non-zero. The flux values of the current scenario are taken into during the optimization.

A typical application would be the maximization of a product yield: When the exchange fluxes for ethanol and glucose are EX_etoh and EX_glc then maximizing $EX_etoh/-EX_glc$ gives the maximal ethanol yield on glucose.

Phase plane plot



Phase plane plotting

Reaction (x-axis):

Reaction (y-axis):

Plot
Close

Figure 8: Phase plane plot dialog.

Creates a plot (in the console) that shows the projection of the solution space onto a two-dimensional plane of two selected reaction rates. The flux values of the current scenario are taken into account during the calculations.

Compute in/out fluxes at a metabolite

If a flux distribution has been calculated (e.g. by FBA) this function shows the magnitude of all fluxes going in/coming out of a specified metabolite. The reaction fluxes are displayed as stacked bar graphs

on the console. You can either call this function from the Analysis menu and specify a metabolite or right-click on a metabolite in the list and call the function from the context menu.

Clipboard calculator

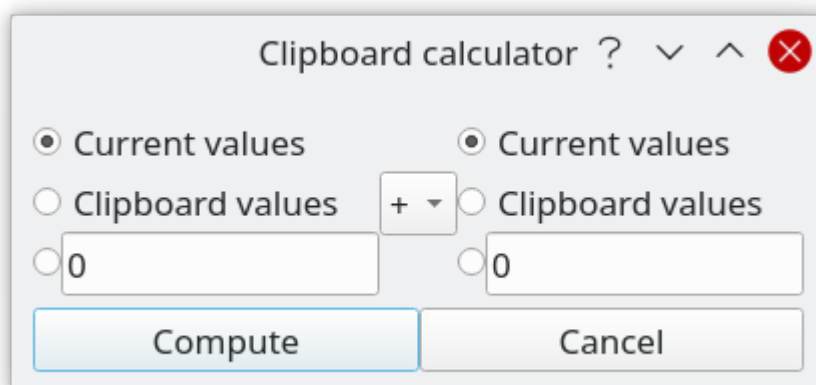


Figure 9: Clipboard calculator.

The clipboard calculator allows you to perform arithmetic operations with the values stored in the clipboard, the current reaction rates or a fixed value that you can enter in this dialog. The result of the operation replaces the current flux values.

Configure COBRApy

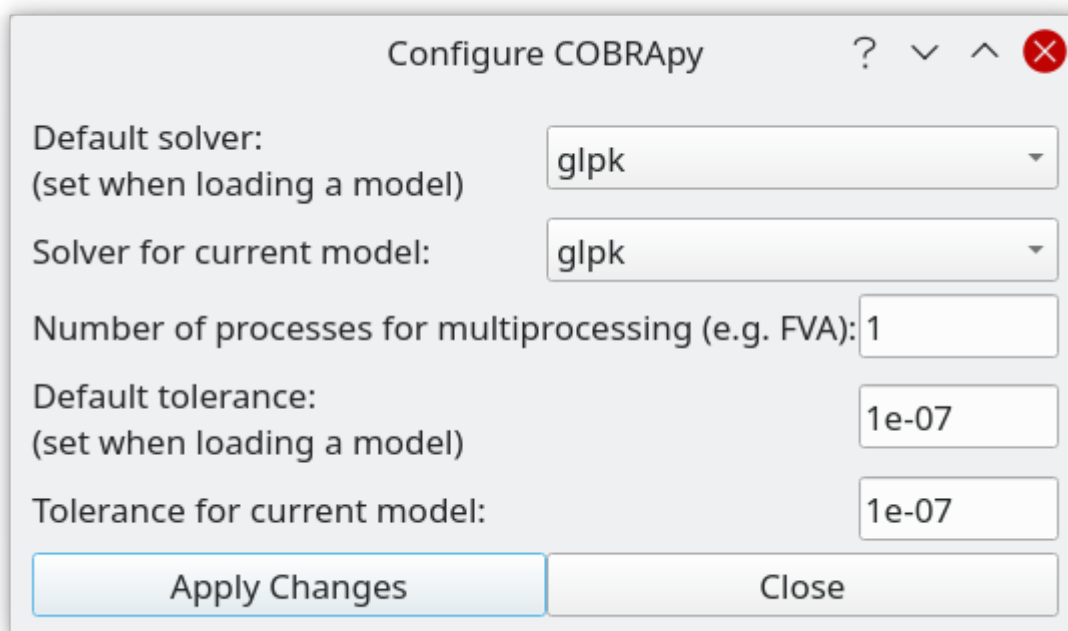


Figure 10: COBRApy configuration dialog.

With this dialog selected global settings (`class cobra.Configuration()`) of the COBRApy toolbox and selected parameters of the current model can be modified. The global settings are saved, but the parameters of the current model are not and will revert to the global default values are (re-)loading a model.

- Default solver (global)

Select which solver to use as global default from the dropdown list. COBRApy supports a variety of solvers but only the ones which are installed for COBRApy on your system will be available on the list. Also, different solvers have different capabilities so you should set a default that has all capabilities that you require (e.g. glpk_exact cannot be used for MILPs).

- Solver for current model

Here you can change the solver for the current model.

- Number of processes for multiprocessing (global)

Number of processes run in parallel when Python multiprocessing is used, e.g. for FVA. Setting this to 1 disables multiprocessing.

IMPORTANT

Should be set to 1 on Windows because Python multiprocessing performs very poorly on this OS.

- Default tolerance (global)

This value is used to distinguish zero from non-zero elements for various purposes. In particular, this value is propagated to the solver (see the COBRApy source code which solver parameters are affected). Most solvers restrict the range of this value, so it must be between 1E-9 and 0.1 For details where this value is used see the COBRApy documentation and source code.

- Tolerance for the current model

Here you can change the tolerance for the current model. You may enter an arbitrary value ≥ 0 here but if it is not compatible with the current solver then “Apply Changes” can lead to an error message.

Programming CNApy

Under Construction

- accessing the COBRApy Model
- accessing scenario and computed values
- accessing the CNApy UI

References
