Supplementary Material

Fernando Izquierdo-Carrasco, John Cazes,
Stephen A. Smith, and Alexandros Stamatakis

November 22, 2013

Contents

1 The PUmPER Framework

PUmPER (Phylogenies Updated PERpertually) works with iterations that combine two different stages: MSA (Multiple Sequence Alignment) construction and ML (Maximum Likelihood) Phylogenetic Inference. In this Section, we provide some additional details on each stage.

Both stages are executed sequentially within each iteration. The first iteration (see Figure 1) is called *initial iteration*, and starts the phylogenetic inference from scratch. Update iterations (see Figure 2) extend alignments and trees from previous iterations.

PUmPER can automatically update reference alignments and phylogenies by new sequence data, once the data become available on GenBank. This is helpful for commonly used datasets such as *rbc*L for seed plants, which has been used for broad scale phylogenies since [1]. As we show in our evaluation in Section 3, our tree update algorithm does not only allow for extending phylogenies at a lower cost (in terms of energy and man hours), but it also yields equally good likelihood scores as *de novo* tree inferences that are conducted from scratch.

PUmPER can be used to run either standalone on a single server or make use of an external cluster to offload the tree searches. Furthermore, it may make use of PHLAWD to generate alignments, or assume alignments are provided by the user. Since the exact requirements are different for each configuration, we provide specific examples for configuration and usage for each use each use case at the PUmPER repository (`https://github.com/fizquierdo/perpetually-updated-trees`). We also provide an installer and tutorials that should help the user to get started with PUmPER.

1.1 MSA Construction/Extension with PHLAWD

We briefly describe the basic PHLAWD [9] procedure here. PHLAWD requires the user to supply a configuration file that specifies for which organisms (as defined by the NCBI taxonomy) and which gene region(s) the dataset shall be constructed. The user can identify the focal gene region by supplying a set of exemplar sequences. These sequences are used for pairwise alignments and homology assessment. Additionally, the user can provide search terms that will be compared against the description of the sequences in the GenBank database to limit the scope of the sequence search. These are used in a Smith-Waterman procedure that filters sequences that are too dissimilar

compared to the exemplars. Using the remaining sequences, PHLAWD attempts to construct MSAs. If the sequences included in the MSA are too divergent to construct a reliable MSA, PHLAWD splits up these sequences based on a guide tree, the default of which is the NCBI taxonomy. These subsets are initially aligned independently, then profile alignments are used to align the subsets based on the guide tree.

We have extended PHLAWD to facilitate the generation of simulated MSAs. Under simulation mode, the user can supply sequences (as an alternative to GenBank downloads), and guide-trees (as an alternative to the NCBI taxonomy). We also changed the underlying PHLAWD file organization. Previously, PHLAWD stored all the intermediate alignments and other information in flat files. Now all files produced by a PHLAWD run are stored in a SQLite database file. The database stores sequences that have been retrieved from GenBank and included in the MSA, and/or sequences that have been added by the user(s).

When PHLAWD has already been executed once (initial iteration) and new sequences are added to the SQLite database, PHLAWD can be executed in update mode (`updatedb` option). Initially, the sequences of each new taxon are aligned to the closest existing subalignment. Then, PHLAWD executes profile-profile alignments in the same order as in the original run.

During an update iteration, each PHLAWD instance is extended independently. The end result of a PHLAWD stage, be it within a initial or update iteration, is a supermatrix stored in *phylip* format, and is used as input data by the subsequent phylogenetic inference stage. Since these two stages remain decoupled, it is straight-forward to replace PHLAWD by an alternative MSA construction method.

1.2 Phylogenetic Inference

The second stage of every iteration is the phylogenetic inference of a set of trees based on the most recent MSA. The number of independent phylogenetic tree searches conducted at each iteration depends on two user parameters: the parsimony parameter (p), and the size of the tree set (b) that shall be selected and kept in the end. We denote the values of p and b for iteration i as $p(i)$ and $b(i)$. In the initial iteration $i := 0$, PUmPER generates p randomized stepwise addition order parsimony starting trees (i.e., Parsimonator is invoked with p distinct random seeds). In an update iteration $i > 0$, given an extended MSA and a set of selected ML trees (from the

preceding iteration), $p(i)$ denotes the number of comprehensive randomized stepwise addition order parsimony starting trees that `PUmPER` generates from the tree set of iteration $i-1$. Thus, `PUmPER` invokes `Parsimonator` to extend the $b(i-1)$ trees from the preceding iteration. In each call, `Parsimonator` generates $p(i)$ distinct comprehensive parsimony trees from the same preceding (non-comprehensive) ML tree. Subsequently, `PUmPER` optimizes each comprehensive parsimony starting tree topology under ML with ExaML [11] or RAxML-Light [12]. Thus, the total number of ML searches conducted per iteration is $p(0)$ for the initial iteration ($i = 0$), and $p(i) \cdot b(i-1)$ for all consecutive iterations ($i > 0$). By default, `PUmPER` will try to run ExaML. Raxml-Light will be used if ExaML is not installed. The choice of the specific RAxML-Light version (e.g., SSE3 versus AVX vectorization, Pthreads versus MPI parallelization) depends on the available hardware. Once all ML searches have been completed, `PUmPER` scores the $p(i) \cdot b(i-1)$ topologies with standard RAxML [10] (`-f J` option) under the Γ model of rate heterogeneity [13]. This option also computes SH-like branch support values as described in [3]. Then, `PUmPER` selects the $b(i)$ best-scoring ML tree topologies, which will then be used as starting trees for iteration $i+1$.

2 Implementation

2.1 Automated Update Iterations

`PUmPER` starts an update iteration if (i) the alignment from the previous iteration has been extended *and* (ii) the phylogenetic analyses of the previous iteration have been completed. `PUmPER` can run a `cron` job that periodically checks if the two conditions are met. The MSA extension using `PHLAWD` is automated via another `cron` job that periodically (default: once per week) queries GenBank and launches `PHLAWD` to extend the MSA if a minimum number of new sequences (default: 100) have become available.

2.2 Using a External Cluster to offload Tree Searches

By using cluster-specific configuration files and batch scheduler templates, the local server creates and submits batch scripts for executing the phylogenetic searches. When all searches have been completed, the batch job transfers the results back to the local system. The ML searches may run for

a long time and require (multiple) restarts from a checkpoint file. ExaML and RAxML-Light offer such a checkpoint and restart capability, which allows for conducting a single tree inference in multiple steps if the run time exceeds the cluster queue limits. RAxML-Light also implements the memory saving techniques described in [4] and [5], which are useful if memory requirements are too large.

We have successfully used the `PUmPER` framework with two widely-used submission engines: `SGE` and `SLURM`. It is straightforward to adapt the current template files to other schedulers. Note that, because cluster setups, security policies, queuing system configurations etc., are different for each individual system, deploying the `PUmPER` framework in conjunction with a cluster using `SGE` or `SLURM` might still need manual configuration.

2.3 Custom iPlant setup

In our testing and development environment, we used a dedicated server to control the workflow on a remote cluster. This server, *Wooster*, is a dedicated virtual machine (VM) provided by iPlant collaborative (`http://www.iplantcollaborative.org/`) to manage the production of perpetually-updated trees. It was configured with 8 Intel Westmere cores and 16 GB of memory. The processes running on this local virtualized server are relatively lightweight.

There were two clusters available. Both are systems at the Texas Advanced Computing Center (**TACC**), and are part of the XSEDE (Extreme Science and Engineering Discovery Environment) program. We used the Linux cluster *Lonestar* during the development phase of the cluster computing component. The Lonestar cluster is composed of just under 2,000 compute nodes each with two 6-core Intel Westmere processors and 24 GB of RAM. Lonestar uses the Sun Grid Engine (SGE) batch facility to schedule jobs and allows users to connect directly via `ssh`. This batch scheduling facility along with the use of `ssh` for remote commands allows the use of a locally managed master server, in this case Wooster, to distribute the computationally intensive tasks. By using template files to describe the cluster and the respective batch system, we have developed the cluster component to be portable to most HPC systems.

Since January of 2013, a newer, more powerful cluster in TACC named *Stampede* is available to iPlant via an XSEDE project. Stampede is composed of 6,400 nodes, each with two 8-core Intel Sandy Bridge processors, 32 GB of

RAM and a Xeon Phi Coprocessor. Although, RAxML, ExaML, RAxML-Light, and Parsimonator, do not yet take advantage of the Xeon Phi, the RAxML family of codes do contain AVX optimizations to take advantage of the Sandy Bridge processors. Stampede differs from Lonestar in a few areas that require changes to the template file.

The cluster and batch template files were ported from Lonestar to Stampede in under an hour. Since we originally developed the cluster file and batch template files for use on Lonestar and the SGE system, we had to modify them to run on the newer system, Stampede. This is easily done for the cluster file, which contains a description of the compute node resources and the path to the appropriate binaries. Below is an example of a cluster template file for Stampede.

```
# Info about cluster
cores_per_node: 16
mem_per_node: 31000
#Project required for batch scheduler
project: TG-MCB110022
submission: slurm

# Installed binaries, absolute paths in remote machine
parsimonator: ~/remote/wooster/bin/parsimonator-AVX
raxmllight: ~/remote/wooster/bin/raxmlLight-AVX
raxmllight_MPI: ~/remote/wooster/bin/raxmlLight-MPI-AVX
raxmllight_pthreads: ~/remote/wooster/bin/raxmlLight-PTHREADS-AVX
raxmlHPC_pthreads: ~/remote/wooster/bin/raxmlHPC-PTHREADS-SSE3
```

Since most of the job control logic is integrated into the local server rather than the scheduler, the batch script templates are simple and can easily be ported to most batch scheduling systems. The only part that must be ported are the batch directives. An example is given below of the SLURM directives required to run the RAxML-Light component of the framework.

```
#SBATCH -J raxmllight_<%=params[:exp_name_run_num]%>
#SBATCH -d singleton  # ensures that each job with this name
                      # will only run one at a time
#SBATCH -n <%=params[:num_tasks]%>
#SBATCH -p normal
```

```
#SBATCH -o raxmllight_<%=params[:exp_name_run_num]%>.o%j
#SBATCH -e raxmllight_<%=params[:exp_name_run_num]%>.o%j
#SBATCH -t 24:00:00
#SBATCH -A <%=params[:project]%>
```

The local server sets the *params* variables to launch individually configured jobs. The only requirement of the scheduler is that it should be able to handle a simple job dependency in which jobs of the same name are run consecutively. The logic of staging input files and starting or restarting the RAxML-Light component is handled in the body of the batch file and should not change from system to system.

In our TACC setup, at the end of each iteration the best tree is uploaded to the iPlant collaborative tree visualization system, which uses *Phyloviewer* (publicly available at `http://portnoy.iplantcollaborative.org/`) to create a tree visualization on the iPlant server, *Portnoy*, with a link to the latest tree.

The most up-to-date tree from Table 2 is available at: `http://portnoy.iplantcollaborative.org/view/tree/d0c35df3256337f1a8649af09d1469ee`.

and the latest best tree from Table 1 in: `http://portnoy.iplantcollaborative.org/view/tree/10b17429d13160ac1cd07e30bb42fd9b`

3 Evaluation and Results

We tested our framework with simulated and real biological datasets. For each experiment (e.g., different clade or gene), we executed several iterations. For each update iteration, we also executed a control run which we denote as *scratch iteration*. A scratch iteration behaves like an initial iteration, that is, it builds the MSA from scratch on all sequences. It also executes the same number of independent ML tree searches as the update iteration, but *without* using previous topologies. We used the CONSEL package [8] to statistically assess if update and scratch iterations yield topologies with significantly different likelihood scores.

3.1 Biological examples

We have constructed two biologically relevant datasets for testing the perpetual tree server. The first dataset consists of the *rbc*L gene region for the

clade of land plants (*Embryophyta*). The second dataset consists of the 18S ribosomal region for the clade *Eukaryota*. Identity and coverage are used by `PHLAWD` to determine what sequences are similar enough based on Smith-Waterman scores. We used values of 0.5 for *rbc*L and 0.4 for 18S. These numbers were based on plots of identity and coverage using the `seqquery` command in `PHLAWD`. In order to simulate the update procedure on real data, each of these two datasets was created for sequences available before January 2008, before January 2010, and before September 2012. Table 1 and Table 2 show the average ML scores for three update iterations. Each update iteration was run with parameters $p := 3$, $b := 10$, resulting in a total of 30 independent ML searches. We concatenated all resulting trees from *scratch* and *update* iterations based on the same biological real MSA, and used the CONSEL package [8] to asses the confidence of phylogenetic tree selection, that is, which trees were statistically significantly better in terms likelihood scores. The confidence set according to the approximately unbiased (AU) test (p-value $p := 0.05$) included in all cases topologies from both the *scratch* and *update* iterations.

3.2 Simulated Data

We used INDELible [2] to generate a simulated dataset with 9097 taxa on a tree inferred on the *rbc*L gene for *Viridiplantae*. We used the simulated sequences and the underlying true tree as guide tree to asses the iterative MSA update procedure in `PHLAWD`. From the 9079 existing sequences in the simulated alignment, we selected 4000 at random as user sequences to generate the initial `PHLAWD`-based MSA. The remaining sequences were randomly distributed in 3 update blocks of 1345, 1396, and 2329 sequences. Each update block was used a user-sequences to extend the MSA, generating extended MSAs of 5345, 6741, and 9079 taxa. We pruned the true tree accordingly such that for each iteration, a corresponding, true tree was available. We used these pruned true trees to determine the topological accuracy of the inferred trees (at each iteration) using the Robinson-Foulds distance [7], denoted as RF-distance. As before, each iteration included a total of 30 independent ML searches. We observed that starting from extended topologies (update iterations) does neither increase nor decrease topological accuracy (see Table 3).

4 Conclusion and Future Work

We have presented and made available a framework named `PUmPER`, which can be used to maintain and perpetually update phylogenetic trees. `PUmPER` can either operate in stand-alone mode on a single server, but also offload computationally expensive ML searches to an external cluster. The perpetually updated phylogenies can be computed slightly faster and are not significantly (in the statistical sense) worse nor better than phylogenies that are inferred from scratch.

The `PUmPER` framework offers the required flexibility to set up automated perpetual phylogenetic analyses such as, for example, the simultaneous perpetual inference of gene/species trees using multi-gene and single-gene inference.

Future development plans include developing a web-service to facilitate use of the automated update pipeline to a broader community. The framework modules can be easily adapted to fit a Model-View-Controller (MVC) based web application. In this context, the user could customize a perpetually updated phylogenetic analysis (genes, taxonomic group, number of phylogenies) and visualize the resulting trees from a web browser. Finally, we also intend to integrate the recently introduced approximation techniques for computing bootstrap values [6], which have already been integrated into standard RAxML, and will be available in the near future in ExaML.

References

[1] M. W. Chase, D. E. Soltis, R. G. Olmstead, D. Morgan, D. H. Les, B. D. Mishler, M. R. Duvall, R. A. Price, H. G. Hills, Y.-L. Qiu, K. A. Kron, J. H. Rettig, E. Conti, J. D. Palmer, J. R. Manhart, K. J. Sytsma, H. J. Michaels, W. J. Kress, K. G. Karol, W. D. Clark, M. Hedren, B. S. Gaut, R. K. Jansen, K.-J. Kim, C. F. Wimpee, J. F. Smith, G. R. Furnier, S. H. Strauss, Q.-Y. Xiang, G. M. Plunkett, P. S. Soltis, S. M. Swensen, S. E. Williams, P. A. Gadek, C. J. Quinn, L. E. Eguiarte, E. Golenberg, J. Learn, Gerald H., S. W. Graham, S. C. H. Barrett, S. Dayanandan, and V. A. Albert. Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcl. *Annals of the Missouri Botanical Garden*, 80(3):pp. 528–548+550–580, 1993.

[2] W. Fletcher and Z. Yang. INDELible: a flexible simulator of biological sequence evolution. *Molecular biology and evolution*, 26(8):1879–1888, August 2009.

[3] S. Guindon, J. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Systematic biology*, 59(3):307–321, 2010.

[4] F. Izquierdo-Carrasco, J. Gagneur, and A. Stamatakis. Trading running time for memory in phylogenetic likelihood computations. In *In Proceedings of Bioinformatics 2012*, 2012.

[5] F. Izquierdo-Carrasco, S. Smith, and A. Stamatakis. Algorithms, data structures, and numerics for likelihood-based phylogenetic inference of huge trees. *BMC Bioinformatics*, 12(1):470, 2011.

[6] B. Q. Minh, M. A. T. Nguyen, and A. von Haeseler. Ultrafast approximation for phylogenetic bootstrap. *Molecular biology and evolution*, 2013.

[7] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(12):131 – 147, 1981.

[8] H. Shimodaira and M. Hasegawa. Consel: for assessing the confidence of phylogenetic tree selection. *Bioinformatics*, 17(12):1246–1247, 2001.

[9] S. A. Smith, J. M. Beaulieu, and M. J. Donoghue. Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. *BMC Evolutionary Biology*, 9(37), 2009.

[10] A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.

[11] A. Stamatakis and A. Aberer. Novel parallelization schemes for large-scale likelihood-based phylogenetic inference. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 1195–1204, 2013.

[12] A. Stamatakis, A. Aberer, C. Goll, S. Smith, S. Berger, and F. Izquierdo-Carrasco. Raxml-light: a tool for computing terabyte phylogenies. *Bioinformatics*, 28(15):2064–2066, 2012.

[13] Z. Yang. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites. *J. Mol. Evol.*, 39:306–314, 1994.

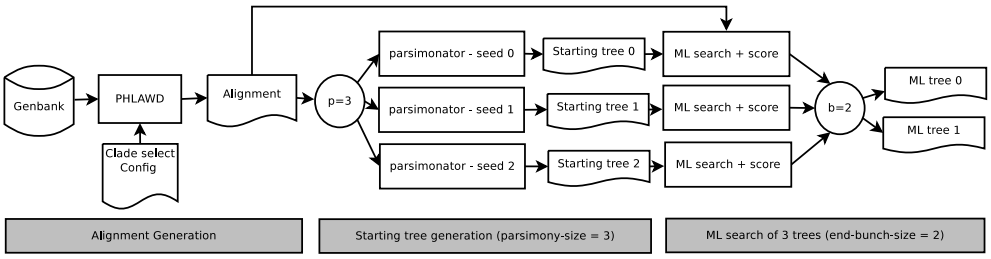

Figure 1: Initial iteration: An initial alignment is built for sequences covering a given clade and description search term (gene). Parsimony starting trees are used for Maximum likelihood searches. The best trees are collected.

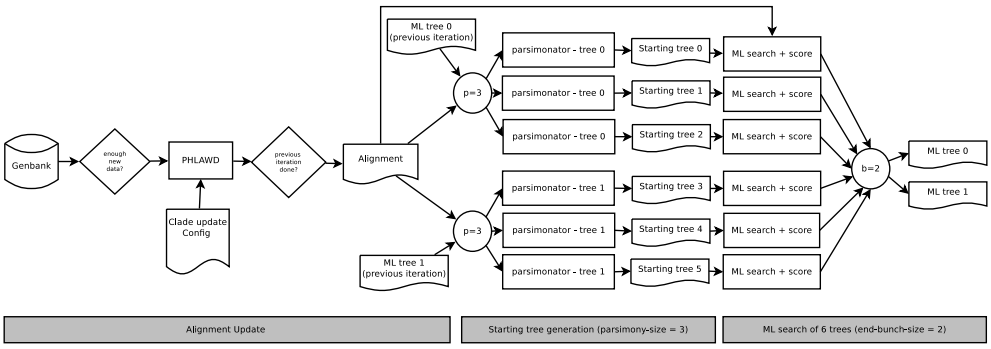

Figure 2: Update iteration: The alignment is extended and previous trees are re-used to continue searching in ML space with a set of different starting trees.

Iteration	Taxa	Sites	Avg LH (30)	Avg LH (10)	Runtime(h)	Avg Branch support
2008	12072	1437	-848794.80	-848745.23	46.55	67.78
2010	16962	1427	-1005824.25	-1005762.81	68.36	64.25
2010 scratch	16962	1427	-1005931.37	-1005863.32	70.89	64.26
2012(Sept)	21791	1424	-1108161.66	-1107598.42	93.40	59.56
2012(Sept)scratch	21791	1424	-1108243.29	-1107774.80	97.42	59.46

Table 1: Original run and two updates of the *rbc*L datasets. Average ML score at the end of each iteration (averaged over all 30 trees and the 10 best trees) and overall run time of all searches. The branch support is the average of SH-like support values on the best tree. The running time is the sum of the 30 ML searches.

Iteration	Taxa	Sites	Avg LH (30)	Avg LH (10)	Runtime(h)	Avg Branch support
2008	14634	2363	-1950468.79	-1950281.56	74.16	70.24
2010	18480	2214	-2340314.06	-2340142.38	88.73	68.73
2010 scratch	18480	2214	-2340563.92	-2340260.37	94.08	68.59
2012(Sept)	23298	2110	-2782234.35	-2781965.38	116.48	68.13
2012(Sept)scratch	23298	2110	-2782132.94	-2781959.11	124.23	68.00

Table 2: Original run and two updates of the 18S datasets. Average ML score at the end of each iteration (averaged over all 30 trees and the 10 best trees) and overall run time of all searches. The branch support is the average of SH-like support values on the best tree. The running time is the sum of the 30 ML searches.

Iteration	Taxa	Sites	Avg LH (30)	Avg LH (10)	RF (true tree)	Runtime(h)	Avg support
0	4000	1500	-589036.97	-589035.58	0.146	2.92	77.63
1	5345	1500	-715683.68	-715682.66	0.163	7.46	76.14
1 scratch	5345	1500	-715682.75	-715681.41	0.162	7.89	76.16
2	6741	1500	-838437.33	-838436.48	0.176	6.21	74.72
2 scratch	6741	1500	-838440.38	-838438.06	0.174	8.17	74.73
3	9079	1500	-1033499.23	-1033498.23	0.184	10.10	73.12
3 scratch	9079	1500	-1033498.15	-1033495.85	0.185	19.27	73.12

Table 3: Original run and three simulated updates of the simulated datasets. Average Likelihood the end of each iteration (30 total trees and 10 best trees) and total run time of all searches. The branch support are the average of SH-like support values of the best tree. The running time is the sum of the 30 maximum likelihood searches.