JMassBalance Reference Manual

Georg Basler and Zoran Nikoloski
contact: georgbasler@yahoo.de
`http://mathbiol.mpimp-golm.mpg.de/massbalance/`

February 7, 2013

Contents

1 Preliminaries

Thank your for downloading *JMassBalance*, a tool for mass-balanced randomization and analysis of metabolic networks. This document describes the installation, general usage, and available parameters. The algorithms used for mass-balanced randomization and their uniformity and complexity properties are discussed in detail in Basler et al. (2011).

1.1 System requirements

The tool is written entirely in Java and should thus run on any common operating system with the Java runtime environment installed. It was developed with Sun Java SE 1.6 on Windows XP, and tested on Windows 7 and Fedora Linux.

1.2 Installation

Extract the files from the download package to a local directory. Make sure that the Java Runtime Environment 1.6 or higher is installed by opening a command line window and typing `java -version`. The output should look similar to `java version "1.6.0_11"`. Otherwise, install Java (see `http://www.oracle.com`).

1.3 SBML support

For randomizing SBML models, `libSBML` (Bornstein et al. (2008)) is required. For Windows, the files `libsbmlj.jar` and `sbmlj.dll` are included in the download package and provide SBML support without the need to install `libSBML`.

If you wish to use SBML models on Linux or Mac, `libSBML` has to be installed with Java support. Note that *JMassBalance* can also run without SBML support, so there is no need to install anything if you do not need SBML.

Refer to `http://sbml.org/Software/libSBML/docs/cpp-api/libsbml-installation.html` for installing `libSBML` with Java support on your system. After the installation, replace the file `libsbmlj.jar` in the directory where you extracted the *JMassBalance* package by the file from the `libSBML` installation. On Linux, the latter is usually located at `/usr/share/java/libsbmlj.jar`.

Additional steps may be required depending on your operating system. Refer to `http://sbml.org/Software/libSBML/docs/cpp-api/libsbml-accessing.html` for more information.

1.4 License

JMassBalance is distributed for free and without any warranty by the authors. If this software is useful for your work, please include one of the following references in your publication or redistribution:

Georg Basler, Oliver Ebenhöh, Joachim Selbig, and Zoran Nikoloski. Mass-balanced randomization of metabolic networks. *Bioinformatics*, 27(10):1397–1403, May 2011. doi: 10.1093/bioinformatics/btr145. URL `http://dx.doi.org/10.1093/bioinformatics/btr145`.

Georg Basler and Zoran Nikoloski. *JMassBalance*: mass-balanced randomization and analysis of metabolic networks. *Bioinformatics*, 27(19):2761–2762, July 2011. doi: 10.1093/bioinformatics/btr448. URL `http://dx.doi.org/10.1093/bioinformatics/btr448`.

The included library `jgrapht-jdk1.6.jar` is distributed under the terms of the GNU Lesser General Public License (see `http://www.jgrapht.org/LGPL.html`). For the library `arpack-combo-0.1.jar`, provided by the netlib-java project, the BSD 2-Clause License applies (see `http://www.opensource.org/licenses/bsd-license.php`). The Windows libraries `libsbmlj.jar` and `sbmlj.dll` are distributed under the terms of the GNU Lesser General Public License (see `http://sbml.org/Software/libSBML/LibSBML_License`).

2 Graphical User Interface

JMassBalance can be used via a graphical user interface (GUI) or from the command line. The GUI is started by opening the file `massbalance.jar` with the Java Runtime Environment (`java` executable). Depending on your system configuration, this may be done by clicking the file or by executing `java -jar massbalance.jar` from the command line.

The GUI is rather self-explanatory. There are two tabs at the top of the window: one for randomization, the other for calculating network properties. On the Randomization tab, you first choose a network file for randomization and the directory where the mass equivalence classes, log files, and randomized network files are placed. See Section 5 for the accepted input file formats.

On the left side, you may choose mass-balanced or switch randomization (see Subsections 3.1.1 and 3.1.2). Below, you may choose the number of randomized networks and their file names (see Subsection 3.1.15).

On the right side, you may select whether unbalanced reactions are fixed (Subsection 3.1.4), compartments are generated (Subsection 3.1.7), reactions are considered reversible (Subsection 3.1.8), reactions are randomized iteratively (Subsection 3.1.9), strict switch randomization (Subsection 3.1.14), the randomization depth (Subsection 3.1.10) and probability (Subsection 3.1.11).

After having generated randomized networks, you may analyze their network properties on the Properties tab. First enter the network name and directory generated during randomization (this is done automatically if you generated randomized networks before). Alternatively to specifying the network name, you may also select the network file. On the left side, select the randomization type and generated networks to analyze. On the right side, you may choose to calculate the average path length (Subsection 4.1.8, clustering coefficient (Subsection 4.1.9), assortativity (Subsection 4.1.16), degree distribution (Subsection 4.1.5), weights distribution (Subsection 4.1.6), scope size distribution (Subsection 4.1.7), or cycles of length `n` (Subsection 4.1.10).

Randomization and calculation of network properties is initiated by pressing the Start button. The console panel on the bottom displays information on the process. Note that, for accessing all features of *JMassBalance*, such as parallelization, you should use the command line, described in the next Section.

3 Randomization

Randomization is executed from the command line by typing

```
java -cp massbalance.jar massbalance/Randomize
```

in the directory where you extracted the download package. Additional parameters are added to specify the file containing the network to be randomized, the output directory, and other parameters for randomization. The syntax is as follows:

```
java -cp massbalance.jar massbalance/Randomize <networkFile>
                    <outputDir> [options]
```

`<networkFile>` specifies the (absolute or relative) path to the network file in BioCyc, SBML, or text file format, *e.g.* `/home/user/ecocyc/reactions.dat` (see Section 5 for a specification of the accepted file formats). `<outputDir>` specifies the directory where the mass equivalence classes, log files, and randomized network files are placed.

Example:

The following command generates 500 mass-balanced randomized networks from the network file `tca` included in the package, and places the mass equivalence classes, log files, and randomized network files in the subdirectory `outdir`:

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                    massbalance 0-499
```

Below is a brief summary of the available options. Detailed descriptions and examples are given in Subsection 3.1.

`massbalance`	Run mass-balanced randomization.
`switch`	Run switch randomization.
`noclasses`	Do not generate equivalence classes.
`nofix`	Do not fix unbalanced reactions.
`removeDuplicates`	Remove duplicate reactions with identical stoichiometry.
`dontRemoveCyclic`	Do not remove cyclic reactions where the substrates are identical to the products.
`compartments`	Generate a network with compartments.
`reversible`	Consider every reaction reversible.
`noniterative`	Reactions are chosen noniteratively for mass-balanced randomization.
`depth=[0,...]`	Mass-balanced randomization depth (0: no randomization, 1: default)
`p=[0,1]`	Probability of mass-balanced randomization of a chosen reaction (1: default).
`substitutability`	Print sample space and substitutability class size distributions.
`D=vertices...`	Preserve the given reactions/compounds.
`strict`	Run strict switch randomization.
`from-to`	Indices of randomized networks to generate and parallelization. (default: 0-999).
`continue`	Continue randomization on existing randomized object files.

3.1 Options

3.1.1 `massbalance`

By passing the `massbalance` argument, mass-balanced randomization will be applied to the network, as described in Basler et al. (2011). The first and most time consuming step is the generation of the mass equivalence classes from the compounds in the network. The mass equivalence classes are written to the file

```
<outputDir>/<name>.classes
```

where `<outputDir>` is the specified output directory, and `<name>` is the name of the network (see Section 5 for how the name is generated from the different file formats). The file is reused in later randomization runs, which significantly speeds up the subsequent executions.

The mass-equivalence classes are generated from the sum formulas of the compounds. To specify the names of chemical elements to consider, change the comma-separated list after `ELEMENTS` in the file `jmassbalance.config` included in the directory where you extracted the package. By default, the chemical elements carbon (C), hydrogen (H), nitrogen (N), oxygen (O), phosphorus (P), and sulfur (S) are considered.

By default, 1000 randomized networks are generated and placed into the subdirectory

```
<outputDir>/<name>-graphs/
```

The graphs are efficiently stored in a binary format, which allows the subsequent calculation of network properties (see Section 4).

As an example, when randomizing the network file `tca`, included in the package, with the command

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                        massbalance
```

the mass equivalence classes and log-files will be placed into the directory `outdir`, and the randomized networks will be placed into the subdirectory `outdir/tca-graphs`.

By passing the `from-to` option, the number and indices of randomized networks can be specified, and the calculation can be distributed among CPU cores (see Subsection 3.1.15).

If neither the `massbalance` nor the `switch` arguments are specified, then the original network is parsed, the mass equivalence classes are generated, but no randomization is carried out.

3.1.2 `switch`

By passing the `switch` argument, switch randomization will be applied to the network. A switch randomized network is obtained by repeatedly shuffling a randomly chosen pair of edges. For example, the two metabolic reactions: glucose isomerase (Glucose → Fructose) and maleate isomerase (Maleate → Fumarate) are randomized by: Glucose → Fumarate and Maleate → Fructose.

By default, 1000 randomized networks are generated and placed into the subdirectory

```
<outputDir>/<name>-graphs/
```

where `<outputDir>` is the specified output directory, and `<name>` is the name of the network (see Section 5 for how the name is generated from the different file formats). The graphs are efficiently stored in a binary format, which allows the subsequent calculation of network properties (see Section 4).

As an example, when randomizing the network file `tca`, included in the package, with the command

```
java -cp massbalance.jar massbalance/Randomize tca outdir switch
```

the log-files will be placed into the directory `outdir`, and the randomized networks will be placed into the subdirectory `outdir/tca-graphs`.

By passing the `from-to` option, the number and indices of randomized networks can be specified, and the calculation can be distributed among CPU cores (see Subsection 3.1.15).

If neither the `massbalance` nor the `switch` arguments are specified, then the original network is parsed, the mass equivalence classes are generated, but no randomization is carried out.

See also Subsection 3.1.14.

3.1.3 `noclasses`

By specifying the `noclasses` option, only the graph is created from the specified network file, but the time-consuming creation of mass equivalence classes is skipped. For subsequent mass-balanced randomization, creation of the mass-equivalence classes is required. This option allows to parse the network and masses files, and print summary statistics or calculate properties of the original network only.

3.1.4 `nofix`

By default, reactions in the original network which are unbalanced only due to one or more hydrogen atoms are fixed by applying the following procedure: (1) if the reaction contains phosphate (PO_4), hydrogen phosphate (HPO_4), dihydrogen phosphate (H_2PO_4), or phosphoric acid (H_3PO_4), and replacing this compound by one of the other hydrogen forms results in a balanced reaction, then the compound is replaced accordingly; (2) if the reaction side which lacks hydrogen atoms contains hydrogen, then its stoichiometric coefficient is increased accordingly; (3) otherwise, hydrogen is added to the corresponding side with appropriate stoichiometric coefficient.

To specify the names of the newly introduced vertices representing hydrogen, phosphate, hydrogen phosphate, dihydrogen phosphate, or phosphoric acid, change the values after `HYDROGEN _NAME` or `PHOSPHATE_NAMES` in the file `jmassbalance.config` included in the directory where you extracted the package.

By specifying the `nofix` option, unbalanced reactions are not fixed.

3.1.5 `removeDuplicates`

Two reactions which have identical substrates and products (within the same compartment) and with the same stoichiometric coefficients are considered duplicates. By default, if duplicate reactions are found in the network file, the corresponding duplicate reactions are created in the network. By specifying the `removeDuplicates` option, duplicate reactions are removed from the network. If one of the duplicates is reversible and the other irreversible, the irreversible reaction is removed.

3.1.6 `dontRemoveCyclic`

A reaction where the substrates are identical to the products (within the same compartment) is considered cyclic. By default, if cyclic reactions are found in the network file, they are removed from the network. By specifying the `dontRemoveCyclic` option, cyclic reactions are left in the network.

3.1.7 `compartments`

If the compounds or reactions in your network are annotated with compartments, you may specify this option to generate a compartment-specific network (see Section 5.3.1 for how to specify compartments in the customizable text file format). In a network without compartments, each compound species is represented by one vertex, regardless of its sub-cellular localization. By including compartments, a separate vertex is created for each compartment, in which the represented compound is located.

When randomizing a network with compartments, compounds will only be substituted within the same compartment, and import/export/transport reactions will be left unchanged.

3.1.8 `reversible`

If specified, every reaction is considered reversible. Consequently, for each reaction, two vertices are created: one which has the substrates as predecessors and the products as successors, and another one for the reverse direction, which has the products as predecessors and the substrates as successors. By default, the reversibility of a reaction is taken from the corresponding annotation in the network file.

3.1.9 `noniterative`

By default, mass-balanced randomization is applied by iterating over all reactions of the network, and applying $(d^2 + d)/2$ substitutions to each side of a reaction, where d is the number of substrates (respectively, products). If `noniterative` is specified, then a reaction is chosen uniformly at random, and one substitution is applied. This process is repeated $t = \lceil |V_r| \cdot \overline{d}(V_r)^2 \rceil$ times, where $\overline{d}(V_r)$ is the average (undirected) reaction degree (see Basler et al. (2011), Supplementary Material Algorithm 2).

In both cases, the number of substitutions can be varied by specifying the randomization depth, see Subsection 3.1.10.

3.1.10 `depth=[0,...]`

The `depth` option allows to specify the mass-balanced randomization depth, *i.e.*, the relative number of substitutions applied to the network. The value after `depth=` gives the percentage of the default number of substitutions to apply. For example,

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                  massbalance depth=0.5
```

applies 50% of the default number of substitutions to the `tca` network, `depth=2` applies 200%, and so on. The default number of substitutions is given in Subsection 3.1.9.

3.1.11 `p=[0,1]`

The value given after `p=[0,1]` specifies the probability of randomizing a reaction. For example,

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                       massbalance p=0.5
```

applies mass-balanced randomization to the `tca` network with a 50% probability of randomization at each iteration step. If `noniterative` is specified (see Subsection 3.1.9), then reactions are chosen at random, so that the probability for applying a substitution is set to 50%. Otherwise, all reactions are iterated, so the probability for randomizing a reaction is set to 50%.

3.1.12 `substitutability`

When this option is specified, the size of the sample space is printed to the standard output, and the substitutability class size distributions are written to files. The sample space is the set of distinct randomized networks which can be generated by applying any number of substitutions of single compounds and pairs of compounds to the network. Only the size of the sample space for single substitutions can be calculated exactly, while the size of the sample space for pair substitutions is approximated.

The substitutability class of a compound is the set of its mass-equivalent compounds. The substitutability class of a pair of substrates or a pair of products is the set of mass-equivalent compound pairs, for which stoichiometric coefficients satisfying mass-balance exist (see Basler et al. (2011) for details). The number of possible substitutions of a reaction is thus the sum of the sizes of substitutability classes of all its substrates and products.

The following files are created in the specified output directory, where `<name>` is the name of the network, as explained in Subsection 3.1.1:

- `<name>.substitutability.single.sizes`
  The substitutability class size distribution of single compounds.

- `<name>.substitutability.double.sizes`
  The substitutability class size distribution of pairs of compounds.

- `<name>.substitutability.reactions.single.sizes`
  The numbers of possible single substitutions of reactions.

- `<name>.substitutability.reactions.double.sizes`
  The numbers of possible pair substitutions of reactions.

3.1.13 `D=vertices...`

Specifies a comma-separated list of compounds or reactions to preserve during randomization. If the graph is created with compartments, then the compounds must be specified by their name and compartment, separated by `DELIMITER_COMPOUND_COMPARTMENT` (see Subsection 5.3.1).

For example, `D=rea1,comp1$c` leaves the reaction `rea1` and the compound `comp1` in compartment `c` unchanged when using the default delimiter, `$`.

3.1.14 `strict`

If the `switch` and `strict` options are specified (see Subsection 3.1.2), then reversible reactions will only be shuffled with reversible reactions, and irreversible reactions only with irreversible reactions, thus ensuring the degree of each compound in the network is strictly preserved. Otherwise, the compound degrees may change by 1, if a reversible reaction is switched with an irreversible reaction.

3.1.15 `from-to`

By passing the `from-to` index range, the number of randomized networks and their file names can be specified, and randomization can be distributed among CPU cores. For example, the command

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                    massbalance 0-1999
```

will generate 2000 mass-balanced randomized networks from the text file `tca`, included in the package, and place them into the subdirectory `outdir/tca-graphs`. Any previously mass-balanced randomized networks with the same indices will be overwritten. The same holds for switch randomized networks. If you want to generate new randomized networks without replacing the previous files, specify an index range different from those of the existing files, *e.g.*, `2000-3999`.

This option also allows to easily distribute the randomization among CPU cores. For example, if you want to randomize a network 1000 times and distribute the calculation on 4 cores, then simply open 4 separate command line windows or screens, and run each of the following commands:

```
java -cp massbalance.jar massbalance/Randomize tca outdir massbalance 0-249
java -cp massbalance.jar massbalance/Randomize tca outdir massbalance 250-499
java -cp massbalance.jar massbalance/Randomize tca outdir massbalance 500-749
java -cp massbalance.jar massbalance/Randomize tca outdir massbalance 750-999
```

3.1.16 `continue`

This option allows to continue randomization on previously randomized networks. For example, if you previously generated 1000 mass-balanced randomized networks by executing

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                    massbalance
```

then you may continue to randomize the already randomized networks by executing

```
java -cp massbalance.jar massbalance/Randomize tca outdir
                    massbalance continue
```

The new randomized networks are placed into the subdirectory `<outputDir>` `/<name>-graphs-<n>/`, where `<name>` is the name of the network (see Subsection 3.1.1), and `<n>` is a counter starting with 1, which is incremented if the output directory already exists.

4 Calculating network properties

JMassBalance can be used to generate randomized networks, as explained in Section 3, and to calculate several network properties of the original and randomized networks. The syntax for calculating network properties is

```
java -cp massbalance.jar massbalance/Properties <name>
                 <inputDir> [options]
```

`<name>` is the name of the network generated during randomization (see Section 5 for how the name is generated from the different file formats). `<inputDir>` specifies the base directory where randomization was carried out, specified previously as `<outputDir>` during randomization. The results, consisting of values or distributions depending on the network property, are written to `<inputDir>`.

Below is a brief summary of the available options. Detailed descriptions and examples are given in Subsection 4.1.

`massbalance`	Calculate properties of mass-balanced randomized networks.
`switch`	Calculate properties of switch randomized networks.
`matrix[-labelled][-sorted][-reversible]`	Print the stoichiometric matrices.
`massMatrix[-labelled]`	Print a matrix of mass vectors.
`degrees`	Print the degree distributions.
`weights`	Print the weight distributions (stoichiometric coefficients).
`scopes[=seedsize]`	Calculate the scope size distributions with the given seed size (default: 1, 2, 4, 8, 16, and 32).
`pathLength`	Calculate the characteristic path lengths.
`clustering`	Calculate the clustering coefficients.
`n-cycles*`	Calculate the number of cycles of length `n`.
`path[-directed]=compounds...`	Test whether an (un-)directed path consisting of the given compounds exists.
`connected=compounds...`	Test whether the compounds are connected by (un-)directed paths.
`deltaGf`	Print the standard Gibbs free energy of all compounds (requires a deltaG annotation file).
`deltaGr`	Print the standard Gibbs free energy distributions of reactions (requires a thermodynamic information file).
`deltaGn`	Print the sum of standard Gibbs free energy distributions of reactions (requires a thermodynamic information file).
`assortativity`	Calculate the in-degree-in-degree and out-degree-out-degree correlation coefficients of compounds.
`transitionDegree`	Calculate the degree of the network node in the transition graph, i.e., the number of possible unique substitutions.
`matrixSubstitutions[-reversible]`	Print all possible substitutions of the stoichiometric matrix to a tab-separated file.
`equationSubstitutions`	Print all possible substitutions as reaction equations to a tab-separated file.
`centrality[-reversible][-double],d=[0,1]`	Calculate the global reaction knockout centralities. Reversible: consider reversible reactions as one; double: double precision; d: damping factor.
`localCentrality[-reversible]`	Calculate the local reaction knockout centralities. Reversible: consider reversible reactions as one.
`knockoutSet[,e=]0,1]],reaction`	Determine the set of reactions affected by the given reaction's knockout up to the specified strength, e.
`D=vertices...`	Preserve the given reactions/compounds.
`write`	Export the networks to a tab-delimited text file.
`from-to`	Indices of randomized networks to read and parallelization. (default: 0-999).

4.1 Options

4.1.1 `massbalance`

Specifies to calculate properties of mass-balanced randomized networks. For example, with the command

```
java -cp massbalance.jar massbalance/Properties tca outdir
                    massbalance pathLength
```

the characteristic path lengths (see Subsection 4.1.8) of the `tca` network, included in the package, and 1000 previously mass-balanced randomized networks are calculated, which are located in the directory `outdir`.

By passing the `from-to` option (see Subsection 4.1.25), the number and indices of randomized networks can be specified, and the calculation can be distributed among CPU cores.

If neither the `massbalance` nor the `switch` arguments are specified, then the property is calculated only for the original network.

4.1.2 `switch`

Specifies to calculate properties of switch randomized networks. For example, with the command

```
java -cp massbalance.jar massbalance/Properties tca outdir
                       switch pathLength
```

the characteristic path lengths (see Subsection 4.1.8) of the `tca` network, included in the package, and 1000 previously switch randomized networks are calculated, which are located in the directory `outdir`.

By passing the `from-to` option (see Subsection 4.1.25), the number and indices of randomized networks can be specified, and the calculation can be distributed among CPU cores.

If neither the `massbalance` nor the `switch` arguments are specified, then the property is calculated only for the original network.

4.1.3 `matrix[-labelled][-sorted][-reversible]`

Prints the stoichiometric matrices of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, to tab-separated text files.

The matrices are written to the subdirectory

```
<name>-matrices
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats). The file containing the stoichiometric matrix of the original network is

```
<name>.stmatrix
```

The files containing the matrices of mass-balanced randomized networks are

```
<name>.massbalance.stmatrix.<n>
```

and the files of switch randomized networks are

```
<name>.switch.stmatrix.<n>
```

where `<n>` is the index of the respective network (see Subsection 4.1.25).

If `-reversible` is appended to the option, then each reversible reaction is represented by two columns, one for each direction of the reaction. Otherwise, only one column represents a reversible reaction, while the names and indices of the reversible reactions are given in the file `<name>.stmatrix.log`.

If `-labelled` is appended to the option, then an additional header row with the reaction names, and an additional header column with the compound names are printed.

If `-sorted` is appended to the option, then the columns are sorted by reaction name and reversibility, and the rows are sorted by compound name and compartment.

4.1.4 `massMatrix[-labelled]`

Prints an $(m \times n)$ matrix containing the mass vectors of compounds in the original network to tab-separated text files, where m is the number of compounds and n is the number of chemical elements specified in the configuration file (see Subsections 3.1.1 and 5.3.2).

The matrix is written to the file

```
<name>-matrices/<name>.massMatrix
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats).

If `-labelled` is appended to the option, then an additional header row with the names of the chemical elements, and an additional header column with the compound names are printed.

The columns are sorted in the order the chemical elements are specified in the configuration file. The rows are sorted by compound name and compartment, corresponding to the row order in the sorted stoichiometric matrix (see Subsection 4.1.3).

4.1.5 `degrees`

Prints the compound degree distributions of the original network, and, if `massbalance` is specified, of the mass-balanced randomized networks, to tab-separated text files. The degree of a compound is the number of reactions it participates in, either as substrate or product (Barabási and Albert, 1999; Newman, 2003b) (note that switch randomization preserves the degree, so the degree distributions of switch randomized networks are all the same).

The degree distributions are written to the files

```
<name>.<from-to>.massbalance.degrees
<name>.<from-to>.massbalance.degrees.in
<name>.<from-to>.massbalance.degrees.out
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), and `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25). The first file contains the distributions of undirected compound degrees, the remaining two files contain the distributions of the in-, respectively out-degree distributions.

Each line of the file contains the degree distribution of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the degree distribution of the original network. The remaining lines contain the distributions of the randomized networks (see Subsection 4.1.25).

4.1.6 `weights`

Prints the distributions of weights (stoichiometric coefficients) of the original network, and, if `massbalance` is specified, of the mass-balanced randomized networks, to tab-separated text files (note that switch randomization does not affect the weights, so the weight distributions of switch randomized networks are all the same).

The weight distributions are written to the file

```
<name>.<from-to>.massbalance.weights
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), and `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25).

Each line of the file contains the weight distribution of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the weight distribution of the original network. The remaining lines contain the distributions of the randomized networks (see Subsection 4.1.25).

4.1.7 `scopes[=seedsize]`

Calculates the scope size distributions (also called network expansion, see Handorf et al. (2005)) of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files. The scope size distribution gives the probability $S(m, n)$, that n compounds can be produced from a random set of m nutrients (the seed) in a metabolic network.

For each network, 5000 sets of randomly chosen seed compounds are generated, and the corresponding scope size distribution is calculated. The number given after `seedsize=` specifies the number of compounds in each seed. If `seedsize` is unspecified, then the scope size distributions are calculated for the seed sizes 1, 2, 4, 8, 16, and 32.

The scope size distributions are written to the file(s)

`<name>.<from-to>.<rtype>.scopes<seedsize>`

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<seedsize>` is the number of compounds in each seed.

Each line of the files contains the scope size distribution of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the scope size distribution of the original network. The remaining lines contain the distributions of the randomized networks (see Subsection 4.1.25).

4.1.8 `pathLength`

Calculates the characteristic path length (also average path length, see Wagner and Fell (2001); Newman (2003b)) of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files. The characteristic path length is the average number of reactions on the shortest path between two compounds in the network.

The characteristic path lengths are written to the file

`<name>.<from-to><rtype>.pathLength`

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains the characteristic path length of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the characteristic path length of the original network. The remaining lines contain the characteristic path lengths of the randomized networks (see Subsection 4.1.25).

4.1.9 `clustering`

Calculates the clustering coefficient (see Wagner and Fell (2001); Newman (2003b)) of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files. The clustering coefficient is the average fraction of mutually connected neighbors of compounds in the corresponding unipartite network, i.e., the average of the local clusterings of all compounds. The local clustering of a compound is $2m/[d(d-1)]$, where d is the degree of the compound, and m is the number of edges between neighbors of the compound. For example, let compound c_1 be only connected to the compounds c_2, c_3, and c_4 in the unipartite metabolite-metabolite network, i.e., $c_1 \to c_2$, $c_1 \to c_3$, and $c_1 \to c_4$, and the neighbors c_2 and c_3 are mutually connected, e.g., $c_2 \to c_3$. Then, the local clustering of c_1 is

1/3, as $d = 3$ and $m = 1$. The clustering coefficient is the average of the local clusterings of all compounds, where the local clustering of compounds with less than two neighbors is 0.

The clustering coefficients are written to the file

```
<inputDir>/<name>.<from-to>.<rtype>.clustering
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains the clustering coefficient of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the clustering coefficient of the original network. The remaining lines contain the clustering coefficients of the randomized networks (see Subsection 4.1.25).

4.1.10 `n-cycles*`

Calculates the number of `n`-cycles (cycles of length `n`) of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files. The cycle length refers to the number of edges contained in a cycle in the corresponding unipartite metabolite-metabolite network (Liu and Wang, 2006). For example, $c_1 \to c_2 \to c_1$ is a cycle of length 2, where c_1, c_2 are compounds.

The cycle lengths are written to the file

```
<name>.<from-to>.<rtype>.<n>cycles
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none), and `<n>` is the cycle length.

Each line of the file contains the cycle counts of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the cycle counts of the original network. The remaining lines contain the cycle counts of the randomized networks (see Subsection 4.1.25).

4.1.11 `path[-directed]=compounds...`

Tests whether a path consisting of the given compound names exists in the original network, and, if `massbalance` or `switch` are specified, in the randomized networks. The compound names are specified as comma-separated list. For example, to test whether there are paths consisting of the compounds $c1, c2, c3$ in any direction, such as $c1 \to c2$, $c2 \to c1$, $c2 \to c3$, and $c3 \to c2$, specify `path=c1,c2,c3`. The results are written to the file

```
<name>.<from-to>.<rtype>.path
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains a `1`, if the path exists, and a `0`, if the path does not exist in the corresponding network. If the index range `<from-to>` starts with `0` (the default), then the first line corresponds to the original network. The remaining lines correspond to the randomized networks (see Subsection 4.1.25).

If `-directed` is specified, then the path is only searched in the given order. For example, to test whether there is a directed path consisting of the compounds $c1, c2, c3$, such as $c1 \rightarrow c2 \rightarrow c3$, specify `path-directed=c1,c2,c3`.

4.1.12 `connected=compounds...`

Tests whether the given compounds are connected in the given directions in the original network, and, if `massbalance` or `switch` are specified, in the randomized networks. If the network was created with the `compartments` option (see Subsection 3.1.7), then the compounds are specified by their name and compartment, separated by the compound-compartment delimiter (by default `$`, see Subsection 5.3). Otherwise, only the compound names are given.

The connections are specified as a list of compounds separated by `>`, specifying a forward connection, and `=`, specifying an undirected connection. For example, assume you want to test whether there is a directed path connecting the compounds $c1, c2$, such as $c1 \rightarrow ... \rightarrow c2$, and an undirected path connecting the compounds $c2, c3$, such as $c2 \leftrightarrow ... \leftrightarrow c3$. If $c1$ is located in the compartment `ext`, and $c2$ and $c3$ are located in the compartment `cyt`, specify `connected=c1$ext>c2$cyt=c3$cyt`. The results are written to the file

```
<name>.<from-to>.<rtype>.connected
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains a list of `1` and `0`, where `1` indicates that a connection exists, and `0` indicates that there is no connection between the given compounds in the corresponding network. If the index range `<from-to>` starts with `0` (the default), then the first line corresponds to the original network. The remaining lines correspond to the randomized networks (see Subsection 4.1.25).

4.1.13 `deltaGf`

Prints the standard Gibbs free energy of formation, Δ_f^0, of all compounds (see Mavrovouniotis (1991)) in the original network to a tab-separated text file. A file containing thermodynamic information must have been provided during randomization (see Subsection 5.4).

The compound names and Δ_f^0 are written to the file

`<name>.deltaGf`

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats).

Each line of the file contains the name of one compound (with corresponding compartment, if the network has compartments), followed by a `TAB` space (`\t`), followed by `D=` and the Δ_f^0 of the compound. The compounds are sorted by their name in Unicode order.

4.1.14 `deltaGr`

Prints the distributions of the standard Gibbs free energy change of reactions, Δ_r^0 (see Mavrovouniotis (1991)), of the original network, and, if `massbalance` is specified, of the mass-balanced randomized networks, to tab-separated text files. A file containing thermodynamic information must have been provided during randomization (see Subsection 5.4).

The Δ_r^0 distributions are written to the file

`<name>.<from-to>.<rtype>.deltaGr`

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains the Δ_r^0 distribution of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the Δ_r^0 distribution of the original network. The remaining lines contain the distributions of the randomized networks (see Subsection 4.1.25).

4.1.15 `deltaGn`

Prints the sum of the standard Gibbs free energy change of all reactions, Δ_n^0 (see Basler et al. (2010)), of the original network, and, if `massbalance` is specified, of the mass-balanced randomized networks, to tab-separated text files. A file containing thermodynamic information must have been provided during randomization (see Subsection 5.4).

The Δ_n^0 values are written to the file

`<name>.<from-to>.<rtype>.deltaGn`

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains the Δ_n^0 of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the Δ_n^0 of the original network. The remaining lines contain the values of the randomized networks (see Subsection 4.1.25).

4.1.16 `assortativity`

Calculates the assortativities (the in-degree/in-degree and out-degree/out-degree correlation coefficients, Newman (2003b)) of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files. The assortativities are the correlation coefficients of the vector of in-/out-degrees of compounds and the in-/out-degrees of their predecessors in the corresponding unipartite metabolite-metabolite network. Note that, in contrast to Newman (2003a), the actual degree of a compound is used, not the excess degree, that equals the actual degree -1.

The assortativities are written to the file

```
<name>.<from-to><rtype>.assortativity
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, `switch`, or none).

Each line of the file contains two values corresponding to one network, (1) the correlation coefficient of the in-degrees of compounds and the in-degrees of their predecessors, and (2) the correlation coefficient of the out-degrees of compounds and the out-degrees of their successors in the corresponding unipartite metabolite-metabolite network. If the index range `<from-to>` starts with `0` (the default), then the first line corresponds to the original network. The remaining lines correspond to the randomized networks (see Subsection 4.1.25).

4.1.17 `transitionDegree`

Calculates the transition degree (the number of possible mass-balanced substitutions, see Basler et al. (2011)) of the original network, and, if `massbalance` is specified, of the mass-balanced randomized networks, to tab-separated text files (the transition degree is not implemented for switch randomization).

The transition degrees are written to the file

```
<name>.<from-to>.massbalance.transitiondegree
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), and `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25).

Each line of the file contains the transition degree of one network. If the index range `<from-to>` starts with `0` (the default), then the first line contains the transition degree of the original network. The remaining lines contain the distributions of the randomized networks (see Subsection 4.1.25).

4.1.18 `matrixSubstitutions[-reversible]`

Writes the possible mass-balanced substitutions of the stoichiometric matrix of the original network to tab-separated text files (not implemented for switch randomized substitutions).

The stoichiometric matrix of the original network is written to

```
<name>.stmatrix
```

with labels and sorted (see Subsection 4.1.3), where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats).

The single and double substitutions are written to the files

```
<name>.matrixsubstitutions.singles
<name>.matrixsubstitutions.doubles
```

respectively.

Each line of the file corresponds to one substitution and starts with the name of the perturbed reaction. The next column contains `D=` and the value $\Delta_r^0 G$ of the original reaction. The third column contains `d=` and the value giving the difference in $\Delta_r^0 G$ between the original and the perturbed reaction. The next two columns (for single substitutions), or the next four columns (for double substitutions) contain the names of the substituted compounds followed by the names of their substitutes, respectively. The remaining columns give the substitutions of the matrix as triplets `i j v`, where `i`, `j` is the modified index of the matrix, and `v` is its new value.

If `-reversible` is appended to the option, then the indices correspond to the stoichiometric matrix where each reversible reaction is represented by two columns, one for each direction of the reaction. Otherwise, the indices correspond to the stoichiometric matrix where only one column represents a reversible reaction (see Subsection 4.1.3).

Note that thermodynamic information is only shown if the corresponding information was included when generating in the network (see Subsection 5.4).

4.1.19 `equationSubstitutions`

Writes the possible mass-balanced substitutions of the original network as reaction equations to tab-separated text files (not implemented for switch randomized substitutions).

The single and double substitutions are written to the files

```
<name>.equationsubstitutions.singles
<name>.equationsubstitutions.doubles
```

respectively.

Each line of the file corresponds to one substitution and is organized as follows:

```
<name> <original-equation> D=<D> <new-equation> d=<d>
```

where `<name>` is the name of the perturbed reaction, `<original-equation>` is the original reaction equation, `<D>` is the $\Delta_r^0 G$ of the original reaction,

`<new-equation>` is the reaction equation after applying the substitution, and `<d>` is the difference in $\Delta_r^0 G$, i.e., $\Delta_{r'}^0 G - \Delta_r^0 G$, where r' is the new reaction.

Note that thermodynamic information is only shown if the corresponding information was included when generating in the network (see Subsection 5.4).

4.1.20 `centrality[-reversible][-double],d=[0,1]`

Calculates the reaction centralities of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files.

The centrality of a reaction is the global impact of its knockout on the network, based on the Hubbell Index of the local dependencies between reactions (see Hubbell (1965)). The value `d=[0,1]` specifies the damping factor for the PageRank transformation (see Langville and Meyer (2003)). If `-reversible` is specified, then a reversible reaction is considered as one, otherwise each direction is considered independently. If `-double` is specified, then the calculations are done in double precision, otherwise in float precision.

The reaction centralities of the original network are written to the file

```
<name>.centrality.<reversible>.d<d>
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats), and `<d>` is the specified damping factor. Each line of the file contains the name of one reaction and its centrality score.

The reaction centralities of the randomized networks are written to the files

```
<name>.<from-to>.<rtype>.centrality.<reversible>.d<d>
```

where `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, or `switch`). Each line contains the reaction centralities of one randomized network (see Subsection 4.1.25).

4.1.21 `localCentrality[-reversible]`

Calculates the local reaction centralities of the original network, and, if `massbalance` or `switch` are specified, of the randomized networks, and writes them to tab-separated text files.

The local centrality of a reaction is the impact of its knockout on successor reactions. If `-reversible` is specified, then a reversible reaction is considered as one, otherwise each direction is considered independently.

The local reaction centralities of the original network are written to the file

```
<name>.centrality.local
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats). Each line of the file contains the name of one reaction and its local centrality score.

The local reaction centralities of the randomized networks are written to the files

```
<name>.<from-to>.<rtype>.centrality.local
```

where `<from-to>` is the specified index range, `0-999` by default (see Subsection 4.1.25), and `<rtype>` is the specified type of randomization (`massbalance`, or `switch`). Each line contains the local reaction centralities of one randomized network (see Subsection 4.1.25).

4.1.22 `knockoutSet[,e=]0,1]],reaction`

Determines the impact of a knockout of the given reaction on other reactions in the original network, up to the given threshold `e`, and writes the results to a tab-separated text file.

The knockout of a reaction may either fully or partly block other reactions in the network. The threshold, by default `e=1`, specifies up to which knockout strength the affected reactions are determined. For example, `knockoutSet,e=0.5,agmatinase` determines all reactions which are blocked by 50% or more by knockout of the agamatinase reaction. The results are written to the file

```
<name>.<reaction>.knockout
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats). Each line of the file corresponds to one reaction and contains three values: (1) the name of the reaction, (2) the percentage, by which the reaction is blocked, and (3) the local centrality of the reaction (see Subsection 4.1.21).

4.1.23 `D=vertices...`

Specifies a comma-separated list of compounds or reactions to preserve when calculating the transition degree (Subsection 4.1.17) or possible substitutions (Subsection 4.1.18). If the graph was created with compartments, then the compounds must be specified by their name and compartment, separated by `DELIMITER_ COMPOUND_COMPARTMENT` (see Subsection 5.3.1).

For example, when using the default delimiter, `$`, then `D=rea1,comp1$c` ignores the reaction `rea1` and the compound `comp1` in compartment `c` for calculating the transition degree or possible substitutions.

4.1.24 `write`

Exports the original network and, if `massbalance` or `switch` are specified, the randomized networks to tab-separated text files. The original file are exported to the subdirectory

```
<name>-graphs
```

where `<name>` is the network name generated during randomization (see Section 5 for how the name is generated from the different file formats). The original network is exported to the file

```
<name>.network
```

and the randomized networks are exported to the files

```
<name>.<rtype>.network.<n>
```

where `<rtype>` is the specified type of randomization (`massbalance` or `switch`), and `<n>` is the index of the respective network (see Subsection 4.1.25).

The customizable file format is defined by the delimiters specified in the configuration file (see 5.3).

4.1.25 `from-to`

By passing the `from-to` index range, the number of randomized networks, for which the respective properties are calculated can be specified, and randomization can be distributed among CPU cores. For example, the command

```
java -cp massbalance.jar massbalance/Properties tca outdir
            massbalance pathLength 0-1999
```

will calculate the characteristic path lengths of 2000 networks, which have been previously mass-balanced randomized from the `tca` network (see Subsection 4.1.8). The same holds for switch randomized networks. Note that the specified index range must fully include the indices of previously randomized networks.

If the `from-to` range starts with `0`, then the property is calculated for the original network and the randomized networks. If the `from-to` range does not start with `0`, then the property is calculated only for the randomized networks.

This option also allows to easily distribute the randomization among CPU cores. For example, if you want to calculate the characteristic path lengths of 1000 mass-balanced randomized networks and distribute the calculation on 4 cores, then simply open 4 separate command line windows or screens, and run the command

```
java -cp massbalance.jar massbalance/Properties tca outdir
              massbalance pathLength
```

with the index ranges `0-249`, `250-499`, `500-749`, and `750-999`, respectively.

5 File formats

JMassBalance accepts three basic types of network files as input: (1) BioCyc flat files (see Subsection 5.1), (2) SBML models (see Subsection 5.2), and (3) a customizable text file format (see Subsection 5.3).

During randomization, a network name is generated, which is used for naming the output files, and for specifying the network when calculating network properties. The network name depends on the provided file format: for BioCyc flat files, the name is obtained by concatenating the `Species`, `Database`, and `Version` strings from the reactions file. For SBML files, the name is the `id` attribute of the model. For text files, the name is simply the name of the provided network file.

5.1 BioCyc flat files

The BioCyc Database Collection (`http://biocyc.org/`) offers a large collection of metabolic networks in a flat-file text format. The networks may either be downloaded directly from BioCyc (see `http://biocyc.org/all-reg.shtml`), or exported from the Pathway Tools Software (see `http://bioinformatics.ai.sri.com/ptools/`).

To randomize a network in BioCyc format, copy the files `reactions.dat` and `compounds.dat` (and optionally `pathways.dat`) to a local directory, and run the following command:

```
java -cp massbalance.jar massbalance/Randomize <networkFile>
                <outputDir> massbalance
```

where `<networkFile>` specifies the (absolute or relative) path to the network file (usually `reactions.dat`), and `<outputDir>` specifies the directory where the mass equivalence classes, log files, and randomized network files will be placed. If a `pathways.dat` file is included, then the reversibilities of reactions are extracted from this file, if no annotation is provided in `reactions.dat`.

If the `compartments` option is specified (Section 3.1.7), then the compartment in which each reaction and compound is located is determined in two parallel ways. First, any compound for which the attribute ^`COMPARTMENT` is specified is placed into the corresponding compartment (as in AraCyc/PlantCyc). However, if the reaction contains the attribute `RXN-LOCATIONS`, then the compartment is specified by this attribute in conjunction with the generic attributes `CCO-IN` and `CCO-OUT` for the substrates and products of the reaction (see `http://ecocyc.org/PGDBConceptsGuide.shtml` for more information).

An additional file with the same name as `<networkFile>`, but with the file extension `.deltaG`, containing thermodynamic information may be placed into the same directory to perform thermodynamic analyses (see Subsection 5.4).

5.2 SBML models

The Systems Biology Markup Language (`http://sbml.org/`) provides an interchange format for computer models of biological processes. Metabolic networks in SBML format are provided as supplementary information by numerous publications and tools in the field of Systems Biology. *JMassBalance* supports models in SBML Levels 2 and 3, as well as the model format used by YeastNet versions 2 and 4 (see `http://www.comp-sys-bio.org/yeastnet/` and Herrgård et al. (2008)).

For randomizing SBML models, `libSBML` is required. For Windows, the files `libsbmlj.jar` and `sbmlj.dll` are included in the download package and provide SBML support without the need to install `libSBML`. For SBML support on Linux or Mac, see Subsection 1.3.

The SBML model may already include the sum formulas for the compound species, which are required for mass-balanced randomization. Otherwise, the sum formulas can be provided in a separate compounds file. To this end, create a compounds file as described in Section 5.3.2 with the same name as the SBML file, but with file name extension `.compounds` instead of `.xml`, and place it into the same directory as the SBML file. For example, if your SBML file is `ecoli.xml`, the compounds file must be named `ecoli.compounds`. In this case, only the sum formulas from the specified compounds file will be used, neglecting any sum formulas in the SBML file.

The recommendation of the SBML community is to include the sum formulas as InChI strings within an `annotation` subelement inside `species` elements (see `http://sbml.org/Community/Wiki/About_annotations_in_Level_2#How_do_I_put_InChI_strings_in_annotations.3F`). However, in YeastNet 2.0, the sum formulas are included as InChI strings within the `speciesType` elements corresponding to the compound species. In YeastNet 4.0, the sum formulas are included as literal strings in the `speciesType` elements corresponding to the compound species. Each of these formats is supported by *JMassBalance*.

To randomize a network in SBML format, copy the SBML file (and optionally the compounds file) to a local directory, and run the following command:

```
java -cp massbalance.jar massbalance/Randomize <networkFile>
                  <outputDir> massbalance
```

where `<networkFile>` specifies the (absolute or relative) path to the SBML file, and `<outputDir>` specifies the directory where the mass equivalence classes, log files, and randomized network files will be placed. Note that an error may occur if the SBML file is located on a remote directory (which seems to be a bug in `libSBML`).

An additional file with the same name as `<networkFile>`, but with the file extension `.deltaG`, containing thermodynamic information may be placed into the same directory to perform thermodynamic analyses (see Subsection 5.4).

5.3 Customizable text file format

JMassBalance accepts metabolic networks represented in a customizable text format, along with a compounds file containing the sum formulas of the compounds. The network file may use arbitrary delimiters denoting the reaction equations (see Subsection 5.3.1). For the compounds file, the considered chemical elements may be specified (see Subsection 5.3.2).

5.3.1 Network file

A metabolic network may be represented in a customizable text format, thus allowing for flexible input formats. The basic structure of the file is pre-defined, but the delimiters for representing reaction equations are customizable by changing the values in the file `jmassbalance.config`, which is included in the download package. In particular, the following delimiters can be specified by the user:

`DELIMITER_PLUS`	Substrate/product-delimiter.
`DELIMITER_EQUALS`	Side-delimiter for reversible reactions.
`DELIMITER_FORWARD`	Side-delimiter for irreversible (forward) reactions.
`DELIMITER_COEFFICIENT`	Two-sided delimiter enclosing stoichiometric coefficients.
`DELIMITER_COMPOUND_COMPARTMENT`	Delimiter for compound compartments.
`DELIMITER_REACTION_COMPARTMENT`	Two-sided delimiter enclosing reaction compartments.
`DEFAULT_COMPARTMENT`	Default compartment.

The default values are shown in the file `jmassbalance.config`. The basic structure of the format is as follows:

- Each line of the file describes one reaction, which starts with the reaction name, followed by a `TAB` space (`\t`), followed by the reaction equation.
- The left and right side of the equation are separated by `DELIMITER_EQUALS` for reversible reactions, or `DELIMITER_FORWARD` for irreversible reactions.
- The substrates and products are separated by `DELIMITER_PLUS`.
- Stoichiometric coefficients are enclosed by the first and second character of `DELIMITER_COEFFICIENT`.
- The compartment of a compound is specified by appending `DELIMITER_COMPOUND_COMPARTMENT` followed by the compartment name to the substrate or product name.
- The compartment of a reaction is specified by prefixing the reaction equation with the compartment string, enclosed by the first and second character of `DELIMITER_REACTION_COMPARTMENT`.
- A line starting with `#` is regarded a comment and skipped.

As an example, the reaction equation for `amidohydrolase` in the compartment `cyt` is commonly written as `alltt + 2 h + 2 h2o` → `co2 + 2 nh4 + urdglyc`. Using the default delimiters specified in `jmassbalance.config`, this reaction is specified as:

```
amidohydrolase  [cyt] alltt $+$ {2} h $+$ {2} h2o $>$ co2 $+$ {2} nh4 $+$ urdglyc
```

The reaction name is followed by a `TAB` space. `[cyt]` specifies that the reaction takes place in the compartment `cyt`. Substrates and products are separated by `$+$`, {2} specifies a stoichiometric coefficient of 2.

An example of a transporter of one `asp-L` molecule and two `h` molecules from the compartment `cyt` to `per` is specified as:

```
aspartate transport    asp-L$per $+$ {2} h$per $>$ asp-L$cyt + {2} h$cyt
```

The compartments `cyt` and `per` are appended to the compound names with a separating `$`.
The following shows an example of the TCA cycle, which is included as file `tca` in the download package:

```
cit synthase       A-CoA $+$ Water $+$ Oxaloacetate $>$ Citrate $+$ CoASH $+$ H+
aconitase          Citrate $=$ Isocitrate
isocit dehyd       Isocitrate $+$ NADP+ $=$ NADPH $+$ CO2 $+$ a-keto
a-keto dehyd       CoASH $+$ NAD+ $+$ a-keto $>$ CO2 $+$ NADH $+$ Succinyl-CoA
suc thiokinase     Succinyl-CoA $+$ ADP $+$ Pi $=$ CoASH $+$ ATP $+$ Succinate
suc dehyd          Succinate $+$ Ubiquinone-8 $>$ Fumarate $+$ Ubiquinol-8
fumarase           Fumarate $+$ Water $=$ L-Malate
malate dehyd       NAD+ $+$ L-Malate $=$ Oxaloacetate $+$ NADH $+$ H+
```

5.3.2 Compounds file

The basic structure of the compounds file is as follows:

- Each line of the file describes one compound, starting with the compound name, followed by a `TAB` space (`\t`), followed by the sum formula.
- The sum formula contains the chemical elements followed by their quantity, such as `CO2`.
- A line starting with `#` is regarded a comment and skipped.

To specify the names of chemical elements to consider, change the comma-separated list after `ELEMENTS` in the file `jmassbalance.config` included in the directory where you extracted the package. By default, the chemical elements carbon (C), hydrogen (H), nitrogen (N), oxygen (O), phosphorus (P), and sulfur (S) are considered. If a sum formula contains an undefined chemical element, then the compound is used without mass information.

An example representing the compounds of the TCA cycle is shown below:

```
A-CoA                     C23H34N7O17P3S
ADP                       C10H12N5O10P2
a-keto                    C5H4O5
ATP                       C10H12N5O13P3
Citrate                   C6H5O7
CO2                       CO2
CoASH                     C21H32N7O16P3S
Fumarate                  C4H2O4
H+                        H
Isocitrate                C6H5O7
L-Malate                  C4H4O5
NAD+                      C21H26N7O14P2
NADH                      C21H27N7O14P2
NADP+                     C21H25N7O17P3
NADPH                     C21H26N7O17P3
Oxaloacetate              C4H2O5
Pi                        HPO4
Succinate                 C4H4O4
Succinyl-CoA              C25H35N7O19P3S
Ubiquinol-8               C49H76O4
Ubiquinone-8              C49H74O4
Water                     H2O
```

5.4 Thermodynamic information

To analyse the thermodynamic properties of metabolic networks, information on the standard Gibbs free energy change of formation ($\Delta_f^0 G$, see Mavrovouniotis (1991)) may be included in the network. To this end, create a file with the name `<networkFile>.deltaG` in the same directory as the network file, `<networkFile>`.

The basic structure of the file is as follows:

- Each line of the file describes one compound, starting with the compound name, followed by a `TAB` space (`\t`), followed by `D=` and the $\Delta_f^0 G$ value, *e.g.* `D=-56.7`.
- A line starting with `#` is regarded a comment and skipped.

For the customizable text file format, the $\Delta_f^0 G$ values may also be included in the compounds file as an additional value, separated from the sum formula by another `TAB` space (see Subsection 5.3.2). The following shows an example which includes the sum formulas and $\Delta_f^0 G$ values of the TCA cycle, which is included as file `tca.compounds` in the download package:

A-CoA	C23H34N7O17P3S	D=-784.6
ADP	C10H12N5O10P2	D=-465.4
a-keto	C5H4O5	D=-188.8
ATP	C10H12N5O13P3	D=-673.5
Citrate	C6H5O7	D=-280.3
CO2	CO2	D=-92.3
CoASH	C21H32N7O16P3S	D=-750.9
Fumarate	C4H2O4	D=-143.7
H+	H	D=-9.5
Isocitrate	C6H5O7	D=-279.0
L-Malate	C4H4O5	D=-201.0
NAD+	C21H26N7O14P2	D=-529.0
NADH	C21H27N7O14P2	D=-523.8
NADP+	C21H25N7O17P3	D=-740.9
NADPH	C21H26N7O17P3	D=-726.2
Oxaloacetate	C4H2O5	D=-190.5
Pi	HPO4	D=-262.0
Succinate	C4H4O4	D=-163.0
Succinyl-CoA	C25H35N7O19P3S	D=-860.9
Ubiquinol-8	C49H76O4	D=124.7
Ubiquinone-8	C49H74O4	D=146.2
Water	H2O	D=-56.7

References

Barabási and Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, Oct 1999.

Georg Basler, Sergio Grimbs, Joachim Selbig, and Zoran Nikoloski. Thermodynamic landscapes of randomized large-scale metabolic networks. *Proceedings of the 7th International Workshop on Computational Systems Biology*, pages 23–26, 2010.

Georg Basler, Oliver Ebenhöh, Joachim Selbig, and Zoran Nikoloski. Mass-balanced randomization of metabolic networks. *Bioinformatics*, 27(10): 1397–1403, May 2011. doi: 10.1093/bioinformatics/btr145. URL `http://dx.doi.org/10.1093/bioinformatics/btr145`.

Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. Libsbml: an api library for sbml. *Bioinformatics*, 24 (6):880–881, Mar 2008. doi: 10.1093/bioinformatics/btn051. URL `http://dx.doi.org/10.1093/bioinformatics/btn051`.

Thomas Handorf, Oliver Ebenhöh, and Reinhart Heinrich. Expanding metabolic networks: Scopes of compounds, robustness, and evolution. *J Mol Evol*, 61 (4):498–512, 2005. doi: 10.1007/s00239-005-0027-1.

Markus J Herrgård, Neil Swainston, Paul Dobson, Warwick B Dunn, K. Yalin Arga, Mikko Arvas, Nils Blüthgen, Simon Borger, Roeland Costenoble, Matthias Heinemann, Michael Hucka, Nicolas Le Novre, Peter Li, Wolfram Liebermeister, Monica L Mo, Ana Paula Oliveira, Dina Petranovic, Stephen Pettifer, Evangelos Simeonidis, Kieran Smallbone, Irena Spasić, Dieter Weichart, Roger Brent, David S Broomhead, Hans V Westerhoff, Betül Kirdar, Merja Penttilä, Edda Klipp, Bernhard Ø Palsson, Uwe Sauer, Stephen G Oliver, Pedro Mendes, Jens Nielsen, and Douglas B Kell. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nat Biotechnol*, 26(10):1155–1160, Oct 2008. doi: 10.1038/nbt1492.

Charles H. Hubbell. An Input-Output Approach to Clique Identification. *Sociometry*, 28(4):377–399, 1965. ISSN 00380431. doi: 10.2307/2785990. URL http://dx.doi.org/10.2307/2785990.

Amy N. Langville and Carl D. Meyer. Deeper Inside PageRank. *Internet Mathematics*, 1(3):335–380, 2003. URL http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.im/1109190965.

Hongbo Liu and Jiaxin Wang. A new way to enumerate cycles in graph. *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications*, page 57, 2006. doi: 10.1109/AICT-ICIW.2006.22. URL http://dx.doi.org/10.1109/AICT-ICIW.2006.22.

M.L. Mavrovouniotis. Estimation of standard gibbs energy changes of biotransformations. *Journal of Biological Chemistry*, 266:14440–14445, 1991.

M. E J Newman. Mixing patterns in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 67(2 Pt 2):026126, Feb 2003a.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003b.

A. Wagner and D.A. Fell. The small world inside large metabolic networks. *Proc R Soc Lond B Biol Sci*, 268:1803–1810, 2001.