COPASI wrapper SpaceScanner

User Manual

version May 2017

Table of Contents

1. SpaceScanner features 2
  1.1 SpaceScanner terminology 2
  1.2 SpaceScanner features 2
  1.3 SpaceScanner required inputs 2
  1.4 Installation 3
2. Creating a model file for *SpaceScanner* 4
3. Running SpaceScanner 4
  3.1 from the web interface 4
    3.1.1 “Select a model” window 5
    3.1.2 “Settings” window 6
    3.1.3 “Parameter sets” 9
    3.1.4 “Status” button 10
    3.1.5 “Export CSV” button 11
  3.2. from command line 11
    3.2.1 "copasi" section 11
    3.2.2 "optimization" section 11
    3.2.3 "parameters" section 12
    3.2.4 "web" section 13
    3.2.5 "output" section 13
    3.2.6 Not in separate sections 13
    3.2.7 Example configuration file 14
  3.3 Stopping SpaceScanner 14
  3.4 Accessing the results 14
References 15

1. SpaceScanner features

1.1 SpaceScanner terminology

There are multiple similar terms with quite different meanings used in the UI and documentation:

- A **run** is a single optimization process. Each active run corresponds to a single Copasi process instance.
- A **job** is a collection of one or more optimization *runs* that all share the same set of parameters and are executed in parallel.
- A **task** is a collection of one or more *jobs* described by a single configuration file. A task corresponds to single command-line execution of SpaceScanner.

1.2 SpaceScanner features

- run multiple parallel optimization tasks on a biological model, and automatically terminate when the tasks have reached a consensus value;
- display optimization history graphically for these parallel runs;
- scan the space of the possible parameters sets to optimize, and determine the minimal subset of parameters that gives "good enough" results for a specific objective function and the minimal number of parameters required for a specific target value.

1.3 SpaceScanner required inputs

- a Copasi model file (`.sbml`)

The file must include:

- the objective function to optimize;
- list of changeable parameters of the model with their minimal and maximal values;
- the optimization methods to use and the configurations of these methods.

SpaceScanner produces these global outputs:

- a `.csv` file with the best (or all) optimization results
- a `.log` file tracing the SpaceScanner execution history;

as well as these outputs for each set of optimization parameters:

- a `.txt` file for each of the several parallel optimization runs; the file includes the history of the Copasi optimization process and the end parameter values;
- a Copasi model file for each if the optimization runs; in this file, the optimization parameters are listed to their best values.

SpaceScanner internally uses Copasi (Hoops et al., 2006; Mendes et al., 2009) to execute the optimizations.

SpaceScanner at the moment supports greedy and exhaustive search strategies when looking for the minimal satisfying number of parameters. "Smarter" search strategies (e.g. global stochastic search, parameter sensitivity-informed search, MFA-value informed search) are planned as future additions.

SpaceScanner is easy to use and configure. There are two ways how to work with SpaceScanner:

- a command-line interface that expects a configuration file in JSON format as the only argument;
- a web interface that allows the user to interactively configure, start, and stop Copasi optimizations, as well as see their results graphically.

1.4 Installation

There's no installation necessary. Download and extract the SpaceScanner source code. Alternatively, get it through Git: `git clone https://github.com/atiselsts/spacescanner.git`.
Prerequisites:

- Python (version 2.7);
- `psutil` Python module.

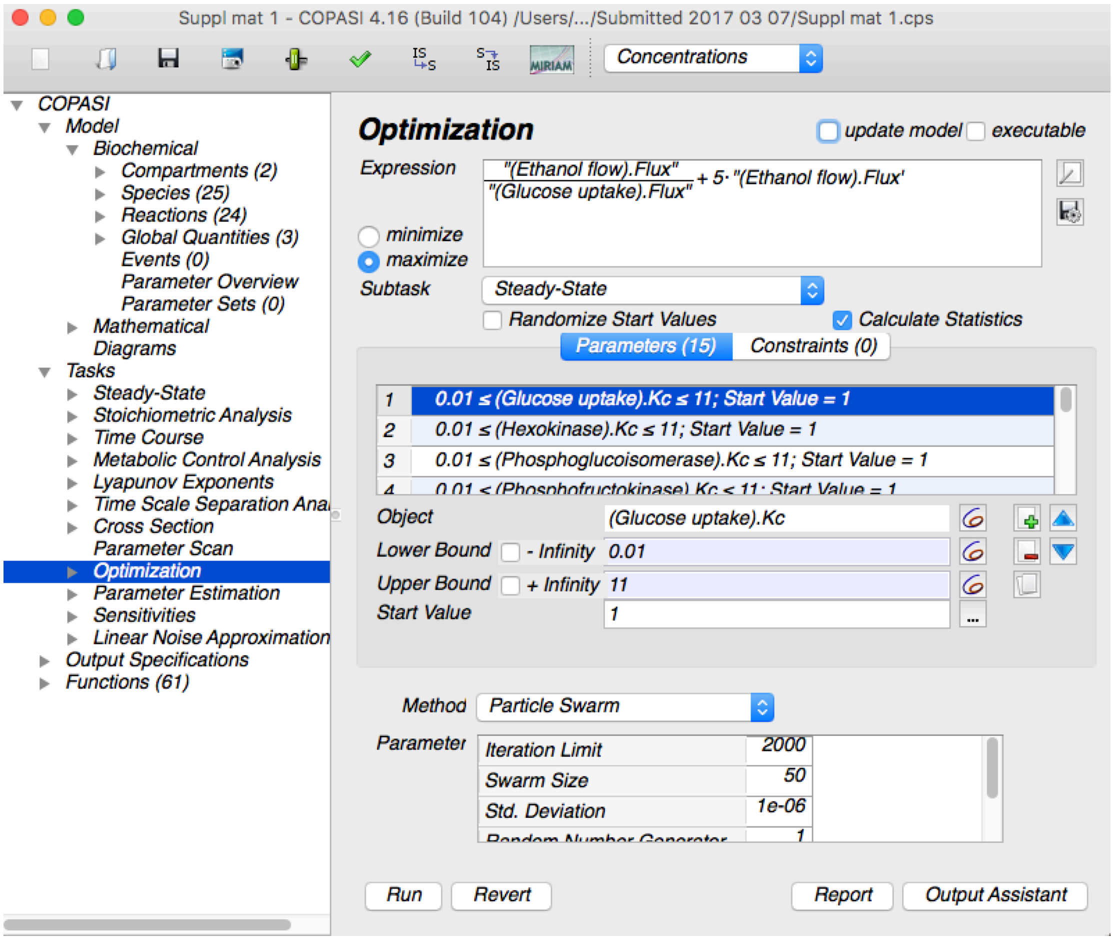
Install `psutil` with PIP, Python package manager, e.g.:
```
sudo pip install psutil
```
To install `pip`, it may be possible to use `easy_install`:
```
sudo easy_install pip
```
Alternatively, run this on Ubuntu Linux to install `psutil`:
```
sudo apt-get install python-psutil
```
SpaceScanner has been successfully tested on 64-bit Linux and Windows, including Cygwin.

In case of installation problems, refer to instructions at https://github.com/atiselsts/spacescanner

2. Creating a model file for *SpaceScanner*

COPASI (Hoops et al., 2006; Mendes et al., 2009) (http://copasi.org/) can work with own format (.cps file) or with SBML (Hucka et al., 2003) (.xml) files. The optimization task has to be set in the model file using COPASI: *SpaceScanner* has to receive a .cps file fully configured for optimization (at least the expression of objective function and adjustable parameters are set).

All adjustable parameters *SpaceScanner* will work with have to be defined. *SpaceScanner* will create the necessary combinations from parameters list automatically.


Suppl mat 1 - COPASI 4.16 (Build 104) /Users/.../Submitted 2017 03 07/Suppl mat 1.cps

Concentrations

- COPASI
  - Model
    - Biochemical
      - Compartments (2)
      - Species (25)
      - Reactions (24)
      - Global Quantities (3)
      - Events (0)
      - Parameter Overview
      - Parameter Sets (0)
    - Mathematical
    - Diagrams
  - Tasks
    - Steady-State
    - Stoichiometric Analysis
    - Time Course
    - Metabolic Control Analysis
    - Lyapunov Exponents
    - Time Scale Separation Ana
    - Cross Section
    - Parameter Scan
    - Optimization
    - Parameter Estimation
    - Sensitivities
    - Linear Noise Approximation
  - Output Specifications
  - Functions (61)

Optimization

update model [ ] executable [ ]

Expression: $\frac{\text{"(Ethanol flow).Flux"}}{\text{"(Glucose uptake).Flux"}} + 5 \cdot \text{"(Ethanol flow).Flux'}$

minimize [✓ maximize]

Subtask: Steady-State

Randomize Start Values [ ] Calculate Statistics [✓]

Parameters (15) | Constraints (0)

1	0.01 ≤ (Glucose uptake).Kc ≤ 11; Start Value = 1
2	0.01 ≤ (Hexokinase).Kc ≤ 11; Start Value = 1
3	0.01 ≤ (Phosphoglucoisomerase).Kc ≤ 11; Start Value = 1
4	0.01 ≤ (Phosphofructokinase).Kc ≤ 11; Start Value = 1

Object: (Glucose uptake).Kc

Lower Bound [ ] - Infinity 0.01

Upper Bound [ ] + Infinity 11

Start Value 1

Method: Particle Swarm

Parameter:

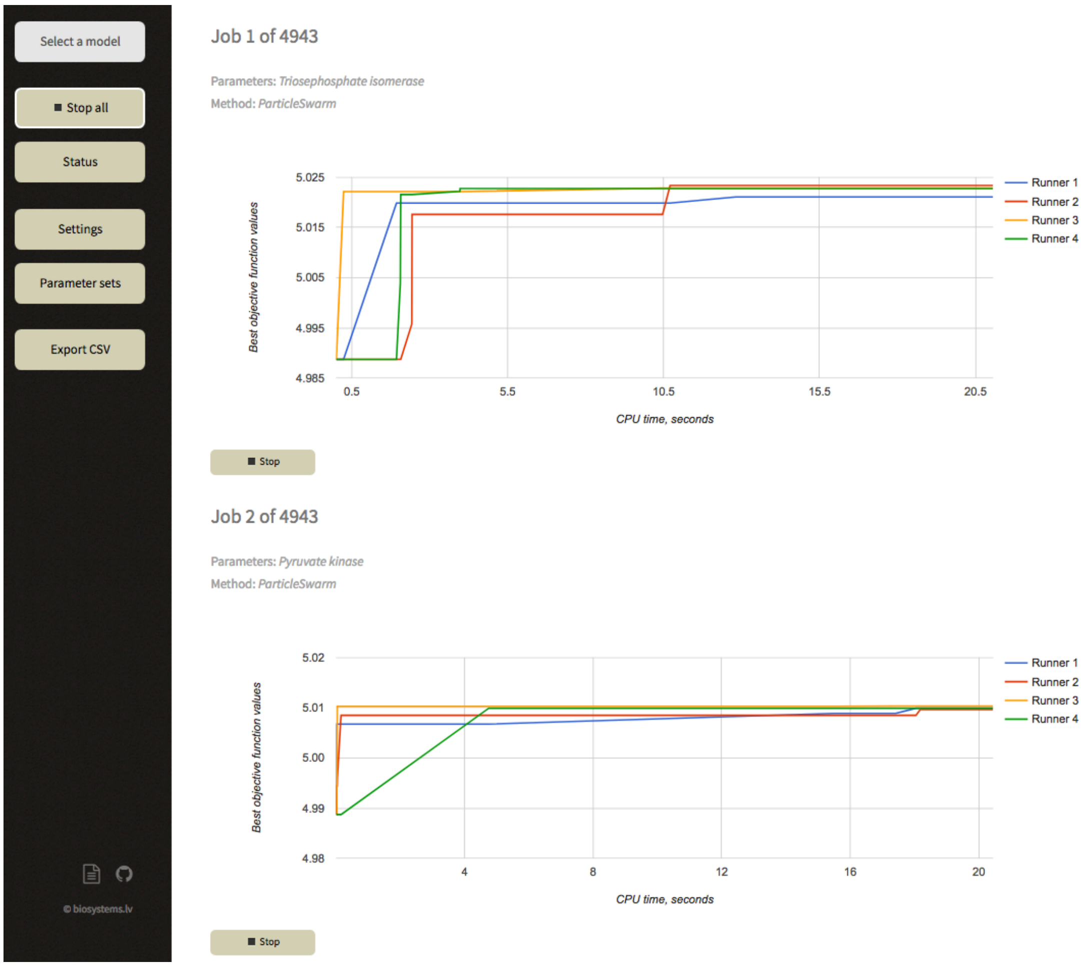
Iteration Limit	2000
Swarm Size	50
Std. Deviation	1e-06
Random Number Generator	1

3. Running SpaceScanner

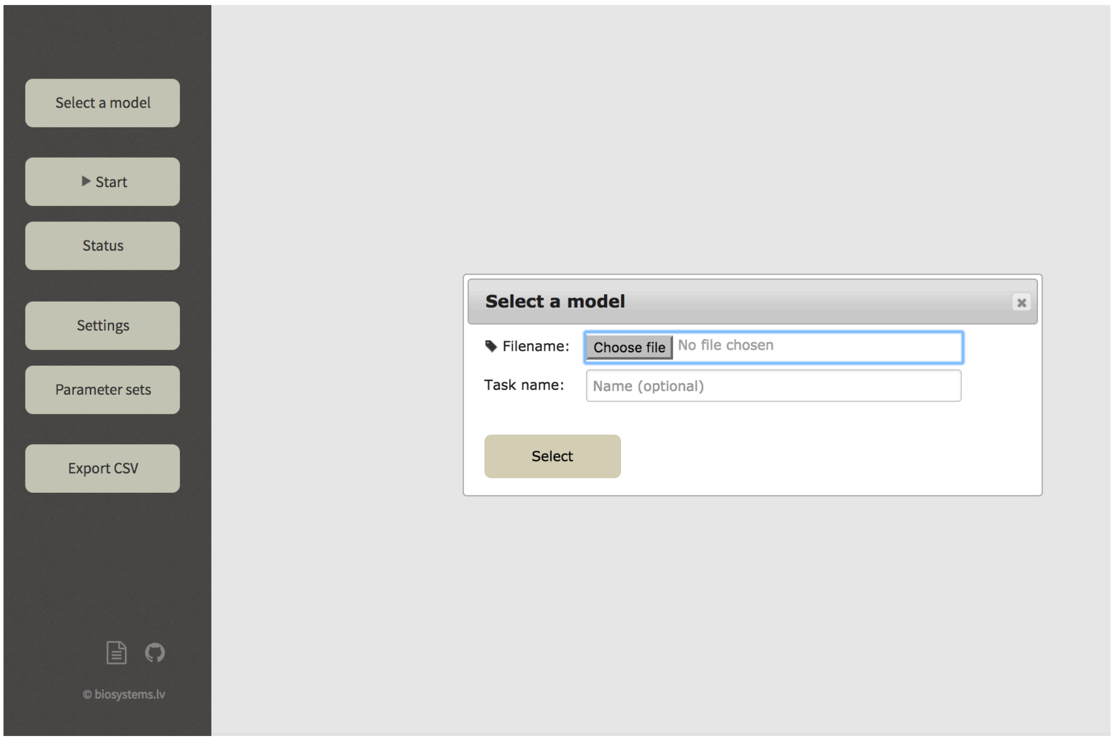
3.1 from the web interface

Double click on `spacescanner/spacescanner_launcher.py` file. This both starts the SpaceScanner executable (if not started yet) and opens the web interface in a web browser.
Alternatively, start `spacescanner/source/spacescanner.py` from a command line, passing the string "web" as the parameter, and open the web interface manually (default URL: `http://localhost:19000`).

SpaceScanner in running mode looks like in the screenshot below.



3.1.1 “Select a model” window



Select a model

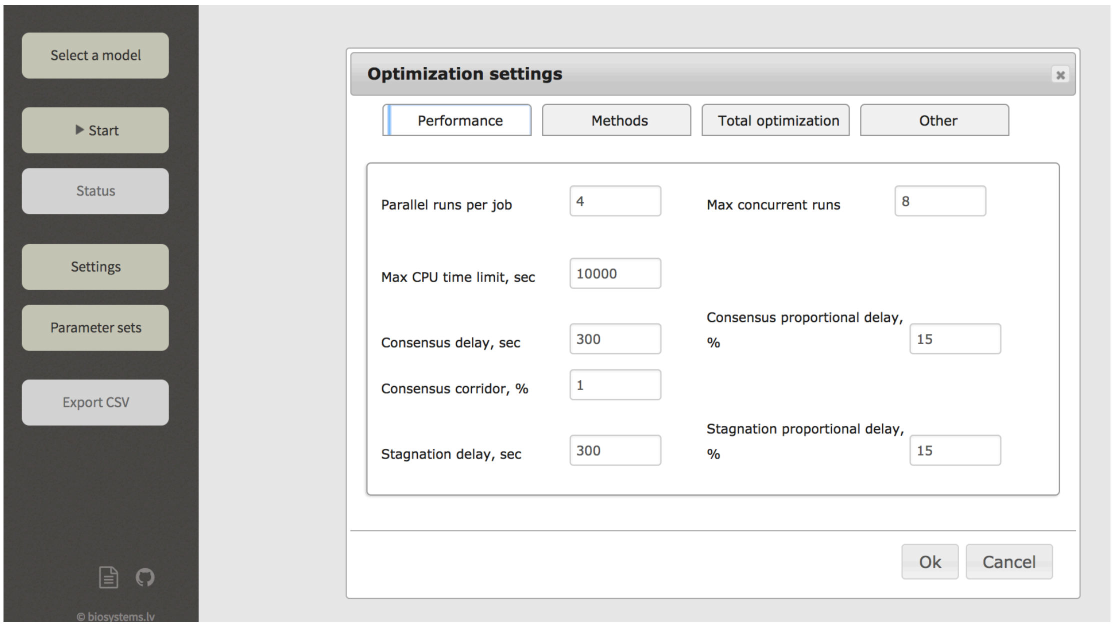
Filename:	Choose file No file chosen
Task name:	Name (optional)

Select

A model can be selected by choosing the prepared model file (see section 2) in .cps format.

3.1.2 "Settings" window

3.1.2.1 "Performance"



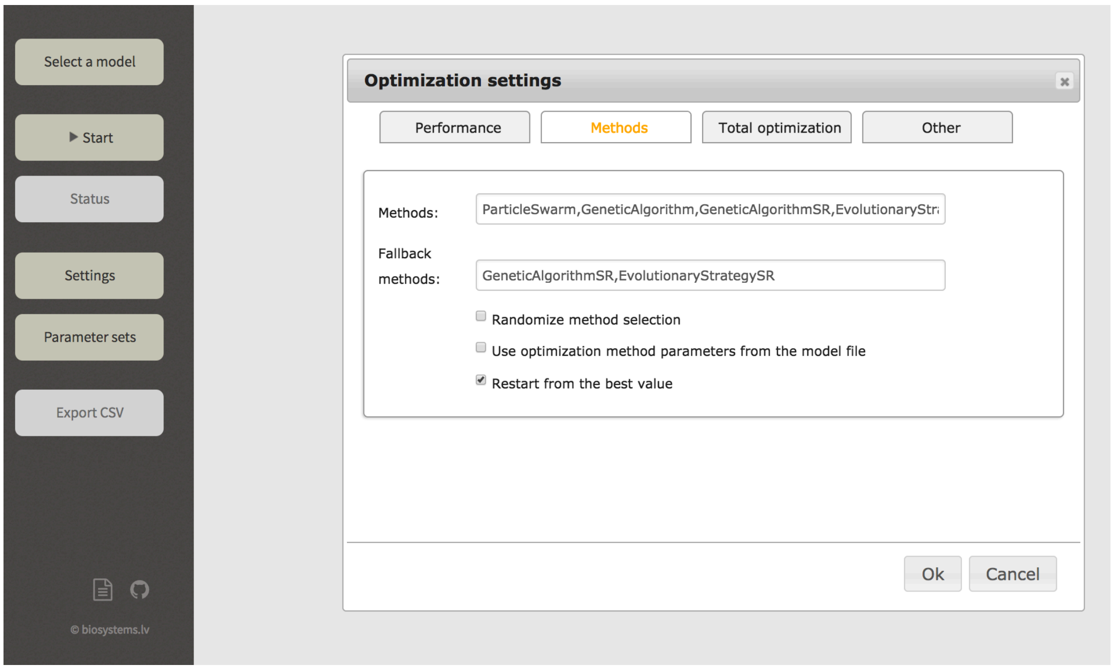
Optimization settings

Parallel runs per job	4	Max concurrent runs	8
Max CPU time limit, sec	10000		
Consensus delay, sec	300	Consensus proportional delay, %	15
Consensus corridor, %	1		
Stagnation delay, sec	300	Stagnation proportional delay, %	15

Parallel runs per job: definition of the number of parallel COPASI executions that will be run per job
Max. concurrent runs: the maximal number of simultaneous COPASI processes
Max CPU time limit, sec: maximal CPU time for optimization in case neither consensus, nor stagnation, nor other termination conditions have been reached.

Consensus delay, sec: minimal time to continue after the consensus criteria has been reached.
Consensus proportional delay, %: minimal time to continue after the consensus criteria has been reached, as a proportion of the runtime so-far.
Consensus corridor, %: "width" of consensus corridor to determine whether the consensus criteria has been reached. Consensus is reached if Objective Function values of all runners are within range [bOF-bOF*(Consensus corrodor,%); bOF] where bOF is Objective Function value of best runner.
Stagnation delay, sec: maximal duration to continue when no runners are finding new Objective Function values.
Stagnation proportional delay, %: maximal duration to continue when no runners are finding new Objective Function values, as a proportion of the runtime so-far.

3.1.2.2 "Methods"



Optimization settings

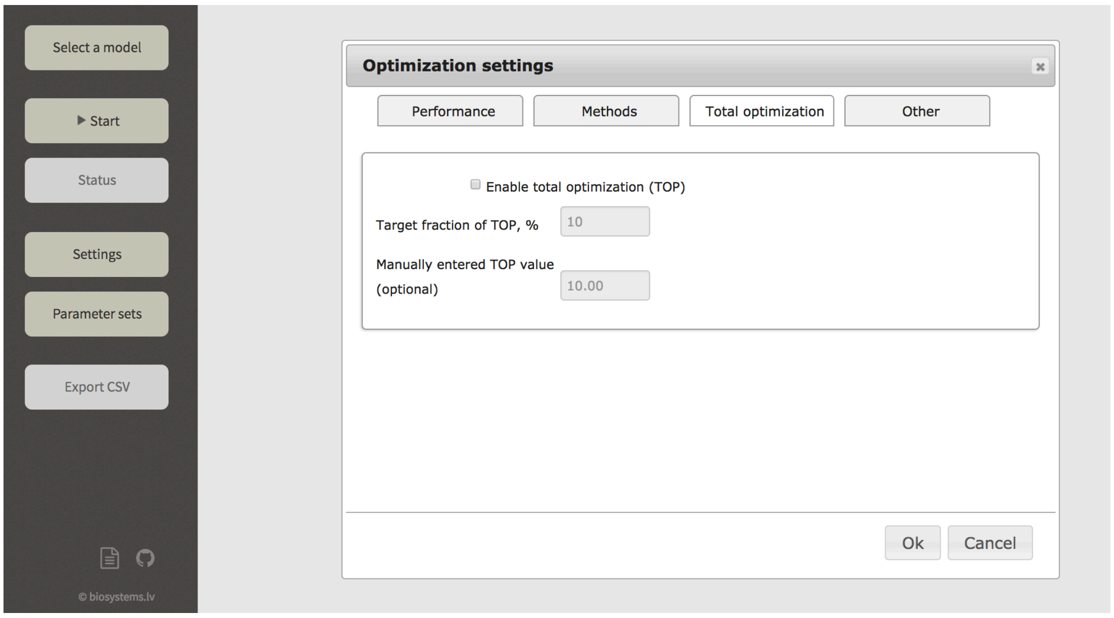
Methods: ParticleSwarm,GeneticAlgorithm,GeneticAlgorithmSR,EvolutionaryStr

Fallback methods: GeneticAlgorithmSR,EvolutionaryStrategySR

☐ Randomize method selection
☐ Use optimization method parameters from the model file
☑ Restart from the best value

Methods: definition of applied optimization methods from the methods list of COPASI. They will be applied in the listing sequence.
Fallback methods: List of optimization methods to use if a method from the first list fails to produce any solutions.
Randomize method selection: pick optimization methods from the list will be chosen in random order.
Use Optimization method parameters from the model file: take optimization method parameters from the model input file.
Restart from the best value: restart each subsequent method will be started from the best point in the search space so far?

3.1.2.3 "Total optimization"



Optimization settings

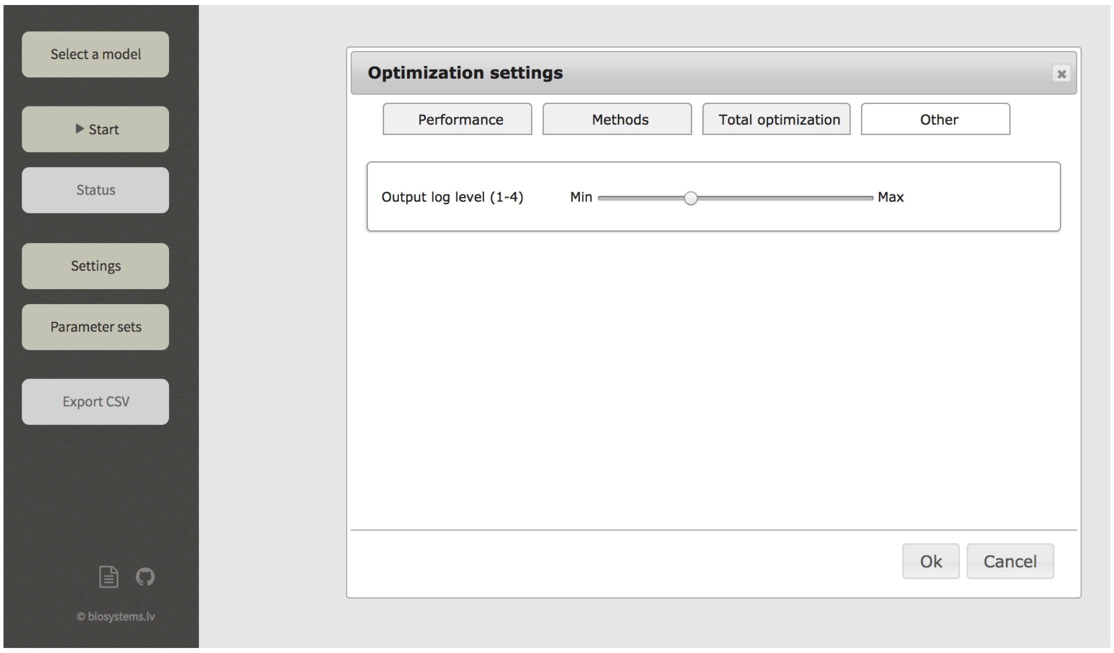
☐ Enable total optimization (TOP)

Target fraction of TOP, % 10

Manually entered TOP value (optional) 10.00

Enable total optimization (TOP) %: enables stopping the optimization when a entered fraction of total optimization potential (TOP) (Stalidzans et al., 2016) has been reached.
Target fraction of TOP, %: maximal difference between the optimal value and a satisfactory value.
Manually entered TOP value (optional): all-parameter optimization value used if this is not set, the best of the two used if set.

3.1.2.4 "Other"

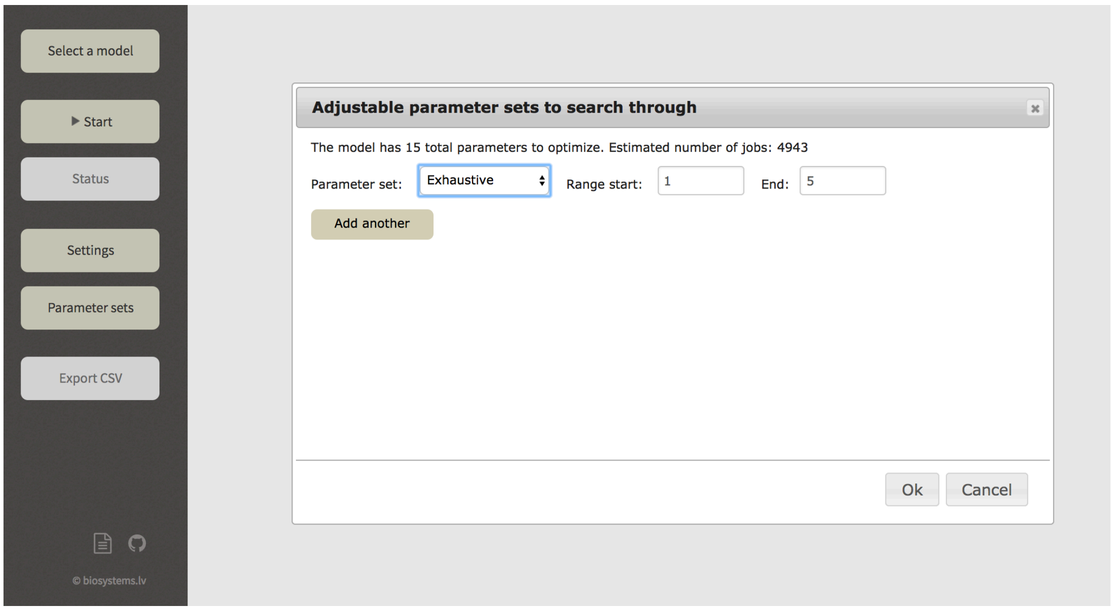


Optimization settings

Output log level (1-4) Min Max

Output log level: console output level (higher means more detailed output).
Stop server: terminates Space Scanner server.

3.1.3 “Parameter sets”



Adjustable parameter sets to search through

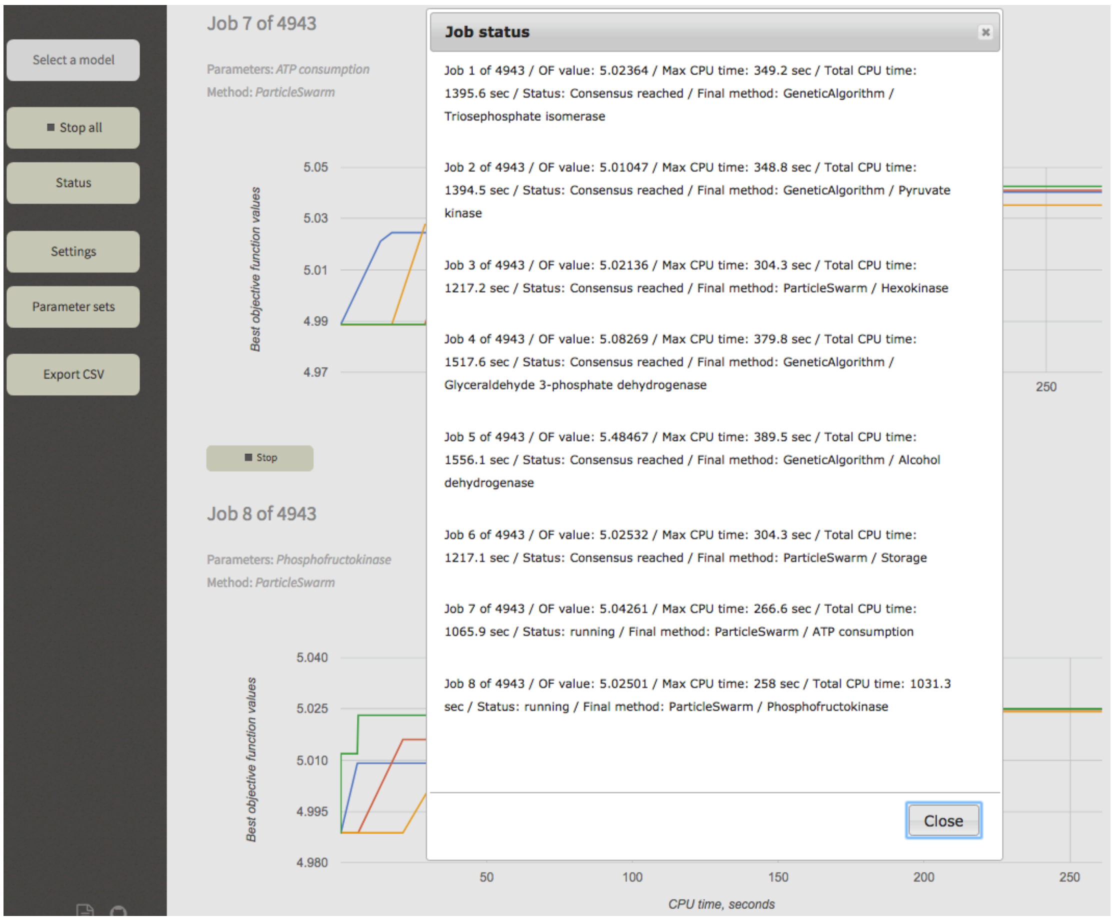
The model has 15 total parameters to optimize. Estimated number of jobs: 4943

Parameter set: Exhaustive Range start: 1 End: 5

In the window “Parameter sets” the fraction of adjustable parameter combination space to be addressed in the optimization can be defined.
In the first row the number of adjustable parameters found in the model and the number of optimization jobs at current parameter choice is indicated. In the example figure above it tells that 15 adjustable parameters are found and doing exhaustive search for 1-5 parameters in combinations the number of jobs (adjustable parameter combinations) is 4943.

Several types of search types are available from dropdown menu.

3.1.4 “Status” button



Job status

Job 1 of 4943 / OF value: 5.02364 / Max CPU time: 349.2 sec / Total CPU time: 1395.6 sec / Status: Consensus reached / Final method: GeneticAlgorithm / Triosephosphate isomerase

Job 2 of 4943 / OF value: 5.01047 / Max CPU time: 348.8 sec / Total CPU time: 1394.5 sec / Status: Consensus reached / Final method: GeneticAlgorithm / Pyruvate kinase

Job 3 of 4943 / OF value: 5.02136 / Max CPU time: 304.3 sec / Total CPU time: 1217.2 sec / Status: Consensus reached / Final method: ParticleSwarm / Hexokinase

Job 4 of 4943 / OF value: 5.08269 / Max CPU time: 379.8 sec / Total CPU time: 1517.6 sec / Status: Consensus reached / Final method: GeneticAlgorithm / Glyceraldehyde 3-phosphate dehydrogenase

Job 5 of 4943 / OF value: 5.48467 / Max CPU time: 389.5 sec / Total CPU time: 1556.1 sec / Status: Consensus reached / Final method: GeneticAlgorithm / Alcohol dehydrogenase

Job 6 of 4943 / OF value: 5.02532 / Max CPU time: 304.3 sec / Total CPU time: 1217.1 sec / Status: Consensus reached / Final method: ParticleSwarm / Storage

Job 7 of 4943 / OF value: 5.04261 / Max CPU time: 266.6 sec / Total CPU time: 1065.9 sec / Status: running / Final method: ParticleSwarm / ATP consumption

Job 8 of 4943 / OF value: 5.02501 / Max CPU time: 258 sec / Total CPU time: 1031.3 sec / Status: running / Final method: ParticleSwarm / Phosphofructokinase

Status button returns information about the finished and running jobs:
Job x gives the number of job
OF value gives information about the best objective function value among all runners within particular job
Max CPU time tells the duration of the longest single runner as the duration of all runs is close to equal but not identical.
Total CPU time gives the total time spent by all runners in the job. Usually “Max CPU time” multiplied by the number of runners per job is close to the “Total CPU time” value.
Status can have several values:

- value ***running*** means the job is still executed
- value ***consensus reached*** means that job is ended because consensus has been reached
- value ***CPU time limit reached*** means that the allowed CPU time (set in window Settings->Performance) has been reached before consensus has been reached.

Final method indicates the method that was used when the job was terminated because when consensus was reached or CPU time was exceeded.
After the last dash in the status report the ***set of adjustable parameters*** is indicated.

3.1.5 “Export CSV” button

A .csv file is exported showing all the information available under Status button (see section 3.1.4) as well as the final values of adjustable parameters.

3.2. from command line

Simply execute the `spacescanner/source/spacescanner.py` script, passing the configuration file name as a parameter.

Configuration file

The name of the file must be passed as command line parameter if run from the command line. If web interface is used, the file is automatically generated.

The configuration file contains a number of fields grouped in a number of sections.

3.2.1 "copasi" section

This section defines the model file and optimization methods to use.

Fields:

- `modelFile` - COPASI model file name; @SELF@ refers to SpaceScanner source directory
- `methods` - list of optimization methods to use; the methods are selected sequentially, each subsequent one is selected when the previous ones fail; can contain a method more than once
- `fallbackMethods` - list of optimization methods to use when a method fails to evaluate the objective function in given time; useful for e.g. highly constrained models on which many methods may not find any solutions at all
- `randomizeMethodSelection` - whether to pick methods from the configuration randomly or in order (default: `false`)
- `methodParametersFromFile` - whether to use optimization method parameters from COPASI model file (default: `false`)

3.2.2 "optimization" section

Defines maximal duration of optimization runs, termination criteria etc.

Fields:

- `timeLimitSec` - maximal CPU time for optimization in case the consensus criteria and other end conditions have not been reached (default: 600 sec)
- `consensusCorridor` - the consensus criteria is satisfied if values of all runs are within this consensus corridor range (default: 1%)
- `consensusAbsoluteError` - to determine whether the consensus criteria has been reached (default: 1e-6)
- `consensusDelaySec` - the minimal time to continue after the consensus criteria has been reached (default: 300 sec)

- `consensusProportionalDelay` - the minimal time to continue after the consensus criteria has been reached as proportion of the runtime so-far (default: 15%)
- `stagnationDelaySec` - the maximal time to continue when no values of any parallel run are changing (default: 300 sec)
- `stagnationProportionalDelay` - the maximal time to continue when no values of any parallel run are changing, as proportion of the runtime so-far (default: 15%)
- `targetFractionOfTOP` - compared to the full-set objective function value or the user-defined TOP value(default: 0.0 (i.e., disabled), range: [0.0 .. 1.0])
- `bestOfValue` - the user-defined best (TOP) objective function's value
- `restartFromBestValue` - restart each subsequent method from the best point in the search space so far (default: `true`)
- `maxConcurrentRuns` - how many COPASI processes to run by parallel (default: max(4, the number of CPU cores); range: [1 .. number of CPU cores])
- `runsPerJob` - how many paraller COPASI processes per each job (i.e. a single set of parameters)

3.2.3 "parameters" section

Defines the way how subsets of paramters are selected. Note that the optimization parameters as such are defined in the `.sbml` model file, not here!
The section is an array of records of arbitrary length. If repeating or overlapping records are specified, an optimization job for a given subset of paramters is still run only once.

Records may have the following type:

- `full-set` - a single optimization job containing all parameters in the model as specified in the `.sbml` file
- `exhaustive - all` possible combinations of `N` to `M` parameters. Field "range" defines the values of `N` and `M`. For example, `"range" : [1, 3]` selects `N` to be equal to 1, `M` to 3. `"range" : [2]` selects `N = M = 2`.
- `greedy` - for `N` parameters, take the best set of `N-1` parameters and run all possible optimizations that add one parameter not yet in the set. Unless `N=1`, there must also be a record describing which strategy to use to for `N-1` parameter sets.
- `greedy-reverse` - similar to "greedy", but takes away a single parameter from the best `N+1` parameter set instead of adding it.
- `explicit` - the names of parameters to use are explicitly named in the "parameters" field of the record.

By default, these records are present:

- `full-set`;
- `exhaustive` with range [1..3];
- `greedy` with range [4..8].

3.2.4 "web" section

Web interface settings.

Fields:

- `enable` - whether to run the web interface (default: `true`). **Warning:** access control is not supported by SpaceScanner! Enable this only in trusted environments.
- `port` - http port number (default: 19000)

3.2.5 "output" section

Logging settings.

Fields:

- `filename` - the file name to use for optimization results (default: "results--.csv")
- `loglevel` - debug log level; from 0 to 4, higher means more messages (default: 2)
- `numberOfBestCombinations` - how many of the best parameter combinations to include in results for each number of parameters; 0 means unlimited (default: unlimited)

3.2.6 Not in separate sections

- `restartOnFile` - .csv file name on which to restart optimization runs, trying to complete timeouted jobs (default: `null`)
- `taskName` - the global name of this optimization task

3.2.7 Example configuration file

```
{
    "copasi" : {
        "modelFile" : "@SELF@/models/simple-6params.cps",
        "methods" : ["ParticleSwarm", "GeneticAlgorithm", "GeneticAlgorithmSR", "EvolutionaryProgram", "EvolutionaryS
        "fallbackMethods" : ["GeneticAlgorithmSR", "EvolutionaryStrategySR"],
        "randomizeMethodSelection" : false,
        "methodParametersFromFile" : false,
        "parameters": []
    },
    "optimization" : {
        "timeLimitSec" : 60,
        "consensusCorridor" : 0.01,
        "consensusAbsoluteError" : 1e-6,
        "consensusDelaySec" : 60,
        "consensusProportionalDelay" : 0.15,
        "targetFractionOfTOP" : 0.9,
        "bestOfValue" : -Infinity,
        "restartFromBestValue" : true,
        "maxConcurrentRuns" : 4,
        "runsPerJob" : 2
    },
    "parameters" : [
        {"type" : "full-set"},
        {"type" : "exhaustive", "range" : [1, 3]},
        {"type" : "greedy", "range" : [4, 8]}
    ],
    "restartOnFile" : null,
    "web" : {
        "enable" : true,
        "port" : 19000
    },
    "output" : {
        "filename" : "results.csv",
        "loglevel" : 2,
        "numberOfBestCombinations" : 0
    },
    "taskName" : null
}
```

3.3 Stopping SpaceScanner

- "Stop" button - terminate a specific, currently running job (stops all runners of that job).
- "Stop all" button - completely stops the analysis of the current model - terminates all running jobs and clears the queue of scheduled jobs.

To terminate the SpaceScanner server, either use the command line (Ctrl+C) or go to *Settings* -> *Other*. Here, a button for that is provided:

- "Stop server" - terminates the SpaceScanner application itself.

3.4 Accessing the results

SpaceScanner stores the results of finished jobs in `spacescanner/results` directory. **Warning:** for tasks with a large number of jobs these directories can take quite a lot of space on the disk!

Each optimization task (i.e., a collection of jobs) is stored in a separate directory. This directory contains the configuration of that task, the `.log` file of the execution, and `.csv` file where the results of finished jobs are stored.
Each job gets its own subdirectory. These subdirectories contain Copasi model files (stored as `.cps`), and Copasi process execution histories (stored as `.log` files). The `.cps` files contain the input model; for all finished runs they also include the parameter values on which that run achieved its best objective function value.

Using web interface the results are downloaded in a .csv file (see section 3.1.5)

References

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U., 2006. COPASI--a COmplex PAthway SImulator. Bioinformatics 22, 3067–74. doi:10.1093/bioinformatics/btl485

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, a. P., Bornstein, B.J., Bray, D., Cornish-Bowden, a., Cuellar, a. a., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, I.I., Hedley, W.J., Hodgman, T.C., Hofmeyr, J.-H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, a., Kummer, U., Le Novere, N., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J., 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19, 524–531. doi:10.1093/bioinformatics/btg015

Mendes, P., Hoops, S., Sahle, S., Gauges, R., Dada, J.O., Kummer, U., 2009. Computational Modeling of Biochemical Networks Using COPASI, in: Maly, I. V (Ed.), Methods in Molecular Biology, Systems Biology, Methods in Molecular Biology. Humana Press, Totowa, NJ, pp. 17–59. doi:10.1007/978-1-59745-525-1

Stalidzans, E., Mozga, I., Sulins, J., Zikmanis, P., 2016. Search for a minimal set of parameters by assessing the total optimisation potential for a dynamic model of a biochemical network. IEEE/ACM Trans. Comput. Biol. Bioinforma. 1–1. doi:10.1109/TCBB.2016.2550451