Boosting BAli-Phy with UPP

Cécile Rottner, Théo Zimmermann, (Nam Nguyen, Tandy Warnow)

December 1, 2013

1 Introduction

The estimation of a phylogeny is always a tremendous project which involves a lot of different technical and scientific skills both in biology and in computer science. It follows the general schema: collection of data, in particular DNA extraction and sequencing, multiple sequence alignment, tree or network reconstruction. Each of these steps is equally important and for the last two ones, increasingly efficient new algorithms have been devised.

Various alignment software are in use today, each presenting specific advantages. BAli-Phy is a Bayesian software which does simultaneously the sequence alignment and the tree estimation. Its advantages are that it includes different and realistic models of evolution, accounting for indels and substitution; it is not biased by the use of a guide tree; it takes uncertainty into account [8]. In a study from some years ago comparing (among others) SATé and BAli-Phy, there were models on which BAli-Phy created better alignments [2]. Unfortunately, Bayesian methods are very slow and consequently, the size of datasets they can address is extremely limited.

The current challenge is to address larger and larger datasets. This challenge threatens Bayesian methods to become useless. Fortunately, some of the new algorithmic techniques that have been devised to address those ultra-large datasets are adaptable. Some software rely on interesting mathematical tricks to leverage the capacity of a basic multiple alignment software. This is the case of two software that have been developed at the University of Texas at Austin: PASTA [4] and UPP [6]. Both rely on a core brick which is a multiple sequence aligner that can be replaced.

PASTA uses a divide-and-conquer approach where only the base case requires the core aligner. Consequently, even a slow aligner like BAli-Phy can be used and run in parallel to compute these preliminary alignments.

UPP proposes an even more powerful leverage because it only asks the core brick to align a backbone of randomly selected sequences. With a backbone of size 100 and the help of Hidden Markov Models (HMMER), it can then compute with very high accuracy multiple sequence alignments up to a million sequences [6].

Until now, the core brick of PASTA has been MAFFT-linsi [1] and the core brick of UPP has been SATé. However, these two aligners can be replaced by others fairly easily.

In this article, we show that it is possible and quite simple to leverage BAli-Phy using UPP. Our boosted version was able to tackle datasets of size 10,000 with very high accuracy. Consequently, this article is the first serious step on the path of Bayesian methods towards handling ultra-large datasets.

2 Results

UPP is often run with backbones of varying sizes between 10 and 100. A first step was to determine which size of backbone would be good for UPP with BAli-Phy.

We worked first with RNAsim, a model containing 10,000 sequences of length 1,000 which were generated from a simulation (see section 6). To chose the good size of backbone we first ran the original UPP (with SATé) to give us an idea of what was enough to get a good accuracy. Results with a backbone of size 10 were not very good so we were planning to test backbones of size 20 and 50.

We ran BAli-Phy on backbones of size 20 but it took around a week on machines with 24Go of RAM to reach the convergence (after more than 20,000 iterations). Given the way the cluster is built we do not really know what CPU speed the machines we used had but it is probably around 2.4GHz with 4 cores.

Given that it took already so much time to reach the convergence on subsets of size 20, we postponed the idea of testing larger backbones. It appears than the limiting factor was the runtime and it was probably due, in particular, to the length of the sequences. Indeed, the authors say that for sequences of length above 1,000, the use of BAli-Phy becomes more and more difficult.

Thus, an idea if we want to test larger backbones would be to shorten the sequences. For instance, we can extract shorter sequences from our previous model.

As said before, we observed that we needed at least 20,000 iterations to reach a good level of convergence.

Once we reached the convergence (see section 6 for details about how it is measured), we asked BAli-Phy to summarize its output. It computed an

	UPP with BAli-Phy	UPP with SATé
SPFN	0.1291	0.1396
SPFP	0.0958	0.1115

Figure 1: False positive and false negative rates for output alignments

	BAli-Phy
SPFN	0.0871
SPFP	0.0870

Figure 2: False positive and false negative rates for backbone alignments

estimate of the backbone alignment using maximum posterior decoding and it computed the majority consensus topology tree. We ran UPP(20,20) (i.e. with no decomposition) with the backbone alignment and tree computed with BAli-Phy. UPP uses the backbone to compute alignments for the whole set of sequences.

What is extremely powerful in UPP is the fact that it only takes a couple of minutes on any standard machine to compute the alignment for 10,000 sequences. It only requires the aligned backbone BAli-Phy had taken more than one week to compute.

We measured the SPFN and SPFP rates of our output with respect to the true alignment. We also ran UPP with SATé and with the same parameters otherwise. We give in Figure 1 a comparison of the SPFN and SPFP rates in the two cases. We have slightly better results with UPP/BAli-Phy.

We also measured the SPFN and SPFP for the backbone alignments given by BAli-Phy, in Figure 2. In particular, we wanted to know how much of error UPP added by scaling up from the backbone to the full dataset. Given these figures, we can conclude that it does not add much error.

We also computed the trees on each of the two alignments (one by UPP with BAli-Phy, another by UPP with SATé) using FastTree [7], and calculated the tree error of each of these trees with respect to the true tree. The results are displayed in Figure 3. The error we get on the tree is also better for UPP/BAli-Phy. The calculation of the tree error using the alignment trees computed by RAxML [9] (instead of FastTree) will probably yield similar results (it has run for 16 days and it is still running). Here, the tree were constructed on masked alignments. We get a higher tree error when trees are computed from unmasked alignments (around 0.17 - 0.18).

We were interested in computing several trees on the backbone alignment

	UPP with BAli-Phy	UPP with SATé
Tree error (FastTree, masked alignments)	0.1337	0.1377
Tree error (RaxML)	...	...

Figure 3: Tree errors for phylogenic tree on the output alignments

	BAli-Phy
Output tree (c50)	0.1765
FastTree tree	0.1765
RaxML tree	0.1765

Figure 4: Tree errors for phylogenic tree on the backbone alignments

and comparing their errors so that we could know if we were right to chose the majority consensus tree produced by BAli-Phy as an input to UPP. BAli-Phy produces various output trees: c50 (majority topology), but also c66, c80, c90 (more restrictive consensus).

We computed the tree error for various trees: the output tree c50 of BAli-Phy, a tree computed with FastTree given the output alignment of BAli-Phy, a tree computed with RaxML given the same input. All these trees are almost fully resolved so the SPFN, SPFP and RF distance error have the same values, shown in Figure 4. These errors are higher than previously since there was no masked alignment produced that we could use (the masked alignment is something specific to UPP). It is however interesting to notice that the tree error on the backbone aligned by BAli-Phy is the same, whatever tree construction method we choose.

Also, we chose to use c50 (one of the output tree of BAli-Phy) in our comparison because it is fully resolved, but another output tree of BAli-Phy is interesting: c80, which has a FP rate of 0.0714, a FN rate of 0.2373 and a RF error of 0.1533 (the other trees yield very high errors). We tried to run UPP with the tree c80 as an input, and we got similar SPFP (0.1292) and SPFN (0.0957), compared to UPP run with c50 (Figure 1).

3 Discussion

UPP allows us to leverage BAli-Phy to handle a very large number of sequences. It does not help with respect to sequences length. Unfortunately, BAli-Phy uses quadratic runtime and memory in terms of the sequences length. Consequently, this is another limitation of Bayesian software that our approach was not designed to tackle and which will require more research before Bayesian software can be uplifted to the same levels as their competitors today.

Since we tried to tackle sequences of length 1,000, it seems possible that, for shorter sequences, the effect would have been even more impressive. In particular, if one wanted to boost BAli-Phy using PASTA, our advice would be to start with sequences of shorter length.

We were concerned about the speed with which BAli-Phy iterates but it was difficult to measure. We were first able to get 1,500 iterations in less than 4 hours on standard cluster machines, 19 sequences whose length was around 1,700 when correctly aligned but around 900 when we removed indels. We have to compare that to the 10 hours on powerful machines we needed to reach the first 1,500 iterations for 20 sequences of length 1,000 when testing RNAsim. The main difference was the model so it probably shows that the model has a big impact on the speed of BAli-Phy.

4 Future Work

BAli-Phy is designed to output different alignments and trees. If we tested the various output trees produced by BAli-Phy, we only managed to get one kind of alignment (the one we used in our work). Thus, we will next try to get those other alignments and assess which ones are best for our analysis.

Our boosted version of BAli-Phy is apparently able to handle large datasets (up to 10,000 taxa) but this needs to be confirmed by more results. We would like to compute new alignments with UPP/Bali-Phy on replicates of the dataset we used. We tried to run new replicates but we encountered many technical difficulties, so no more replicates are running for the moment. Also, since BAli-Phy is really slow to run, we will try other dataset models with a slower rate of evolution (like the 1000-taxa dataset used in SATé, or some Gutell datasets), since this could help BAli-Phy to run a little bit faster. Once this is done, we could test UPP/BAli-Phy on real-world datasets.

Then, there are some directions in which we would like to go afterwards:

1. testing UPP/Bali-Phy with a backbone decomposition (with for example UPP(20,10)) on datasets such as RNASim or 16SBAl.

2. testing UPP/BAli-Phy with a larger backbone (for example of size 50), aligned with BAli-Phy boosted with PASTA.

5 Conclusion

Our work proves that it is possible to leverage a widely-used Bayesian method to handle very large datasets for MSA. Moreover, we proved the interest of using BAli-Phy rather than SATé on some datasets.

A first remark will be to remind that this advantage is in terms of accuracy but definitely not in terms of running time. Indeed, BAli-Phy took more than a week where SATé was done in a few hours.

Besides, we do not take advantage of all the properties of BAli-Phy. BAli-Phy generates trees and alignments that we use for the backbone. But when we compute the final tree given the alignments we computed for the whole set of sequences, we use our own method to do it (FastTree or RAxML). It does not mean that the advantage that BAli-Phy does join estimation of the tree and the alignment is lost but simply that it is not used at the end of the process. Another advantage of BAli-Phy is that it constructs the tree from all the possible alignments and not from a consensus alignment. We lose this advantage at the last step of our construction but it was still useful for constructing the backbone alignment.

It would be interesting to see if there is a way to adapt divide-and-conquer approaches, such as DACTAL [5], to a software like BAli-Phy in order to never lose those interesting properties.

6 Methods

We give here full details about our experiments so that it will be very easy for anyone to replicate. All the software and datasets we used are available online for free. We give the details on where to find them.

The initial RNAsim datasets can be downloaded from

http://kim.bio.upenn.edu/software/csd.shtml

Our 10,000 taxa dataset was randomly subsampled from the 1 million taxa dataset at this address.

We ran BAli-Phy with the following command:

```
bali-phy - -randomize-alignment backbone.fas
```

We used a cluster to run 10 MCMC in parallel. 10 is the recommended number by the authors of BAli-Phy.

Contrary to what is explained in [3], we did not experienced any problem with BAli-Phy memory usage. In fact, we did not even needed the large amount of memory the machines we used provided but we used them because they were also conveniently powerful machines.

To measure convergence we primarily used the command

```
trees-bootstrap directory-1/C1.trees directory-2/C1.trees ···
                        directory-n/C1.trees > partitions.bs
```

to calculate the ASDSF value, which is suggested in the Users manual of BAli-Phy. Some other statistics are also proposed and everything can be summarized, when we are confident enough that we reached the end, by using

```
bp-analyze.pl directory-1/ directory-2/ ···
```

All these values are computed by comparing several MCMCs.

We waited until we had reached the recommended value for the different statistics. In particular, our main criterion, the ASDSF value, had to pass under the bar of 0.01. When we waited for this parameter to converge, we observed that using a subset of all the MCMCs we had could help lower the value, especially by removing chains which had done a very different number of iterations (a lower number or even a larger one). We did not entirely understood this strange observation.

BAli-Phy can be downloaded from

http://faculty.biomath.ucla.edu/msuchard/bali-phy/

and the Users manual is available at

http://faculty.biomath.ucla.edu/msuchard/bali-phy/README.html

We ran UPP with the command

```
python bin/exhaustive_upp.py -t c50.tree -a P1-max.fasta
                        -A 20 -s query.fas -o upp_20_20
```

c50.tree is the majority consensus tree produced by BAli-Phy on all the samples from our MCMCs (minus a "burn-in" of the 10 percents first samples) and P1-max.fasta is one of the summary alignment it should produce. As explained earlier, we also tried to look at other consensus trees (but the results were not better). We did not try other alignments because surprisingly, this one was the only one BAli-Phy effectively produced. The option

-A serves to express that UPP should create only one HMM for the whole backbone. By default, it would have splitted the backbone in two and created two HMMs. The option -o only gives the prefix of the output file.

UPP can be found at

https://github.com/smirarab/sepp

The file README.UPP.md contains the detailed instructions on how to install it and where to find SATé, which is required prior to installing UPP.

We aligned the backbones with SATé then we called UPP as above.

You should have already installed SATé if you correctly installed UPP.

We compared the alignments thanks to the software FastSP which can be found at

http://www.cs.utexas.edu/~phylo/software/fastsp/

The command to run was

```
java -jar FastSP_1.5.jar -r true.fasta -e upp_20_20_alignment.fasta
                                                    > upp_20_20_stats.txt
```

We constructed trees from alignments using FastTree and RAxML. To run FastTree we used the command:

```
FastTree -nt -gtr < upp_20_20_alignment_masked_capitalized.fasta
                                                 > upp_20_20_fasttree.tree
```

UPP produces two alignments, a masked and an unmasked alignment. As said earlier, FastTree produced way better trees by using the masked alignment. Before calling FastTree, we had to make sure every lowercase u in the alignment file was first replaced by a uppercase U because of some bug in FastTree which prevent it from recognizing lowercase u. The option -nt means nucleotides and the model we used was GTR. FastTree can be downloaded from

http://www.microbesonline.org/fasttree/#Install

FastTree was very fast on the backbone sequences but took several hours on the whole set of sequences aligned with UPP.

To run RAxML, we used the command:

```
raxmlHPC-PCTHREADS-SSE3 -m GTRCAT -T 8 -p 32585475
                    -s upp_20_20_alignment.fasta -n upp_20_20
```

We used a multi-threaded version of RAxML. The model was GTRCAT. We ran 8 threads in parallel. The value given with the option -p is just a random seed given by the user so that the result can be reproduced. The option -s gives a suffix to all the files RAxML produces. This suffix must have not been used for a previous call to RAxML. We kept the "RAxML_bestTree" output file.

RAxML took several GB of memory and more than 16 days (cumulative cpu time) to compute a tree with 10,000 taxa. On the 20 taxa backbone, it took 7 seconds.

The most recent version of RAxML can be obtained from

https://github.com/stamatak/standard-RAxML

To compute the tree error, we used a script of spruce which can be found here:

http://www.cs.utexas.edu/ phylo/software/spruce/

We ran the following:

```
python getFpFn.py -t true.tt -e upp_20_20_fasttree.tree
                        > upp_20_20_fasttree_stats.txt
```

References

[1] K Katoh, K Kuma, H Toh, and T Miyata. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucl Acids Res, 33(2):511:518*, 2005.

[2] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C Randal Linder, and Tandy Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science (New York, N.Y.)*, 324(5934):1561–4, June 2009.

[3] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C Randal Linder, and Tandy Warnow. Supporting Online Material for Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science (New York, N.Y.)*, 324(5934):1561–4, June 2009.

[4] Siavash Mirarab, Nam Nguyen, and Tandy Warnow. PASTA : ultra-large multiple sequence alignment. Submitted.

[5] Serita Nelesen, Kevin Liu, Li-San Wang, C Randal Linder, and Tandy Warnow. DACTAL: divide-and-conquer trees (almost) without alignments. *Bioinformatics (Oxford, England)*, 28:i274–82, 2012.

[6] Nam Nguyen, Siavash Mirarab, and Tandy Warnow. UPP: Ultra-large alignment using SEPP. Article in preparation.

[7] Price, Dehal, and Arkin. Fasttree 2 approximately maximum-likelihood trees for large alignments. *PLoS ONE 5(3): e9490. doi:10.1371/journal.pone.0009490*, 2010.

[8] Benjamin D Redelings and Marc a Suchard. Joint Bayesian estimation of alignment and phylogeny. *Systematic biology*, 54(3):401–18, June 2005.

[9] Alexandros Stamatakis. Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics (2006) 22 (21): 2688-2690. doi: 10.1093/bioinformatics/btl446*, 2006.