MIST: Model Inference from Sampled Trees

Yujin Chung and Jody Hey

Center for computational genetics and genomics, Temple University

Email: yujin.chung@temple.edu

August 11, 2016

1 Introduction

MIST is a program to analyze a multi-locus data set for estimating demographic parameters in isolation with migration models. Figure 1 shows the schematic of MIST. In step 1, the coalescent trees are sampled from a Markov chain Monte Carlo (MCMC) simulation under an importance sampling distribution, independent of a demographic model of interest. In step 2, the joint posterior probability under the demographic model of interest is calculated from the sampled coalescent trees and a differential evolution (DE) algorithm [Price et al., 2005] is applied to find the maximum a posteriori (MAP) estimate of the model parameters. The approximation of the joint posterior density includes the numerical integration over coalescent trees using the sampled coalescent trees, and the analytical integration over migration paths. By using a demography-free importance sampling distribution in step 1, it is possible to study diverse demographic models without having to repeat step 1.

2 Installation and Compilation

MIST is a command-line controlled program written in OpenMPI-C++. It should be easily compiled and run on any Linux system. The source files are available at `https://github.com/yujin-chung/MIST`. MIST used `Eigen`, c++ linear algebra library, and the source folder contains the library.

Installation from source code

Pick a directory where you want the code to be. This directory will be called $HOME in this document. Download the tar ball and put it in $HOME

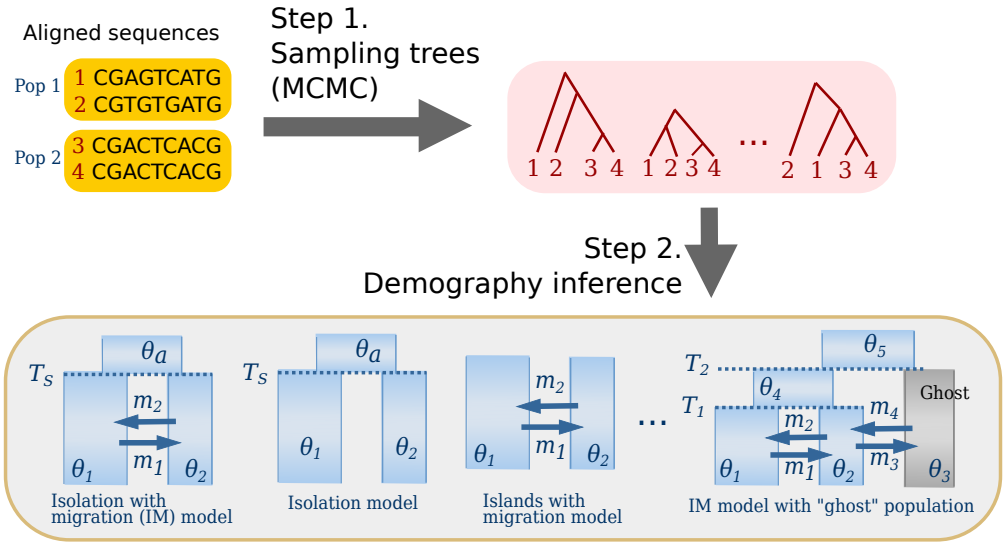
```
> tar -xzvf MIST.tar
```

This creates a directory named MIST with subdirectories MIST/example and MIST/src.

Compilation

MIST has been written using the OpenMPI-C++ framework, and can be compiled using standard MPI favors of the GNU compiler (including mpicc and mpicxx). OpenMPI must be installed. Also once installed, please make sure that the "bin" directory of OpenMPI is added to your machine's path. Then compile the software with these commands.

Figure 1: The MIST's schematic. In step 1 coalescent trees for the aligned DNA sequences are sampled from an MCMC simulation using an importance sampling prior distribution of trees that is free of a demographic model. In step 2, the set of sampled coalescent trees is used for the approximation of the joint posterior density under a demographic model of interest. Optimization of the joint posterior density provides an estimate of model parameters. The same set of trees from step 1 can be used repeatedly to study different demographic models.




```
> cd $HOME/MIST/src
> mpicxx -std=c++11 *.cpp *.cc -o MIST
```

If your machine has different compiler versions, for example `c++0x`, then use these commands.

```
> cd $HOME/MIST/src
> mpicxx -std=c++0x *.cpp *.cc -o MIST
```

This will compile program MIST. It is suggested that a copy of MIST be put in ∼/bin if this directory is in your path.

3 Syntax and options

Step 1: Sampling trees from MCMC simulation

To run the step 1 of MIST, type:

```
> mpirun -np <No. CPUs> MIST -r 1 -i <input data file> [-options]
```

The option -r 1 executes a MCMC simulation to simulate gene trees. For example, we use 2 CPUs for parallel computing to analyze DNA alignments in the input file, toy_data_10loci.u:

```
> mpirun -np 2 MIST -r 1 -i toy_data_10loci.u
```

At default, 100 trees for each locus are removed for burn-in period and 100 trees are sampled for every 10th iteration for thinning.

Here is a description of the basic options for step 1 only.

-r run	1 for step 1 (MCMC); 2 for step 2 (model inference)
-i input-file	input data file
-l number-of-trees	the number of iterations for sampling. default: 100
-b burn-in	the number of iterations for burn-in. default: 100
-n thinning	the number of iterations for thinning. default: 10
-u mutation-model	1 if mutation rate per site is constant; 2 if mutation rate per locus are various. Default: 1
-g prior	prior of coalescent trees. -g 0 if improper prior; -g 1 if coalescent prior of a single population; Default: 0 (improper prior)

Here is another example of a MCMC simulation using 3 CPUs of 1,000-tree sample from each locus after 2,000 iterations for burn-in.

```
> mpirun -np 3 MIST -r 1 -i toy_data_10loci.u -l 1000 -b 2000
```

Here is more advanced options for step 1 only.

-s number-of-pairs	the number of random pairs of mutation scalars to update ($s \leq [\ell/2]$), ℓ: the number of loci; 0 if $[\ell/2]$ random pairs of mutation scalars are tried to update. This option is called only if -u 1 is selected. Default 0.
-c number-of-chains	the number of chains per process. default: 1.
-q upper-bound	the upper bound of a single population size. This option is called only if -g 1 is selected. Default: 10
-d no-data	1 if likelihood is constant (no data is used); 0 if likelihood of a given data is computed. Debugging purpose. Default: 0

Step 2: Model inference

To run the step 2 of MIST, type:

```
> mpirun -np <No. CPUs> MIST -r 2 [-options]
```

The option -r 2 executes the differential evolution (DE) optimization procedure to find the maximum a *posterior* demographic parameters. MIST automatically read the output files of the step 1 in the same directory. For example, we take as an input the result files (100 trees per locus) from the previous example in step 1 analysis and estimate demographic parameters of 2-population IM model using 5 CPUs for parallel computing:

```
> mpirun -np 5 MIST -r 2 -n 10 4 -l 100 -m 100 -q 1 -t 0.2 -h "(1,2):3;"
```

The DNA data file is not used anymore and hence users should specify the number of loci is 10 and the number of gene copies per locus is 4 with option -n 10 4, and the number of sampled trees per locus is 100 with option -l 100. The underlying demographic model is specified with option -h "(1,2):3;". Population tree is expressed in newick format. That is, sampling populations are labeled starting with 1 and ancestral population label should be specified after ":" sign. The prior distribution of each parameter is a Uniform from 0 to a value which user specifies. The option -q 1 indicates the upper bound of population sizes is 1, -m 100 for 100 upper bound for migration rates and -t 0.2 for 0.2 upper bound for splitting times.

-n num-of-loci num-of-gene-copies	the number of loci and the number of gene copies
-l number-of-trees	the number of trees per locus
-m upper-bound-migration	the upper bound of migration rates
-q upper-bound-population-size	the upper bound of population sizes
-t upper-bound-splitting-time	the upper bound of splitting time
-h population-tree	population tree in newick-like format within quotation marks. In order to estimate demographic parameters in 2-population IM model, for example, -h "(1,2):3;".
-u mutation-model	0 if all the loci have the same mutation rates; 1 otherwise. Default: 0 (same per-site mutation rate)
-g prior	the prior used in MCMC. -g 0 for improper prior; -g 1 if non-uniform prior was used. Default: 0 (improper prior)
-s newick-tree	0 if the output of step 1 is the input files for step 2; 1 if trees in newick format are the input for step 2. Two files should follow -s 1: a filename of newick trees and a filename of sampling population labels. That is, -s 1 ¡treesfile¿ ¡population-labels¿. Default: -s 0.

Here is more advanced options for step 2 only.

-a ancestral-pop	0 if island model (no ancestral populations); 1 if isolation model. Default: -a 1
-c constant-pop-size constant-mig	two numbers indicating whether population sizes and migrations are constant or not, respectively. For example, 1 0 means population sizes are constrained to be the same but migration rates may different. Default: -c 0 0 (different population sizes and different migration rates)
-w checkpoint	indicator for checkpoint of DE and how often it writes a checkpoint. There are four indicators for checkpoint: 0 for no checkpoint (default), 1 for writing checkpoint, 2 for reading checkpoint but not writing checkpoint anymore, 3 for reading the checkpoint and writing checkpoint. For example, -w 1 100 means that it writes a checkpoint at every 100 iteration of DE. -w 0 means that checkpoint option is not invoked. -w 2 means that the DE starts from the checkpoint. -w 3 100 means that the DE starts from the checkpoint and also write a checkpoint at every 100 iterations. Default -w 0.
-i number-individuals	the number of individuals in the optimization using the differential evolution (DE) algorithm (default: 30)

At default, all population sizes and migration rates are different. However, we can estimate the parameters with the restriction of same population sizes and same migration rates as follows:

```
> mpirun -np 5 MIST -r 2 -n 10 4 -l 100 -m 100 -q 1 -t 0.2 -h "(1,2):3;" -c 1 1
```

When a file of trees in newick format is given, the filename should be specified with option -s 1. Moreover, the sampling population information of sequences should be provided as a file. For example "trees.txt" contains newick trees:

```
> more trees.txt
0 ((1:1.016114,2:1.016114):0.820494,(3:0.156677,4:0.156677):1.679931);
1 ((1:0.757583,(3:0.375963,4:0.375963):0.381620):2.190959,2:2.948542);
2 ((1:0.125980,2:0.125980):1.303508,(3:0.696509,4:0.696509):0.732979);
3 ((1:1.194771,2:1.194771):1.217341,(3:0.391543,4:0.391543):2.020569);
4 ((1:1.207241,2:1.207241):3.051563,(3:0.397384,4:0.397384):3.861421);
```

and "poplabels.txt" contains the sampling population labels of sequences:

```
> more poplabels.txt
1 1 2 2
```

which indicates that sequences 1 and 2 are sampled from population 1 and sequences 3 and 4 are from population 2. Then, the step 2 is executed as follows:

```
> mpirun -np 2 MIST -r 2 -n 10 4 -l 1 -m 100 -q 1 -t 0.2 -s 1 trees.txt poplabels.txt -h "(1,2):3;"
```

4 DNA sequence file formats

An snapshot of "toy_data_10loci.u" is

```
4 10
100 I
1 1 AGCCTAGCATTGCCATATATAGTTCGCGGTGTTACGGAGCACGGTTTTAAGGGTAGAGTAGTTACCAAATATGTTAACTGAGCACAAA
1 2 AGCCTAGCATTGCCATATATAGTTCGCGGTGTTACGGAGCTCGGTTTTAAGGGTAGAGTAGTTACCAAATATGATAACTGAGCACAAA
2 3 AGCCTAGCATTGCCATATATAGTTCGCGGTGTTACGGAGCACGGTATTAAGGGTAGAGTAGTTACCATAAATGATAACTGAGCACAAA
2 4 AGCCTAGCATTGCCATATATAGTTCGCGGTGTTACGGAGCACGGTATTAAGGGTAGAGTAGTTACCATAAATGATAACTGAGCACAAA
100 I
1 1 TCGCAAGTCCCCTTAGAGAGTCGACTAGTCGCCTGCAGTAAGTACCGATTCTTCGTCGTTGCTGGAGACAGGTTATACATGATACAGT
1 2 TCGCTAGTCCCCTTAGAGAGTCGACTAGTCGCCTGCTGTAAGTACCGAATCTTCGTCGTTGCTGGAGACAGGTTATACATGTTACAGT
....
```

The first line of the data file contains two integers for the number of gene copies and the number of loci. In the following lines, basic information and DNA sequences for each locus are grouped in the same format and listed. The first line in a group of lines for a locus contains the number of sites and the letter indicating the mutation model (I for infinite-site model and H for HKY). In the following lines, a single line contains the information of one gene copy in the locus: the sampling population label, gene copy ID and DNA sequence. The same format is repeated for other loci.

5 Output file format

Output file of step 1

Running MIST with option `-r 1` (MCMC) will create a various output files. The output files are explained below:

`MCMCsample_listTopo.txt`	contains the list of unique ranked tree topologies with *population labels* from the sampled coalescent trees through MCMC. The ancestral nodes have IDs and old nodes have larger IDs. The ranked topologies are in newick format and their IDs are in the file as well.
`MCMCsample_treeIDs.txt`	contains the tree IDs of sampled coalescent trees for each locus. The tree IDs are defined in `MCMCsample_listTopo.txt`
`MCMCsample_tipIDs.txt`	contains the tip IDs of sampled coalescent trees for each locus. Combining with the information in `MCMCsample_listTopo.txt` and `MCMCsample_treeIDs.txt`, investigators can reconstruct the sampled topologies of the coalescent trees. For example, the first sampled tree for a locus has tree ID 0 which represents `((2,2):5,(1,1):6):7` and the associated tip IDs are 3, 4, 1, and 2 in order. The *population labels* are replaced by tip IDs from left to right. Then the topology of the coalescent tree is `((3,4):5,(1,2):6):7`.
`coalescentTimes.txt`	contains coalescent even times (time to internal nodes). Since ancestral nodes in a ranked topology have larger IDs, the times to the ancestral nodes are fully determined.
`MCMCsample_eachLogPrior.txt`	contains log prior of each sampled coalescent tree. If improper prior was applied, the log of priors are all zero.
`MCMCsample_logEachLikelihoods.txt`	contains log-likelihood of each sampled coalescent tree
`mutationScaler_MCMCsample.txt`	contains the sampled mutation scalars of each locus. This file is created only if the option `-u 2` is executed.

Output of step 2

Running MIST with option `-r 2` (model inference) will print the result on the screen. When the optimization (DE algorithm) is done, it will print out the maximum a *posteriori* of all demographic parameters in the following order: population sizes (in the order of *population labels*), migration rate from population label 1 to 2, migration rate from 2 to 1 (both are defined backward in times) and splitting time. For example, MIST with the following options,

```
> mpirun -np 5 MIST -r 2 -n 10 4 -l 100 -m 100 -q 1 -t 0.2 -h "(1,2):3;"
```

will print out the following result at the end:

```
Maximum a posterior estimates: 0.0122853 0.00463582 0.000203102 20.9225 0.000338902 0.0335491
log(posterior density) = 103.245
```

The MAP estimations of population 1, 2 and 3 sizes (ancestral population) are 0.0122853, 0.00463582 and 0.000203102, respectively, migration rate are 20.9225 from 1 to 2 and 0.000338902 from 2 to 1, and the splitting time is 0.0335491. The log of the posterior density at MAPs is 103.254.

References

K. Price, R.M. Storn, and J.A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer, New York, 2005.