Evo^{2}Sim User Manual

for version 1.0.1 (October 29, 2017)

Contents

1 Introduction **5**
- 1.1 Introducing Evo²Sim . . . 5
- 1.2 License . . . 5
- 1.3 Community . . . 5
- 1.4 Download . . . 6
- 1.5 Contact . . . 6

2 Installation instructions **8**
- 2.1 Supported platforms . . . 8
  - 2.1.1 Required dependencies . . . 8
  - 2.1.2 Optional dependencies (for graphical outputs) . . . 8
  - 2.1.3 HTML viewer dependencies . . . 9
- 2.2 Software compilation . . . 9
  - 2.2.1 User mode . . . 9
  - 2.2.2 Debug mode . . . 9
  - 2.2.3 Executable files emplacement . . . 9

3 Typical usage **10**
- 3.1 Creating a simulation . . . 10
- 3.2 Generating viable initial conditions with a bootstrap . . . 11
- 3.3 Running a simulation . . . 11

4 Simulation viewer **12**
- 4.1 Population . . . 12
- 4.2 Best lineage . . . 12
- 4.3 Best individual . . . 12
- 4.4 Environment . . . 12
- 4.5 Phylogeny . . . 13
- 4.6 Parameters . . . 13

A Main executables description **14**
- A.1 `evo2sim_create` executable . . . 14
- A.2 `evo2sim_bootstrap` executable . . . 14
- A.3 `evo2sim_run` executable . . . 15
- A.4 `evo2sim_generate_figures` executable . . . 16
- A.5 `evo2sim_recover_parameters` executable . . . 16
- A.6 `evo2sim_unitary_tests` executable . . . 16

A.7 `evo2sim_integrated_tests` executable . . . 17
A.8 Other executables . . . 17

B Parameters description **18**
B.1 Pseudorandom numbers generator . . . 18
B.2 Parallel computing . . . 18
B.3 Simulation schemes . . . 18
B.3.1 Energy costs scheme . . . 18
B.3.2 Membrane permeability scheme . . . 19
B.3.3 Metabolic inheritance scheme . . . 19
B.3.4 Enzymatic inheritance scheme . . . 19
B.3.5 Co-enzymes scheme . . . 19
B.3.6 Score scheme . . . 19
B.3.7 Selection threshold . . . 20
B.4 Space . . . 20
B.4.1 Grid width . . . 20
B.4.2 Grid height . . . 20
B.5 Output . . . 20
B.5.1 Simulation backup step . . . 20
B.5.2 Figures generation step . . . 21
B.6 Genome . . . 21
B.6.1 Load the genome from file . . . 21
B.6.2 Metabolite tags initial range . . . 21
B.6.3 Binding site tags initial range . . . 22
B.6.4 Co-enzyme tags initial range . . . 22
B.6.5 Transcription factor tags initial range . . . 22
B.6.6 Transcription factors binding window . . . 22
B.6.7 Initial number of non-coding units . . . 22
B.6.8 Initial number of enzyme coding units . . . 22
B.6.9 Initial number of transcription factor coding units . . . 23
B.6.10 Initial number of binding site units . . . 23
B.6.11 Initial number of promoter units . . . 23
B.6.12 Point mutation rate . . . 23
B.6.13 Duplication rate . . . 23
B.6.14 Deletion rate . . . 23
B.6.15 Translocation rate . . . 23
B.6.16 Inversion rate . . . 23
B.6.17 Transition rate . . . 24
B.6.18 Breakpoint rate . . . 24
B.6.19 Substrate tag mutation size . . . 24
B.6.20 Product tag mutation size . . . 24
B.6.21 k_{cat} mutation size . . . 24
B.6.22 k_{cat}/k_M ratio mutation size . . . 24
B.6.23 Binding site tag mutation size . . . 24
B.6.24 Co-enzyme tag mutation size . . . 25
B.6.25 Transcription factor tag mutation size . . . 25

B.6.26 Basal expression level mutation size . . . 25
B.7 Genetic regulation network . . . 25
B.7.1 Genetic regulation network time-steps ratio . . . 25
B.7.2 Hill function theta parameter . . . 25
B.7.3 Hill function n parameter . . . 25
B.7.4 Protein degradation rate . . . 26
B.8 Metabolic network . . . 26
B.8.1 Metabolism time-steps . . . 26
B.8.2 Essential metabolites toxicity threshold . . . 26
B.8.3 Non-essential metabolites toxicity threshold . . . 26
B.8.4 Initial metabolite amount in cells . . . 26
B.8.5 Maximum reaction size . . . 26
B.9 Energy . . . 27
B.9.1 Energy transcription cost . . . 27
B.9.2 Energy degradation cost . . . 27
B.9.3 Energy enzymatic cost . . . 27
B.9.4 Energy pumping cost . . . 27
B.9.5 Energy dissipation rate . . . 28
B.9.6 Energy toxicity threshold . . . 28
B.9.7 Initial energy amount in cells . . . 28
B.10 Cell . . . 28
B.10.1 Membrane permeability . . . 28
B.11 Population . . . 28
B.11.1 Death probability . . . 28
B.11.2 Migration rate . . . 28
B.11.3 HGT rate . . . 29
B.12 Environment . . . 29
B.12.1 Environment initialization cycles . . . 29
B.12.2 Environment species tags range . . . 29
B.12.3 Environment concentrations range . . . 29
B.12.4 Environment number of species range . . . 29
B.12.5 Environment interaction scheme . . . 30
B.12.6 Environment renewal scheme . . . 30
B.12.7 Environment variation scheme . . . 30
B.12.8 Environment localization scheme . . . 30
B.12.9 Environment metabolic scheme . . . 31
B.12.10 Environment introduction rate . . . 31
B.12.11 Environment diffusion coefficient . . . 31
B.12.12 Environment degradation rate . . . 31

Chapter 1

Introduction

1.1 Introducing Evo²Sim

Evo²Sim is a multi-scale model of *in silico* experimental evolution, the virtual pendant of experimental evolution in wet laboratory (see Fig 1.1). The software is equiped with the whole tool case of experimental setups, competition assays, phylogenetic analysis, and, most importantly, allowing for evolvable ecological interactions. Digital organisms with an evolvable genome structure, encoding evolvable genetic regulation and metabolic networks are evolved for tens of thousands of generations in environments mimicking the dynamics of real controlled environments, including chemostat or batch culture.

Evo²Sim was developed under EvoEvo (http://www.evoevo.eu/), a FP7-ICT project funded by the European Commission (FP7-ICT-610427). The source code is written in C++.

You can find more details on software description and development on Github page charlesrocabert/Evo2Sim. A website fully dedicated to Evo²Sim is coming soon.

1.2 License

1.3 Community

Evo²Sim was developed by Charles Rocabert, Carole Knibbe and Guillaume Beslon, under the EvoEvo project. The list of contributors is displayed in text-file `AUTHORS` of Evo²Sim package. You shall find more details on http://www.evoevo.eu/community/.

1.4 Download

Evo^{2}Sim last releases are available on Github page charlesrocabert/Evo2Sim.

1.5 Contact

For any question about the software, do not hesitate to contact us at http://www.evoevo.eu/contact-us/.

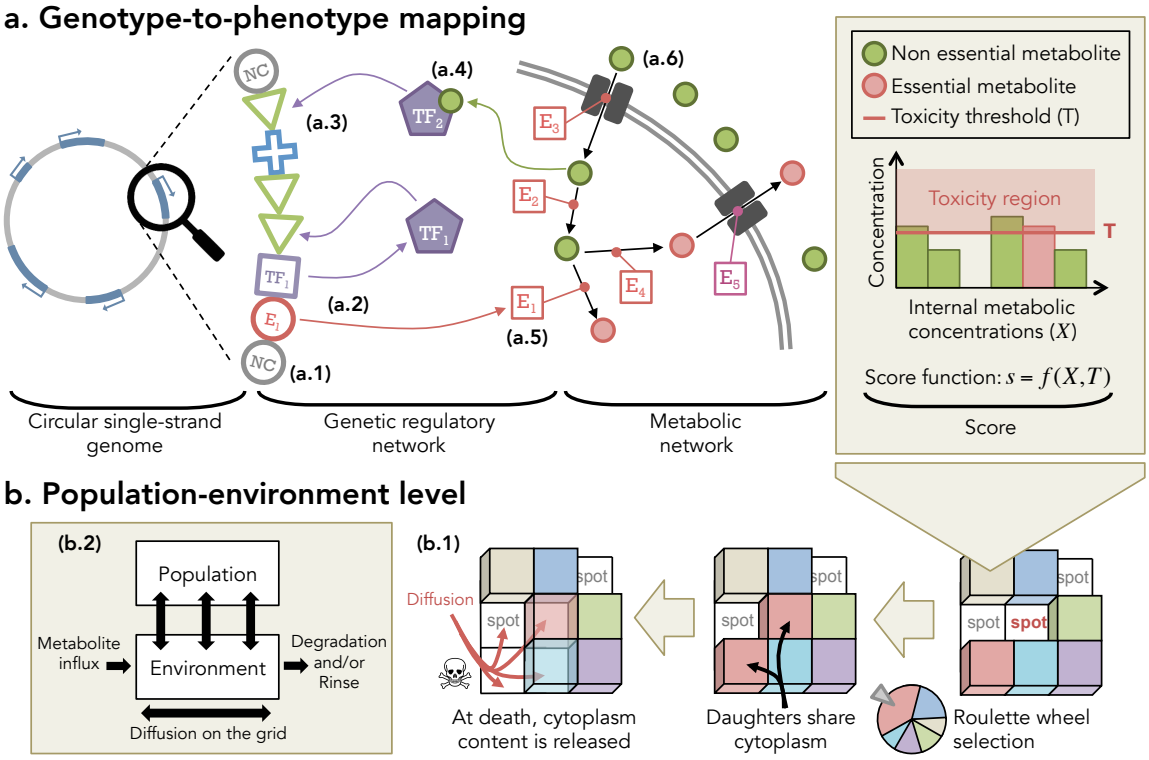

Figure 1.1: **Global picture of Evo²Sim. a. Description of the genotype-to-phenotype mapping.** Organisms own a coarse-grained genome made of units. This genome is a circular single-strand sequence, with a unique reading frame. Non coding **(NC)** units are not functional **(a.1)**. The arrangement of the units on the sequence defines functional regions, where a promoter (**P**, blue cross) controls the expression of enzyme coding units (**E**, red circles) or transcription factor coding units (**TF**, purple squares), thereby allowing for operons (here, one E and one TF). When coding units are expressed **(a.2)**, they contribute to the genetic regulatory network (for TFs) and the metabolic network (for Es). Depending on their attributes, transcription factors bind on binding sites. **(a.3)** If they bind on the enhancer sequence (binding sites flanking the promoter upstream), the promoter activity is up-regulated. If they bind on the operator sequence (binding sites flanking the promoter downstream), the promoter activity is down-regulated. **(a.4)** Metabolites can bind on a transcription factor as co-enzymes, and activate or inhibit it, depending on transcription factor attributes. Enzymes perform metabolic reactions in the cytoplasm **(a.5)**, or pump metabolites in or out **(a.6)**. The score of an organism is computed from its "essential metabolites" (usually the score is the sum of essential metabolite concentrations). Lethal toxicity thresholds are applied to each metabolic concentration and forbid organisms to accumulate resources. **b. Description of the population and environment levels.** Organisms are placed on a 2D toroidal grid, and compete for resources and space. When an organism dies, it leaves its grid cell empty and organisms in the Moore neighborhood (if any) compete to divide in available space. The competition is based on scores, a minimal threshold being applied on scores to forbid worst organisms to divide. At division, daughters share cytoplasm content (enzymes and metabolites). At death, metabolites from the cytoplasm are released in the local environment, and diffuse on the grid **(b.1)**. On the largest scale, the population evolves on the environment by up-taking, transforming and releasing metabolites. Metabolites then diffuse and are degraded. This strong interaction between the population and the environment allows for the evolution of complex ecological situations, depending on environmental properties **(b.2)**.

Chapter 2

Installation instructions

Download the latest release of Evo^{2}Sim on Github page charlesrocabert/Evo2Sim and save it to a directory of your choice. Open a terminal and use the `cd` command to navigate to this directory. Then follow the steps below to compile and build the executables.

2.1 Supported platforms

Evo^{2}Sim software has been successfully tested on Ubuntu 12.04 LTS, Ubuntu 14.04 LTS, OSX 10.9.5 (Maverick) and OSX 10.10.1 (Yosemite).

2.1.1 Required dependencies

- A `C++` compiler (GCC, LLVM, ...)
- `CMake` (command line version)
- `zlib`
- `GSL`
- `CBLAS`
- `TBB`
- `R` (packages `ape` and `RColorBrewer` are needed)

2.1.2 Optional dependencies (for graphical outputs)

- `X11` (or `XQuartz` on latest OSX version)
- `SFML 2`
- `matplotlib` (this python library is needed for the script `track_cell.py` (see below)

2.1.3 HTML viewer dependencies

- Javascript must be activated in your favorite internet browser

Note, however, that Evo^{2}Sim can be compiled without graphical outputs, and hence no need for X and SFML libraries (see compilation instructions below for more information). This option is useful if you want to run Evo^{2}Sim on a computer cluster, for example.

2.2 Software compilation

2.2.1 User mode

To compile Evo^{2}Sim, run the following instructions on the command line:

```
$ cd cmake/
```

and

```
$ bash make.sh
```

To gain performances during large experimental protocols, or on computer cluster, you should compile the software without graphical outputs:

```
$ bash make_no_graphics.sh
```

2.2.2 Debug mode

To compile the software in `DEBUG` mode, use `make_debug.sh` script instead of `make.sh`:

```
$ bash make_debug.sh
```

When Evo^{2}Sim is compiled in `DEBUG` mode, a lot of tests are computed on the fly during a simulation (*e.g.* integrity tests on phylogenetic trees, or on the ODE solver ...). For this reason, this mode should only be used for test or development phases. Moreover, unitary and integrated tests must be ran in `DEBUG` mode (see below).

2.2.3 Executable files emplacement

Binary executable files are in `build/bin` folder.

Chapter 3

Typical usage

Evo^{2}Sim includes three main executables (`evo2sim_create`, `evo2sim_bootstrap` and `evo2sim_run`), and a set of executables dedicated to post-treatments, data recovery or tests.

Everything in Evo^{2}Sim relies on an ad-hoc file organization where all the data for a simulation is stored: populations in the `population` directory, environments in `environment`, phylogenetic and lineage trees in `tree` and so on. It is not recommended to manually modify these files since this may cause some inconsistency leading to undefined behavior. Besides, most of these files are compressed.

Open a terminal and use the `cd` command to navigate to Evo^{2}Sim directory. A typical parameters file is provided in Evo^{2}Sim package, in folder `example` (an exhaustive description of the parameters is available in chapter "Parameters description"). Navigate to this folder using the `cd` command. Then follow the steps below for a first usage of the software.

3.1 Creating a simulation

Create a fresh simulation from the parameters file (by default `parameters.txt`):

```
$ ../build/bin/evo2sim_create
```

Several folders have been created. They mainly contain simulation backups (population, environment, trees, parameters, ...). Additional files and folders have also been created:

- `version.txt`: this file indicates the version of the software. This information is useful to ensure that the code version is compatible with the backup files (*e.g.*, in case of post-treatments).
- `track_cell.py`: when executed, this python script displays on the fly the internal protein and metabolic concentrations of the cell at position 0×0 on the grid. This script is useful to get an idea of internal cell's dynamics (metabolic fluxes, regulation, ...).
- `viewer` folder: the viewer is central to the usage of Evo^{2}Sim (see chapter "Simulation viewer"). To access the viewer, open the html page `viewer/viewer.html` in an internet browser.

3.2 Generating viable initial conditions with a bootstrap

Alternatively to the `evo2sim_create` executable, use a bootstrap to find a simulation with good initial properties from the parameters file:

```
$ ../build/bin/evo2sim_bootstrap
```

A fresh simulation with an updated parameters file will be automatically created if a suitable seed is found.

3.3 Running a simulation

In Evo^{2}Sim, running a simulation necessitates to load it from backup files. Here, we will run a simulation from freshly created backups (see above):

```
$ ../build/bin/evo2sim_run -b 0 -t 10000 -g
```

with `-b` the date of the backup, here 0 (fresh simulation), `-t` the simulation time, here 10,000 time-steps. Option `-g` activates the graphical output (not works if the software has been compiled with the no-graphics option). At any moment during the simulation, you can take a closer look at the evolution of the system by opening `viewer/viewer.html` in an internet browser. You can track internal cell's dynamics by executing the script `track_cell.py`.

Other main executables are described below in section "Main executables description". You can also obtain help by running the executable with the `-h` option (e.g. `evo2sim_create -h`)

Chapter 4

Simulation viewer

Evo^{2}Sim comes with an HTML viewer displaying a very complete set of live statistics. Each new simulation owns a dedicated viewer, which is frequently actualized on the fly (by default, every 500 simulation time-steps). This viewer has been developed using `Bootstrap`, `DyGraph`, `CytoscapeJS`, `ChartJS` and `JQuery`.

To access the viewer, simply open `viewer/viewer.html` in an internet browser (Javascript must be enabled). The different tabs are described below.

4.1 Population

This page displays the evolution of main population statistics (population size, mean genome size, mean score, ...), as well as the evolution of the trophic network.

4.2 Best lineage

This page displays the evolution of last best individual statistics. These informations are the most representative of evolutionary dynamics, since they contains all the mutations fixed since the beginning of the simulation.

4.3 Best individual

This page displays some informations about the last best individual at the moment of the visualization (genome state, genetic regulation network, metabolic network, internal metabolic state, ...).

4.4 Environment

This page displays the evolution of main environment statistics, as well as its current state.

4.5 Phylogeny

This page displays various rendering of the current phylogenetic tree, as well as some evolution statistics (number of nodes, common ancestor age, ...).

4.6 Parameters

This page displays the parameters file used to create the simulation, as well as a short description of parameters usage.

Appendix A

Main executables description

A.1 `evo2sim_create` executable

Create a fresh simulation from a parameters file.

Usage:

$ `evo2sim_create -h` or `--help`

or

$ `evo2sim_create [options]`

Options are:

`-h, --help`: print this help, then exit (optional)

`-v, --version`: print the current version, then exit (optional)

`-f, --file`: specify the parameters file (default: `parameters.txt`)

`-rs, --random-seed`: the prng seed is drawn at random (optional)

Be aware that creating a simulation in a folder completely erases previous simulation.

A.2 `evo2sim_bootstrap` executable

Run a bootstrap to find viable initial conditions.

Usage:

$ `evo2sim_bootstrap -h` or `--help`

or

$ `evo2sim_bootstrap [options]`

Options are:

`-h, --help:` print this help, then exit (optional)

`-v, --version:` print the current version, then exit (optional)

`-f, --file:` specify the parameters file (default: `parameters.txt`)

`-min, --minimum-time:` specify the minimum time the new population must survive (default: 100)

`-pop, --minimum-pop-size:` specify the minimum size the new population must maintain (default: 500)

`-t, -trials:` specify the number of trials (default: 1000)

`-g, --graphics:` activate graphic display (optional)

A simulation is automatically created if good conditions are found. The parameters file is also edited to include the corresponding prng seed value. Be aware that creating a simulation in a folder completely erases previous simulation.

A.3 `evo2sim_run` executable

Run a simulation from backup files.

Usage:

$ `evo2sim_run -h` or `--help`

or

$ `evo2sim_run [options]`

Options are:

`-h, --help:` print this help, then exit (optional)

`-v, --version:` print the current version, then exit (optional)

`-b, --backup-time:` set the date of the backup to load (default: 0)

`-t, --simulation-time:` set the duration of the simulation (default: 10000)

`-g, --graphics:` activate graphic display (optional)

Statistic files content is automatically managed when a simulation is reloaded from backup to avoid data loss.

A.4 `evo2sim_generate_figures` executable

Extract statistics and generate viewer figures from backup files.

Usage:

$ `evo2sim_generate_figures -h` or `--help`

or

$ `evo2sim_generate_figures [options]`

Options are:

`-h, --help`: print this help, then exit (optional)

`-v, --version`: print the current version, then exit (optional)

`-b, --backup-time`: set the date of the backup to load (mandatory)

A.5 `evo2sim_recover_parameters` executable

Recover the parameters file from backup files.

Usage:

$ `evo2sim_recover_parameters -h` or `--help`

or

$ `evo2sim_recover_parameters [options]`

Options are:

`-h, --help`: print this help, then exit (optional)

`-v, --version`: print the current version, then exit (optional)

`-f, --file`: specify the name of the parameters file to save (mandatory)

A.6 `evo2sim_unitary_tests` executable

Run unitary tests.

Usage:

$ `evo2sim_unitary_tests -h` or `--help`

or

$ `evo2sim_unitary_tests [options]`

Options are:

`-h, --help`: print this help, then exit (optional)

`-v, --version`: print the current version, then exit (optional)

`-f, --file`: specify the parameters file (default: `parameters.txt`)

To use the unitary tests, the software must be compiled in DEBUG mode (see installation instructions below).

A.7 `evo2sim_integrated_tests` executable

Run integrated tests.

Usage:

$ `evo2sim_integrated_tests -h` or `--help`

or

$ `evo2sim_integrated_tests [options]`

Options are:

`-h, --help`: print this help, then exit (optional)

`-v, --version`: print the current version, then exit (optional)

`-f, --file`: specify the parameters file (default: `parameters.txt`)

`-tests, --number-of-tests`: specify the number of tests with different seeds (default: 1)

`-steps, --number-of-steps`: specify the number of steps by test (default: 1)

`-rs, --random-seed`: the prng seed is drawn at random for each test (optional)

`-rp, --random-parameters`: the parameters are drawn at random for each test (optional)

To use the unitary tests, the software must be compiled in DEBUG mode (see installation instructions below).

A.8 Other executables

For all the other executables, you can obtain help by running the executable with the `-h` option (e.g. `evo2sim_create -h`)

Appendix B

Parameters description

All the parameters of the parameters file are described in details below. Each parameters receive at least on value. There is three types of values:

- **integer**: integer number
- **float**: floating point number
- **string**: characters string

For each parameter, the type is possibly bounded. In this case, boundaries are indicated.

B.1 Pseudorandom numbers generator

SEED <seed> (integer > 0)

Simply set the seed of the pseudorandom numbers generator (prng). The seed value is important since it allows to exactly replay a simulation if needed.

B.2 Parallel computing

PARALLEL_COMPUTING <choice> (YES/NO)

This parameter allows to activate, or deactivate, parallel computing at will. Parallel computing is managed by the external library `TBB`.

B.3 Simulation schemes

B.3.1 Energy costs scheme

ENERGY_COSTS_SCHEME <choice> (YES/NO)

Choose the energy scheme. By default, biochemical reactions are energy free in Evo^{2}Sim. When energy costs are activated, inner cell's chemical reactions produce or consume energy (an abstract view of energy carriers, like ATP). Transcription, enzymatic reactions and pumps else produce or cost energy to the cell, which must maintain its energy level to survive. Specific parameters are used to precisely set energy costs (see below).

B.3.2 Membrane permeability scheme

MEMBRANE_PERMEABILITY_SCHEME <choice> (YES/NO)

Choose membrane permeability scheme. If membrane permeability is activated, metabolites diffuse through the cell's membrane at a specific rate (see MEMBRANE_PERMEABILITY parameter below).

B.3.3 Metabolic inheritance scheme

METABOLIC_INHERITANCE_SCHEME <choice> (YES/NO)

Choose metabolic inheritance scheme. If this parameter is activated, the two daughter cells share the metabolic content of their parent. Each daughter cell inherits half of metabolic concentrations.

B.3.4 Enzymatic inheritance scheme

ENZYMATIC_INHERITANCE_SCHEME <choice> (YES/NO)

Choose enzymatic inheritance scheme. If this parameter is activated, the two daughter cells share the enzymatic content of their parent. Each daughter cell inherits half of enzymatic concentrations.

B.3.5 Co-enzymes scheme

CO_ENZYME_ACTIVITY <choice> (YES/NO)

Choose co-enzyme scheme. If this parameter is activated, some metabolites act as co-enzymes. Each transcription-factor owns a site where a specific metabolite can bind, activating or inhibiting the transcription factor depending on its properties. Activating this parameter increases the complexity of the genetic regulation network, and more importantly, allows cells to evolve environmental sensing.

B.3.6 Score scheme

SCORE_SCHEME <choice> (SUM/SUM_MINUS_DEV/COMBINATORIAL)

Choose the score scheme. The score of a cell is computed from its internal metabolic concentrations:

- SUM scheme: the score is simply the sum of essential metabolite concentrations;
- SUM_MINUS_DEV scheme: the score is the sum of essential metabolite concentrations, minus the standard deviation of the concentrations. This score adds an homeostatic constraint on cells.

- COMBINATORIAL scheme: the score is computed depending on relative essential metabolite concentrations. Basically, essential metabolites are considered to form complex molecules similar to RNA polymerases. Bigger is the polymerase, higher is its contribution to the score. Then, the bigger polymerase including all the essential metabolites is defined by the lowest concentration. Since the lowest metabolite is exhausted for this polymerase, the next one is the contribution of remaining metabolites, and so forth.

B.3.7 Selection threshold

SELECTION_THRESHOLD <threshold> (float $\in [0, 1]$)

Define a score threshold, above which cell's division is forbidden. When neighboring cells compete for a gap in the environment, one cell is elected at random by a roulette wheel draw, based on relative scores. However, a minimum threshold is mandatory to avoid individuals owning a very low score to divide, and drive the population in an artificial dead-end (where everybody is very bad, but nobody dies).

B.4 Space

B.4.1 Grid width

WIDTH <width> (integer > 0)

Simply define the width of the environmental grid.

B.4.2 Grid height

HEIGHT <height> (integer > 0)

Simply define the height of the environmental grid.

B.5 Output

B.5.1 Simulation backup step

SIMULATION_BACKUP_STEP <step> (integer ≥ 0)

Define the frequency at which backups of the simulation are saved. The resolution is in simulation time-steps. It is possible to exactly replay a simulation from backup files. Be aware that backup files size is large, the backup frequency must be reasonable (*e.g.*, 1,000 time-steps).

B.5.2 Figures generation step

FIGURES_GENERATION_STEP <step> (integer ≥ 0)

Define the frequency at which figures are generated for the html viewer. Some scripts used to generate figures may take more time to execute for very long simulations, the backup frequency must be reasonable (*e.g.*, 1,000 time-steps).

B.6 Genome

B.6.1 Load the genome from file

LOAD_GENOME_FROM_FILE <choice> (YES/NO)

Choose to generate genomes at random (NO, in this case, random generation depends on parameters below), or load a handcrafted genome from a file (YES). In case the handcrafted genome is loaded, it must be encoded in a file named `initial_genome.txt`. The structure of this file is specific and must respect the following scheme:

1. To encode non-coding units (NC), insert the following line: `NC <number of units>`. The specified number of random NC units will be inserted (`<number of units>` > 0);
2. To encode a promoter unit (P), insert the following line: `P <basal expression level>`. A promoter unit with a basal expression level $\beta =$ `<basal expression level>` will be inserted ($\beta \in [0, 1]$);
3. To encode a binding site unit (BS), insert the following line: `BS <TF tag>`. A binding site unit owning the specified transcription factor tag value will be inserted (`<TF tag>` $\in \mathbb{Z}$);
4. To encode a transcription factor coding unit (TF), insert the following line: `TF <BS tag> <CoE tag> <free activity> <bound activity> <binding window>`. A transcription factor coding unit with specified attributes will be inserted (`<BS tag>` $\in \mathbb{Z}$, `<CoE tag>` $\in \mathbb{N}^*$, `<free activity>` $\in \{true, false\}$, `<bound activity>` $\in \{true, false\}$, `<binding window>` ≥ 0);
5. To encode an enzyme coding unit (E), insert the following line: `E <substrate> <product>` `<`k_{cat}`>` `<`K_M`>`. An enzyme coding unit with specified attributes will be inserted (`<substrate>` > 0, `<product>` > 0, `<`k_{cat}`>` and `<`K_M`>` $\in$ specified boundaries).

B.6.2 Metabolite tags initial range

METABOLITE_TAG_INITIAL_RANGE <min> <max> (integer > 0; min $\leq$ max)

Define the initial distribution of metabolite tags encoded in the initial random genome (*i.e.*, in transcription factor and enzyme units). `min` and `max` values define the boundaries of a uniform law, used to draw the metabolite tags.

B.6.3 Binding site tags initial range

BINDING_SITE_TAG_INITIAL_RANGE <min> <max> (float > 0; min $\leq$ max)

Define the initial distribution of binding site tags encoded in the initial random genome (*i.e.*, in transcription factor units). `min` and `max` values define the boundaries of a uniform law, used to draw the binding site tags.

B.6.4 Co-enzyme tags initial range

CO_ENZYME_TAG_INITIAL_RANGE <min> <max> (float > 0; min $\leq$ max)

Define the initial distribution of co-enzyme tags encoded in the initial random genome (*i.e.*, in transcription factor units). `min` and `max` values define the boundaries of a uniform law, used to draw the co-enzyme tags.

B.6.5 Transcription factor tags initial range

TRANSCRIPTION_FACTOR_TAG_INITIAL_RANGE <min> <max> (float > 0; min $\leq$ max)

Define the initial distribution of transcription factor tags encoded in the initial random genome (*i.e.*, in binding site units). `min` and `max` values define the boundaries of a uniform law, used to draw the transcription factor tags.

B.6.6 Transcription factors binding window

TRANSCRIPTION_FACTOR_BINDING_WINDOW <window> (integer $\geq$ 0)

Define the "binding window" of a transcription factor on a binding site. If transcription factors and binding site tags are similar enough, the binding is allowed. More precisely if $\text{tag}_{TF} \in [\text{tag}_{TF} - \text{window} \ , \ \text{tag}_{TF} + \text{window}]$, the binding is possible.

B.6.7 Initial number of non-coding units

INITIAL_NUMBER_OF_NON_CODING_UNITS <number> (integer ≥ 0)

Define the number of random non-coding units in the initial random genome.

B.6.8 Initial number of enzyme coding units

INITIAL_NUMBER_OF_ENZYME_UNITS <number> (integer ≥ 0)

Define the number of random enzyme units in the initial random genome.

B.6.9 Initial number of transcription factor coding units

INITIAL_NUMBER_OF_TRANSCRIPTION_FACTOR_UNITS
<number> (integer ≥ 0)

Define the number of random transcription factor units in the initial random genome.

B.6.10 Initial number of binding site units

INITIAL_NUMBER_OF_BINDING_SITE_UNITS <number> (integer ≥ 0)

Define the number of random binding site units in the initial random genome.

B.6.11 Initial number of promoter units

INITIAL_NUMBER_OF_PROMOTER_UNITS <number> (integer ≥ 0)

Define the number of random promoter units in the initial random genome.

B.6.12 Point mutation rate

POINT_MUTATION_RATE <rate> (float $\in [0, 1]$)

Define the point mutation rate (in attribute^{-1}.replication^{-1}).

B.6.13 Duplication rate

DUPLICATION_RATE <rate> (float $\in [0, 1]$)

Define the duplication rate (in genomic-unit^{-1}.replication^{-1}).

B.6.14 Deletion rate

DELETION_RATE <rate> (float $\in [0, 1]$)

Define the deletion rate (in genomic-unit^{-1}.replication^{-1}).

B.6.15 Translocation rate

TRANSLOCATION_RATE <rate> (float $\in [0, 1]$)

Define the translocation rate (in genomic-unit^{-1}.replication^{-1}).

B.6.16 Inversion rate

INVERSION_RATE <rate> (float $\in [0, 1]$)

Define the inversion rate (in genomic-unit^{-1}.replication^{-1}).

B.6.17 Transition rate

TRANSITION_RATE <rate> (float $\in [0, 1]$)

Define the transition rate (in genomic-unit^{-1}.replication^{-1}).

B.6.18 Breakpoint rate

BREAKPOINT_RATE <rate> (float $\in [0, 1]$)

Define the breakpoint rate (in attribute^{-1}.breakpoint^{-1}).

B.6.19 Substrate tag mutation size

SUBSTRATE_TAG_MUTATION_SIZE <size> (integer ≥ 0)

Define the size of the uniform distribution used to mutate substrate tags (in enzyme units). The mutation is defined as tag + $\mathcal{U}$(-size, +size).

B.6.20 Product tag mutation size

PRODUCT_TAG_MUTATION_SIZE <size> (integer ≥ 0)

Define the size of the uniform distribution used to mutate product tags (in enzyme units). The mutation is defined as tag + $\mathcal{U}$(-size, +size).

B.6.21 k_{cat} mutation size

KCAT_MUTATION_SIZE <size> (float ≥ 0.0)

Define the standard deviation of the gaussian distribution used to mutate k_{cat} constant (in enzyme units). The mutation is defined as $log_{10}(k_{cat})$ + $\mathcal{N}$(0, size).

B.6.22 k_{cat}/k_M ratio mutation size

KCAT_KM_RATIO_MUTATION_SIZE <size> (float ≥ 0.0)

Define the standard deviation of the gaussian distribution used to mutate k_{cat}/k_M ratio (in enzyme units). The mutation is defined as $log_{10}(k_{cat}/k_M)$ + $\mathcal{N}$(0, size).

B.6.23 Binding site tag mutation size

BINDING_SITE_TAG_MUTATION_SIZE <size> (integer ≥ 0)

Define the size of the uniform distribution used to mutate binding site tags (in transcription factor units). The mutation is defined as tag + $\mathcal{U}$(-size, +size).

B.6.24 Co-enzyme tag mutation size

CO_ENZYME_TAG_MUTATION_SIZE <size> (integer ≥ 0)

Define the size of the uniform distribution used to mutate co-enzyme tags (in transcription factor units). The mutation is defined as tag + $\mathcal{U}$(-size, +size).

B.6.25 Transcription factor tag mutation size

TRANSCRIPTION_FACTOR_TAG_MUTATION_SIZE <size> (integer ≥ 0)

Define the size of the uniform distribution used to mutate transcription factor tags (in binding site units). The mutation is defined as tag + $\mathcal{U}$(-size, +size).

B.6.26 Basal expression level mutation size

BASAL_EXPRESSION_LEVEL_MUTATION_SIZE <size> (float ≥ 0.0)

Define the standard deviation of the gaussian distribution used to mutate β constant (in promoter units). The mutation is defined as $\beta + \mathcal{N}(0, \text{size})$.

B.7 Genetic regulation network

B.7.1 Genetic regulation network time-steps ratio

GENETIC_REGULATION_NETWORK_TIMESTEP <time-step> (float > 0.0)

Define the number of ODE time-steps used to solve the genetic regulation network per simulation time-step.

B.7.2 Hill function theta parameter

HILL_FUNCTION_THETA <theta> (float $\in [0, 1]$)

Define the parameter θ of the Hill function used to compute the contribution of the regulation on each promoter transcription.

B.7.3 Hill function n parameter

HILL_FUNCTION_N <n> (float ≥ 0.0)

Define the parameter n of the Hill function used to compute the contribution of the regulation on each promoter transcription.

B.7.4 Protein degradation rate

PROTEIN_DEGRADATION_RATE <rate> (float $\in [0, 1]$)

Define the protein degradation rate per genetic regulation ODE time-step.

B.8 Metabolic network

B.8.1 Metabolism time-steps

METABOLISM_TIMESTEP <time-step> (float > 0.0)

Define the number of ODE time-steps used to solve the metabolic network per simulation time-step.

B.8.2 Essential metabolites toxicity threshold

ESSENTIAL_METABOLITES_TOXICITY_THRESHOLD <threshold> (float > 0.0)

Define the maximum cell's toxicity threshold of essential metabolites. If one essential metabolite overreaches this threshold in cell's cytoplasm, the cell dies.

B.8.3 Non-essential metabolites toxicity threshold

NON_ESSENTIAL_METABOLITES_TOXICITY_THRESHOLD <threshold> (float > 0.0)

Define the maximum cell's toxicity threshold of non-essential metabolites. If one non-essential metabolite overreaches this threshold in cell's cytoplasm, the cell dies.

B.8.4 Initial metabolite amount in cells

INITIAL_METABOLITES_AMOUNT_IN_CELLS <initial_amount> (float ≥ 0.0)

Define the initial amount of metabolites found in cells when the simulation is created from scratch.

B.8.5 Maximum reaction size

MAXIMUM_REACTION_SIZE <size> (integer ≥ 0)

Define the maximum jump size of a metabolic reaction in the metabolic space. Considering s and p to be resp. the tags of the substrate and the product of a metabolic reaction (catalyzed by an enzyme), the reaction only occurs if $|s - p| \leq$ size.

B.9 Energy

B.9.1 Energy transcription cost

ENERGY_TRANSCRIPTION_COST <cost> (float ≥ 0)

Define the cost of producing proteins (mainly by transcription). When a enzyme or transcription factor unit is transcribed at a certain rate e, energy cost is $c = e * cost$. For computation reasons, the energy is not coupled to transcription equations (*i.e.*, the reaction speed of the transcription does not depend on energy concentration). If the cost is set to 0.0, the transcription comes with no energy cost. If the energy becomes negative, the cell dies.

B.9.2 Energy degradation cost

ENERGY_DEGRADATION_COST <cost> (float ≥ 0)

Define the cost of degrading proteins. When proteins are degraded at rate d, energy cost is $c = d * cost$. For computation reasons, the energy is not coupled to degradation equations (*i.e.*, the speed of the degradation does not depend on energy concentration). If the cost is set to 0.0, the degradation comes with no energy cost. If the energy becomes negative, the cell dies.

B.9.3 Energy enzymatic cost

ENERGY_ENZYMATIC_COST <cost> (float ≥ 0)

Define the cost or the production of energy when performing metabolic reactions. Metabolic reactions are performed by enzymes needing or producing energy carrier molecules. Let's consider s and p the tags of resp. the substrate and the product of a metabolic reaction catalyzed by enzyme E. If $s < p$, the reaction consumes energy at rate $c = (p - s) * cost$. If $s > p$, the reaction produces energy at rate $c = (s - p) * cost$. For computation reasons, the energy is not coupled to metabolic reaction equations (*i.e.*, the reaction speed does not depend on energy concentration). If the cost is set to 0.0, metabolic reactions come with no energy cost. If the energy becomes negative, the cell dies.

B.9.4 Energy pumping cost

ENERGY_PUMPING_COST <cost> (float ≥ 0)

Define the cost of pumping in or out metabolites. When metabolites are pumped in or out at rate r by a pump, energy cost is $c = r * cost$. For computation reasons, the energy is not coupled to pump equations (*i.e.*, the reaction speed does not depend on energy concentration). If the cost is set to 0.0, the pumping activity comes with no energy cost. If the energy becomes negative, the cell dies.

B.9.5 Energy dissipation rate

ENERGY_DISSIPATION_RATE <rate> (float $\in [0, 1]$)

Define the rate at which a cell looses its energy stock by dissipation.

B.9.6 Energy toxicity threshold

ENERGY_TOXICITY_THRESHOLD <threshold> (float ≥ 0)

Define a maximum threshold to cell's energy. If a cell energy stock overreaches this threshold, the cell dies.

B.9.7 Initial energy amount in cells

INITIAL_ENERGY_AMOUNT_IN_CELLS <amount> (float ≥ 0)

Define the initial energy amount available in cells when the simulation is created from scratch. This parameter allow for random initialization of complex cells needing energy production to survive.

B.10 Cell

B.10.1 Membrane permeability

MEMBRANE_PERMEABILITY <permeability> (float $\in [0, 1]$)

Define the membrane permeability. Metabolites in cell's cytoplasm or in the local environment diffuse through the cell's membrane depending on their concentrations and the permeability.

B.11 Population

B.11.1 Death probability

DEATH_PROBABILITY <probability> (float $\in [0, 1]$)

Define the probability to die at random per simulation time-step. This probability is the same for every cell, and is constant during cell life. This rate is applied in addition to other death events linked to toxicity thresholds.

B.11.2 Migration rate

MIGRATION_RATE <rate> (float $\in [0, 1]$)

If the migration rate is not null, pairs of random cells exchange their location at a defined rate per simulation time-step. Depending on the strength of the random mixing, cell's behavior evolve differently (*e.g.*, to evolve cooperation).

B.11.3 HGT rate

HGT_RATE <rate> (float $\in [0, 1]$)

Define the probability for a genome to receive alien genetic sequences at replication. Genetic sequences are generated at random, and do not come from the simulated population.

B.12 Environment

B.12.1 Environment initialization cycles

ENVIRONMENT_INITIALIZATION_CYCLES <cycles> (integer ≥ 0)

Define the number of initialization loops applied to a newly created environment. Initialization loops are based on environment parameters defined below. For example, If a concentration $c = 0.1$ of metabolite 10 is introduced in the environment at every simulation time-step, and if 5 initialization cycles are requested, the initial concentration will be 0.1×5. This parameter is useful to allow the environment to reach dynamic equilibrium before introducing new cells.

B.12.2 Environment species tags range

ENVIRONMENT_SPECIES_TAG_RANGE <min> <max> (integer > 0; min $\leq$ max)

Define the boundaries of the uniform law used to draw a new metabolite introduced in the environment.

B.12.3 Environment concentrations range

ENVIRONMENT_CONCENTRATION_RANGE <min> <max> (float > 0.0; min $\leq$ max)

Define the boundaries of the uniform law used to draw the concentration of each new metabolite introduced in the environment.

B.12.4 Environment number of species range

ENVIRONMENT_NUMBER_OF_SPECIES_RANGE <min> <max> (integer > 0.0; min $\leq$ max)

Define the boundaries of the uniform law used to draw the number of metabolites introduced in the environment.

B.12.5 Environment interaction scheme

ENVIRONMENT_INTERACTION_SCHEME <choice>
(NO_INTERACTION/INTERACTION)

Define the interaction scheme between the population and the environment.

- NO_INTERACTION: environment concentrations are not modified by cells. Cells grow on ressources with constant concentrations.
- INTERACTION: cells modify their environment by uptaking or releasing food.

B.12.6 Environment renewal scheme

ENVIRONMENT_RENEWAL_SCHEME <choice>
(KEEP_MATTER/CLEAR_MATTER)

Define the renewal scheme of the environment at each new variation.

- CLEAR_MATTER: the environment is rinsed at each variation.
- KEEP_MATTER: the environment is NOT rinsed at each variation.

B.12.7 Environment variation scheme

ENVIRONMENT_VARIATION_SCHEME <choice>
(RANDOM/PERIODIC/CYCLIC)

Define the variation scheme of the environment.

- PERIODIC: variation periodically occurs with frequency INTRODUCTION_RATE
- RANDOM: variation occurs with probability INTRODUCTION_RATE
- CYCLIC: variation occurs at each time-step, but is pondered by a sinus function of period 1/INTRODUCTION_RATE

B.12.8 Environment localization scheme

ENVIRONMENT_LOCALIZATION_SCHEME <choice>
(GLOBAL/RANDOM/SPOT/CENTER)

Define the localization scheme of the environment.

- GLOBAL: the variation affects the whole environment at once (the same concentration(s) of the same new metabolite(s) is introduced everywhere).
- RANDOM: the variation affects the whole environment, but new concentrations and new metabolites are drawn for each location.
- SPOT: the variation affects only one random spot
- CENTER: the variation affects the center of the environment.

B.12.9 Environment metabolic scheme

ENVIRONMENT_VARIATION_SCHEME <choice>
(UNIQUE/MULTIPLE/BOUNDARIES)

Define the metabolic scheme of the environment.

- UNIQUE: only one metabolite is introduced at each new variation.
- MULTIPLE: multiple metabolites introduction is possible.
- BOUNDARIES: restricted multiple scheme: only boundaries of the environment species range are chosen.

B.12.10 Environment introduction rate

ENVIRONMENT_INTRODUCTION_RATE <rate> (float $\in [0, 1]$)

Define the rate at which environmental variations occur (depends on the variation scheme).

B.12.11 Environment diffusion coefficient

ENVIRONMENT_DIFFUSION_COEFFICIENT <coefficient> (float $\in [0, 1]$)

Define the diffusion coefficient in the environment grid. Diffusion is based on a simple algorithm diffusing every metabolites at the same rate in the Moore neighborhood. No ODEs are used here. For this reason, the algorithm becomes unstable for coefficient $>$ 0.1. Thus, if coefficient > 1, diffusion is infinite (well-mixed environment).

B.12.12 Environment degradation rate

ENVIRONMENT_DEGRADATION_RATE <rate> (float $\in [0, 1]$)

Define the rate at which metabolites are degraded. All metabolites degrade at the same rate. Degradation products are implicit, meaning that degraded metabolites simply disappear from the environment.