Contents

SELVa: overview

The Simulator of Evolution with Landscape Variation (SELVa) is a simulator of sequence evolution that allows the fitness landscape to vary according to user-specified rules.

Abstract

Evolution of an organism is governed by a fitness landscape, which relates genotype to phenotype. The organism evolves towards optima in the fitness landscape. At the same time, fitness landscapes themselves are subject to change, either due to changes in the environment, to host immune response in case of pathogens, or, if we are considering the fitness landscape for a single position in the genome, due to changes in other positions (epistasis). Computer simulation methods have established themselves as invaluable tools in the study of molecular evolution, enabling scientists to replay evolution with different scenarios and to test models and parameters. SELVa, the Simulator of Evolution with Landscape Variation, is a novel evolutionary simulator specifically geared towards

exploring the effects of landscape change on molecular sequence evolution. SELVa has a variety of options for specifying the rules of landscape change, allowing the user to tailor the simulation to his or her needs and to explore various settings.

What SELVa does

SELVa simulates point mutations (no indels yet) along a user-provided phylogenetic tree (given in a separate file in the Newick format). These substitutions are governed by a fitness landscape that is specified by a vector giving the scaled fitness of each allele. The fitness landscape can change discretely according to rules set by the user in the config file. The config files additionally specifies everything else about the simulation, including the sequence alphabet, the length of the sequence, the number of processors used, whether to print the intermediate fitness values, etc. Detailed information on the config file options is given below.

How to run SELVa

1. SELVa is distributed as a Java jar file.
2. Simulation settings are given in a config file.
3. SELVa runs the simulation along the user-provided phylogenetic tree. The tree should be wprovided in a Newick-format file, and the config file should contain the path to this file (as the value of the TREE_FILE parameter).
4. To run the simulator, open the command line prompt, go to the directory where the above-mentioned files are and type
   % java -jar Selva.jar config.txt
   Currently, the jar is built using Java 1.8, so you have to have the corresponding JDK on your system.

Technical overview

SELVa is an event-driven simulator where at any point in time along any branch of the rooted phylogenetic tree, one of the following events can occur: a point mutation in the sequence (governed by the current fitness landscape), or an instantaneous change in the fitness landscape. The mutational process follows the description in Chapter 12.5.4 of *Molecular Evolution: A Statistical Approach (*Yang 2014): briefly, the mutation events are modeled as a Markov chain with inter-event times described by an exponential distribution with the mean parameter derived from the current fitness landscape. Once a mutation event occurs, the allele that is transitioned to is chosen probabilistically according to the stationary distribution of the current scaled fitness vector. Meanwhile, the landscape-change events can occur either deterministically at equal time intervals, or stochastically (in which case the landscape change becomes just another event of the simulation, along with point mutations), and the value of the new landscape is chosen according to user-specified rules. The various options for changing landscapes are described below.

Some things to keep in mind are that the branch lengths of the Newick tree are interpreted as evolution time for the purposes of the simulation. Also, the probability of a sequence position changing to a character depends only on its “fitness” according to the current landscape; it does not depend on the character that is mutated away from. This limitation may be addressed in future versions of the simulator.

The config file

The config file is a whitespace-delimited text file containing parameter-value pairs. All lines preceeded with a hash sign ("#") are ignored as comments.

The config file options are described in detail in the following sections.

Global and landscape-specific settings

SELVa permits the user to evolve different landscapes in parallel. There are three possibilities regarding the extent to which landscape data is shared among positions of a sequence.

1. All positions in a sequence are subject to the same landscape, which can change with time.

2. Multiple landscapes are run simultaneously and independently, but under the same rules (e.g., the times of landscape change are sampled independently from the same distribution). The lengths of the sequences goverened by each landscape (and the initial ancestral sequence, if supplied) are also the same. This is the preferred setting for running multiple simulations governed by the same set of rules.

3. There are multiple landscapes, each governing a separate partition of the sequence. The landscapes do not need to share any parameters. This is the preferred setting for having different parts of the sequence evolve under different rules.

There are threfore "global" parameters that apply to all landscapes in a simulation, and landscape-specific parameters that can vary between the landscapes.   In the third scenario, the global parameters must come first and include the parameter `NUM_LANDSCAPES`, which informs SELVa of the number of "unrelated" landscapes to simulate. The global parameters must be followed by the line `[END GLOBAL PARAMETERS]`

Each landscape's parameters are then preceded by the line `[LANDSCAPE 1]`… `[LANDSCAPE 2]`… etc. Below, we give an example of a config file. In this config, there are two landscapes, the first governs the first 4 characters of the sequence, the second governs the next 2. The two landscape settings differ in the value of the `CHANGE_BRANCH_AND_TIME_FILE`, but they can differ in any of the fields.

In the first and second scenarios (either there is only one landscape or the rules are the same for all landscapes), it is possible to omit the tags in square brackets `[]` and give just the one set of landscape rules.

Example config file for two separate landscapes:

```
ALPHABET  ACGT
TREE_FILE simple.tre
ROOT_SEQUENCE_FILE rootseq.txt
PRINT_LANDSCAPE_INFO true
CONSTANT_RATE true
SCALE_LANDSCAPE_CHANGE_TO_SUBSTITUTION_RATE true
NUM_LANDSCAPES 2
[END GLOBAL PARAMETERS]
```

```
[LANDSCAPE 1]
LENGTH 4
INITIAL_FITNESS file
FITNESS_FILE fitness_20aa_1_0.1.txt
NEW_FITNESS_RULE shuffle
LANDSCAPE_CHANGE_TIMING specified_branch_and_time
CHANGE_BRANCH_AND_TIME_FILE change_branch_time1.txt
[LANDSCAPE 2]
LENGTH 2
INITIAL_FITNESS file
FITNESS_FILE fitness_20aa_1_0.1.txt
NEW_FITNESS_RULE shuffle
LANDSCAPE_CHANGE_TIMING specified_branch_and_time
CHANGE_BRANCH_AND_TIME_FILE change_branch_time2.txt
```

Global rules governing the structure of the simulation

The tree file

The user must specify the phylogenetic tree in the Newick file format and give its name (and path) as the value of the `TREE_FILE` parameter in the config file.

Example:

```
TREE_FILE /path/to/tree/file.tre
```

The sequence alphabet

The user must also provide the allele alphabet as the `ALPHABET` string in the config file.

Example:

```
ALPHABET ARNDCEQGHILKMFPSTWYV
```

The root sequence (optional)

The user also has the option of providing a root sequence in a separate file. In that case, the file name should be provided in the `ROOT_SEQUENCE_FILE` parameter (if the parameter is not set, the root sequence is generated from the stationary distribution, as stated above). The file should contain either just the sequence, or the sequence with a FASTA header (which will be ignored). The sequence must be at least as long as LENGTH (any characters exceeding LENGTH) will be ignored, and contain only characters from ALPHABET.

Example:

config.txt:

```
ALPHABET  ACGT
ROOT_SEQUENCE_FILE rootseq.txt
```

roootseq.txt:

```
>test
ACTGCT
```

The mutation rate matrix (optional)

The user has the option of providing the mutation rate matrix in a separate file. The file name should be given as the `MUTATION_RATE_FILE` parameter. If the parameter is not set, all mutation rates are assumed to be one.

The mutation rate matrix should be an |ALPHABET|x|ALPHABET| matrix. The diagonal entries will be ignored. It is up to the user to make sure that the matrix conforms to the desired model.

Example:

config.txt:

```
ALPHABET ACGT
MUTATION_RATE_MATRIX_FILE mutation_rates.txt
```

mutation_rate.txt:

```
0.5 1 1 1
0.5 2 1 1
0.5 1 2 1
0.5 1 1 2
```

Parallelization/multiple landscapes

Multiple instances mode

The user can have multiple *instances* of the same landscape rules/distribution run in parallel by settting the value of the `NUM_INSTANCES` parameter. Other than sharing input and output files and the probability distributions (if any) that are used to generate fitness landscapes or landscape change times, these simulations are independent.

Multiple landscapes mode

Alternatively, the user may have landscapes that are not related in any way. Then, the number of such landscapes must be stated as the value of the `NUM_LANDSCAPES` parameter.

`NUM_LANDSCAPES` and `NUM_INSTANCES` are mutually exclusive.

Parallelization

If more than one processor is available, the instances may be be divided among multiple threads by setting the `NUM_THREADS` parameter to a desired number. Note: a single run cannot be multithreaded, so `NUM_THREADS` should be `<= NUM_INSTANCES` or `NUM_LANDSCAPES`, whichever is used.

Example:

```
NUM_INSTANCES 10
NUM_THREADS 2
```

In this case, 10 parallel simulation instances are run on two threads.

Advanced global parameters

Some global parameters which are not likely to be useful to the majority of users are described in the section Advanced global parameters.

Landscape-specific parameters

Each landscape specification must be preceded with a line `[LANDSCAPE N]`, whre N is a number. If only one landscape specification is used, this line can be skipped.

Length of the modeled sequence

The LENGTH parameter defines the length of the sequence whose evolution will be simulated. All elements of the sequence will evolve independently on the same fitness landscape. Under default behavior, an ancestral sequence of this length will be generated at random by sampling from the stationary distribution of the initial fitness vector (described in further detail below).

Example:

```
LENGTH 1000
```

If `ROOT_SEQUENCE_FILE` is set, and `NUM_LANDSCAPES` > 1, then the ROOT_SEQUENCE is partitioned in such a way that each landscape applies to its LENGTH of the sequence (subsequences are allocated to landscapes in the order of appearance of the landscape specifications).

Example:

rootseq.txt:

```
AACCGGTT
```

config.txt:

```
…
[LANDSCAPE 1]
LENGTH 6
…
[LANDSCAPE 2]
LENGTH 2
```

Then, the first landscape applies to the root subsequence AACCGG, and the second, to the root subsequence TT.

Rules governing the fitness landscape

The fitness landscape is specified as a vector of the same length as the ALPHABET string, such that the value of *i*th element of the fitness vector corresponds to the (population size-) scaled additive fitness of the *i*th character in the ALPHABET string. The fitness vector is used to generate both the stationary allele distribution (from which the ancestral sequence is sampled) and the transition matrix (the mathematical details are given in the paper).

Specifying the initial fitness landscape

The initial fitness landscape can be read from a file or generated by sampling the scaled fitness of each allele from one of the supported distributions (the parameter governing the distribution is set by the user – see below).

Providing the initial fitness vector in a file

In this case, the config should contain the lines:

```
INITIAL_FITNESS file
```

```
FITNESS_FILE /path/to/fitness/file.txt
```

Where `file.txt`, whose path on your system is given by `path/to/fitness/file.txt` should contain a list of space-delimited numbers that correspond to the scaled fitnesses of each allele given in the `ALPHABET` parameters in the order in which they are listed in the `ALPHABET` string.

Example:

```
ALPHABET ARNDCEQGHILKMFPSTWYV
INITIAL_FITNESS file
FITNESS_FILE fitness_20aa_1_0.1.txt
```

and the file `fitness_20aa_1_0.1.txt` resides in the working directory and contains a single line:

```
1.0 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

Then, the scaled additive fitness of the A allele is 1.0, and for all the other alleles, it is 0.1.

Generating the fitness vector at random by sampling from a probability distribution

Another option is to draw the scaled fitness of each allele independently from one of the supported distributions. The currently supported distributions are the **lognormal** and the **gamma** distributions.

The **lognormal** disbribution is parameterized by a single parameter σ that is set as the value of the `SIGMA` parameter in the config file. For simplicity, μ is fixed at `0`.

An example: the config file may contain the lines:

```
INITIAL_FITNESS lognorm
SIGMA 1.0
```

In this case, the scaled (additive) fitness of each allele in the ALPHABET is drawn from the lognormal distribution with $\mu = 0$ and $\sigma = 1.0$.

The **gamma** distribution is parameterized by two parameters `GAMMA_ALPHA` and `GAMMA_BETA`.

An example:

```
INITIAL_FITNESS gamma
GAMMA_ALPHA 1.0
GAMMA_BETA 0.5
```

In this case, the scaled fitness of each allele in the ALPHABET is drawn at random from the gamma distribution with α = `1.0` and β = `0.5`.

Rules governing changes in the fitness landscape.

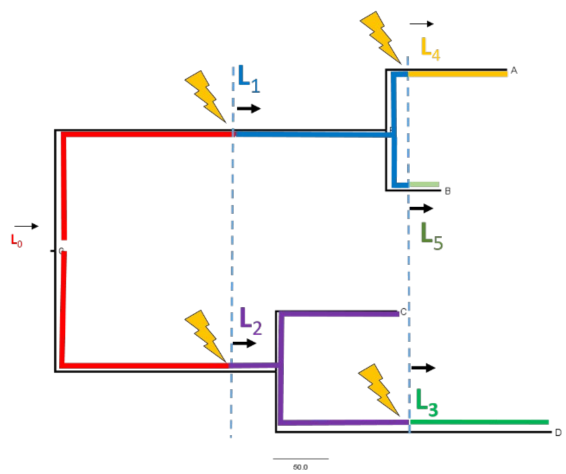
As the ability to vary the fitness landscape is the core feature of SELVa, the simulator provides a number of options for when the fitness landscape changes, what it changes to, and whether the changes are shared by all branches that are in existence at a given time, or each branch has its own landscape history.

Rules governing when the landscape change occurs

The landscape change can take place at regular time intervals, stochastically as a Poisson process, or at user-specified positions on the tree.

Changing the landscape at regular time intervals

This option is illustrated by the figure below where "lightning bolts" represent landscape changes and landscapes are denoted as L_0, L_1, … and represented with colors.



The interval of time between consecutive landscape changes can be given either directly as the length of the interval, or indirectly by specifying the desired number of landscape changes. In the latter case, the interval length will be computed based on the longest path from the root to a leaf (tip node) in the tree. The choice between these two options is governed by the `LANDSCAPE_CHANGE_TIMING` parameter, and the interval length or number of intervals is given by the overloaded `LANDSCAPE_CHANGE_PARAMETER`, whose interpretation depends on the value of `LANDSCAPE_CHANGE_TIMING`.

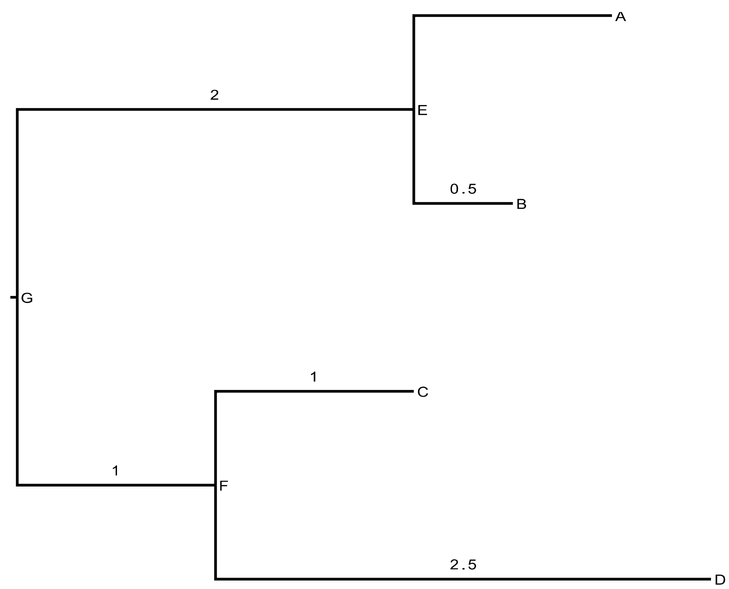
- To specify the interval length directly, include in the config file the lines:
  `LANDSCAPE_CHANGE_TIMING fixed_interval_length`
  `LANDSCAPE_CHANGE_PARAMETER` *`interval_length`*
  where interval_length is a nonnegative real number.
- To specify the desired number of landscape changes (along the longest path in the tree), include in the config file the lines:
  `LANDSCAPE_CHANGE_TIMING fixed_num_changes`
  `LANDSCAPE_CHANGE_PARAMETER` *`num_changes`*
  where `num_changes` is a non-negative integer.

As an example, consider the tree below where the longest path from root to leaf (node D) is 3.5, and the desired landscape change times are shown by vertical red lines (spaced 0.5 time units apart). We can specify these change times either by setting the interval length:

```
LANDSCAPE_CHANGE_TIMING fixed_interval_length
LANDSCAPE_CHANGE_PARAMETER 0.5
```

or by setting the number of landscape changes (6):

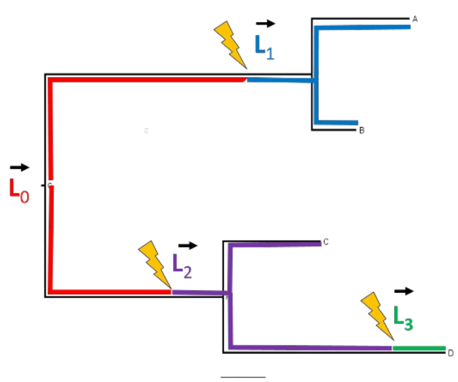
```
LANDSCAPE_CHANGE_TIMING fixed_num_changes
LANDSCAPE_CHANGE_PARAMETER 6
```



Changing the landscape at stochastic intervals

In this case, the landscape change timing is a Poisson process whose mean rate parameter is specified as the `LANDSCAPE_CHANGE_PARAMETER`, which in this case is interpreted as the rate of the Poisson process.

This option is illustrated by the following figure.



To use this option, the config file should contain the lines:

```
LANDSCAPE_CHANGE_TIMING stochastic
LANDSCAPE_CHANGE_PARAMETER lambda
```

where *lambda* is a real number giving the mean rate λ of the landscape change process.

No landscape changes

If landscape change should *never* occur (useful for testing the baseline model or using SELVa as a conventional molecular evolution simulator), these parameters can be set appropriately (e.g., set the `LANDSCAPE_CHANGE_TIMING` to `stochastic` or `fixed_num_changes` and `LANDSCAPE_CHANGE_PARAMETER` to 0.

A note regarding numerical issues. The small loss of precision inherent in using floating-point numbers may lead to artefacts in the landscape `timing` calculation. One can occur when branch lengths are multiples of the length of the (fixed) inter-change interval. In this case, it is possible for the landscape change to take place sometimes *just before* the branching event (in which case both daughter branches start with the same landscape), and sometimes *just after* the branching event (in which case each daughter branch starts with its own landscape).

Changing the landscape at user-specified positions

To specify the specific positions in the tree where the landscape change should occur, set `LANDSCAPE_CHANGE_TIMING` to `specified_branch_and_time`. The positions of the landscape changes should be specified in a text file in the format

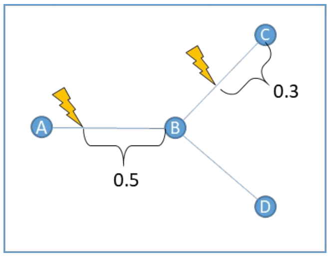
```
branch1_endpoint  time_till_branch1_endpoint
branch2_endpoint  time_till_branch2_endpoint
branch3_endpoint  time_till_branch3_endpoint
```

Where `branch_endpoint` is the exact name of the tree node completing the branch on which the change is to occur (in other words, the node that occurs immediately after the desired change) and `time_till_branch_endpoint` is the remaining time on that branch after the desired change.

The name of the file should be given in the config file as the value of `CHANGE_BRANCH_AND_TIME_FILE`

This option is incompatible with shared landscapes (see below).

As an example, consider the following tree where the “lightning bolts” mark the desired landscape change times:



Then we might have a file named change `coordinates.txt` which would contain the lines:

```
B 0.5
C 0.3
```

And the config file would have the lines:

```
LANDSCAPE_CHANGE_TIMING specified_branch_and_time
CHANGE_BRANCH_AND_TIME_FILE coordinates.txt
```

If this option is chosen, the user can additionally specify the new scaled fitness vector next to the branch and time in the `CHANGE_BRANCH_AND_TIME_FILE`. In that case, the vector is given on the same line, following the branch and time, for example:

```
C 0.5 10 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
0.1 0.1 0.1
```

Additionally, the `NEW_FITNESS_RULE` parameter should be set to `user_set`

Calculating the new fitness vector

There are several options for how the new (scaled) fitness vector is obtained when the landscape changes. Currently, these are: randomly permuting the current fitness vector (guaranteeing that the old and the new vector are different); randomly sampling another vector from the same distribution as the initial fitness vector; linearly increasing or decreasing the fitness of the current allele (the latter option only works under a subset of landscape change settings).

New fitness vector as a reshuffled (randomly permuted) old fitness vector

In this case, the config file should contain the line

```
NEW_FITNESS_RULE shuffle
```

The new vector may turn out to be the same as the old one.

New fitness vector generated by drawing from the same distribution as the initial vector.

The config file should then contain the line:

```
NEW_FITNESS_RULE iid
```

This option only works if `INITIAL_FITNESS` has been set to `lognorm` *OR* `gamma` with `DIST_PARAM` set appropriately.

New fitness vector generated from the old one by increasing or decreasing the fitness of the current allele.

This option is designed to model situations where the fitness of the current allele changes, e.g., due to host immune response (the current allele may become less fit with time as the host immunity learns to fight it).

NOTE: This option only works if the landscape changes at regular intervals (`LANDSCAPE_CHANGE_TIMING` is NOT `stochastic`), `LENGTH` is set to `1`, and `SHARED_LANDSCAPE` is set to `false`. (The latter two restrictions are due to the need to have a unique current allele to use this option).

In this case, the fitness values of all alleles but the current one remain unchanged. If the old fitness vector is $<f_1^{old},\ldots f_n^{old}>$ and the current allele has index i, then the new fitness vector $<f_1^{new},\ldots f_n^{new}>$ has values:

> $f_j^{new} = f_j^{old}$ for $i \neq j$
> $f_i^{new} = f_i^{old} + \mathrm{k} \cdot \Delta t$, where k = `AGE_DEPENDENCE_COEFFICIENT` and Δt is the length of the interval between consecutive landscape changes (they need to be regularly spaced and determined by `LANDSCAPE_CHANGE_TIMING` and `LANDSCAPE_CHANGE_PARAMETER` described above). If k is negative, the current allele's fitness decreases with time.

The config file should contain the lines:

```
NEW_FITNESS_RULE current_allele_dependent
AGE_DEPENDENCE_COEFFICIENT k
```

where `k` is the real-valued change in fitness per time unit

Advanced global parameters

Random seed

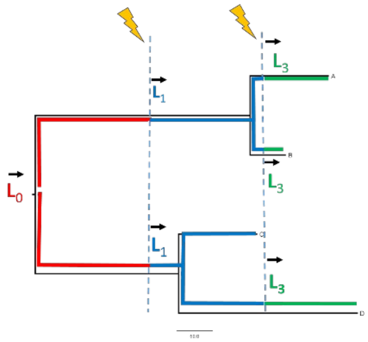
For debugging purposes, it is possible to specify the random seed by setting the `SEED` parameter.

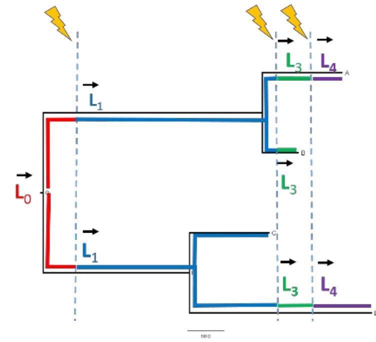
Specifying whether the fitness landscapes and their change times are shared among branches, or are branch-specific.

In some cases, it is desirable for all tree branches to have the same fitness landscape; in other, for the landscape on each branch to be independent (and change independently).

- If the config file contains the line
  `SHARED_LANDSCAPE true`
  then all points in the tree that are at the same distance from the root share the same landscape (this option, naturally, does not work with the allele-specific landscape change option that is described below).

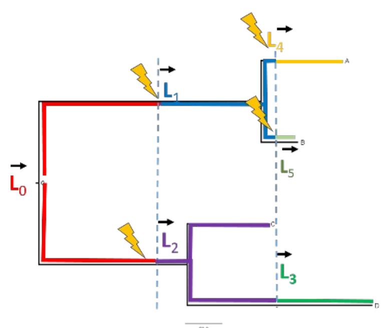
The figures below illustrate this option for deterministic regularly-spaced landscape changes (left) and for stochastic landscape changes (right); landscape changes are shown as “lightning bolts”, and the landscapes are denoted by L_0, L_1, etc. and color-coded.

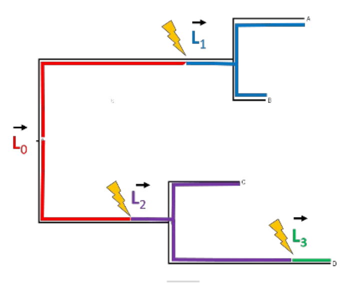




- If the config file contains the line
  `SHARED_LANDSCAPE false`
  then non-overlapping paths in the tree have independent fitness landscapes. If the landscape is updated stochastically, landscape change times will be different as well. **This is the default behavior.**

  The figures below illustrate this option: landscapes (marked by L_0, L_1, etc.) are independent for each path from root to leaves. In the figure on the left, the landscape change times (but not the values of the fitness vectors) are deterministic and therefore necessarily the same for all branches; in the figure on the right, the landscape change times are stochastic and therefore independent for all paths from root to leaf.





Advanced parameters concerning the rate of substitution

Understanding these parameters and their explanation is easier if one is familiar with the concept of a substitution-rate matrix and the corresponding stationary distribution vector. The reader is referred to, for example, Molecular Evolution: A Statistical Approach (*Yang 2014*)

Different fitness landscapes (fitness vectors) can lead to different rates at which substitutions take place. It is often desirable, however, to fix the expected substitution rate always to be 1, so that tree branch lengths would correspond to the expected number of substitutions. This is the approach taken in the `evolver` program of the PAML package. It is the default behavior of SELVa and can also be explicitly selected by setting the `CONSTANT_RATE` parameter to `true`.

If CONSTANT_RATE parameter is set to false, the substitution rate is not normalized, and tree branch lengths do not correspond to the expected number of substitutions.

The technical details of the meaning of this parameter and additional options are given in the Apendix.

Output

Output of the simulated sequences

SELVa prints out the sequence(s) generated for each tree node over the course of all simulation instances that were run simultaneously into a FASTA-format file allnodes.merged.fasta. If multiple simulation instances were run at the same time, the sequence at each node is a concatenation of the sequences generated by each instance (the order of instances is the same for all nodes).

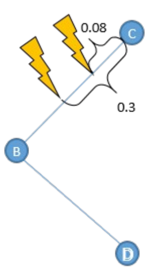
Landscape change information

The user has the additional option of printing out the information about all landscape changes, namely, the time of change and the value of the new vector. This option is turned on by setting the **global** parameter PRINT_LANDSCAPE_INFO to true. **Since keeping track of this information significantly increases memory usage and may therefore impact the running time as well (by causing the simulation to use virtual memory), this user is advised to turn on this option only if he or she plans to use this information.**

Landscape change times

The landscape change times are printed in FASTA-like format in the file changetimes.merged.fasta. Since any branch on a rooted tree can be uniquely identified by the node that it leads up to (its endpoint), we identify each point on the tree by the node terminating the branch on which it is located and the time remaining between that time point and that node.

For illustration, consider the figure below. In this example, two landscape changes take place on the branch leading up to the node C at 0.3 and 0.08 time units before the end of the branch, respectively.



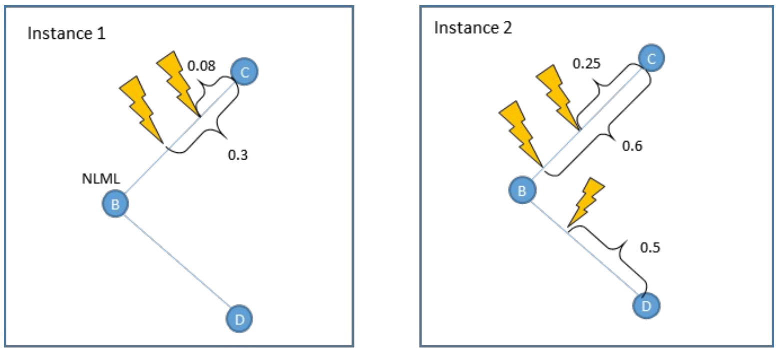
These changes are written to the changetimes.merged.fasta, with the terminal node of a branch written to the “FASTA header” and the change times encoded as the times remaining on that branch and written down on the line following the “header”:

```
>C
[0.3; 0.08];
>D
[];
```

```
>B
[];
```

As shown in this example, the (potentially multiple) landscape change times that occur on the same branch are printed in chronological order in brackets as a comma-and-space separated vector.

Change times for different instances of the simulation (different independent landscape histories) are separated by a semicolon followed by a space, e.g, a run with two instances whose landscape changes took place as shown in the figure below:



will be recorded in the `changetimes.merged.fasta` as

```
>C
[0.3; 0.08]; [0.6; 0.25];
>D
[]; [0.5];
>B
[]; []
```

Landscape values

Similarly to the change time information, the actual (scaled) fitness vectors are printed in the `fitnesses.merged.fasta` file following the "header" for the endpoint of the branch on which they occur. The fitness vectors are printed as bracketed comma-separated vectors. When multiple landscape changes occur on the same branch, the different landscape vectors are separated by colons (no space); they are listed in the same order as the corresponding change times in the `changetimes.merged.fasta` file, i.e., chronologically (within the given branch). All fitness vectors that occur on a given branch in one simulation instance are placed inside curly braces, with the curly brace-enclosed lists of fitness vectors for different instances separated by a semicolon followed by a space, e.g.:

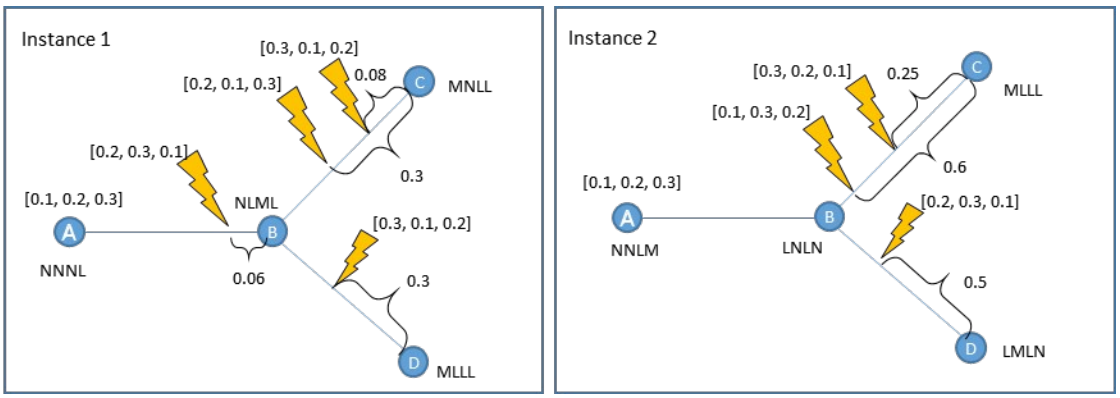
```
>C
{[0.2, 0.1, 0.3]:[0.3, 0.1, 0.2]}; {[0.1, 0.3, 0.2]:[0.3, 0.2, 0.1]};
```

The order of simulation instances is the same in all output files.

The initial landscape is postulated to occur at time 0.0 before the root node

Example

Consider the course of a hypothetical simulation summarized in the figure below (not drawn to scale). Two simultaneous instances of the simulation (`NUM_INSTANCES 2`) are run on a three-letter alphabet “LMN”; in both cases, the initial landscape is given by the (scaled) fitness vector (0.1, 02, 0.3) (read from a file). The landscape change times are determined stochastically and independently for each branch, and the new landscape is obtained by permuting the previous fitness vector. Landscape changes are marked by a “lightning bolt” with the new fitness vector given next to the “bolt”. The distance from the landscape change time to the branch endpoint is given in the diagrams. The four-letter sequence (`LENGTH 4`) of each of the four (ancestral and extant) species is given next to the corresponding tree node.



The contents of the output files corresponding to this example are given below. For purposes of illustration, the values from the first instance are shown red, and the values from the second instance are shown blue.

allnodes.merged.fasta:

```
>A
NNNLNNLM
>B
NLMLLNLN
>C
MNLLMLLL
>D
MLLLLMLN
```

changetimes.merged.fasta:

```
>A
[0.0]; [0.0];
>B
[0.06]; [];
>C
[0.3, 0.08]; [0.6, 0.25];
>D
[0.3]; [0.5];
```

fitnesses.merged.fasta:

```
>A
{[0.1, 0.2, 0.3]}; {[0.1, 0.2, 0.3]};
>B
{[0.2, 0.3, 0.1]}; {};
>C
{[0.2, 0.1, 0.3]:[0.3, 0.1, 0.2]}; {[0.1, 0.3, 0.2]:[0.3, 0.2, 0.1]};
>D
{[0.3, 0.1, 0.2]}; {[0.2, 0.3, 0.1]};
```