Parsimonator v1.0.1 Manual

Alexandros Stamatakis
Scientific Computing Group
Heidelberg Institute for Theoretical Studies
Alexandros.Stamatakis@h-its.org

What is Parsimonator?

Parsimonator is a no-frills light-weight implementation for building starting trees under parsimony for RAxML-Light. **It currently only works on DNA data!** It deploys a randomized stepwise addition order algorithm to build trees and thereafter conducts a couple of SPR (Subtree Pruning Re-Grafting moves) to further improve the parsimony score. Right now, it can only compute trees on DNA datasets. It uses SSE3 128-bit wide vector instructions to significantly accelerate parsimony computations. As of version 1.0.1 it also supports 256-bit AVX vector instructions as available on new Intel core i7 desktop computers. This means that, if you have a AVX-capable processor it will run much faster.
Although it is significantly slower than TNT, I think that it is the fastest open-source parsimony function implementation, albeit the search algorithm itself is rather naïve. It can also extend given trees that do not comprise all taxa of an input alignment by using a randomized stepwise addition order algorithm and for those taxa that are not contained in the starting tree.

What's new in Parsimonator?

Not much, it has been extracted from standard RAxML, with a lot of RAxML overhead removed. In contrast to standard RAxML it nonetheless uses the SSE3-optimized parsimony function implementation for tree extension and unlike in RAxML only for de novo tree building. It is meant to be used with RAxML-Light v1.0.2 for building very large trees.

How do I compile Parsimonator?

The distribution comes with three Makefiles for the standard, SSE3-based and AVX-based versions. **Note that, whenever you can, you should use the SSE3 or, even better AVX versions, because they are about factor 4 to almost factor 8 faster respectively.** SSE3 are SIMD (Single Instruction Multiple Data) vector instructions which are offered by all more recent AMD and Intel x86 architectures. SSE3 instructions allow you to execute instructions on 128 bits within one clock cycle of your CPU. AVX are also SIMD instructions, but the vector unit is larger, that is, 256 bit wide. Note that, AVX instructions are currently only available with new Intel desktop processors, but they should become available for Intel server systems and AMD processors in 2011 I guess. I have only tested this with the gcc compiler so far.

To compile the standard version, type:

```
make -f Makefile.gcc
```

This will generate an executable called `parsimonator`. If you want to compile the SSE3 version next, first type "`rm *.o`" in your terminal to remove the object files generated for compiling and linking the sequential program and type:

```
make -f Makefile.SSE3.gcc
```

This will produce an executable called parsimonator-SSE3

For the AVX version type:

```
make -f Makefile.AVX.gcc
```

This will produce an executable called parsimonator-AVX

Command line options

Here are the Parsimonator program options, many are similar to the standard RAxML options.

[parsimonator|parsimonator-SSE3|parsimonator-AVX]
-s sequenceFileName
-n outputFileName
-p randomNumberSeed
[-t incompleteTree]
[-N numberOfTreesToCompute]

-n Specifies the name postfic of the output files.

-s Specify the name of the alignment data file in relaxed PHYLIP format

-t Specify a user starting tree file name in Newick format. Note that, this tree is supposed not to contain all taxa of the input alignment.

-p Specify a random number seed doe the randomized stepwise addition process.

-N Specify how many randomized stepwise addion order trees you want to compute.
DEFAULT: one tree

Usage Examples

Before using parsimonator you should check that the alignment is correctly formated, doesn't contain duplicate taxon names etc. with standard RAxML by typing ./raxmlHPC -f c ....., i.e., the alignment file checking option. Then, usage is easy, suppose we want to compute a starting tree from scratch using the example alignment contained in the source archive, we'd just type:

```
./parsimonator-SSE3 -s 354.phy -p 12345 -n T1
```

This will just generate one parsimony starting tree that will be stored in a file called RAxML_parsimonyTree.T1.0

The terminal output will look like this:

```
stamatak@exelixis:~/Desktop/GIT/raxml-hpc/parsimonator/Parsimonator-1.0.0$ ./parsimonator-SSE3 -s 354.phy -p 12345 -n T1

Parsimonator version 1.0.0 a fast open-source code for builiding DNA parsimony start trees

Released under GNU GPL in February 2011 by Alexandros Stamatakis

Alignment has 460 sites and 354 taxa

1 randomized stepwise addition order parsimony trees with a couple of SPR moves will be computed

Parsimony tree [0] computed in 0.425553 seconds written to file: RAxML_parsimonyTree.T1.0
```

To compute multiple parsimony starting trees, just type:

```
./parsimonator-SSE3 -s 354.phy -N 5 -p 12345 -n T2
```

and you will obtain the following output:

```
stamatak@exelixis:~/Desktop/GIT/raxml-hpc/parsimonator/Parsimonator-1.0.0$ ./parsimonator-SSE3 -s 354.phy -N 5 -p 12345 -n T2

Parsimonator version 1.0.0 a fast open-source code for builiding DNA parsimony start trees

Released under GNU GPL in February 2011 by Alexandros Stamatakis

Alignment has 460 sites and 354 taxa

5 randomized stepwise addition order parsimony trees with a couple of SPR moves will be computed

Parsimony tree [0] computed in 0.419410 seconds written to file: RAxML_parsimonyTree.T2.0
Parsimony tree [1] computed in 0.300762 seconds written to file: RAxML_parsimonyTree.T2.1
Parsimony tree [2] computed in 0.300196 seconds written to file: RAxML_parsimonyTree.T2.2
Parsimony tree [3] computed in 0.205048 seconds written to file: RAxML_parsimonyTree.T2.3
Parsimony tree [4] computed in 0.391552 seconds written to file: RAxML_parsimonyTree.T2.4
```

This will generate 5 randomized stepwise addition parsimony trees stored in files RAxML_parsimonyTree.T2.0, ..., RaxML_parsimonyTree.T2.5.

Now, if we want to extend a given ML tree for instance, that contains 100 out of the 354 taxa of the example alignment, we just need to type:

./parsimonator-SSE3 -s 354.phy -t RAxML_bestTree.incompleteTree -N 5 -p 12345 -n T3

This will once again compute 5 randomized stepwise addition parsimony trees, randomly adding those taxa that are not contained in the starting tree stored in file RAxML_bestTree.incompleteTree. The terminal output will look like this:

```
Parsimony tree [2] computed in 0.300196 seconds written to file: RAxML_parsimonyTree.T2.2
Parsimony tree [3] computed in 0.205048 seconds written to file: RAxML_parsimonyTree.T2.3
Parsimony tree [4] computed in 0.391552 seconds written to file: RaxML_parsimonyTree.T2.4

stamatak@exelixis:~/Desktop/GIT/raxml-hpc/parsimonator/Parsimonator-1.0.0$ ./parsimonator-SSE3 -s 354.phy -t RAxML_bestTree.incompleteTree -N 5 -p 12345 -n T3

Parsimonator version 1.0.0 a fast open-source code for builiding DNA parsimony start trees

Released under GNU GPL in February 2011 by Alexandros Stamatakis

Alignment has 460 sites and 354 taxa

5 randomized stepwise addition order parsimony trees with a couple of SPR moves will be computed

Read Starting tree with 100 tips, total 354
Parsimony tree [0] computed in 0.231452 seconds written to file: RAxML_parsimonyTree.T3.0
Read Starting tree with 100 tips, total 354
Parsimony tree [1] computed in 0.411879 seconds written to file: RAxML_parsimonyTree.T3.1
Read Starting tree with 100 tips, total 354
Parsimony tree [2] computed in 0.217426 seconds written to file: RAxML_parsimonyTree.T3.2
Read Starting tree with 100 tips, total 354
Parsimony tree [3] computed in 0.306844 seconds written to file: RAxML_parsimonyTree.T3.3
Read Starting tree with 100 tips, total 354
Parsimony tree [4] computed in 0.294811 seconds written to file: RAxML_parsimonyTree.T3.4
```

Frequently Asked Questions

Q: How much memory does this consume?
A: Not much, for a tree with 19,000 sites and 116,000 taxa it needs about 4GB of main memory.

Q: Will you implement a protein parsimony function?
A: Maybe, but it depends on my time.

Q: Can I use this to infer high-quality parsimony trees?
A: No, you should use TNT by Pablo Goloboff for this, the program is meant to be used with RAxML for Maximum Likelihood analyses and thus just strives to infer reasonably good starting trees for a subsequent ML search without trying to hard to optimize the parsimony score.

Q: How much faster is the AVX version?
A: It's pretty fast, here is a table with execution times for a dataset with 125 taxa and 29149 sites (included in the distribution) on my brand-new toy, an Intel Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz:

Command: ./parsimonator -p 12345 -s 125.phy -n T1
execution time: 4.725 seconds

Command: ./parsimonator-SSE3 -p 12345 -s 125.phy -n T2
execution time: 1.597 seconds

Command: ./parsimonator-AVX -p 12345 -s 125.phy -n T3
execution time: 0.926 seconds