NAME

fitfluxes − fit a flux distribution of a network in order to reproduce a set of isotope labeling measurement values.

SYNOPSIS

fitfluxes [*options*]

DESCRIPTION

Fitfluxes is a program for intelligently adjusting a flux distribution of a metabolic network in order to reproduce a set of isotope labeling measurement values.

For this purpose **fitfluxes** reads the metabolic network and measurement configuration from a FluxML file, analyzes the network's stoichiometry and uses an optimization algorithm for adjusting certain 'free' variables in the flux distribution. Given the resulting new flux distribution the network and the measurements are simulated and the resulting synthetic measurement values are compared to the real ones. This process is repeated until synthetic and real measurement values agree or a certain termination condition is fulfilled (e.g. the maximum number of iterations is exceeded).

The resulting flux distribution and the corresponding synthetic measurement values are written to a **fwdsim**(5) XML document, i.e. a document in the same format also generated by the **fwdsim**(1) command.

COMMON OPTIONS

The following options are common to **fwdsim** and **fitfluxes**:

−h, −−help
Show a brief help for all command line options.

−i, −−in <FILE> [default: stdin]
The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

−o, −−out <FILE> [default: stdout]
The name of the FWDSIM (XML) output file. If omitted, the generated FWDSIM document is written to standard output.

−L, −−list
Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

−c, −−configure <CFG> [default: 'default']
Because FluxML documents may contain several **<configuration/>** elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

−l, −−log DEST
Specify the destination for the internal logging. In the most simple case **DEST** is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by **fd:[num]**, where **[num]** is the number of the file descriptor.

A unix domain socket in the local file system is specified by **unix:[name]**, where **[name]** is the name of the socket file

A UDP or (connectionless) SCTP port is specified by **[proto]:[host]:[port]**, where **[proto]** is either "udp" or "sctp" and **[host]** is the name of the destination host and **[port]** is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size − log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination **@gui@**. The GUI

requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

−v, −−verbose 0..10 [default: 5]
Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- **0 (QUIET)** do not emit log messages at all.
- **1 (ERROR)** only emit severe error messages.
- **2 (WARNING)** only report severe errors and warnings.
- **3 (NOTICE)** report all errors and warnings including important informal messages.
- **4 (INFO)** report all errors, warnings and all informal messages.
- **5 (THROW)** in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- **6 (DEBUG0)** emit the more important debugging messages.
- **7 (DEBUG1)** emit the less important debugging messages
- **8 (DEBUG2)** emit the superfluous debugging messages
- **9 (DEBUG3)** emit annoying debugging messages.
- **10 (DEBUG4)** don't dare to use it!

−t, −−tolerance <val> [default: 1e−9]
Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

SOLVER OPTIONS

There are three options affecting the behavior of the solver. The same options are supported by the **fwdsim** command:

−a, −−apsolve
Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round−off for the solution of the individual network levels.

−A, −−exact
In addition to option **−a** absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (**−g analytic**) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

−d, −−dbgsolve
When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

OPTIMIZATION OPTIONS

−O, −−optimizer <arg> [default: IPOPT]
The desired optimization algorithm. Currently available are **Ipopt**, **CFSQP**, and **NAGNLP** (the latter two are distributed under a commercial license and may not be available at your site).

−P, −−properties <arg>
This option allows fine−grained control over the optimization algorithm's behavior and settings. The argument to option **−P** is a comma−separated list of key−value pairs. The general syntax is: **[optimizer].[property]=[type]([value])**.

Valid **[optimizer]**'s are '**ipopt**' and '**cfsqp**'.

Valid **[type]**'s are:

- **integer** − in this case **[value]** has to be an integer.
- **string** − the contents of **[value]** are allowed to be any string.
- **real** − for real (double precision) values.
- **boolean** − restricts the contents of **[value]** to the values **true** and **false**.

Using this interface all of the numerous settings of Ipopt can be accessed. A full list of valid options can be obtained by setting the boolean property **ipopt.print_options_documentstion** to **true**. See the examples below. Other interesting properties include:

- **ipopt.max_cpu_time** − allows to abort Ipopt after a specified number of CPU seconds is exceeded.
- **ipopt.print_level** − the verbosity of Ipopt. Valid settings are integers from 0 to 12. The default setting is 5.
- **ipopt.max_iter** − the maximum number of iterations before termination. The default value is 3000.
- **ipopt.linear_solver** − allows to choose the linear solver used by Ipopt. The default setting is '**mumps**'. Using another solver may improve convergence speed and optimization results.

CFSQP supports significantly fewer properties (a more detailled description can be found in the CFSQP documentation):

- **cfsqp.mode** − the type and mode of the used solver. The default setting is 200.
- **cfsqp.iprint** − the verbosity of CFSQP. The default setting is 1. Other settings are 0 (quiet), 2, or 3 (more verbose output).
- **cfsqp.miter** − the maximum number of iterations before termination. The default value is 1000.
- **cfsqp.bigbnd** − allows setting the value which plays the role of infinity. The default value is 1e20.
- **cfsqp.eps** − final norm of the newton gradient. Must be bigger than the machine epsilon. The default value is 1e−10.

NAGNLP refers to the "nag_opt_nlp" ("e04ucc") optimizer found in the commercial NAGC library. Most important settings are accessible via the following properties (cf. NAGC documentation):

- **nag_opt_nlp.max_iter** − the maximum number of iterations before termination. The default value is 250.
- **nag_opt_nlp.list** − print settings. The default value is "false".
- **nag_opt_nlp.print_level** − the major print level. The default setting is "Nag_Soln_Iter". See the NAGC documentation for other available settings.
- **nag_opt_nlp.minor_print_level** − the minor print level. The default setting is "Nag_NoPrint". See the NAGC documentation for other available settings.
- **nag_opt_nlp.verify_grad** − allows to enable the gradient checker. The default setting is "Nag_NoCheck". Set to "Nag_SimpleCheck" to turn on gradient checking.

−S, −−shrink <eps> [default: 5e−7]
This option may be used to tighten the complete set of constraints exposed to the optimization packages. If you think of the constraints as a convex polyhedron this option causes the polyhedron to be shrunk symmetrically about the specified argument eps. Specify this option if you notice warning

messages reporting that the optimizer is violating constraints. Because this seems to be a common problem in the used optimization packages (probably due to numerical inaccuracies) fitfluxes uses a default shrinking of 5−e7. Specify −S 0 to disable this feature.

−g, −−gradient <arg> [default: fd4]
The desired method for computaton of partial derivatives in case statistics are to be computed (option **−s**). Possible values are '**fd1**', '**fd2**', '**fd3**', '**fd4**' for finite difference approximations, and '**analytic**' for exact derivatives. The default value is '**fd4**', i.e. the gradient is approximated using an O(h^4) finite difference formula (which is slightly faster than using the exact derivatives).

EXAMPLES

Find a new flux distribution for a metabolic network specified in a FluxML file in order to reproduce a set of istope labeling measurement values. Safe the XML output to file network.fwd (most simple case):

fitfluxes −i network.fml −o network.fwd

Use the optimization package CFSQP instead of Ipopt and compute the required gradient analytically (instead of fourth order finite differences):

fitfluxes −i network.fml −o network.fwd −O CFSQP −g analytic

Convert an old FTBL to FluxML, find a new flux distribution using Ipopt, and filter out the flux distribution in net / echange01 coordinates (the flux coordinate system of 13CFLUX 1). Send all log messages of fitfluxes to a GUI (requires a running X server):

ftbl2fml −i network.ftbl | fitfluxes −l @gui@ | fwdsimflt −s −X

Request the documentation of all settings from Ipopt. Also set option **max_iter** to 0 in order to abort as soon as possible. Redirect the output to text file ipopt_settings.txt:

fitfluxes −i network.fml −o /dev/null −P "ipopt.print_options_documentation=boolean(true), ipopt.max_iter=integer(0)" > ipopt_settings.txt

Use the commercial NAG NLP SQP optimizer (nag_opt_nlp, e04ucc) and restrict the maximum number of iterations to 50 (default: 1000):

fitfluxes −i network.fml −o network.fwd −O nagnlp −P "nag_opt_nlp.max_iter=integer(50)"

Request a list of available options along with their default settings from the NAG NLP optimizer (the options and their meaning is discussed in the NAG manual of e04ucc):

fitfluxes −i network.fml −o /dev/null −O nagnlp −P "nag_opt_nlp.print_available_options=boolean(true)"

SEE ALSO

fwdsim(1), fwdsim(5), fwdsimflt(1), ftbl2fml(1), simreport(1), setfluxes(1), multifit(1)

AUTHOR

Michael Weitzel

This manpage was written by Michael Weitzel <mich@el−weitzel.de>.