Getting started with the integrated evolutionary model

Charles Rocabert (charles.rocabert@inria.fr)

October 13, 2015

Contents

1 Short introduction

1.1 The model

The *integrated evolutionary model* includes the features described in the deliverable 2.7 of the EvoEvo project (http://www.evoevo.eu/deliverable-2-7-specifications-of-the-integrated-evolutionary-model/):

(A) Each cell owns a circular single strand genome made of elements of 5 types: elements coding for enzymes catalyzing metabolic reactions (type E), elements coding for transcription factors (type TF), elements coding for binding sites (type BS), elements coding for promoters (type P), and non coding elements (type NC). At replication, the genome undergoes point mutations, but also large chromosomic rearrangements: duplications, large deletions, inversions, and translocations. The various types of mutation can modify existing genes, but also create new genes, delete some existing genes, modify the length of the intergenic regions, modify gene order...

(B) Each cell owns a genetic regulation network, regulating the evolution of transcription factor and enzyme concentrations during the life of the cell,

(C) Each cell also owns a metabolic network, defined by the whole set of enzymes produced by the cell. Some enzymes act as inflowing or outflowing pumps,

(D) A dynamic environment is displayed on a two dimensional toroidal grid, on which cells grow. The environment contains various metabolites at various concentrations, which diffuse and are degraded. Cells interact with the environment by pumping metabolites in or out, or by releasing their content at death. In the current version, the environment only contains metabolites (cells cannot exchange DNA or proteins).

1.2 The source package

The source package is displayed as following:

- The `src` folder contains the source code of the project,
- The `example` folder contains a typical parameters file (default name : `parameters.txt`) used to create a new experiment. Parameters are described in `doc/parameters.html`. See the `README` file for a typical usage of the software,
- The `doc` folder contains documentation on the model,
- The `build` folder contains compilation products (binary executables being in `build/bin` folder),
- The `cmake` folder contains compilation tools,
- Instructions files (including `INSTALL` and `README` files).

1.3 License

The software is distributed under the open source GNU General Public License (see http://www.gnu.org/licenses/gpl-3.0.en.html).

1.4 Download

Release(s) of the `Integrated Evolutionary Model` can be downloaded at http://www.evoevo.eu/softwares/.

1.5 Contact

For any question about the software, do not hesitate to contact us via http://www.evoevo.eu/contact-us/ or charles.rocabert@inria.fr.

2 Installation instructions

2.1 Requirements

- CMake (command line version, please do not use the GUI version)
- zlib
- GSL
- CBLAS
- SFML 2
- TBB
- R
- R packages : ape, RColorBrewer (and dependencies)

2.2 Supported platforms

The program has been successfully tested on Ubuntu 12.04 LTS, Ubunutu 14.04 LTS, OSX 10.9.5 (Maverick) and OSX 10.10.1 (Yosemite).

2.3 Installation instructions

To compile the project, run the following instructions on the command line:

```
$ cd (/path/to/project)/cmake/
```

(and)

$ `sh cmake_debug.sh` to compile the software in `DEBUG` mode

(or)

$ `sh cmake_release.sh` to compile the software in `RELEASE` mode

(or)

$ `sh cmake_no_graphics.sh` to compile the software without graphics

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

3 Typical usage

For a first usage, please take the following steps.

(A) First, place yourself in the examples folder:

$ `cd /path/to/project/examples`

(B) Create a fresh simulation with the parameters file (`parameters.txt`):

$ `../build/bin/create`

Several folders have been created. They mainly contain simulation backups (population, environment, trees, parameters, ...).
Parameters meaning is detailed in `doc/parameters_description.html`.

(C) Alternatively to the `create` executable, use a bootstrap to find a simulation with good initial properties from the parameters file:

$ `../build/bin/bootstrap`

A fresh simulation will be automatically created if a suitable seed is found.

(D) Run the simulation:

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

with `-b` the date of the backup, here 0 (fresh simulation), `-t` the simulation time, here 10000 time steps, `-g` display the graphic window. At any moment during the simulation, one can take a closer look at the evolution of the system by opening `viewer/viewer.html` in an internet browser.

For more information, run any executable with the `-h` option (e.g. `create -h`) and ultimately visit www.evoevo.eu

4 Executables description

4.1 `create` executable

Create a fresh simulation from a parameters file.

Usage:

$ `create -h` or `--help`

or

$ `create [-f param-file] [options]`

Options are:

`-h, --help:` print this help, then exit

`-v, --version:` print the current version, then exit

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

`-rs, --random-seed:` specify if the prng seed should be at random

Be aware that creating an experiment in a folder erases previous files.

4.2 `bootstrap` executable

Find a viable initial population for a given parameters set, via bootstrapping.

Usage:

$ `bootstrap -h` or `--help`

or

$ `bootstrap [-f param-file] [-min minimum-time] [-pop minimum-pop-size] [-t, trials] [options]`

Options are:

`-h, --help:` print this help, then exit

`-v, --version:` print the current version, then exit

`-f, --file:` specify 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

`-g, --graphics:` activate graphic display

4.3 `run` executable

Run a simulation from a backup.

Usage:

$ `run -h` or `--help`

or

$ `run [-b backup-time] [-t simulation-time] [options]`

Options are:

`-h, --help:` print this help, then exit

`-v, --version:` print the current version, then exit

`-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

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

4.4 `experiment` executable

Generate an experiment, containing repetitions on a specified parameters file.

Usage:

$ `experiment -h` or `--help`

or

$ `experiment [-f param-file] [options]`

Options are:

`-h, --help:` print this help, then exit

`-v, --version:` print the current version, then exit

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

`-rep, --repetitions:` specify the number of repetitions

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

`-rs, --random-seed:` specify if the prng seed should be drawn at random for each repetition

4.5 `unitary_tests` executable

Run unitary tests.

Usage:

$ `unitary_tests -h` or `--help`

or

$ `unitary_tests [-f param-file]`

Options are:

`-h, --help:` print this help, then exit

`-v, --version:` print the current version, then exit

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