Syst. Biol. 57(5):758–771, 2008

ISSN: 1063-5157 print / 1076-836X online
DOI: 10.1080/10635150802429642

A Rapid Bootstrap Algorithm for the RAxML Web Servers

ALEXANDROS STAMATAKIS,[1] PAUL HOOVER,[2] AND JACQUES ROUGEMONT[3]

[1]*The Exelixis Lab, Teaching and Research Unit Bioinformatics, Department of Computer Science, Ludwig-Maximilians-University Munich, Amalienstr. 17, D-80333, Munich, Germany; E-mail: stamatakis@bio.ifi.lmu.de*
[2]*San Diego Supercomputer Center, La Jolla, California, USA*
[3]*School of Life Sciences, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*

Abstract.—Despite recent advances achieved by application of high-performance computing methods and novel algorithmic techniques to maximum likelihood (ML)-based inference programs, the major computational bottleneck still consists in the computation of bootstrap support values. Conducting a probably insufficient number of 100 bootstrap (BS) analyses with current ML programs on large datasets—either with respect to the number of taxa or base pairs—can easily require a month of run time. Therefore, we have developed, implemented, and thoroughly tested rapid bootstrap heuristics in RAxML (Randomized Axelerated Maximum Likelihood) that are more than an order of magnitude faster than current algorithms. These new heuristics can contribute to resolving the computational bottleneck and improve current methodology in phylogenetic analyses. Computational experiments to assess the performance and relative accuracy of these heuristics were conducted on 22 diverse DNA and AA (amino acid), single gene as well as multigene, real-world alignments containing 125 up to 7764 sequences. The standard BS (SBS) and rapid BS (RBS) values drawn on the best-scoring ML tree are highly correlated and show almost identical average support values. The weighted RF (Robinson-Foulds) distance between SBS- and RBS-based consensus trees was smaller than 6% in all cases (average 4%). More importantly, RBS inferences are between 8 and 20 times faster (average 14.73) than SBS analyses with RAxML and between 18 and 495 times faster than BS analyses with competing programs, such as PHYML or GARLI. Moreover, this performance improvement increases with alignment size. Finally, we have set up two freely accessible Web servers for this significantly improved version of RAxML that provide access to the 200-CPU cluster of the Vital-IT unit at the Swiss Institute of Bioinformatics and the 128-CPU cluster of the CIPRES project at the San Diego Supercomputer Center. These Web servers offer the possibility to conduct large-scale phylogenetic inferences to a large part of the community that does not have access to, or the expertise to use, high-performance computing resources. [Maximum likelihood; phylogenetic inference; rapid bootstrap; RAxML; support values.]

Phylogenetic trees are used to represent the evolutionary history of a set of n organisms. An alignment of DNA or protein sequences that represent these n organisms can be used as input for phylogenetic inference. In a phylogeny the organisms of the input dataset are located at the tips (leaves) of the tree and the inner nodes represent extinct common ancestors. Due to the rapid growth of sequence data over the last years, which is further accelerated by new sequencing techniques, there exists an increasing demand to compute large trees, which often comprise more than 1000 organisms and/or sequence data from several genes (so-called multigene alignments). Because alignments are becoming ever larger in the number of organisms *and* in sequence length, there is an increasing need for efficient phylogeny programs. The fundamental algorithmic problem lies in the immense number of alternative tree topologies, which grows exponentially with the number of organisms n; e.g., for $n = 50$ there exist 2.84×10^{76} alternative trees. It has recently been shown that the ML phylogeny problem is NP-hard (Chor and Tuller, 2005).

In addition to the algorithmic problem, ML-based inference of phylogenies (Felsenstein, 1981) is memory and floating point intensive. Thus, the application of high-performance computing techniques can significantly contribute to the reconstruction of larger trees (see Stamatakis et al., 2005b; Charalambous et al., 2005; Blagojevic et al., 2007; Ott et al., 2007). More emphasis needs to be placed on the so-called "memory gap" problem; i.e., the fact that memory access speeds have been growing at a significantly lower rate than CPU speeds over the last 20 years (see Wilkes, 2001 for a summary). This means that execution times of the ML function for alignments with increasing length will not scale linearly due to a larger average number of cache misses per instruction. Despite the high computational cost, significant progress has been achieved over the last few years in the field of heuristic ML search algorithms with the release of several new programs. Nonetheless, the main computational burden consists in computing BS support values on trees, which can require more than 1 month of sequential execution time for a probably insufficient number of 100 replicates on a reasonably fast CPU.

There has been a long-lasting controversial discussion on support value interpretation and the differences between Bayesian and BS methods as well as their interpretation, which is summarized; e.g., in the introduction of Anisimova and Gascuel (2006). Nonetheless, the computation of BS values (Felsenstein, 1985) will remain a standard technique in the foreseeable future and represents a useful way to assess the topological stability of trees under slight alterations of the input data. From a computational point of view, one might argue in favor of topology-based resampling strategies because they better account for the vastness of the topological search space. Moreover, it has become common practice in recent phylogenetic analyses to use and compare the results and support values obtained via multiple methods (maximum parsimony, ML, Bayesian). Therefore, rather than taking the debate on support metrics further, the focus of this article is on the development and experimental assessment of novel rapid bootstrap (RBS) heuristics in RAxML that

yield execution time improvements of more than one order of magnitude while returning BS support values that are qualitatively comparable to those obtained via the standard, well-established search algorithms. The current version of RAxML (Randomized Axelerated Maximum Likelihood, version 7.0.4; Stamatakis, 2006b) is a tool for large-scale ML-based inference of evolutionary trees for DNA or AA (amino acid) data under plain or partitioned (sometimes referred to as mixed) models. The code used to conduct computational experiments for this article is an earlier, pre-release version.

Algorithms and Web Servers

The main design goal was to implement a "quick and dirty" bootstrap procedure (RBS) that runs significantly faster than standard bootstrapping (SBS), while returning highly correlated support values and comparable average bipartition frequencies. In order to achieve this, an algorithmic engineering approach (Moret, 2002) was adopted that consisted of several iterations through search algorithm modifications and large-scale computational experiments on a broad and diverse range of real datasets. This approach is well-suited for the design of ML-based search heuristics, because they prove to be hard to analyze using theoretical tools due to the complexity of the non-discrete optimality criterion underlying this NP-complete problem. As an example, consider the fact that NP hardness of ML could be demonstrated only as recently as 2005 (Chor and Tuller, 2005), nearly 25 years after the introduction of the model.

The rationale for using real-world data to steer algorithmic design and assess performance is that state-of-the-art simulated data generation tools are unable to reproduce algorithmic behavior on real data. Trees for perfect simulated data tend to be "easier" to infer than trees for real-world data. To this end, we have made an effort to assemble a set of alignments that covers a broad range of organisms and dataset sizes. In addition to the RBS algorithm, we also implemented an improved ML search algorithm (compared to standard ML searches in RAxML), which is executed after the RBS search to infer an ML tree on the original alignment; i.e., to conduct a *full* ML analysis, including BS and ML searches in a single program run. The main reason for this was to facilitate the design of the Web servers that thus need to invoke the program only once for each individual job submission. In addition, this implementation allows for future exploration of ML search algorithms that use information gathered during the BS analysis to steer a more efficient exploitation of search space. Therefore, the ML search on the original alignment is conducted *after* the RBS search.

Likelihood Cutoff Heuristics

We briefly describe recently introduced (Stamatakis et al., 2007) heuristics in RAxML that are required as a prerequisite to outline the rapid BS and ML search algorithms. The standard RAxML SBS searches to which we refer throughout this article already include these heuristics. The basic concept of the likelihood cutoff heuristics is outlined in Figure 1.

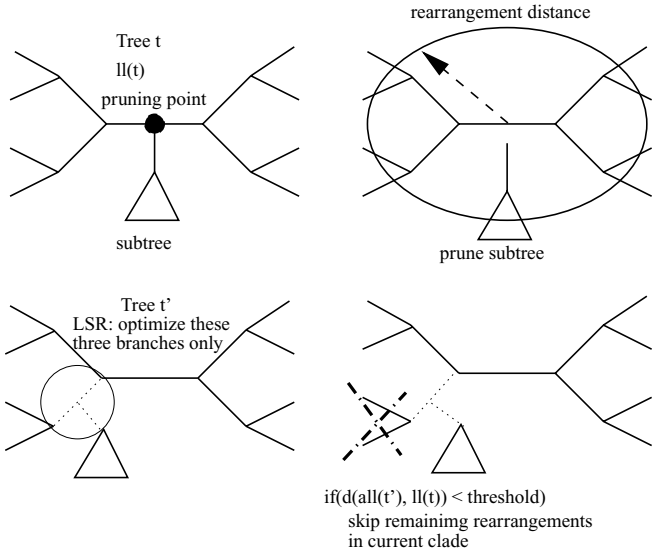

FIGURE 1. Outline of lazy subtree rearrangements with cutoff procedure.

The fundamental mechanism that is used to search the tree space with RAxML is called Lazy Subtree Rearrangement (LSR; for details see Stamatakis et al., 2005a). Similar mechanisms are used in GARLI (Zwickl, 2006) and have been proposed for PHYML (Hordijk and Gascuel, 2005). Here we describe an extension of the specific LSR search mechanism implemented in RAxML. An LSR consists in pruning/removing a subtree from the currently best tree t and subsequently reinserting it into all neighboring branches up to a certain distance/radius (rearrangement distance) of n nodes from the pruning point (n typically ranges from 5 to 25). For each possible subtree insertion within the rearrangement distance, RAxML computes an approximate log likelihood score of the alternative topology. This computation is performed in a lazy fashion because only the length of the three branches adjacent to the insertion point/node will be optimized. Thus, an LSR only yields an approximate log likelihood $all(t')$ score for each alternative topology t' constructed by applying an LSR to t. This approximate $all(t')$ score can be used to pre-score and sort the potential alternative topologies accordingly. After this fast pre-scoring of a large number of alternative topologies, only a small fraction of the best-scoring topologies needs to be optimized more exhaustively to improve the overall tree score. One iteration of the RAxML hill-climbing algorithm consists in performing LSRs on all subtrees for a given topology t and a fixed rearrangement distance n. Thereafter, the branch lengths of the 20 best-scoring trees are thoroughly optimized. This procedure of conducting LSRs on all subtrees and then optimizing the 20 best-scoring trees is performed until no further

likelihood improvement can be achieved by application of LSRs (see Stamatakis et al., 2005a, for a more detailed description).

The main idea of the recently introduced heuristics (Stamatakis et al., 2007) consists in reducing the number of LSRs that are performed per subtree. This is achieved by deploying an empirical cutoff rule that stops the recursive descent of an LSR into deeper branches at a higher rearrangement distance n from the pruning position, if they do not appear to be promising. Thus, if the approximate log likelihood $all(t')$ for the rearranged tree t' is worse than the log likelihood $ll(t)$ of the currently best tree t *and* if the difference $\delta(all(t'), ll(t))$, where $\delta(x, y) = x - y$, is larger than a certain—dynamically determined—threshold lh_{cutoff}, the remaining LSRs beyond that node are omitted. The threshold lh_{cutoff} is determined as follows: during the first iteration of the RAxML search algorithm $lh_{\text{cutoff}} = \infty$, which means that no cutoffs are performed. In the course of this first iteration, the differences $\delta_i(all(t_i), ll(t))$ for all $i = 1 \ldots m$ alternative tree topologies t_i where $all(t_i) \leq ll(t)$ are stored. The threshold lh_{cutoff} for the next iteration is then set to the average of δ_i, i.e., $lh_{\text{cutoff}} = (\sum_{i=1}^{m} \delta_i)/m$. If the search computes an LSR for which $all(t') \leq ll(t)$ and $\delta(all(t'), ll(t)) \geq lh_{\text{cutoff}}$, it will omit the remaining LSRs below the current node (see Fig. 1). Thus, each iteration k of the search algorithm uses a threshold value lh_{cutoff} that has been obtained during the previous iteration $k - 1$. This allows for dynamic adaptation of lh_{cutoff} to the specific dataset and to the progress of the search. The omission of a large number of unnecessary LSRs, which will most probably not improve the tree, yields substantial run time improvements (average 2.5) but returns equally good trees. For a detailed performance analysis refer to Stamatakis et al. (2007).

Rapid Bootstrap Search

The rapid bootstrap search starts with the computation of a stepwise random addition order MP starting tree on the original alignment (Fig. 2). Thereafter, the model parameters and branch lengths are optimized on this starting tree for the original alignment. Maximum likelihood model parameters will not be reoptimized for any of the RBS replicates after this initial optimization on the original alignment. To accommodate rate heterogeneity among sites, we used the GTR+CAT approximation, which represents an efficient computational work-around for the GTR+Γ model (Stamatakis, 2006a). The GTR+CAT method is denoted as *approximation* because although it provides a "quick and dirty" way to accommodate rate heterogeneity during the tree search, the log likelihood values calculated for trees must not be used for likelihood-based comparisons of topologies. This means that GTR+CAT is just used as a means to rapidly navigate into an area of the topological search space where trees score well under GTR+Γ. RAxML does not offer the usage of the plain GTR model without rate heterogeneity under any search algorithm, because the vast majority of current-day phylogenetic analyses use models of rate heterogeneity (Ripplinger and Sullivan, 2008).

We exclusively use the GTR+CAT approximation of rate heterogeneity for RBS for two reasons: *Firstly*, the computational advantages with respect to memory footprint and execution times (fourfold reduction in memory consumption, three to four times lower execution times; Stamatakis , 2006a), and *secondly*, because the rapid bootstrapping procedure, which does not conduct a per replicate model parameter optimization, is less sensitive to GTR+CAT as opposed to GTR+Γ, because every alignment pattern is assigned 1 of the 25 default rate categories that can be drawn along with the columns from the original alignment for each RBS replicate.

Once the initial tree and optimized model parameters on the original alignment have been computed, the RBS method conducts searches on BS replicates as follows: every 10th BS replicate $r_0, r_{10}, r_{20}, \ldots$ is seeded with a new stepwise random addition order MP starting tree that is computed on the original alignment to reduce the risk of navigating into local optima. This means that at every 10th replicate, the program reloads the original alignment to compute a randomized stepwise addition MP starting tree. Once this starting tree has been computed, the program calculates and loads a bootstrapped alignment, which is then optimized under ML. The searches for the remaining replicates $r_1 - r_9, r_{11} - r_{19}, \ldots$ start on the *final trees* of the respective preceding BS replicates; i.e., the search on r_1 starts on the final tree of r_0, the search for r_2 on the final tree of r_1, etc. (r_i refer to individual RBS samples). In the context of large-scale phylogenetic analyses, one drawback of this algorithm compared to SBS consists in the slightly limited degree of parallelism due to sequential dependencies from, e.g., $r_0 \rightarrow r_1 \rightarrow \ldots \rightarrow r_9$; i.e., 100 replicates can be computed in parallel on 10 CPUs.

In the following, we describe the actual "quick and dirty" per replicate search in more detail. In contrast to the standard RAxML search procedure, where the rearrangement radius/setting is determined automatically (see the on-line supplement of Stamatakis, 2006b, for details), we use a randomly assigned radius between 5 up to and including 15. The rationale for not using higher settings is that 90% of the searches start on the final tree of the previous search using a distinct alignment pattern composition, such that it becomes unlikely that all 10 replicates computed on one starting tree will get trapped in local optima. The scalability of our approach to datasets with several thousand sequences (see Tables 1 and 2) without a decrease in correlation values justifies this approach. In addition, we have set the upper limit k of iterative LSR applications (LSR for all subtrees of the current tree) for each replicate to 2 and use a more strict likelihood cutoff value $lh_{\text{cutoff}} = 0.5 \times (\sum_{i=1}^{m} \delta_i)/m$ that skips a larger number of LSR descents into subtrees. Finally, at the end of each LSR cycle, we only evaluate the five most promising pre-scored trees as opposed to 20 during the standard searches. The above modifications yield a total run time improvement by a factor of approximately 15 for bootstrapping.

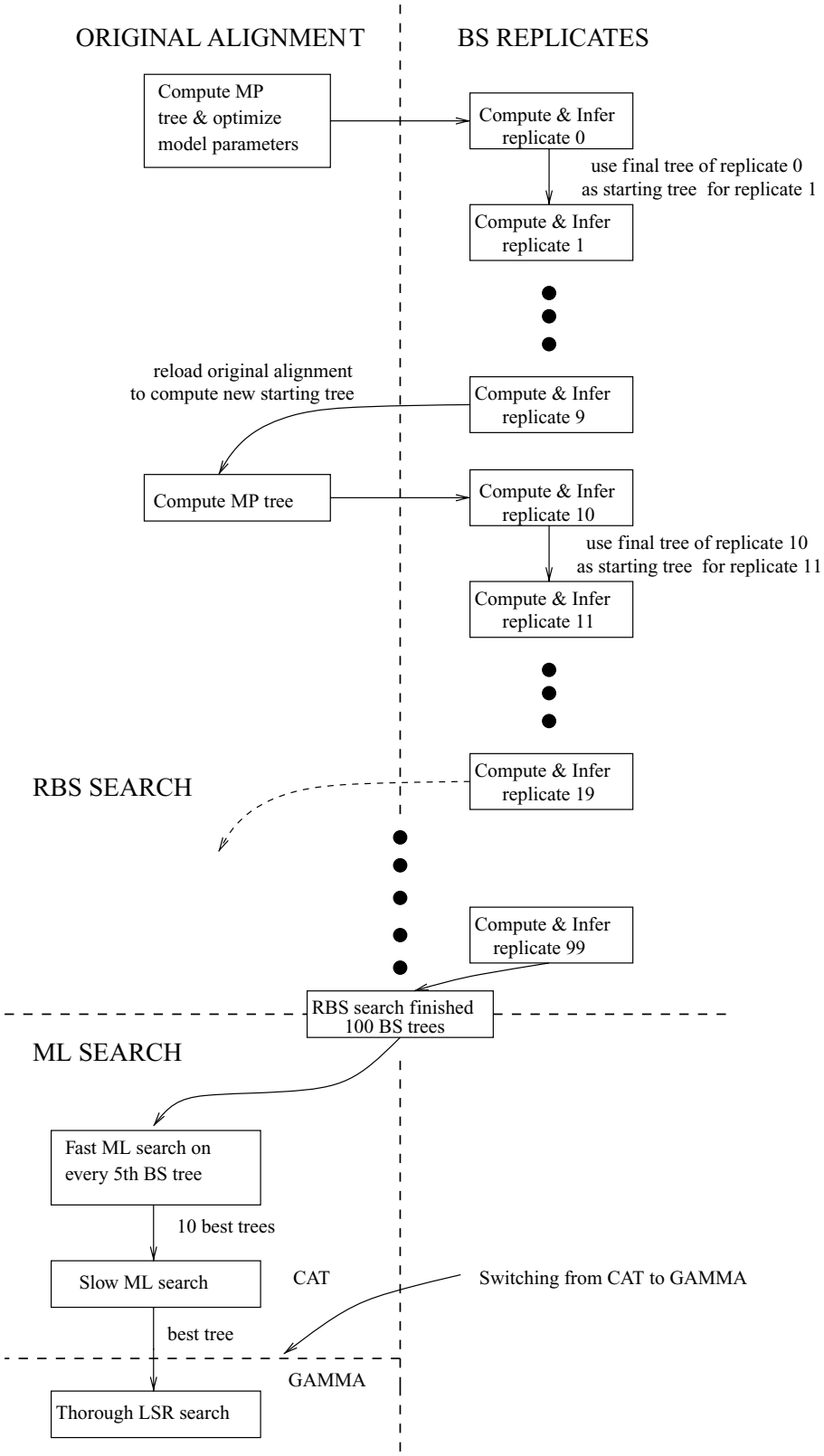

FIGURE 2. Outline of the RBS (rapid bootstrap) search algorithm and the successive ML search procedure.

TABLE 1. Experimental data used and execution time comparison between RBS (rapid bootstrap) and SBS (standard bootstrap). Column *#SEQS* indicates the number of sequences, *#PATT* the number of distinct patterns in the alignment that correspond to the length of the likelihood vectors and compute-intensive for-loops, *% Gaps* the percentage of completely undetermined character states (e.g., N,O,X,?,- for DNA data) in an ML context, *SBS(hrs)* indicates the execution time of SBS in hours for 100 BS replicates, *RBS(hrs)* the execution time for 100 RBS searches, and *Speedup* the acceleration achieved by RBS.

# SEQS	# PATT	% Gaps	SBS (hrs)	RBS (hrs)	Speedup
d125	19,436	32.72	128.45	10.52	12.21
d140_AA	1,041	0.60	51.80	5.17	10.02
d140_AA_P	1,057	0.60	63.55	5.34	11.89
d150	1,130	4.77	5.31	0.37	14.46
d218	1,846	35.33	18.33	1.18	15.49
d354	348	14.71	4.45	0.30	14.63
d404	7,429	78.92	236.10	16.91	13.96
d404_P	7,444	78.92	259.23	24.08	10.77
d500	1,193	2.48	31.09	1.86	16.72
d628	1,033	36.44	26.47	1.88	14.11
d714	1,231	5.83	48.32	2.86	16.89
d775_AA	3,838	19.35	2673.74	332.67	8.04
d994	3,363	71.39	255.25	14.72	17.34
d1288	1,132	7.53	218.06	14.63	14.91
d1481	1,241	26.58	137.28	9.09	15.10
d1512	1,576	3.02	198.44	13.43	14.77
d1604	1,275	5.71	159.23	8.61	18.48
d1908	1,209	58.38	224.72	12.05	18.64
d2000	1,251	12.98	422.23	21.02	20.08
d2308	1,184	12.71	379.01	28.68	13.21
d2554	1,232	5.81	386.04	29.39	13.13
d4114	1,263	2.00	583.58	39.09	14.93
d6718	1,122	20.87	1235.75	76.02	16.26
d7764	851	20.60	1273.77	72.90	17.47
Averages	2,655	23.26	375.84	30.95	14.73

Rapid ML Search

The accelerated ML search procedure, which is executed after the RBS search, was designed using an analogous algorithmic engineering approach. The ML search is executed *after* the RBS inference because the ML search can be accelerated by using tree topologies inferred during the RBS phase (see below). Thus, once the RBS analyses are finished, the program reloads the original alignment to conduct an ML search.

Initially, every fifth final RBS tree is used as a starting tree for a fast ML search (always on the original alignment); i.e., if 100 RBS replicates have been computed, 20 fast ML searches will be conducted under GTR+CAT. After this initial fast search, all 20 final trees are scored under GTR+Γ and a more thorough search is applied to the best-scoring 10 trees. This more thorough search is once again conducted under GTR+CAT. The final 10 trees of these searches are then scored under GTR+Γ again and the best-scoring tree undergoes a final and more thorough LSR-based optimization under GTR+Γ. The thoroughness is due to the usage of a less lazy LSR mechanism, where a region of branches up to four nodes beyond the insertion point of a subtree is reoptimized. The above modifications yield an average speedup of factor 2.2 compared to 20 standard RAxML searches for the best-scoring tree.

The RAxML Web Servers

In order to thoroughly test the functionality of the RAxML Web servers prior to submission of the manuscript, the availability of the prototype at the Vital-IT unit of the Swiss Institute of Bioinformatics was announced at the end of August 2007 via Evoldir, the RAxML mailing list, and was also posted on the iPhylo blog of Roderic Page (http://iphylo.blogspot.com/). From September 3, 2007, to May 18, 2008, 7080 jobs have been submitted from 766 distinct IP addresses. During this period, we fixed several minor bugs and integrated additional features with the help of the user community. The main goal of the Web servers is to keep things as simple as possible; i.e., provide the capability to infer trees for nonexpert users, hence the name RAxML Black Box. On the Vital-IT job submission page (see http://phylobench.vital-it.ch/raxml-bb/), one can select between AA- and DNA-based inference, partitioned models with joint and per partition optimization of branch lengths, usage of a proportion of invariant sites estimate, and execution of RBS or a combined RBS and rapid ML search. Analyses can also be conducted with multifurcating constraint trees or bifurcating backbone trees (for details see RAxML manual). The server, which is attached to a 200-CPU cluster located at the Vital-IT unit of the Swiss Institute of Bioinformatics, will return the following result files:

- A file containing all RBS trees.
- An extended majority rule RBS consensus tree.
- The `RAxML_info.RUN_ID` file containing information on model parameter estimates and execution times.
- The best-scoring ML tree (if ML inference was selected).
- The best-scoring ML tree with RBS values (if ML inference was selected).
- Copies of the best-scoring ML tree with individual branch lengths for each partition (if ML inference and per partition branch length optimization were selected).

The consensus and best-scoring ML trees can be viewed and browsed online by using a customized version of the PHY.FI display engine, http://cgi-www.daimi.au.dk/cgi-chili/phyfi/go (Fredslund, 2006). The CIPRES RAxML Web server has been set up in an analogous way (see http://8ball.sdsc.edu:8889/cipres-web/Bootstrap.do).

Availability: http://icwww.epfl.ch/~stamatak/index-Dateien/software/RAxML-VI-HPC-4.0.0.tar.gz
Web Servers: http://phylobench.vital-it.ch/raxml-bb/ http://8ball.sdsc.edu:8889/cipres-web/Bootstrap.do

RESULTS

To test the performance and *relative* accuracy of RBS with respect to SBS as well as to BS values obtained via competing programs, we used 22 real-world AA and

TABLE 2. BS support value comparison RBS versus SBS. Column *#SEQS* indicates the dataset, *CorrBest* denotes the Pearson correlation coefficient ρ on the best-scoring ML tree, *Slope* the slope of the linear regression function, *Intercept* the intercept point of the linear regression function (the worst possible intercept would be ± 100), *CorrAll* the Pearson correlation between all bipartitions/splits induced by the BS replicates, *Slope* the slope of the linear regression function for all splits (the intercept was omitted because it ranged only between -0.013 and $+0.001$), *SBS(Bips)* and *RBS(Bips)* the total number of bipartitions detected by SBS and RBS, respectively, *RF* the relative RF-distance (Robinson-Foulds distance) between extended majority rule consensus trees for RBS, and SBS, and finally *WRF* the analogous weighted RF distance.

# SEQS	CorrBest	Slope	Intercept	CorrAll	Slope	SBS(Bips)	FBS(Bips)	RF	WRF
d125	0.917	1.06	−6.82	0.988	1.02	164	156	0.02	0.03
d140_AA	0.990	0.99	−0.18	0.997	1.01	425	387	0.06	0.03
d140_AA_P	0.985	1.05	−5.16	0.996	1.00	435	417	0.07	0.03
d150	0.984	0.99	−1.02	0.989	1.02	1,785	1,419	0.13	0.06
d218	0.978	0.99	−3.23	0.981	1.03	3,427	2,555	0.17	0.05
d354	0.976	0.97	−1.55	0.972	1.02	8,811	7,781	0.23	0.05
d404	0.958	0.96	2.67	0.973	0.98	5,394	5,245	0.16	0.04
d404_P	0.967	0.97	1.01	0.977	0.99	5,480	5,501	0.14	0.04
d500	0.975	0.99	−2.38	0.988	1.02	5,751	4,637	0.10	0.04
d628	0.963	0.96	0.23	0.977	0.99	8,405	7,616	0.15	0.05
d714	0.972	0.99	−2.76	0.985	1.02	7,593	6,484	0.11	0.04
d775_AA	0.978	0.97	2.36	0.993	1.01	3,806	3,506	0.07	0.03
d994	0.972	1.03	−4.75	0.988	1.02	8,352	6,648	0.08	0.03
d1288	0.964	0.99	−2.69	0.983	1.02	15,150	12,141	0.11	0.03
d1481	0.969	0.95	1.77	0.974	0.99	27,882	27,122	0.20	0.04
d1512	0.985	0.99	−1.66	0.987	1.01	25,277	22,783	0.11	0.02
d1604	0.973	0.99	−1.79	0.981	1.00	22,320	21,027	0.13	0.03
d1908	0.982	0.99	−1.64	0.987	1.01	26,835	24,333	0.12	0.03
d2000	0.980	0.99	−1.95	0.981	1.02	39,891	33,965	0.13	0.03
d2308	0.979	1.00	−1.73	0.992	1.01	17,252	14,945	0.07	0.02
d2554	0.980	0.99	−2.04	0.987	1.02	34,555	30,139	0.09	0.02
d4114	0.977	0.98	−1.71	0.975	1.00	88,766	74,802	0.18	0.04
d6718	0.975	0.96	0.10	0.972	0.99	170,176	155,342	0.19	0.03
d7764	0.977	0.96	−0.021	0.966	0.99	248,130	229,290	0.22	0.03
Averages	0.972	0.99	2.33	0.983	1.008	32,335	29,093	0.13	0.04

DNA alignments containing 125 up to 7764 sequences. Although analyses of datasets with more than 500 taxa are still relatively uncommon, from an algorithmic point of view, it is important to demonstrate the scalability of both RBS accuracy as well as execution times with respect to the number of taxa. Eight out of 173 jobs submitted to the Vital-IT Web server in February 2008 contained more than 500 taxa, with the largest dataset comprising 5319 sequences (the average number of sequences was 125, whereas the average number of sequences per alignment out of 400 jobs on the CIPRES Web server amounted to 134). The number of unique alignment site patterns in our test data ranges from 348 to 19,436 and the percentage of completely undetermined characters (`N,O,X,?,-` for DNA data and `X,?,-,*` for protein data) from 0.6% to 79%. The number of alignment columns has deliberately been omitted, because it is irrelevant with respect to the computational cost and does not represent an important measure of alignment size, especially with respect to memory consumption. The taxon names in the alignments have been anonymized to protect unpublished data. We removed all duplicate sequences as well as all entirely undetermined columns from the alignments. For the sake of simplicity, alignments will henceforth be referenced by the number of taxa, preceded by a "d" as provided in the first column of Table 1. The experimental data span a broad range of organisms including rbcL genes (d500, d2554), mammalian sequences (d125, d1288, d2308), bacterial and archaeal sequences (d714, d994, d1481, d1512, d1604, d2000, d4114, d6718, d7764), ITS sequences (d354), fungal sequences (d628, d1908), grasses (d404), as well as AA sequences (denoted as _AA) of papillomaviruses (d140) and fishes (d775). For two multigene alignments (d140 and d404), we also assessed RBS performance under a partitioned model with joint branch length estimate (denoted as _P).

Computational experiments were conducted on the CIPRES project (http://www.phylo.org) cluster at the San Diego Supercomputer Center and the Infiniband cluster (http://www.lrr.in.tum.de/Par/arch/infiniband/) at the Technische Universität München, which are equipped with 16 eight-way SMP (symmetric multiprocessing) nodes (128 CPUs total) and 36 four-way SMP nodes (144 CPUs total), respectively. Both systems are based on AMD 2.4-GHz Opteron processors. All result files, scripts, and datasets used in this study are available for download at http://icwww.epfl.ch/~stamatak/RAPID-RESULTS.tar.bz2.

Computational experiments were performed as follows: for each dataset we conducted 100 sequential SBS runs with RAxML and a fixed BS random number seed via `-b 12345`. Thereafter, we conducted RBS runs on the same 100 BS replicates with `-x 12345`. Although not required, these random number seeds were kept constant across datasets (not only across RBS/SBS runs for the same dataset) to simplify the scripts as well as to facilitate reproducibility. The only exceptions are datasets d4114, d6718, d7764, and d775_AA where SBS was executed using the parallel version of RAxML due to the

long run times. In these four cases SBS trees were inferred for different BS replicates than the respective RBS trees.

All BS inferences were conducted under the GTR+CAT (DNA) and WAG+CAT (AA) approximation of rate heterogeneity, which represents an efficient computational work-around for GTR+Γ (Stamatakis, 2006a, 2006b). In addition, we executed 20 independent ML searches on 20 randomized stepwise addition maximum parsimony trees under GTR+CAT using the standard RAxML search algorithm. Final trees were scored under GTR+Γ; e.g., `raxmlHPC -m GTRMIX -s 500 -# 20 -n 500.ML` (for details refer to the RAxML Manual at http://icwww.epfl.ch/ stamatak). The best-scoring tree out of these 20 standard ML searches was determined by analyzing the result files.

ML searches in conjunction with the RBS algorithm were conducted in a single run with RBS; e.g., `raxmlHPC -f a -x 12345 -p 12345 -m GTRCAT -s 500 -# 100 -n 500.RAPID`. The option `-p` provides a random number seed for the—in this case 100/10 = 10—MP starting trees, such that our RBS experiments can be fully reconstructed. The RBS algorithm directly returns the best ML tree (scored under GTR+Γ) on the original alignment in a file called, e.g., `RAxML_bestTree.500.RAPID`. Finally, we conducted additional experiments on selected datasets to assess the effects of using, e.g., GTR+CAT instead of GTR+Γ, the likelihood cutoff procedure, and distinct sets of BS replicates. These experiments show that no bias is introduced by the plethora of approximations used in RAxML.

Result analysis.—The resulting BS trees from the RBS and SBS analyses were compared as follows: we mapped the bipartition support values (using the respective RAxML option `-f b`) to the best-scoring ML tree found by RAxML either during the 20 standard ML searches or via the rapid ML search after the RBS phase; i.e., the best out of these 21 trees. We then computed the Pearson correlation coefficient ρ between corresponding pairs of RBS and SBS support values on the best-scoring ML tree. In addition, we computed the slope and intercept of the respective linear regression function. We also extracted the RBS- and SBS-based frequencies for *all bipartitions inferred* either by RBS or SBS (using the RAxML option `-f m` and passing the SBS and RBS BS tree files as parameters) and computed the Pearson correlation coefficient of the resulting bipartition frequency pairs. In addition, we counted the total number of distinct bipartitions induced by the SBS and RBS replicates. Finally, we computed the extended majority-rule consensus tree (a bifurcating tree) for SBS and RBS replicates using `consense` from the PHYLIP package. The relative Robinson-Foulds (RF, Robinson and Foulds, 1981) distance between the consensus trees was computed with `treedist` from PHYLIP and the weighted relative RF distance (WRF; Robinson and Foulds, 1979) with `partitionMetric.pl` by Olaf Bininda-Emonds (available at http://www.personal.uni-jena.de/b6biol2/ProgramsMain.html; see also Hillis et al., 2005, for an application of RF and WRF distances).

The standard, unweighted RF distance between two topologies counts the number of clades found in one tree or the other but not in both of them. Each unique clade counts one in the unweighted case. The weighted RF distance uses the support value of the clade instead; i.e., it uses the complete information provided by a majority rule consensus tree. Thus, a clade with a bootstrap value of 0.4 counts 0.4 instead of 1. Therefore, poorly supported clades do not increase the RF as much as better supported ones. This weighted topological distance measure between extended majority-rule consensus trees takes into account the support values and penalizes incongruent subtrees with low support to a lesser extent. When RF distances, as in our experiments (see columns *RF* and *WRF* in Table 2), are significantly larger (approximately a factor 3 on average) than their weighted counterparts, this indicates that the differences in the consensus trees are induced by subtrees with low support. An extreme example for this phenomenon is d354, a relatively short and hard to analyze ITS dataset (Grimm et al., 2006), where the weighted RF distance is almost five times lower than the respective plain RF distance.

Comparison of RBS versus SBS

Tables 1 and 2 provide the relevant alignment dimensions and data on inference times as well as the aforementioned metrics for quantification of differences between SBS and RBS support values. Scatterplots of RBS and SBS values on the best-scoring tree as well as for all bipartitions (splits) detected by RBS and SBS replicates for *all test alignments* are available at http://icwww.epfl.ch/~stamatak/results.html. We also provide plots for the RBS and SBS support value distribution and indicate the average support values on the respective best-scoring tree. Therefore, we only include exemplary plots for the dataset with the worst correlation coefficient (d125; Fig. 3a to c), the best correlation coefficient (d140; Fig. 4a), the largest dataset analyzed in terms of inference times (d775; Fig. 4b), and the dataset with the worst weighted RF distance (d150; Fig. 4c).

The only dataset (d125) that shows $\rho < 0.95$ (denoted as *CorrBest* in Table 2) for SBS and RBS values on the best-scoring tree is a long multi-gene alignment, which has an unusually high average node support of 0.966 (RBS) and 0.948 (SBS), respectively. In addition, as depicted in Figure 3a, the deviation in BS proportions that causes the relatively low ρ value is due to the few nodes that have a BS value < 0.95. The distribution of support values for alignment d125 is provided in Figure 3c. Figure 3b provides a scatterplot for the frequencies of all bipartitions (all splits) detected by SBS or RBS: the correlation is 0.988. Whereas the correlation coefficient on the best-scoring tree is low, the plain and weighted RF distances (0.02 and 0.03) between majority-rule consensus trees are among the lowest recorded in our experiments. Dataset d125 exhibits the smallest difference between the weighted and unweighted RF distances of consensus trees; i.e., the BS consensus topology is stable.

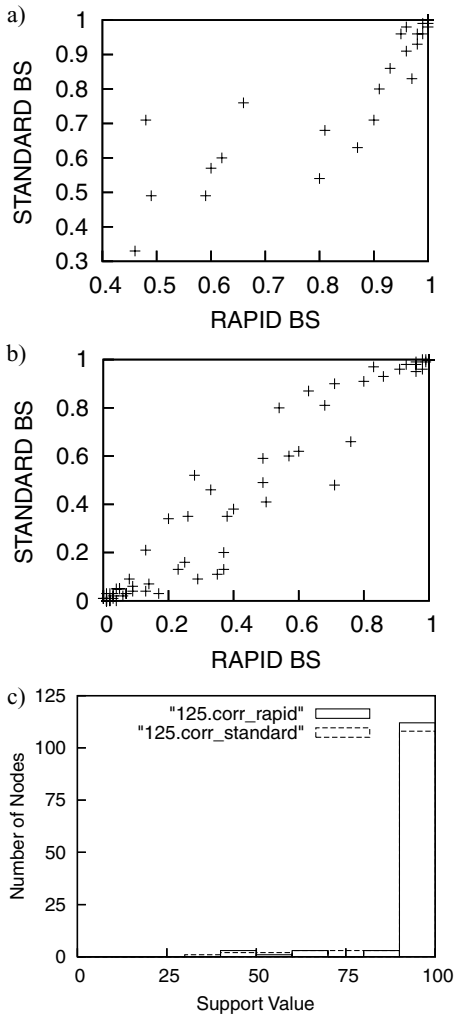

FIGURE 3. Detailed analysis of RBS and SBS support values for dataset d125. (a) Scatterplot for SBS (standard bootstrap) and RBS values drawn on the best–scoring ML tree for d125 (DNA). (b) Scatterplot for SBS and RBS bipartition frequencies for all bipartitions found by SBS or RBS for d125 (DNA). (c) Distribution of SBS and RBS support values on best-scoring ML tree for d125 (DNA).

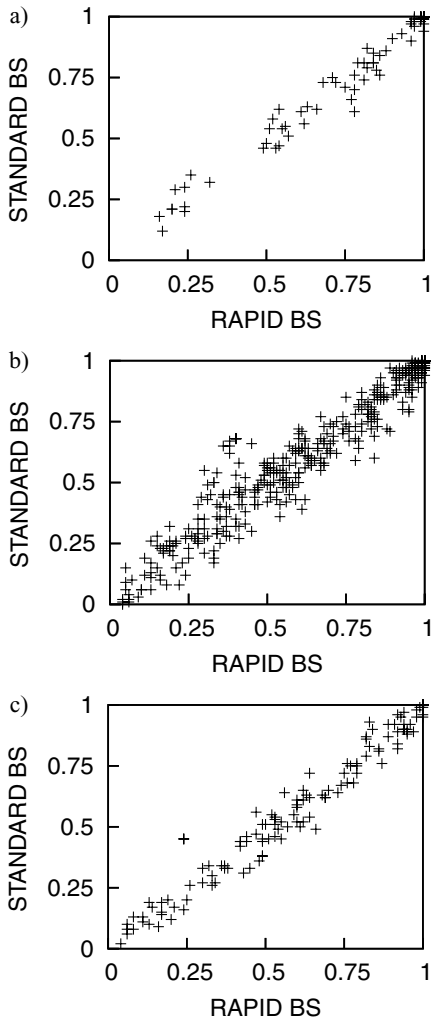

FIGURE 4. Scatterplot for SBS and RBS values drawn on the best-scoring ML tree for (a) d140 (AA), (b) d775 (AA), (c) d150 (DNA).

As further examples, we included support value plots on the best-scoring tree for datasets d775 (AA; Fig. 4b) and d150 (DNA; Fig. 4c). Dataset d775 (AA) represents the largest matrix analyzed in this study in terms of memory footprint and inference times, whereas d150 exhibits the largest weighted RF distance. Finally, the 140-sequence protein dataset of papillomaviruses yielded the highest correlation between RBS and SBS values on the best-scoring tree (Fig. 4a). An interesting observation is that both AA alignments have relatively high average support (> 0.85) under partitioned as well as unpartitioned models, which appears to confirm the empirical observation that AA alignments yield more stable topologies than DNA data. Except for dataset d404 under a plain, unpartitioned model, the average support values obtained via RBS are slightly higher than those obtained via SBS (see http://icwww.epfl.ch/~stamatak/results.html). The higher amount of total bipartitions (columns *SBS(Bips)* and *RBS(Bips)* in Table 2) detected by SBS compared to RBS as well as the slightly higher average support obtained via RBS is due to the increased locality of RBS searches, which only initiate 10% of total searches on a new starting tree and conduct a smaller number of less exhaustive LSR cycles per search/replicate. Another important conclusion that can be drawn from Tables 1 and 2 is that RBS execution times as well as measures of relative accuracy scale well with increasing number of taxa.

Impact of Parameters and Approximations on RBS and SBS Values

We assessed the effect of RAxML program parameters and approximations, particularly the usage of GTR+CAT, on the support values. The partially "odd" number of replicates used in these experiments is due to a run-time limitation of 60 hours on the cluster located

at the Technical University of Munich. Initially, we analyzed the effect of an increased number of replicates on relative accuracy. The correlation between RBS and SBS for 1000 RBS replicates on dataset d714 increases to 0.985 compared to 0.972 on 100 replicates. We make the same observation for 1000 replicates on alignment d628, where ρ increases by 0.012 to 0.975. The correlation on the best tree for d2000 with 934 SBS and RBS replicates also increases by 0.012 to 0.992. In general, there is a trend for the correlation between RBS and SBS to increase with the number of replicates. Thereafter, we investigated the impact of the GTR+CAT approximation compared to analyses under GTR+Γ. The correlation of 359 SBS replicates under GTR+Γ on dataset d1288 with 359 RBS replicates (always GTR+CAT) amounts to 0.970; i.e., improves by 0.006 with respect to 100 SBS/RBS replicates under CAT. The correlation on the best-scoring tree of 1000 RBS replicates on data set d125 compared to 802 SBS replicates under GTR+Γ is 0.98, which represents a significant improvement compared to 0.917 on 100 replicates. The general tendency for correlation values to improve with the number of replicates does not seem to depend on whether Γ or CAT is used. We also computed the correlation between 100 replicates under the SBS algorithm with GTR+Γ and 100 replicates under GTR+CAT for SBS as well as RBS for the following datasets: d150, d218, d404, d500, d628, d714, d1481, d1604. The average correlation between SBS analyses under GTR+Γ and GTR+CAT was 0.980 (minimum: 0.976, maximum: 0.987). The average correlation between SBS analyses under GTR+Γ and RBS analyses under GTR+CAT was 0.964 (minimum: 0.953, maximum 0.980). The correlation between Γ-based SBS analyses and CAT-based RBS analyses decreases insignificantly compared to the results in Table 2.

In addition, we also determined the impact of the likelihood cutoff heuristics. We computed 100 SBS replicates without cutoff heuristics and compared them to the respective SBS/RBS analyses *with* cutoff on the best-scoring trees for datasets d500 (SBS: 0.997, RBS: 0.972) and d628 (SBS: 0.995, RBS: 0.963). In this experiment, correlation coefficients also change insignificantly when RBS values are compared to SBS values inferred with and without cutoff. Finally, we quantified the effect of using a distinct set of BS replicates. The variation among SBS support values on two distinct sets of 100 bootstrap replicates (different random number seed passed via `-b`) was analyzed on the following 11 datasets: d125, d140, d150, d218, d354, d404, d500, d628, d714, d994, d1908. The average correlation for SBS inferences on distinct sets of 100 replicates was 0.988 (minimum: 0.987, maximum 0.990). These values do not differ significantly from those obtained by comparing SBS to RBS support values on 100 replicates. Thus, the variation observed between 100 RBS and SBS samples is similar to the effect of using a distinct set of SBS replicates.

Performance of Rapid ML Search

Table 3 reports performance data for the accelerated ML search that is conducted *after* RBS compared to 20

TABLE 3. Performance of standard and rapid RAxML ML searches and full SBS/RBS ML analyses. Column *SLH* provides the log likelihood of the best scoring tree found during these 20 searches, whereas column *RLH* indicates the respective log likelihood obtained by the accelerated ML search. Column *SLH(hrs)* and *RLH(hrs)* report the respective ML inference times in hours, *Speedup* shows the acceleration factor achieved for the ML search, whereas *Better?* indicates whether the tree obtained via the rapid search had a better log likelihood score than the best tree of the slow searches. Execution times for the combined BS and ML searches are provided in column *SF(hrs)* for SBS and standard ML, *RF(hrs)* for RBS and rapid ML. Finally, column *SpeedupF* provides the overall speedup of RBS over SBS for a *full* ML analysis, including BS inference and ML search.

#SEQS	SLH	RLH	SLH(hrs)	RLH(hrs)	Speedup	Better?	SF(hrs)	RF(hrs)	SpeedupF
d125	−825,204.83	−825,204.83	66.57	14.01	4.75	equal	195.02	24.54	7.95
d140_AA	−121,810.45	−121,809.52	15.80	11.46	1.38	yes	67.60	16.63	4.06
d140_AA_P	−121,482.21	−121,481.58	19.08	12.93	1.48	yes	82.62	18.28	4.52
d150	−39,601.59	−39,600.94	1.75	0.77	2.26	yes	7.06	1.14	6.19
d218	−134,157.15	−134,152.30	5.78	3.32	1.74	yes	24.11	4.50	5.36
d354	−6,561.96	−6,560.88	1.42	0.77	1.84	yes	5.87	1.08	5.46
d404	−156,128.23	−156,115.25	133.31	35.16	3.79	yes	369.41	52.07	7.09
d404_P	−154,067.20	−154,054.75	98.23	54.19	1.81	yes	357.46	78.27	4.57
d500	−85,787.22	−85,778.11	8.54	4.22	2.02	yes	39.63	6.08	6.52
d628	−50,866.93	−50,858.99	7.54	4.54	1.66	yes	34.01	6.42	5.30
d714	−148,516.26	−148,491.64	15.73	6.74	2.33	yes	64.05	9.60	6.67
d994	−348,825.66	−348,803.74	103.97	46.38	2.24	yes	359.22	61.10	5.88
d1288	−395,859.80	−395,926.96	71.53	28.57	2.50	no	289.59	43.20	6.70
d1481	−197,450.53	−197,371.61	48.71	27.70	1.76	yes	185.99	36.78	5.06
d1512	−273,396.87	−273,409.64	60.11	42.35	1.42	no	258.56	55.78	4.64
d1604	−167,320.22	−167,286.47	51.28	27.18	1.89	yes	210.52	35.79	5.88
d1908	−149,607.38	−149,573.19	65.84	26.25	2.51	yes	290.57	38.30	7.59
d2000	−364,813.48	−364,847.60	114.60	60.31	1.90	no	536.82	81.34	6.60
d2308	−449,780.13	−449,928.70	114.66	51.35	2.23	no	493.67	80.03	6.17
d2554	−318,436.87	−318,463.68	96.14	46.99	2.05	no	482.19	76.38	6.31
d4114	−325,441.86	−325,420.85	250.39	101.99	2.46	yes	833.97	141.07	5.91
d6718	−481,080.68	−481,203.10	466.87	178.36	2.62	no	1,702.62	254.38	6.69
d7764	−498,458.40	−498,235.53	467.83	164.85	2.84	yes	1,741.60	237.75	7.33
Averages	−252,811.13	−252,807.82	99.38	41.32	2.24		375.31	59.15	6.02

standard RAxML tree searches on 20 distinct MP starting trees. In addition, it provides the accumulated inference times; i.e., SBS plus time for standard ML search and RBS plus time for rapid ML search, as well as the overall speedup attained by the new algorithm for *full* ML searches. In 18 out of 25 cases, the rapid ML search algorithm yields a tree with a better score than the 20 standard ML searches. The average speedup with respect to these 20 standard ML searches is 2.2. We did not execute a rapid ML search on the d775 AA dataset, due to the long sequential execution time, even under the accelerated algorithm. The average speedup for complete BS and ML analyses over all datasets is 6.28. For the two largest datasets (d6718 and d7764), this represents a reduction of inference times from more than 2 months to 10 days on a single CPU. It is important to note that the overall speedup for a *full* ML analysis will further increase in favor of RBS if an appropriate, larger number of BS replicates is conducted.

Comparison with PHYML and GARLI

Tables 4 and 5 provide an execution time and support value comparison of RBS/SBS with PHYML v2.4.5 (Guindon and Gascuel, 2003) and GARLI v0.951 (Zwickl, 2006) on several randomly selected smaller datasets that kept the required execution times for 100 replicates within acceptable limits. PHYML and GARLI BS searches were conducted on the same BS replicates as the RAxML RBS/SBS searches that were generated with RAxML via `-f j -b 12345 -# 100`. In Table 1 we also provide a comparison of GARLI and PHYML support values with RAxML SBS support under GTR+Γ in order to assess whether the usage of GTR+CAT has a notable effect on the correlation.

As can be derived from the correlation coefficients (*CBest-SBS*/*CBest-RBS*) and weighted RF (*WRF-SBS*/*WRF-RBS*) distances in Tables 4 and 5, GARLI support values correlate better with RAxML RBS as well as SBS support values than PHYML BS values. This is due to the NNI search algorithm in PHYML, which tends to get trapped in local optima earlier than RAxML and GARLI (Stamatakis et al., 2005a; Stamatakis, 2006b; Morrison, 2007). In addition, the average support obtained by PHYML (column *AVG(P)* in Table 4) is significantly lower than the values obtained via RAxML and GARLI. The data in Table 5 (columns *CBest-RBS*, *CBest-SBS*, *CBest-SBS(GAMMA)*) indicate that the usage of GTR+CAT or GTR+Γ in RAxML has no notable effect on the correlation between RAxML and PHYML support values; i.e., the differences in the degree of exhaustiveness of the respective search algorithms have a higher impact on BS values than model details. GARLI and RAxML yield qualitatively comparable trees on medium-sized alignments up to 1000 to 1500 taxa. The only datasets where the correlation between GARLI and SBS/RBS support values is below 0.90 has more than 1500 sequences. The usage of SBS under GTR+Γ yields a slightly better correlation for d1288 (column *CBest-SBS(GAMMA)* in Table 5), whereas it is slightly worse than either SBS under GTR+CAT or RBS on datasets d404, d714, d1604. An important result from

TABLE 5. Support value comparison between 100 SBS/RBS and PHYML/GARLI bootstrap analyses. Column *CBest-SBS*/*CBest-RBS* provides the correlation of SBS/RBS and PHYML/GARLI support values on the best-scoring ML tree, column *CBest-SBS(GAMMA)* the correlation of SBS support values under GTR+Γ with PHYML and GARLI on the best-scoring ML tree. Finally, in columns *Slope* and *Intercept* we report the slope and intercept of the linear regression function between SBS under GTR+Γ and PHYML/GARLI respectively.

#SEQS	CBest-SBS	CBest-RBS	CBest-SBS (GAMMA)	Slope	Intercept
d150	0.935	0.923	0.933	1.03	−4.40
d628	0.904	0.896	0.898	0.99	−2.88
d714	0.829	0.782	0.834	0.97	−6.11
d1,481	0.900	0.884	0.891	0.98	−5.70
d1,604	0.923	0.898	0.913	1.01	−4.24
#SEQS	CBest-SBS	CBest-RBS	CBest-SBS (GAMMA)	Slope	Intercept
d404	0.968	0.940	0.964	0.95	4.61
d714	0.923	0.912	0.922	0.97	3.24
d714 SPR=12	0.923	0.908	0.923	0.96	3.57
d1288	0.958	0.947	0.975	0.99	−0.74
d1604	0.872	0.890	0.882	0.95	−0.05

TABLE 4. Computational performance and support value comparison between 100 SBS/RBS and PHYML/GARLI bootstrap analyses. Column *PHYML(hrs)*/*GARLI(hrs)* provides the execution time for 100 BS replicates, *RF-SBS*/*RF-RBS* the RF distance between SBS/RBS and the respective PHYML/GARLI consensus trees, and *WRF-SBS*/*WRF-RBS* the weighted topological distance between SBS/RBS and PHYML/GARLI consensus trees. Finally, column *Speedup* provides the run time acceleration achieved by RBS over PHYML and GARLI, respectively. We also indicate the average support values for SBS *Avg(S)*, RBS *Avg(R)*, PHYML *Avg(P)*, and GARLI *Avg(G)*.

#SEQS	PHYML (hrs)	RF-SBS	WRF-SBS	RF-RBS	WRF-RBS	Speedup	Avg(P)	Avg(S)	Avg(R)
d150	6.93	0.24	0.11	0.24	0.12	18.87	0.59	0.63	0.65
d628	61.66	0.23	0.09	0.26	0.11	32.88	0.52	0.58	0.59
d714	67.29	0.34	0.14	0.33	0.14	23.52	0.54	0.63	0.66
d1481	820.73	0.35	0.10	0.36	0.11	90.29	0.44	0.52	0.53
d1604	307.11	0.25	0.08	0.27	0.09	35.65	0.52	0.56	0.59
#SEQS	GARLI (hrs)	RF-SBS	WRF-SBS	RF-RBS	WRF-RBS	Speedup	Avg(G)	Avg(S)	Avg(R)
d404	1258.98	0.16	0.05	0.17	0.05	74.44	0.59	0.58	0.58
d714	182.91	0.16	0.06	0.17	0.07	63.92	0.63	0.63	0.66
d714 SPR=12	219.17	0.16	0.06	0.19	0.08	76.60	0.63	0.63	0.66
d1288	771.87	0.13	0.04	0.14	0.05	53.78	0.68	0.65	0.68
d1604	1085.07	0.27	0.07	0.26	0.08	125.96	0.53	0.56	0.59

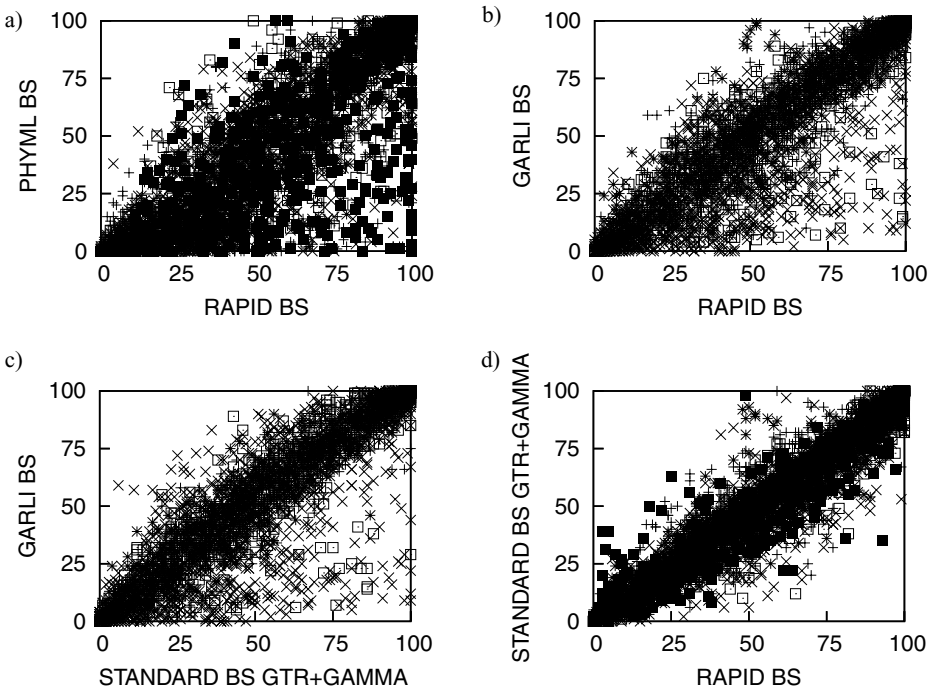

FIGURE 5. Comparison of RBS support values with RAxML SBS, GARLI, and PHYML support values under GTR+Γ. (a) Superimposed scatterplots for RBS and PHYML BS values for datasets d150, d628, d714, d1481, and d1604 on the respective best-scoring trees. (b) Superimposed scatterplots for RBS and GARLI BS values for datasets d404, d714, d1288, and d1604 on the respective best-scoring trees. (c) Superimposed scatterplots for SBS BS values under GTR+Γ and GARLI BS values for datasets d404, d714, d1288, and d1604 on the respective best-scoring trees. (d) Superimposed scatterplots for RBS and SBS BS values under GTR+Γ for datasets d150, d404, d628, d714, d1288, and d1604 on the respective best-scoring trees.

the comparisons of RBS/SBS with PHYML is that the effect of the approximations used to obtain RBS support values is less prevalent than the differences induced by the distinct search strategies. The comparison with GARLI shows that the average variation of support values among SBS(GTR+CAT)/SBS(GTR+Γ) and RBS support values is within the same order of magnitude as the variation induced by using the different, though equally thorough, GARLI search algorithm. This justifies the claim that the deviations between RBS and SBS as well as SBS under GTR+Γ support values are small. In Figure 5a we provide superimposed scatterplots of RBS versus PHYML support values for datasets d150, d628, d714, d1481, and d1604 on the respective best-scoring trees. In the analogous Figures 5b and c, we plot GARLI BS over RBS values and SBS(GTR+Γ) values, respectively, for datasets d404, d714, d1288, and d1604. In Figure 5d we provide superimposed plots for RBS versus SBS under GTR+Γ on datasets d150, d404, d628, d714, d1288, and d1604. Figures 5a to d show that the overall correspondence of support values is best between RBS (always under GTR+CAT) and SBS under GTR+Γ.

Finally, we summarize RBS execution time improvements over SBS, PHYML, and GARLI on all DNA datasets used in this study in Figure 6. Figure 6 shows the development of RBS, SBS, PHYML, and GARLI execution times over the number of taxa in the alignments (note the log-scale on the y-axis). For some of the larger datasets, PHYML (d1908, d2000, d2308, d2554) and GARLI (d2000, d4114, d6718) BS inference times were estimated on the basis of the time required for five replicates due to excessively long run times. PHYML crashed for d4114, d6718, and d7764, despite

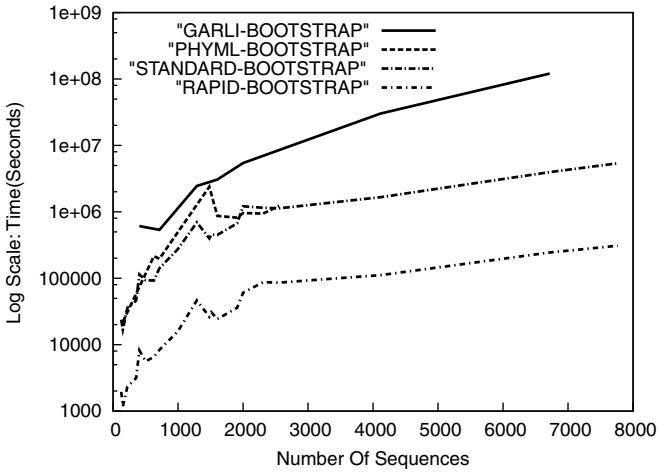

FIGURE 6. Inference times for 100 BS replicates with RAxML(SBS), RAxML(RBS), PHYML, and GARLI on DNA datasets; times scaled linearly to 1000 distinct alignment patterns for easier comparability. Note the log-scale on the *y*-axis.

setting `N_MAX_OTU 10000` in the PHYML source file `utilities.h`; dataset d4114 crashed at the start of the BS inference, d6718 and d7764 crashed during model parameter optimization.

All execution times have been scaled up or down linearly to a length of 1000 distinct column patterns to facilitate comparability. This is not an exact scaling, because inference times do not increase linearly with alignment length due to decreased cache efficiency (Stamatakis et al., 2005b), but serves illustrative purposes well at this point. Another issue that is highlighted by the relatively uneven plot is that certain real-world alignments are "harder" to infer than others. Figure 6 illustrates that RAxML scales well with the number of taxa and that the RBS algorithm is two orders of magnitude faster than the qualitatively comparable GARLI algorithm.

Discussion

Significant progress has been achieved over the last years in the field of heuristic ML search algorithms with the release of programs, such as IQPNNI (Minh et al., 2005), PHYML (Guindon and Gascuel, 2003; Hordijk and Gascuel, 2005), TREEFINDER (Jobb et al., 2004), TREE-PUZZLE (Strimmer and von Haeseler, 1996; Schmidt et al., 2002), GARLI (Zwickl, 2006), DPRML (Keane et al., 2005), PHYNAV (Vinh et al., 2005), LeaPhy (Whelan, 2007), and RAxML (Stamatakis et al., 2005a; Stamatakis, 2006b). A recent performance study (Stamatakis, 2006b) reveals that RAxML outperforms PHYML, GARLI, IQPNNI, and MrBayes (Ronquist and Huelsenbeck, 2003) in terms of speed, accuracy, and memory consumption on alignments with more than 1000 sequences. GARLI performs equally well with respect to accuracy and speed for up to 1000 to 2000 sequences but scales significantly worse on datasets with several thousand taxa. With the rapid bootstrap (RBS) heuristics introduced here, RAxML required about 24 hours on a *single* CPU to compute 100 BS replicates for a multigene alignment of 404 sequences (d404) with 11 partitions and a joint branch length estimate, whereas an unpartitioned analysis of the same dataset required 52 days with GARLI. Although analyses of large single-gene datasets of more than 1000 sequences as used in our computational experiments are still relatively uncommon, memory requirements of ML programs are becoming increasingly important in the context of multi-gene analyses (see, for instance, McMahon and Sanderson, 2006, or Dunn et al., 2008). Memory-wise, such single-gene analyses with thousands of taxa roughly correspond to large multi-gene analyses with hundreds of sequences and several genes. For example, PHYML cannot handle the large real-world AA dataset of 775 protein sequences of fishes. The accuracy of PHYML is expected to be improved by the integration of the fast SPR move technique (Hordijk and Gascuel, 2005) into PHYML v3.0. However, the prerelease version of PHYML v3.0 could not be used in the current study because it was still unstable at the time the computational experiments were conducted (September/October 2007). Therefore, the results in this article are all based on PHYML v2.4.5. The usage of the less exhaustive topological search mechanism implemented in PHYML v2.4.5 also allowed us to compare the RBS support values with SBS and GARLI BS values on one hand and values obtained via a significantly less powerful NNI-based search strategy on the other; i.e., to determine whether RBS values are more similar to SBS/GARLI values or PHYML values and thereby devise a notion of relative accuracy.

Web servers.—There already exist several Web servers for inference of phylogenetic trees. The Cipres portal service (http://www.phylo.org/sub_sections/portal/) currently offers computations with GARLI, MrBayes, PAUP* (Swofford, 2002), and RAxML. However, the initial version of the portal did not provide the possibility to conduct bootstrap analyses due to run time and resource limitations. There also exist several Web servers based on PHYML. The server located at LIRMM (Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier; http://atgc.lirmm.fr/phyml/) offers the current official release of PHYML (Guindon et al., 2005). This Web server uses 16 CPUs and appears to be heavily loaded. Two more comprehensive approaches that use PHYML as well as numerous related programs, including complete analysis pipelines, are Phylogeny.fr (Serious Phylogenetic Analysis For The Non-Specialist; http://www.phylogeny.fr/) and PHYLEMON (Tarraga et al., 2007; http://phylemon.bioinfo.cipf.es/). Finally, Keane et al. (2007) have recently launched a Web server for MultiPhyl. The main advantage of our Web services in comparison to the services mentioned above is that we can achieve a significantly higher throughput for full ML analyses on alignments of almost arbitrary size, by using two medium-sized clusters with a total of 328 CPUs in combination with the RBS algorithm. However, those 328 CPUs are not exclusively used for phylogenetic inferences with RAxML; i.e., scheduling delays can occur. In addition, as outlined by our results, RBS yields qualitatively similar results to SBS and GARLI as well as better ML trees compared to PHYML v2.4.5.

Fast methods for computation of support values.—Because BS analysis represents a well-known computational bottleneck, faster approaches for computation of support values or statistics under maximum likelihood, such as the usage of randomized estimated log likelihoods (RELLs, Waddell et al., 2002) to approximate BS support values or the approximate likelihood-ratio test (aLRT; Anisimova and Gascuel, 2006) have been proposed. Although RELLs can also be used for likelihood-based tests of topologies (Goldman et al., 2000), Waddell et al. (2002) explicitly analyzed its usage as a fast approximation to compute ML and Bayesian support values. The only implementation of RELLs we are aware of in this specific context is provided in Treefinder (Jobb et al., 2004), but unfortunately neither implementation details nor a respective performance study are available. After more than a year of algorithmic and computational experiments, we were not able to devise an RELL-based solution that produces highly correlated support values with standard BS on large datasets.

More recently, Anisimova and Gascuel (2006) proposed the approximate likelihood-ratio test (aLRT), which provides a computationally efficient way to compute support statistics. The inference of aLRT statistics is based on the computation of $n - 3$ (where n is the number of sequences) nearest neighbor interchanges (NNIs) on an NNI-optimal tree. The test must be applied to the respective best-scoring ML tree. The standalone aLRT test is less than an order of magnitude faster than RBS. It requires, for instance, approximately 2 hours compared to 16 hours with RBS on dataset d404 or 40 minutes compared to 111 minutes on the single-gene d500 alignment. However, the current implementation of aLRT, when executed in combination with an NNI search for the best-scoring ML tree, requires almost 26 hours on d404 and is thus slower than RBS. A comparison of aLRT with RBS was omitted because it currently remains unclear how to compare aLRT statistics with BS support values, in particular on real-world datasets where the true tree and the true support values are unknown.

CONCLUSION

We have presented a novel rapid bootstrapping method that yields support values that are highly correlated to standard BS values obtained by RAxML as well as the qualitatively comparable GARLI program (see Tables 4 and 5). The average improvement in execution times over standard RAxML BS and competing programs (GARLI, PHYML) exceeds one order of magnitude and solves—to a large extent—the computational problems associated with present–day full ML analyses with a couple of hundred or a few thousand taxa. The RBS algorithm allows systematists to conduct large ML analyses in less than a week on their workstation. In addition, we offer the program as a Web service at the Vital-IT unit of the Swiss Institute of Bioinformatics and on the CIPRES project cluster at the San Diego Supercomputer Center.

The significant speed improvement also opens up new possibilities for improvements in current phylogenetic methodology. For example, one can easily assess the impact of different alignments obtained by distinct alignment programs on the final trees and associated support values. This is particularly important, because the quality of the alignment can have a significant impact on the inferred trees. Due to the high computational cost, this important issue is rarely addressed in current real-world studies. Moreover, one can design an iterative or genetic alignment-refinement tree-building method that uses trees and support values computed on an initial tree as input for the next alignment refinement step. Another useful application is the integration into pipelines that automatically update backbone trees for large sequence alignment databases, such as greengenes (DeSantis et al., 2006), when sequences become available.

The RBS algorithm may also, due to its speed, contribute to the development of a bootstopping criterion that determines at which number of replicates one might stop the BS process. The value of RBS is that it allows for empirical assessment of such a bootstopping procedure, because an extremely large number of 10,000 BS reference replicates can easily be computed for datasets up to 2500 sequences. Finally, we will also focus on further accelerating the ML search algorithm, which has now become the main computational bottleneck, by using information from the preceding RBS search; e.g., one might assign probabilities for conducting LSR moves based on RBS support values or build a starting tree for the ML search based on an appropriate RBS consensus tree.

ACKNOWLEDGMENTS

We would like to thank Tandy Warnow, who initiated this work by requesting a quick and dirty BS algorithm, Peter Waddell for discussions about RELL, and Jakob Fredslund for providing us the PHY.FI code. Furthermore, we would like to thank Olaf Bininda-Emonds, Jun Inoue, Nikos Poulakakis, Usman Roshan, Marc Gottschling, Chuck Robertson, and Nicolas Salamin for providing real-world test datasets. Sergios-Orestis Kolokotronis, Maria Anisimova, and Markus Göker provided helpful comments on the manuscript. Finally, we would like to thank Olivier Gascuel and an anonymous reviewer for useful comments on the manuscript. This work was funded by the Emmy Noether program of the German Science Foundation (AS), the NFS ITR program "Building the Tree of Life" (EF 03-31648, PH), and financial support from HP/Intel to Vital-IT (JR).

REFERENCES

Anisimova, M., and O. Gascuel. 2006. Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. Syst. Biol. 55:539–552.

Blagojevic, F., D. S. Nikolopoulos, A. Stamatakis, and C. D. Antonopoulos. 2007. Dynamic multigrain parallelization on the cell broadband engine. Pages 90–100 *in* Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. San Diego, California.

Charalambous, M., P. Trancoso, and A. Stamatakis. 2005. Initial experiences porting a bioinformatics application to a graphics processor. LNCS 3746:415–425.

Chor, B., and T. Tuller. 2005. Maximum likelihood of evolutionary trees: Hardness and approximation. Bioinformatics 21:97–106.

DeSantis, T. Z., P. Hugenholtz, N. Larsen, M. Rojas, E. L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu, and G. L. Andersen. 2006. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. Appl. Env. Microbiol. 72:5069–5072.

Dunn, C. W., A. Hejnol, D. Q. Matus, K. Pang, W. E. Browne, S. A. Smith, E. Seaver, G. W. Rouse, M. Obst, G. D. Edgecombe, M. V. Sorensen, S. H. D. Haddock, A. Schmidt-Rhaesa, A. Okusu, R. M. Kristensen, W. C. Wheeler, M. Q. Martindale, and G. Giribet. 2008. Broad phylogenomic sampling improves resolution of the animal tree of life. Nature 452:745–749.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. J. Mol. Evol. 17:368–376.

Felsenstein, J. 1985. Confidence limits on phylogenies: An approach using the bootstrap. Evolution 39:783–791.

Fredslund, J. 2006. PHY.FI: Fast and easy online creation and manipulation of phylogeny color figures. BMC Bioinformatics 7:315.

Goldman, N., J. P. Anderson, and A. G. Rodrigo. 2000. Likelihood-based tests of topologies in phylogenetics. Syst. Biol. 49:652–670.

Grimm, G. W., S. S. Renner, A. Stamatakis, and V. Hemleben. 2006. A nuclear ribosomal DNA phylogeny of acer inferred with maximum likelihood, splits graphs, and motif analyses of 606 sequences. Evol. Bioinformatics Online 2:279–294.

Guindon, S., and O. Gascuel. 2003. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. 52:696–704.

Guindon, S., F. Lethiec, P. Duroux, and O. Gascuel. 2005. PHYML online—A Web server for fast maximum likelihood-based phylogenetic inference. Nucleic Acids Res. 33:557–559.

Hillis, D. M., T. A. Heath, and K. S. John. 2005. Analysis and visualization of tree space. Syst. Biol. 54:471–482.
Hordijk, W., and O. Gascuel. 2005. Improving the efficiency of SPR moves in phylogenetic tree search methods based on maximum likelihood. Bioinformatics 21:4338–4347.
Jobb, G., A. von Haeseler, and K. Strimmer. 2004. Treefinder: A powerful graphical analysis environment for molecular phylogenetics. BMC Evol. Biol. 4:18.
Keane, T. M., T. J. Naughton, and J. O. McInerney. 2007. MultiPhyl: A high-throughput phylogenomics Webserver using distributed computing. Nucleic Acids Res. 35:W33–W37.
Keane, T. M., T. J. Naughton, S. A. A. Travers, J. O. McInerney, and G. P. McCormack. 2005. DPRml: Distributed phylogeny reconstruction by maximum likelihood. Bioinformatics 21:969–974.
McMahon, M. M., and M. J. Sanderson. 2006. Phylogenetic supermatrix analysis of Genbank sequences from 2228 papilionoid legumes. Syst. Biol. 55:818–836.
Minh, B. Q., L. S. Vinh, A. von Haeseler, and H. A. Schmidt. 2005. PIQPNNI: Parallel reconstruction of large maximum likelihood phylogenies. Bioinformatics 21:3794–3796.
Moret, B. M. E. 2002. Towards a discipline of experimental algorithmics. Pages 197–213 *in* Data structures, near neighbor searches, and methodology: Fifth and Sixth DIMACS Implementation Challenges (M. H. Goldwasser, D. Johnson, and C. McGeoch, eds.), volume 59 of DIMACS Monographs. American Mathematical Society. Providence, Rhode Island.
Morrison, D. A. 2007. Increasing the efficiency of searches for the maximum likelihood tree in a phylogenetic analysis of up to 150 nucleotide sequences. Syst. Biol. 56:988–1010.
Ott, M., J. Zola, S. Aluru, and A. Stamatakis. 2007. Large-scale maximum likelihood-based phylogenetic analysis on the IBM blueGene/L. *In* On-Line Proceedings of IEEE/ACM Supercomputing Conference 2007. Reno, Nevada, USA.
Ripplinger, J., and J. Sullivan. 2008. Does choice in model selection affect maximum likelihood analysis? Syst. Biol. 57:76–85.
Robinson, D. F., and L. R. Foulds. 1979. Comparison of weighted labelled trees. Lecture Notes Math. 748:119–126.
Robinson, D. F., and L. R. Foulds. 1981. Comparison of phylogenetic trees. Math. Biosci. 53:131–147.
Ronquist, F., and J. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574.
Schmidt, H. A., K. Strimmer, M. Vingron, and A. von Haeseler. 2002. TREE-PUZZLE: Maximum likelihood phylogenetic analysis using quartets and parallel computing. Bioinformatics 18:502–504.
Stamatakis, A. 2006a. Phylogenetic models of rate heterogeneity: A high performance computing perspective. *In* Proceedings of 20th IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS2006). Rhodos, Greece.
Stamatakis, A. 2006b. RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22:2688–2690.
Stamatakis, A., F. Blagojevic, C. D. Antonopoulos, and D. S. Nikolopoulos. 2007. Exploring new search algorithms and hardware for phylogenetics: RAxML meets the IBM cell. J. VLSI Sig. Proc. Sys. 48:271–286.
Stamatakis, A., T. Ludwig, and H. Meier. 2005a. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics 21:456–463.
Stamatakis, A., M. Ott, and T. Ludwig. 2005b. RAxML-OMP: An efficient program for phylogenetic inference on SMPs. LNCS 3606:288–302.
Strimmer, K. and A. von Haeseler. 1996. Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies. Mol. Biol. Evol. 13:964–969.
Swofford, D. L. 2002. PAUP*: Phylogenetic analysis using parsimony (*and other methods). Version 4.0b10. Sinauer Associates, Sunderland, Massachusetts.
Tarraga, J., I. Medina, L. Arbiza, J. Huerta-Cepas, T. Gabaldon, J. Dopazo, and H. Dopazo. 2007. Phylemon: A suite of Web tools for molecular evolution, phylogenetics and phylogenomics. Nucleic Acids Res. 35:W38.
Vinh, L. S., H. A. Schmidt, and A. von Haeseler. 2005. PhyNav: A novel approach to reconstruct large phylogenies. Pages 386–393 *in* Classification, the Ubiquitous Challenge. (Proceedings of the 28th Annual Conference of the GfKl 2004). Studies in classification, data analysis, and knowledge organization Springer, Heidelberg/New York.
Waddell, P. J., H. Kishino, and R. Ota. 2002. Very fast algorithms for evaluating the stability of ML and Bayesian phylogenetic trees from sequence data. Gen. Informatic 13:82–92.
Whelan, S. 2007. New approaches to phylogenetic tree search and their application to large numbers of protein alignments. Syst. Biol. 56:727–740.
Wilkes, M. 2001. The memory gap and the future of high performance memories. ACM SIGARCH Computer Architecture News 29:2–7.
Zwickl, D. 2006. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. PhD thesis, University of Texas at Austin.

First submitted 23 December 2007; reviews returned 6 March 2008; final acceptance 20 May 2008
Associate Editor: Susanne Renner