EVE v2.1

Evolution in Variable Environments

Vadim Mozhayskiy and Ilias Tagkopoulos

April 10, 2012

The program is available for download at
`www.tagkopouloslab.cs.ucdavis.edu/eve.html`

Contents

Overview

`EVE (Evolution in Variable Environments)` simulator employs abstract, multi-scale models of basic sub-cellular phenomena related to expression (transcription, translation, protein modification, degradation, etc.), evolution (mutation, gene duplication, gene deletion, etc.), network regulation and other evolutionary processes such as natural selection. Bacteria are some of the most ubiquitous, simple and fastest evolving life forms in the planet, yet even in their case, evolution is painstakingly difficult to trace in a laboratory setting. EVE simulator is a tool to study and analyze hypotheses regarding microbial evolution dynamics *in silico*. The evolutionary "fossil record" is recorded in each run for the later analysis. This dataset includes all environmental and cellular parameters, cellular (division, death) and evolutionary events (mutations, horizontal gene transfer).

The serial code (EVE v1.0) has been used successfully in the past to generate hypotheses related to regulatory network evolution in nutrient-limited microbial communities [1], and it has been documented elsewhere [2]. The progress of the current code development was published at TeraGrid'11 meeting [5] and at BioViz: IEEE Symposium [6]. The latest version of the code was used to investigate the effect of the Horizontal Gene Transfer in microbial evolution [3] and to address the hypothesis that the rate of evolution can both increase or decrease, depending on the similarity and complexity of the intermediate and final environments [4].

Please contact Prof. Ilias Tagkopoulos (iliast@ucdavis.edu) and Vadim Mozhayskiy (mozhaysk@ucdavis.edu) if you have suggestions, questions, or bugs to report. Have fun!

1 Quick start

The following versions of the EVE are available for download. Each package includes this manual and sample inputs.

Source code (serial and parallel MPI versions)
`www.tagkopouloslab.cs.ucdavis.edu/files/EVE-v2.1.source.tar.gz`

Compiled code for a generic Linux 32bit or 64bit system (serial version)
`www.tagkopouloslab.cs.ucdavis.edu/files/`
`EVE-v2.1.Linux32.serial.tar.gz`

Compiled code for an OS X 32bit or 64bit system (serial version)
`www.tagkopouloslab.cs.ucdavis.edu/files/EVE-v2.1.OSX32.zip`

Compiled code for a generic Windows 32bit or 64bit system (serial version)
`www.tagkopouloslab.cs.ucdavis.edu/files/EVE-v2.1.Windows32.zip`

1.1 Run a sample job

Once you compile EVE (see Section 6 for building instructions) or download a precompiled version for your system, you can run EVE in a serial mode as:

```
./eve _work_dir/ _work_dir/input_serial.in
```

The parallel code compiled with the openMPI library can be executed as following (in this example 16 MPI processes will be started):

```
mpiexec -n 16 ./eve _work_dir/ _work_dir/input.in
```

1.2 Job parameters

Two input files are required for this sample run (included in the package):

(1) `work_dir/input.in` which sets basic parameters of the run (for more details on the file format see Section 3.1), and
(2) `work_dir/input_signals.in`, which described the environment: input signals and the nutrient abundance over one epoch (for more details on the file format see Section 3.2)

1.3 Output

The "fossil record" (evolutional history) of the run is saved in `_work_dir/` directory. The phylogenetic history (history of cell growth and divisions) is printed to the standard output. To redirect the EVE output to a file use for example:

```
./eve _work_dir/ _work_dir/input_serial.in > _work_dir/log
```

For more details on the format of other output files see Section 7.

2 The model description

2.1 Biological model

An evolving population is composed of a fixed number of organisms. Each cell is described by its gene regulatory and biochemical network with abstract molecular representations. The network comprises of a number of "triplets" (three nodes): Gene/mRNA, Protein, and Modified Protein (Figure 1A). The Promoter/Gene/RNA node captures gene regulation and transcription, while the Protein and Modified Protein nodes capture translation and post-translational modification (acetylation, phosphorylation, etc.), respectively. Therefore triplets capture the "central dogma" of molecular biology. Each organism has its own distinct gene regulatory and biochemical network (i.e. a collection of various triplets and weighted regulatory edges) that can be depicted as a directed weighted graph (Figure 1B).

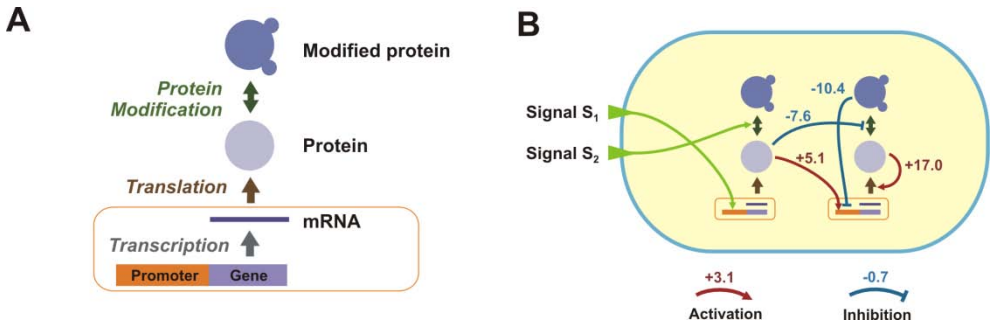

Figure 1. (A) "Triplet" – mRNA, protein, and modified protein model the central dogma of molecular biology; (B) gene regulatory network of the cell consists of "triplets"; nodes activate or inhibit each other, the network mutates over time to adapt to the external signals from the environment.

The probability of molecule creation at each node and at each time step is a function of the regulatory effect of other nodes (activation or inhibitions) on that specific node, and the availability of substrate. The molecule production probability is modeled by a two-level sigmoid function that captures a threshold and saturation effects for any given regulator and for the expression of any given node:

$$G_i = basal_i + (1 - basal_i) \cdot \tanh\left(\frac{\sum_{j=1}^{n} \left(w_{ij} \cdot F\left(v_j, \tilde{m}_{ij}, \tilde{s}_{ij} \right) \right) - m_i}{s_i} \right)$$

where the sigmoid function F_{ij} describes the regulatory effect of node j on node i:

$$F\left(v_j, \tilde{m}_{ij}, \tilde{s}_{ij} \right) = \frac{1}{2} \cdot \left[1 + \tanh\left(\frac{v_j - \tilde{m}_{ij}}{\tilde{s}_{ij}} \right) \right]$$

where w_{ij} is the regulatory matrix element (i.e. the strength and direction that exerts node j to node i), v_j is the value of node j, m_i and s_i the midpoint and slope of the target-specific sigmoid function, $\tilde{m}_{ij}$ and $\tilde{s}_{ij}$ the midpoint and slope of the regulator specific sigmoid function, n is number of regulating nodes, $basal_i$ is the basal expression parameter.

In addition to its regulatory network, each organism has a unique metabolic pathway which, when expressed, can metabolize available resources in the environment.

Mutational events (e.g. transcription rate changes, node duplications, node deletions, etc.) occur stochastically at any time point and on any node or edge, thus changing its internal network and potentially its phenotype, which in this context is synonymous to the regulatory and metabolic pathway expression. The production and destruction of any molecule has an energy cost, as does the maintenance of molecular species (nodes). Organisms cannot directly sense the presence of resources; however they can potentially infer their future presence, if they are able to process information from various environmental signals through biochemical and regulatory interactions. Once an organism reaches a certain energy level, it undergoes division, increasing its genotype representation in the population, while its progeny replaces an existing organism so that the fixed size of the population is preserved (probability of an organism being replaced is inversely proportional to its energy level in our model).

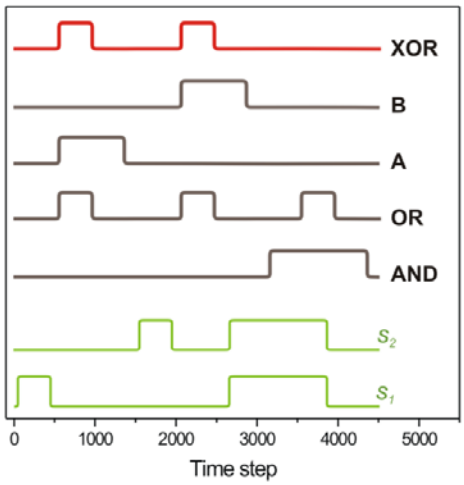

Figure 2. Environmental signals (green) and nutrient abundance for three sample environments (AND, OR, and XOR). Nutrient availability in these examples is a delayed function of two signals. However any arbitrary shape for signals and nutrients can be supplied as an input. One epoch is shown for each environment, which by default consists of 4,500 time steps.

Default environments consist of two signals, s1 and s2, which carry information regarding the presence of nutrients in the environment over one epoch. The number of environmental signals can be increased as needed. The nutrient availability can be any predefined function of the environment. Several

standard logical environments are shown in Figure 2. A delay in the signal/nutrient correlation is usually introduced to further increase the evolutionary complexity of the environment, as organisms have to account for it through the topology and dynamics of the respective underlying networks.

The fitness level of each organism is evaluated as the Pearson correlation between the nutrient abundance and the response protein expression level over a predefined interval of time, which we call an "epoch" (4,500 time steps by default). We stress that this similarity measure is used for visualization purposes as a proxy to each organism's fitness, and at no point participates or interferes with the selection or evolutionary trajectory of cells during the simulation. High correlation between nutrients and response protein concentration implies an efficient underlying mechanism to metabolize nutrients, as activation of this costly pathway takes place only when it confers an advantage to the organism.

The model also incorporates horizontal gene transfer (HGT) in addition to the other cellular (transcription, translation, modification, growth, death, etc.) and evolutionary (mutation and natural selection) processes. In our model a gene and its products are represented by triplets, and therefore HGT can be treated as inter-cellular transfer of one or more triplets. For every HGT event a random subset of triplets (sub-network) is copied from the donor cell and inserted into the regulatory network of the recipient cell. Original regulation of the metabolic pathway RP0 and triplet T0 by the transferred sub-network is preserved. In our model triplets with preserved regulatory network are transferred from one organism to another, and the fragment size for an established HGT event is chosen using a probability density function as a normalized sigmoid function:

$$P(n) = \frac{1 - \tanh\left(\frac{n-m}{s}\right)}{m \cdot \left(2 + \ln(e^{-\frac{2m}{s}} + 1)\right)}$$

where n is the fragment size in triplets, m and s are the middle point and slope of the probability density function, respectively; the denominator is a normalization coefficient.

2.2 Parallel framework

The code is based on a stochastic simulation algorithm where mutational events occur randomly based on predefined probability distributions. A simulation environment consists of independent organisms each described by a directed weighted graph. All organisms interact with each other through the common environment by consuming a limited supply of nutrients, sensing environmental proprieties and secreting chemicals. Organisms stochastically evolve and their gene regulatory and biochemical network changes both in size and topology.

A general structure of the serial version of the code is as follows:

```
Input: population and environment
For (each Epoch)
 For (each Time_step)
  For (each Cell)
    Update_cell()
    if ( Energy == 0 )
      Replace_cell() // new random cell
    if ( Energy > Division_threshold )
      // dividing cell replaces the weakest
      Cell_divide( this-->weakest)
  Output: Phylogenetic information
 Output: Cell fossil history and statistics
```

We use an MPI model to distribute a population of cells to a set MPI processes (Figure 3). At every time step organisms mutate with predefined probabilities and node values are updated using the stochastic expression model described above. Cells which exhausted their energy are removed from the population and replaced with new random cells (to start from a new point on the fitness landscape). Cells which reach energy above the division threshold are duplicated, and daughter cells replace cell with low energies to maintain a constant population size.

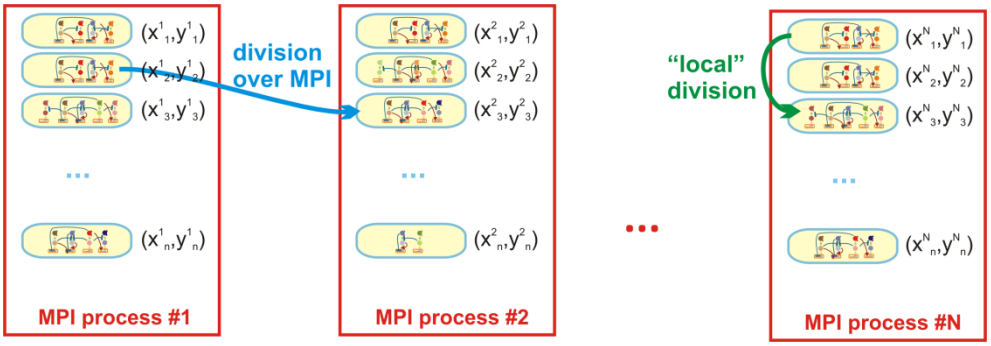

Figure 3. Parallel implementation of simulation framework. Diagram shows data distribution between MPI processes: cell population is divided to run on N MPI processes with n cells per process (total population size n·N).

3 Input format

3.1 Input file

A sample input file can be found in the samples directory: `_work_dir/input.in` and it includes the following parameters:

`Number of epochs`: the overall length of the experiment;

`Time steps per epoch`: number of time steps in each epoch, cell is updated every time step;

`Number of cells per process`: in a serial run this is the population size. In a parallel MPI run it is number of cells per MPI process, and the total population size is equal to "Number of cells per process"·"number of MPI processes";

`Number of signals`: number of signals in the environment;

`Initial energy of each cell`: in energy units, default value 800,000;

`Division threshold`: once cell reaches this threshold it divides, default 1,600,000 (in energy units);

`Death threshold`: once cell's energy is below this threshold, cell is removed from the population;

`Maintenance cost per response protein`: cost (in energy units per time step) to maintain each response protein, default 10;

`Maintenance cost per network node`: cost (in energy units per time step) to maintain each node response protein default 1;

`Metabolic gain per response protein`: gain by each expressed response protein (in energy units per time step), default 50;

`Creation`: probability of triplet duplication (per triplet, per time step);

`Destruction`: probability of triplet deletion (per triplet, per time step);

`Mild mutation`: probability of mild network mutation (per triplet, per time step);

`Strong mutation`: probability of strong network mutation (per triplet, per time step);

`Energy synchronization frequency`: (units time steps) frequency of population synchronization;

`Save cell info frequency`: (units epochs) population snapshots are saved with this frequency;

`Import cell`: possible values: 0 – generate a random initial population; 1 – create a clonal initial population from a single cell; 2 – load previously saved population;

2D grid

`Grid points along X&Y`: number of grid points for a square 2D population;

`Cell diffusion probability`: per time step;

`Horizontal gene transfer probability`: per cell, per time step;

`Horizontal gene transfer network incorporation probability`: per edge;

`Horizontal gene transfer distance middle`: midpoint of the distance function;

`Horizontal gene transfer distance slope`: slope of the distance function;

`Horizontal gene transfer size middle`: midpoint of the fragment size function;

`Horizontal gene transfer size slope`: slope of the fragment size function;

`Keep metabolic protein regulation`: 1 if regulation is maintained for a transferred fragment, 0 otherwise;

Additionally three sample inputs `input_hm.in`, `input_mm.in`, and `input_lm.in` are provided in `_work_dir/` directory for runs with high, medium, and low mutation rates, respectively.

3.2 Input environment file

A sample environmental input file can be found in the samples directory: `_work_dir/input_signals.in` and the format is as follows:

```
01    Comment line
02    Comment line
03    <one line per time step, each line k+1 space separated
      floats with values of k input signals and nutrient
      abundance for each time step. Number of time steps is
      equal to the length of the epoch defined in input.in>
```

Additionally five sample input environments: `input_signals_AND.in`, `input_signals_NAND.in`, `input_signals_NOR.in`, `input_signals_OR.in`, and `input_signals_XOR.in` are provided in `_work_dir/` directory for runs in AND, NAND, NOR, OR, and XOR environments, respectively.

3.3 Input population format

As an option, the simulation can be started not from a random population, but from a previously evolved set of cells. Two options are available:

(1) Start from a clonal population, i.e. all organisms in the start population are identical and loaded from a file containing a single cell.
(2) Start from a previously saved population.

A sample file with a single saved organism can be found in the samples directory: `_work_dir/input_cell.in`. The format is as following:

```
01      Comment line
02      Cell: <unique ID, string> <label, string>
03      Number of nodes: <N number of nodes, integer>
04      Fitness: <initial fitness, float>
05      Energy: <initial energy, float>
06      Mutational bias: <five floats: probability of (1)
        creation, (2) destruction, (3) mild mutation, (4)
        strong mutation, and (5) evolvability>
07      Connection to any signal (input sensor protein):
08      <N space separated integers, values can be:
        0-node is not connected to any signal,
        n-node is connected to n-th signal.
        Each node can be connected to only one of the signals>
09      Slope:
10      <N float numbers, slope s_i for each node>
11      Middle point:
12      <N float numbers, midpoint m_i for each node>
13      Basal:
14      <N float numbers, basal_i for each node>
15      Degradation:
16      <N float numbers, degradation rate for each node>
17      Weight matrix
        <N lines, each line N floats for w_ij weights>
17+N+1  Regslope
        <N lines, each line N floats for s̃_ij values>
17+2N+1 Regmiddle
        <N lines, each line N floats with m̃_ij values>
```

Where N is number of nodes in the organism, w_{ij} is the regulatory matrix element (i.e. the strength and direction that exerts node j to node i), m_i and s_i the midpoint and slope of the target-specific sigmoid function, $\tilde{m}_{ij}$ and $\tilde{s}_{ij}$ the midpoint and slope of the regulator specific sigmoid function, $basal_i$ is the basal expression parameter (see Section 2.1).

Additional samples files with a single cells evolved under various environments can be found in the samples directory: `_work_dir/input_cell_XXX_Nm.in`, where `XXX` is `XOR` or `AND` environment, and `Nm` is `hm`, `mm`, or `lm` for high, medium, and low mutation rates, respectively.

4 Output format

Phylogenetic history of divisions and organism replacements in the population is logged from all processes into the single standard output. For a serial run on a Linux/Unix based system the standard output can be redirected into a file, for example:

```
./eve _work_dir/ _work_dir/input_serial.in > _work_dir/log
```

Each event is described on a separate line. Cells participating in the event are identified:

```
epoch_number.timestep: process#[local#].localID (X,Y)
```

where X and Y are cell coordinated on a 2D grid, process# is the MPI rank of the process with that cell, local # and localID are the cell number and cell ID within that MPI process.

For example the cell lysis and replacement with a new random cell is logged as:

```
5.3578: 2[0].0 (1,4) -|- 2[0].2 (1,4) Label=753
```

which means that at epoch 5, at time step 3578 on MPI process #2 cell #0 with ID=0 at coordinate (1,4) was replaced with a new random cell, at the same process, at the same coordinate, with new ID=2

Similarly cell division event is logged as:

```
13.3462: 1[0].5 (2,2) --> 2[1].7 (2,3) {replaced cell: 2[1].6 (2,2)}
```

which means that at epoch 13, at time step 3462 on MPI process #1 cell #0 with ID=5 at coordinate (2,2) divided, and the offspring was placed at the coordinate (2,3) on MPI process #2 with new ID=7. The offspring replaced poorly fit cell with ID=6 at MPI process #2.

Population snapshots are saved into **./_work_dir/cell_info__rank#.data**, one file per MPI process (number #) in the cell format described in Section 3.3.

File **./_work_dir/output_statistics.data** contains global population statistics for every epoch (average and maximum fitness, cell size, and growth rates).

File **./_work_dir/cell_density.data** contains snapshots of 2D population densities.

Files **./_work_dir/mutation_history__rank#.data** contains mutation history for every epoch (one file per MPI process number #).

5 Running the code

In addition to the minimal command mentioned in Section 1.1:

```
mpiexec -n 16 ./eve _work_dir/ _work_dir/input.in
```

The code can be executed with additional (optional) arguments:

```
mpiexec -n 16 ./eve _work_dir/ {_work_dir/input.in {experiment#
{seed}}}
```

If experiment # and seed (long unsigned integer) are provided, they are used to initialize the random number generator, and that can be used to reproduce previous experiments. Note that the input file is also an optional argument, if none provided all parameters assume default values.

6 Compiling the code

To compile the parallel version of code you need to have one of the MPI libraries installed (such as openMPI or any other supporting MPI 1.0 specification). Once installed, use the `make` command inside the EVE directory to create an executable ./eve:

```
make eve
```

File `Makefile` contains several machine dependant variables that need to be adjusted according to the local setup:

(1) **`CC=mpiCC`** Defines `C++` compiler; use `CC=mpiCC` (or any other MPI library wrapper) to compile a parallel version of the code, or `CC=g++` to compile a serial executable;

(2) **`IF_MPI=1`** Set to 0 if serial version of the code is desired (regardless of the compiler used);

7 Common problems and FAQ

Here are we will address common problems. If you have suggestions, questions, or bugs to report, please contact Prof. Ilias Tagkopoulos (iliast@ucdavis.edu) and Vadim Mozhayskiy (mozhaysk@ucdavis.edu).

1. **How to run EVE under MS Windows operating system?**

   EVE is compiled for a command line usage. In all examples throughout this manual replace `./eve` with `eve.exe` and run from a Windows command line. To open a command line interface type `cmd` in the `Run` field of the `Start` menu and navigate to the folder containing `eve.exe`.

2. **How to run EVE under OS X operating system?**

   EVE is compiled for a command line usage. To open a command line interface select `Terminal` under `Utilities` in `Applications` folder and navigate to the folder containing `eve`.

Bibliography

1. Tagkopoulos I, Liu YC, Tavazoie S: Predictive behavior within microbial genetic networks. *Science* 2008, 320(5881):1313-1317.
2. Tagkopoulos I: Emergence of Predictive Capacity within Microbial Genetic Networks, PhD Thesis. Princeton University; 2008.
3. Mozhayskiy V, Tagkopoulos I: In Silico Evolution of Multi-Scale Microbial Systems in the Presence of Mobile Genetic Elements and Horizontal Gene Transfer. *Lect Notes in Comput Sc* 2011, 6674:262.
4. Mozhayskiy V, Tagkopoulos I: Guided evolution of in silico microbial populations in complex environments accelerates evolutionary rates through a step-wise adaptation. *BMC Bioinformatics (Accepted)* 2012.
5. Mozhayskiy V, Miller R, Ma K-L, Tagkopoulos I: A Scalable Multi-scale Framework for Parallel Simulation and Visualization of Microbial Evolution. In: *TeraGrid'11; Salt Lake Sity, UT* 2011.
6. Miller R, Mozhayskiy V, Tagkopoulos I, Ma K-L: EVEVis: A Multi-Scale Visualization System for Dense Evolutionary Data BioVis. In: *BioViz: IEEE Symposioum on Biological Data Visualization; Providence, RI* 2011.