Ultrafast Approximation for Phylogenetic Bootstrap

Bui Quang Minh,*[1] Minh Anh Thi Nguyen,[2] and Arndt von Haeseler*[1]

[1]Center for Integrative Bioinformatics Vienna, Max F. Perutz Laboratories, University of Vienna, Medical University of Vienna, Vienna, Austria

[2]Groningen Bioinformatics Centre, University of Groningen, Groningen, The Netherlands

***Corresponding authors:** E-mail: minh.bui@univie.ac.at; arndt.von.haeseler@univie.ac.at.

Associate editor: Naruya Saitou

Abstract

Nonparametric bootstrap has been a widely used tool in phylogenetic analysis to assess the clade support of phylogenetic trees. However, with the rapidly growing amount of data, this task remains a computational bottleneck. Recently, approximation methods such as the RAxML rapid bootstrap (RBS) and the Shimodaira–Hasegawa-like approximate likelihood ratio test have been introduced to speed up the bootstrap. Here, we suggest an ultrafast bootstrap approximation approach (UFBoot) to compute the support of phylogenetic groups in maximum likelihood (ML) based trees. To achieve this, we combine the resampling estimated log-likelihood method with a simple but effective collection scheme of candidate trees. We also propose a stopping rule that assesses the convergence of branch support values to automatically determine when to stop collecting candidate trees. UFBoot achieves a median speed up of 3.1 (range: 0.66–33.3) to 10.2 (range: 1.32–41.4) compared with RAxML RBS for real DNA and amino acid alignments, respectively. Moreover, our extensive simulations show that UFBoot is robust against moderate model violations and the support values obtained appear to be relatively unbiased compared with the conservative standard bootstrap. This provides a more direct interpretation of the bootstrap support. We offer an efficient and easy-to-use software (available at http://www.cibiv.at/software/iqtree) to perform the UFBoot analysis with ML tree inference.

Key words: **phylogenetic inference, nonparametric bootstrap, tree reconstruction, maximum likelihood.**

Article

Introduction

Since the groundbreaking work of Felsenstein (1985), nonparametric bootstrapping (Efron 1979) has become one of the widely used tools to estimate the phylogenetic support of certain clades or splits in an inferred phylogenetic tree. Here, the sequence alignment sites are sampled with replacement resulting in a number of pseudoreplicates. For every replicate, one applies a method of interest such as maximum likelihood (ML; Felsenstein 1981) to reconstruct a bootstrap tree. One then either constructs a consensus tree from the bootstrap trees or places the support values onto the reconstructed ML tree.

Because of the enormous computation time required for the standard bootstrap (SBS) with ML, several approaches have been published to approximate SBS. Resampling estimated log-likelihoods (RELL; Kishino et al. 1990; Hasegawa and Kishino 1994) was the first attempt to avoid a full ML inference per bootstrap replicate; it reuses the log-likelihood scores calculated for individual sites in the original alignment, given the tree. RELL was used to infer local bootstrap probabilities (LBP; Adachi and Hasegawa 1996) of every internal branch of the ML tree by comparing the three nearest neighbor interchange (NNI) tree topologies around the branch of interest. The approximate likelihood-ratio test (aLRT; Anisimova and Gascuel 2006) and its nonparametric variant (SH-aLRT; Guindon et al. 2010) differ slightly from the method used to calculate LBP by employing the SH test (Shimodaira and Hasegawa 1999) on these three NNI trees. Although RELL and SH-aLRT are very fast, it is currently unclear how they perform if the four subtrees incident to that branch are not fixed. The RAxML rapid bootstrap (RBS; Stamatakis 2006; Stamatakis et al. 2008) is a recent method to resemble SBS while performing 8–20 times faster on large data sets.

It has been shown that the SBS probabilities typically underestimate the true probabilities of a clade to be correct (Felsenstein and Kishino 1993; Hillis and Bull 1993). SBS is therefore biased but conservative. Efron et al. (1996) proposed a method to correct for this bias which, however, requires considerably more computation. Other methods include quartet puzzling (Strimmer and von Haeseler 1996; Schmidt et al. 2002) and Bayesian Markov chain Monte Carlo (MCMC) analysis (Yang and Rannala 1997; Huelsenbeck and Ronquist 2001). Bayesian MCMC methods, however, tend to overestimate the true probabilities in case of model misspecification or polytomies (Suzuki et al. 2002; Douady et al. 2003; Lewis et al. 2005; Anisimova et al. 2011). Both quartet puzzling and Bayesian MCMC methods are very time consuming for large data sets.

New Approaches

Here, we present an ultrafast bootstrap approach (UFBoot) as an alternative to the other nonparametric bootstrap approaches. To this end, we utilize the RELL concept with an efficient way of sampling plausible trees using the important quartet puzzling (IQP) with NNI (IQPNNI) algorithm (Vinh and von Haeseler 2004; Minh et al. 2005). In short, IQPNNI samples the local maxima and their neighborhoods in the tree space defined by the NNI operations. Because the number of trees encountered during the IQPNNI search might be excessively large, we adaptively estimate a log-likelihood threshold ℓ_{min} such that we only investigate the trees with the RELL bootstrapping if their log-likelihoods are higher than ℓ_{min}. Taken together, UFBoot first generates a number of bootstrap alignments (typically 1,000) and initializes the corresponding bootstrap trees as null. UFBoot then performs the IQPNNI tree sampling on the original alignment. Whenever a new tree T whose log-likelihood exceeds ℓ_{min} is found, UFBoot quickly computes the RELL score of T for each bootstrap alignment. If T has a higher RELL score than that of the current bootstrap tree, UFBoot updates the current bootstrap tree as T for the corresponding bootstrap alignment. That way, UFBoot gradually rectifies the set of bootstrap trees. UFBoot stops collecting candidate trees when the correlation coefficient c_F of the split occurrence frequencies computed from the first half of the analysis and from the full analysis is larger than 0.99 (more details in Materials and Methods). Finally, UFBoot computes a consensus tree from the set of bootstrap trees and also maps the split support values onto the ML tree reconstructed during the IQPNNI sampling.

We provide an implementation of the whole framework in the IQ-TREE package (Nguyen L-T, Minh BQ, Schmidt HA, von Haeseler A, in preparation). In the following, we compare the performance of UFBoot against other bootstrap approaches in terms of accuracy (Hillis and Bull 1993) and computational time.

Results

Accuracy

We used simulated data (table 1; Materials and Methods) to compare four different methods (SBS with RAxML, RBS with RAxML, SH-aLRT with PhyML, and UFBoot) with respect to their accuracy defined in Hillis and Bull (1993). To this end, we plot the number of true splits (i.e., splits that occur in the true trees) having support of $x\%$ divided by the number of all splits with support of $x\%$ (eq. 2; fig. 1). This ratio gives the estimated probability of a split to be true. Curves above the dashed diagonal line indicate that the inferred support values underestimate this probability, and thus the corresponding method exhibits a conservative behavior. In contrast, curves below the diagonal indicate that the method overestimates the true probabilities. Methods that generate curves around the diagonal are almost unbiased.

Table 1. Simulation Settings.

True Tree	Data Type	No. Sequences	No. Sites	No. Alignments
Yule–Harding	DNA	100	500	200
		200	1,000	200
		500	1,000	200
	Protein	100	300	200
		200	500	200
PANDIT	DNA	4–403	24–6,891	6,222
	Protein	4–545	12–2,297	6,182

Figure 1 summarizes the results for the Yule–Harding and PANDIT-based simulations (see Materials and Methods for more details). Note that the curves look similar for the seven simulation settings (table 1) and are thus not shown. SBS (blue curves) is the most conservative approach by substantially underestimating the probabilities of splits being correct for both Yule–Harding and PANDIT-based simulations. For example, a split with SBS support of 80% has indeed a probability of 0.95 to be correct. This biased but conservative behavior of SBS corroborates previous studies (Hillis and Bull 1993; Anisimova et al. 2011), which led to the widely accepted interpretation of "trusting" splits with SBS supports $\geq 80\%$. RBS (fig. 1, yellow curves) performs very similarly to SBS but with a tendency of being less conservative.

SH-aLRT (fig. 1, black curves) is generally as conservative as SBS and RBS in the Yule–Harding simulations but becomes apparently less conservative in the PANDIT-based simulations. Moreover, low SH-aLRT split supports ($\leq 50\%$) are not informative with respect to the true probabilities. For example, splits with SH-aLRT support of 20% are as correct as those with support of 50%.

UFBoot (fig. 1, red curves) appears to be almost unbiased compared with the other methods for both simulations (i.e., the split support values obtained closely reflect the probabilities of the split being correct). UFBoot is unbiased for support values higher than 70%. On the other side, UFBoot support values smaller than 70% slightly overestimate the true probability. Such unbiased behavior simplifies the interpretation of support values reported by UFBoot. For example, a split with support of 95% will have a probability of 0.95 to be correct.

Moreover, we assessed the impact of model misspecification on the accuracies by repeating the analysis on the simulated DNA alignments using the simpler JC + Γ model (Jukes and Cantor 1969; Yang 1994) and the simplest JC model (Jukes and Cantor 1969) for phylogenetic inference. Note that we could not repeat the same analysis with RBS and SBS, because RAxML supports only the GTR + Γ model (Lanave et al. 1984; Yang 1994). Alternatively, we performed SBS with 100 replicates using IQ-TREE. Figure 2 shows that model violations have almost no influence on SBS estimates with IQ-TREE (green curves) in PANDIT-based simulations (Yule–Harding data not shown). Similarly, the accuracies of SH-aLRT and UFBoot do not change under moderate model violations (JC + Γ). However, the split support values are inflated under severe model violations (JC). This agrees with previous studies showing that accounting for the rate heterogeneity among sites is more important than varying substitution rates (Sullivan and Swofford 2001; Nguyen et al. 2012).

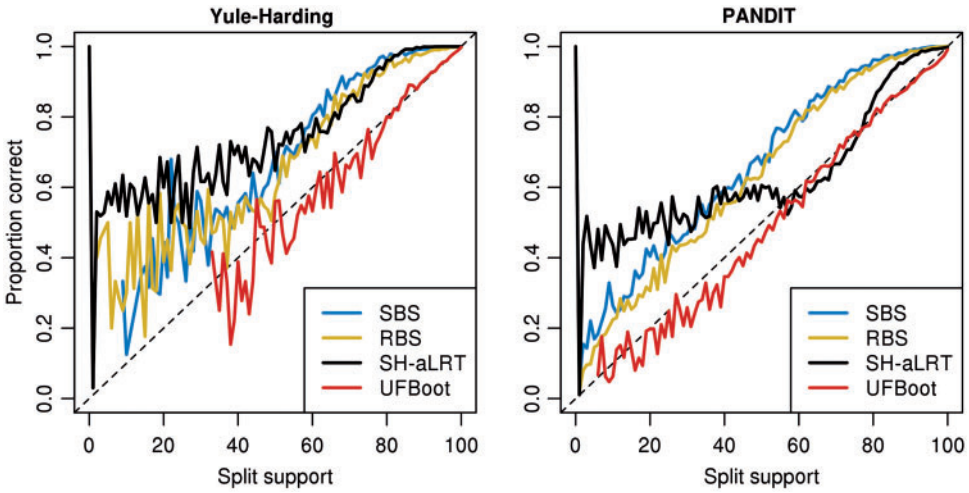

FIG. 1. Accuracies of SBS, RBS with RAxML, SH-aLRT with PhyML, and UFBoot approximation from the Yule–Harding (left panel) and the PANDIT-based simulations (right panel).

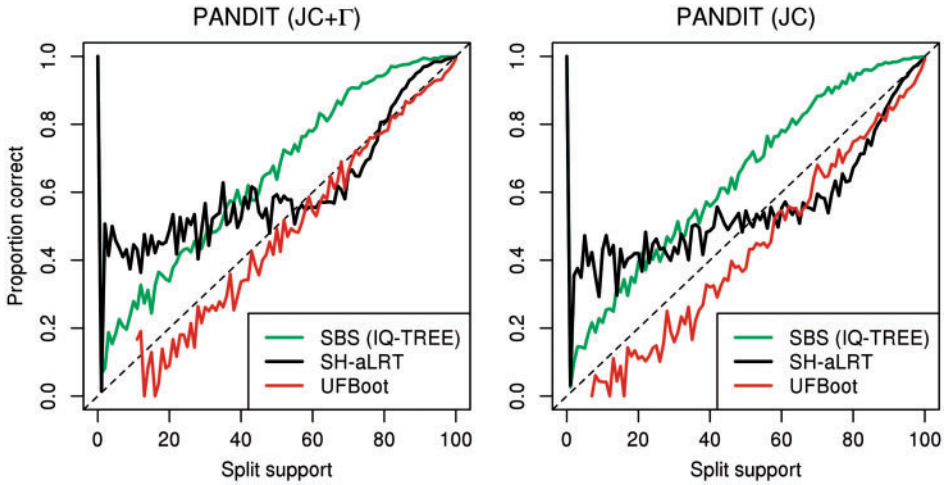

FIG. 2. Impact of moderate (JC + Γ) and severe model violations (JC) on the accuracies of SBS, SH-aLRT, and UFBoot in the PANDIT-based simulations.

Computational Time

For more than 96% of the Yule–Harding and PANDIT-based simulations the UFBoot stopping rule (see Materials and Methods) suggested to stop after 100 IQPNNI iterations. The remaining runs finished after at most 800 iterations. Thus, $Q_{max} = 1{,}000$ is a conservative upper bound for the number of iterations to achieve high accuracy.

A more detailed picture emerges from the real PANDIT data. We compared the computational times of RBS and UFBoot on 308 large (≥100 sequences) DNA- and AA-PANDIT alignments. For a fair comparison of computing times, we apply the bootstopping criterion (-N autoMRE) (Pattengale et al. 2010) in the RBS search to automatically determine the number of bootstrap replicates required. For eight AA-PANDIT alignments (PF01261, PF00149, PF01546, PF01547, PF01636, PF00496, PF00501, and PF07690) RAxML did not finish after more than 1 week of computation, the runs were then stopped by our computing system. These alignments were excluded from our analysis, leaving us with 300 alignments. The bootstopping criterion of RBS yielded an average of 528 bootstrap replicates. The number of bootstrap replicates varied between 250 and 1,000 (the default upper limit in RAxML), where 5 alignments needed 250 replicates and 1 alignment hit the upper limit.

Our UFBoot stopping rule suggested on average 453 IQPNNI iterations for all alignments. We observed that for 80 (27%) alignments 100 iterations sufficed to obtain stable bootstrap estimates and for 69 (23%) alignments we hit the maximum of 1,000 iterations, indicating that the resulting split supports from these runs did not meet our convergence criterion. Among these 69 alignments 49, 15, and 5 alignments achieved a correlation coefficient c_F of at least 0.95, between 0.9 and 0.95, and less than 0.9, respectively. However, the five alignments with $c_F < 0.9$ comprise very divergent sequences and possibly nonalignable sequences. The percentages of alignment sites with low alignment confidence (Whelan et al. 2006) are ranging between 32% and 52%. Therefore, the nonconvergence in such cases is not surprising.

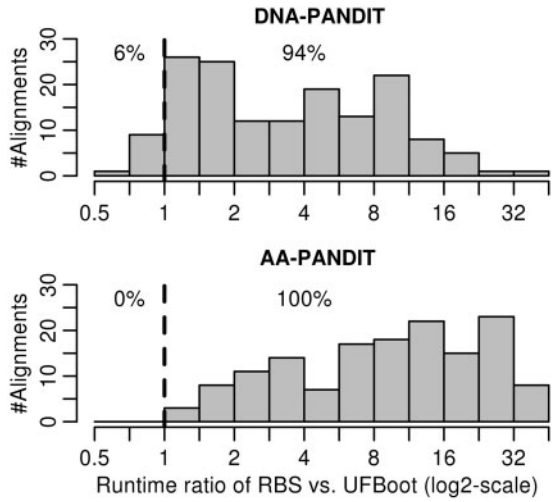

FIG. 3. Distributions of run-time ratios ($\log_2$-scale) between RBS and UFBoot for 300 DNA and AA PANDIT alignments. The percentages of alignments where UFBoot runs slower (left from the dashed line) or faster (right from the dashed line) than RBS are shown.

Finally, we computed the distribution of the ratio between the computational times of RBS and UFBoot for the 300 alignments (fig. 3). UFBoot was always faster than RBS except for 10 DNA alignments. The 69 alignments where UFBoot did not converge (discussed earlier) also caused the slowest UFBoot runs. UFBoot runs 3.1 times (median, range: 0.66–33.3) and 10.2 (median, range: 1.32–41.4) times faster than RBS for DNA and AA alignments, respectively. More impressive is the total computing time for the full PANDIT data analysis: UFBoot required 797 CPU core hours (1.1 month) on a computer cluster equipped with 2.2-GHz CPUs, whereas RBS needed 4,293 CPU hours (~6 months).

Discussion

We have suggested a very fast bootstrap approximation, namely UFBoot, and compared the performance with a collection of widely used methods. Although SBS and RBS estimates of clade support are conservative (see also Hillis and Bull 1993; Anisimova et al. 2011), the clade support estimated by UFBoot appears less biased according to our large-scale simulations. This leads to a different and easy-to-understand interpretation of the support values. For example, a support of at least 95% should be used if one wants to control the false-positive rate of 5%. The fact that UFBoot is a hybrid of parametric sampling of the tree space and the nonparametric bootstrap sampling of the alignment may be one explanation for reduction of the bias of the bootstrap probabilities. Parametric methods (aLRT, Bayesian MCMC) are unbiased if the true substitution model is known (Anisimova et al. 2011). UFBoot inherits this property as shown in our simulations. Moreover, UFBoot partly overcomes model misspecifications by applying the nonparametric RELL correction (Anisimova et al. 2011). However, we have to acknowledge that a thorough theoretical explanation for our observation is missing.

The interpretation of support values as unbiased has been used in Bayesian inference. However, Bayesian inference has been known to be sensitive even against mild model violations (Suzuki et al. 2002; Anisimova et al. 2011). In contrast, UFBoot appears robust against moderate model violations during phylogenetic inference (fig. 2). However, caution is advised under severe model violations (i.e., wrongly assumed rate homogeneity among sites) then UFBoot (also SH-aLRT) tends to infer unduly high support values. Here, methods to detect model violations (Goldman 1993; Weiss and von Haeseler 2003; Nguyen et al. 2011) should be applied before the UFBoot analysis (or any other analysis). At present it is not clear, if the number of IQPNNI iterations necessary to achieve bootstrap support convergence may be helpful to detect such artifacts.

Apart from oversimplified substitution models, other types of model violations such as polytomies and heterotachy (i.e., varying substitution rates among different tree branches and alignment sites) (Lopez et al. 2002) are known to cause systematic bias in the ML and Bayesian methods (Kolaczkowski and Thornton 2004; Lewis et al. 2005). For example, polytomies often lead to a tree space with a lot of local optima. This may hamper the underlying IQPNNI algorithm in exploring the tree space (Whelan and Money 2010; Money and Whelan 2012), which might in turn inflate UFBoot support values. It is necessary to investigate these and other factors (e.g., by looking at the support of conflicting splits) to understand further the mechanism of bias correction in UFBoot and under which conditions the correction might fail. Currently, these are still unclear to us. However, a more thorough analysis is beyond the scope of this study. Nevertheless, as our methodology works on any set of input candidate trees, it might be worthwhile to exploit UFBoot with other tree sampling strategies such as the genetic algorithm (Zwickl 2006) or the Bayesian MCMC (Drummond et al. 2012; Ronquist et al. 2012). We provide such an option in our implementation.

SH-aLRT behaves very differently between the Yule–Harding and PANDIT-based simulations (fig. 1), implying that there is no easy rule of thumb how to interpret SH-aLRT support values. This may be due to the fact that SH-aLRT computes the support value for every branch by only comparing the tree log-likelihood with the log-likelihoods of the two alternative NNI trees around the branch of interest (Adachi and Hasegawa 1996). That way, SH-aLRT ignores all other trees that may show higher log-likelihoods than the two NNI trees, which may result in an overconfidence of SH-aLRT support values. Nevertheless, SH-aLRT, being a very quick branch test method, is useful for extremely large data sets. In our implementation, we offer an option to report both SH-aLRT and UFBoot support values per branch so that users can directly compare them.

Our built-in UFBoot stopping rule provides an intuitive statistic c_F, the correlation coefficient of the split support values inferred from the first half of the analysis and from the full analysis. c_F values close to 1.0 imply that an extended tree search will not substantially change the

resulting support values and we can therefore stop. Similar ideas have been employed in the bootstopping criterion (Pattengale et al. 2010). The fact that the UFBoot stopping rule suggested only 100 iterations for most simulated data are not surprising because the tree space for simulated data typically contains only a few local maxima and is therefore easy to sample. The situation is different for real data where our convergence criterion was not always met. But these cases were also characterized by low phylogenetic information (Money and Whelan 2012). This reinforces the observation that one should assess the phylogenetic signals in the data with, for example, the likelihood mapping (Strimmer and von Haeseler 1997) and saturation plots (Van de Peer et al. 2002; Xia et al. 2003) before carrying out an expensive bootstrap analysis. If the data appear to be appropriate for phylogenetic reconstruction, then UFBoot is a time-saving option compared with the other bootstrap inference tool.

Conclusion

We have presented the UFBoot approximation approach that 1) outperforms the RAxML RBS in terms of the computational time, 2) achieves almost unbiased support values like Bayesian methods, and 3) is relatively robust against moderate model violations. We provide an implementation of UFBoot within the IQ-TREE software package available from http://www.cibiv.at/software/iqtree. IQ-TREE is a substantially improved reimplementation of the IQPNNI algorithm with additional features (Nguyen L-T, Minh BQ, Schmidt HA, von Haeseler A, in preparation). IQ-TREE allows users to reconstruct the ML tree (with support values), the bootstrap trees, and the consensus tree by UFBoot within one single run. Users can also perform UFBoot from a user-defined set of trees sampled by other methods (e.g., genetic algorithm or MCMC sampling).

Materials and Methods

ML Principle

Let A denote a multiple sequence alignment with n sequences and m sites (columns), where sites in A are grouped into k site-patterns $D_1, \ldots, D_k$ of identical sites. Hence, we represent A by a vector of site-pattern frequencies $(d_1, \ldots, d_k)$, where d_i is the number of sites having site-pattern D_i ($d_i > 0$ and $\sum_{i=1}^{k} d_i = m$).

Under the assumption of independence of the sites, the log-likelihood ℓ of a tree T (with branch lengths) given A is computed by:

$$\ell(T \mid A) = \sum_{i=1}^{k} d_i \times \ell(T \mid D_i),$$

where $\ell(T \mid D_i)$ is the log-likelihood of T at site-pattern D_i.

Under the ML principle, the objective is to identify the most likely tree $T_{ML} = \text{argmax}_T\, \ell(T \mid A)$. Note that the computation of ℓ is implicitly based on a predefined substitution model, which we omit in this notation for the sake of simplicity.

RELL Method Revisited

A bootstrap sample A^* of A is simply a resampled frequency vector $A^* = (d_1^*, \ldots, d_k^*)$, where d_i^* is the frequency of D_i in A^* ($d_i^* \geq 0$ and $\sum_{i=1}^{k} d_i^* = m$). To compute $\ell(T \mid A^*)$ for a given tree T under the SBS, one has to re-estimate the branch lengths and model parameters based on A^*. To save computation, RELL (Kishino et al. 1990) approximates $\ell(T \mid A^*)$ by using $\ell(T \mid D_i)$ (i.e., keeping branch lengths and model parameters fixed). Hence, the log-likelihood scores of individual sites remain the same, implying that calculating

$$\hat{\ell}(T \mid A^*) = \sum_{i=1}^{k} d_i^* \times \ell(T \mid D_i) \quad (1)$$

for many bootstrap alignments on a fixed tree will be computationally inexpensive. In addition, one can quickly select an approximate ML tree for A^* from a collection $\mathbf{C}$ of candidate trees by computing $T_{ML}^* = \text{argmax}_{T \in \mathbf{C}}\, \hat{\ell}(T \mid A^*)$ if $\ell(T \mid A)$ is known for all $T \in \mathbf{C}$.

RELL was used to infer the LBP (Adachi and Hasegawa 1996) for every internal branch of a fixed tree T as follows: For each internal branch one computes $\ell(T_{NNI1} \mid A)$ and $\ell(T_{NNI2} \mid A)$ of the two NNI trees around this branch. Next, one generates B bootstrap alignments $A_1^*, A_2^*, \ldots, A_B^*$ and computes the three corresponding RELL scores $\hat{\ell}(T \mid A_i^*), \hat{\ell}(T_{NNI1} \mid A_i^*), \hat{\ell}(T_{NNI2} \mid A_i^*)$ for each A_i^* according to equation (1). The local support of the branch in question is the percentage of A_i^* where $\hat{\ell}(T \mid A_i^*) > max\{\hat{\ell}(T_{NNI1} \mid A_i^*), \hat{\ell}(T_{NNI2} \mid A_i^*)\}$. In other words, the LBP method considers the set $\mathbf{C}$ of exactly three candidate trees and may overlook other "good" tree topologies (Adachi and Hasegawa 1996, p. 49). For that reason, we pursue another approach described in the following sections.

Tree Proposal

The applicability of RELL crucially depends on the collection of candidate trees. The naive way of evaluating all tree topologies of n taxa (Waddell et al. 2002) only works for small n. Here, we exploit a strategy of sampling trees using the IQPNNI algorithm (Vinh and von Haeseler 2004; Minh et al. 2005). In principle, IQPNNI does a sampling of local maxima in the tree space defined by the NNI operations (fig. 4). To this end, IQPNNI iteratively moves through the tree space in which the IQP operations help to escape local optimal regions and subsequently NNI moves toward the local optima within regions (T_1, T_2, and T_3 in fig. 4). To escape local optima the IQP step randomly deletes a fraction p_{del} of the leaves of the tree and re-inserts the leaves using the quartet puzzling method (Strimmer and von Haeseler 1996).

The IQPNNI algorithm (fig. 4) works as follows. IQPNNI starts with the BIONJ (Gascuel 1997) tree T_0 and moves to T_1 via a series of NNIs. Here T_1 represents a local maximum of the tree space. This completes the first IQPNNI iteration. In the second IQPNNI iteration, IQPNNI applies the IQP operation to propose T_2' from T_1 and subsequently moves to T_2 (via NNI), which locates another local maximum.

As $\ell(T_2 \mid A) < \ell(T_1 \mid A)$, we keep T_1 as the current best tree. In the third iteration, T_3' is generated from T_1 and then T_3 reflects another local optimum. Now, as $\ell(T_3 \mid A) > \ell(T_1 \mid A)$, T_3 becomes the new ML tree as it has a higher likelihood. In other words, the IQPNNI algorithm allows us to escape the local optimum T_1. Because this search is carried out for many iterations, IQPNNI samples many local optima and thus provides a rough picture of the tree space.

As a by-product IQPNNI also samples the trees that are a few NNIs away from local optima. To get a collection $\boldsymbol{C}$ of candidate trees, we collect all distinct trees encountered during the IQPNNI search.

Restricting the Number of Candidate Trees

As we might encounter millions of distinct trees during the IQPNNI search and as we are interested in plausible trees (i.e., those in the vicinity of local optima), we introduce the parameter τ to consider only trees in $\boldsymbol{C}$ exceeding a certain log-likelihood threshold. In other words, based on τ we empirically determine a log-likelihood threshold $\ell_{\min}$ during the search such that a tree T will only be investigated with the RELL bootstrapping if $\ell(T \mid A) \geq \ell_{\min}$. This works as follows: Let Q be the total number of IQPNNI iterations and q the current iteration. On average, we aim to collect $\frac{\tau}{Q}$ trees per iteration. Hence, we expect $\frac{\tau}{Q} \cdot q$ trees after q iterations. If $|\boldsymbol{C}| < \frac{\tau}{Q} \cdot q$ after the qth iteration, we have collected fewer trees than we aimed for, so we set $\ell_{\min} = -\infty$ to accept all subsequent trees. If however $|\boldsymbol{C}| \geq \frac{\tau}{Q} \cdot q$, then the expected number of trees after Q iterations might exceed τ. To avoid this, we set $\ell_{\min}$ equal to the log-likelihood of the $\left\lceil \frac{\tau}{Q} \cdot q \right\rceil$-th best tree in $\boldsymbol{C}$. In the subsequent iteration q+1, a tree T is assigned to $\boldsymbol{C}$ only if it is not yet in $\boldsymbol{C}$ and if $\ell(T \mid A) \geq \ell_{\min}$. At the end of iteration q+1, we update $\ell_{\min}$ as shown earlier. $\ell_{\min}$ will decrease or increase depending on the number of trees added to $\boldsymbol{C}$ during iteration q+1. We therefore adaptively adjust $\ell_{\min}$ based on the number of trees encountered during the search. Note that because we do not remove any trees from $\boldsymbol{C}$, the size of $\boldsymbol{C}$ might slightly exceed τ at the end.

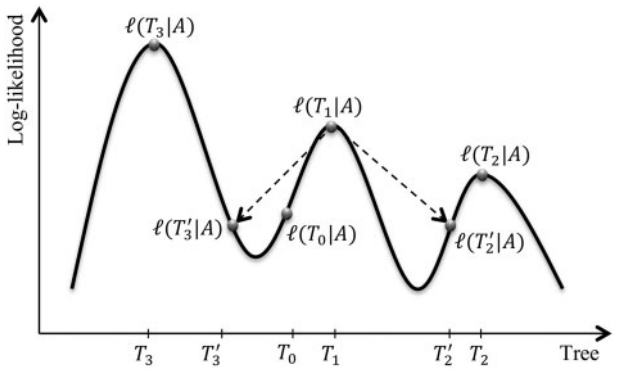

Fig. 4. Schematic view of the tree space sampled by the IQPNNI algorithm. The solid curve reflects the log-likelihood surface on the tree space. The structure of tree space is defined by the NNI operations where each $\boldsymbol{n}$-taxon tree has exactly $2\boldsymbol{n} - 6$ neighboring trees.

UFBoot Approximation

The UFBoot works as follows:

1) Initialization step: Initialize the collection of trees $\boldsymbol{C}$:=∅ and the log-likelihood cutoff $\ell_{\min} := -\infty$. Generate B (typically 1,000 or 10,000) bootstrap alignments $A_1^*, A_2^*, \ldots, A_B^*$. For each A_b^* initialize the bootstrap tree $T_b^* := \emptyset$ and $\hat{\ell}(T_b^* \mid A_b^*) := -\infty$.
2) Exploration step: Perform the IQPNNI search with A (fig. 4). Every time a new tree $T \notin \boldsymbol{C}$ with $\ell(T \mid A) \geq \ell_{\min}$ is encountered during the search:
   a) Compute the approximate log-likelihood $\hat{\ell}(T \mid A_b^*)$ for each bootstrap replicate A_b^* using RELL (eq. 1). If $\hat{\ell}(T \mid A_b^*) > \hat{\ell}(T_b^* \mid A_b^*)$, update $T_b^* := T$.
   b) Update $\boldsymbol{C} := \boldsymbol{C} \cup \{T\}$.

   Re-estimate $\ell_{\min}$ as explained upon finishing an IQPNNI iteration.
3) Summarization step: Construct a consensus tree from the bootstrap trees $\{T_1^*, T_2^*, \ldots, T_B^*\}$ or map the support values onto the ML tree reconstruced by the IQPNNI search.

The exploration step is the main step that simultaneously explores the tree space and updates the bootstrap trees. The computation of $\hat{\ell}(T \mid A_b^*)$ represents the only additional computation compared to the original IQPNNI algorithm and has a time-complexity of $O(|\boldsymbol{C}|kB)$, where k is the number of site-patterns in the input alignment. We implement the collection of distinct trees $\boldsymbol{C}$ as a hash table for computational efficiency, implying that we compute the approximate likelihoods for trees encountered during the search exactly once. Moreover, if the probability of revisiting a tree during the search is small (which often happens for large data), one can safely omit storing the trees in $\boldsymbol{C}$, and thus substantially reducing the memory consumption. We provide both options in our implementation.

UFBoot Stopping Rule

In principle, the more IQPNNI iterations (Q) are carried out during the exploration step, the more candidate trees ($\boldsymbol{C}$) are considered and the better UFBoot performs. However, Q should not be too large since our goal is to provide an UFBoot approximation method. Q should also not be unrealistically small because we want to achieve high accuracy. Thus, we introduce a so-called "UFBoot stopping rule" that automatically assesses the convergence of the split support values and stops collecting candidate trees once convergence is achieved.

To this end, we start with $Q := 100$ and $\tau := n \cdot Q$, where n is the number of sequences. That means, τ is no more an independent parameter and we collect on average n trees per IQPNNI-iteration. This is motivated by the fact that each IQPNNI iteration generates $O(n)$ trees, and we will therefore consider a constant factor (<1) of the number of trees encountered. During the exploration step, once $\frac{Q}{2}$ iterations have been completed, we compute the vector of split occurrence frequencies $F\left(\frac{Q}{2}\right)$ for all splits in

the current set of bootstrap trees $\{T_1^*, T_2^*, \ldots, T_B^*\}$. At the end of the Q-th iteration we compute $F(Q)$ and the Pearson's correlation coefficient c_F between the two vectors $F(\frac{Q}{2})$ and $F(Q)$. For splits occurring in one split set but not the other, a corresponding zero entry is added into the other vector. If $c_F \geq 0.99$, then more IQPNNI iterations do not substantially change the split support values. In such case, we stop and output the split support values in $F(Q)$. Otherwise, we continue the exploration step with 100 more iterations (i.e., we increase $\tau := \left\lceil \frac{Q+100}{Q} \cdot |\boldsymbol{C}| \right\rceil$ and $Q := Q + 100$). Therefore, we compute the bootstrap split support every 50 iterations and evaluate the convergence every 100 iterations. Finally, we provide an option to specify a maximum number of iterations Q_{max} such that we will also stop once $Q \geq Q_{max}$. This ensures that the analysis will finish in case a c_F of 0.99 is unlikely to be reached.

Performance Study with Yule–Harding Simulation

We simulated data with varying number of sequences and sites (table 1) to assess the performance of UFBoot. For each setting, we used IQ-TREE (Nguyen L-T, Minh BQ, Schmidt HA, von Haeseler A, in preparation) to generate 200 random trees (true trees) under the Yule–Harding model (Harding 1971) where the branch lengths follow an exponential distribution with the mean of 0.1. Seq-Gen (Rambaut and Grassly 1997) was used to evolve the DNA or protein sequences along the tree under the GTR + Γ (Lanave et al. 1984; Yang 1994) and WAG + Γ (Yang 1994; Whelan and Goldman 2001) model, respectively. The GTR model parameters are: $r_{AC} = 3$, $r_{AG} = 5$, $r_{AT} = 7$, $r_{CG} = 4$, $r_{CT} = 6$, $r_{GT} = 2$, $\pi_A = 0.25$, $\pi_C = 0.15$, $\pi_G = 0.2$, $\pi_T = 0.4$. The Γ distribution parameter is $\alpha = 0.5$. In total, we simulated 600 DNA alignments and 400 amino acid alignments for five settings (table 1).

For each simulated alignment, we then performed UFBoot with $B = 1{,}000$, $Q_{max} = 1{,}000$, and $p_{del} = 0.5$. To compare the UFBoot results, we conducted SBS as implemented in RAxML-SSE3 7.3.0 with 100 replicates (Stamatakis 2006), RBS with 1,000 replicates (Stamatakis et al. 2008), and PhyML SH-aLRT (Guindon et al. 2010). For each bootstrap method, the inferred split support values were mapped onto the ML tree reconstructed by IQ-TREE.

Finally, we collected the set Σ of unique splits occurring in the 1,000 ML trees reconstructed from the 1,000 alignments generated and classified them as true or false splits (i.e., splits that occur in the corresponding true tree or not). Each split $\sigma \in \Sigma$ was associated with four support values: $s_{SBS}(\sigma)$, $s_{RBS}(\sigma)$, $s_{SH-aLRT}(\sigma)$, and $s_{UFBoot}(\sigma)$ rounded as integers between 0% and 100%. Then, we computed the fraction, $f_\star(x)$, of true splits with support value x% against all splits with the same support value x%, thus we computed:

$$f_{SBS}(x) = \frac{\left|\{\text{true splits } \sigma \in \Sigma : s_{SBS}(\sigma) = x\}\right|}{\left|\{\text{splits } \sigma \in \Sigma : s_{SBS}(\sigma) = x\}\right|}. \quad (2)$$

Similarly, we computed $f_{RBS}(x)$, $f_{SH\text{-}aLRT}(x)$, and $f_{UFBoot}(x)$. This ratio is coined "accuracy" (Hillis and Bull 1993) and was used recently by Anisimova et al. (2011).

PANDIT-Based Simulation

Moreover, we performed a large-scale simulation based on the PANDIT database (Whelan et al. 2006) to examine the performance of different bootstrap strategies on trees inferred from biological data. To this end, we retrieved 6,491 DNA and 6,617 protein alignments with at least four sequences from the PANDIT website. Following the recommendation of Whelan et al. (2006), we removed all short alignments ($m < 3n$ for DNA and $m < n$ for protein alignments). For the remaining 6,222 DNA and 6,182 protein alignments, we selected the best-fit models with the Bayesian information criterion using ModelTest (Posada and Crandall 1998) and ProtTest (Darriba et al. 2011), respectively. We then reconstructed an ML tree for each alignment using IQ-TREE under the selected model. The reconstructed ML trees were treated as true trees to generate alignments. We again used Seq-Gen to simulate alignments with the same alignment lengths as the original PANDIT alignments and under the estimated model parameters. We then superimposed the gap positions from the original PANDIT alignments onto corresponding simulated alignments. The use of PANDIT trees and the introduction of gaps into the simulated alignments are to reflect as much reality as possible in the simulation.

Finally, we compared the bootstrap strategies with respect to the accuracy as in the Yule–Harding simulations (eq. 2). Moreover, for 5,688 DNA alignments, where the selected best-fit models are more complex than JC + Γ (Jukes and Cantor 1969; Yang 1994), we assessed the impact of model misspecification on the accuracy (i.e., when the trees are reconstructed under JC + Γ and JC models representing moderate and severe model violations, respectively).

Acknowledgments

The authors thank Dirk Metzler for discussions, Tina Koestler for helpful comments on the manuscript, and Manuel Gil for proofreading. They also thank Lars Jermiin and two anonymous reviewers for their constructive comments on the manuscript. This work was supported by the Austrian Science Fund—FWF (I760) to B.Q.M. and A.v.H. and the EU EURATRANS consortium (HEALTH-F4-2010-241504) to M.A.T.N. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

References

Adachi J, Hasegawa M. 1996. MOLPHY version 2.3—programs for molecular phylogenetics based on maximum likelihood. Minato-ku (Tokyo): Institute of Statistical Mathematics.

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

Anisimova M, Gil M, Dufayard JF, Dessimoz C, Gascuel O. 2011. Survey of branch support methods demonstrates accuracy, power, and robustness of fast likelihood-based approximation schemes. *Syst Biol.* 60:685–699.

Darriba D, Taboada GL, Doallo R, Posada D. 2011. ProtTest 3: fast selection of best-fit models of protein evolution. *Bioinformatics* 27: 1164–1165.

Douady CJ, Delsuc F, Boucher Y, Doolittle WF, Douzery EJP. 2003. Comparison of Bayesian and maximum likelihood bootstrap measures of phylogenetic reliability. *Mol Biol Evol.* 20:248–254.

Drummond AJ, Suchard MA, Xie D, Rambaut A. 2012. Bayesian phylogenetics with BEAUti and the BEAST 1.7. *Mol Biol Evol.* 29: 1969–1973.
Efron B. 1979. Bootstrap methods—another look at the kackknife. *Ann Stat.* 7:1–26.
Efron B, Halloran E, Holmes S. 1996. Bootstrap confidence levels for phylogenetic trees. *Proc Natl Acad Sci U S A.* 93:13429–13434.
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.
Felsenstein J, Kishino H. 1993. Is there something wrong with the bootstrap on phylogenies—a reply. *Syst Biol.* 42:193–200.
Gascuel O. 1997. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol Biol Evol.* 14:685–695.
Goldman N. 1993. Statistical tests of models of DNA substitution. *J Mol Evol.* 36:182–198.
Guindon S, Dufayard JF, Lefort V, Anisimova M, Hordijk W, Gascuel O. 2010. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Syst Biol.* 59:307–321.
Harding EF. 1971. The probabilities of rooted tree-shapes generated by random bifurcation. *Adv Appl Prob.* 3:44–77.
Hasegawa M, Kishino H. 1994. Accuracies of the simple methods for estimating the bootstrap probability of a maximum-likelihood tree. *Mol Biol Evol.* 11:142–145.
Hillis DM, Bull JJ. 1993. An empirical-test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Syst Biol.* 42: 182–192.
Huelsenbeck JP, Ronquist F. 2001. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 17:754–755.
Jukes TH, Cantor CR. 1969. Evolution of protein molecules. In: Munro HN, editor. Mammalian protein metabolism. New York: Academic Press. p. 21–132.
Kishino H, Miyata T, Hasegawa M. 1990. Maximum-likelihood inference of protein phylogeny and the origin of chloroplasts. *J Mol Evol.* 31: 151–160.
Kolaczkowski B, Thornton JW. 2004. Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature* 431:980–984.
Lanave C, Preparata G, Saccone C, Serio G. 1984. A new method for calculating evolutionary substitution rates. *J Mol Evol.* 20:86–93.
Lewis PO, Holder MT, Holsinger KE. 2005. Polytomies and Bayesian phylogenetic inference. *Syst Biol.* 54:241–253.
Lopez P, Casane D, Philippe H. 2002. Heterotachy, an important process of protein evolution. *Mol Biol Evol.* 19:1–7.
Minh BQ, Vinh LS, von Haeseler A, Schmidt HA. 2005. pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies. *Bioinformatics* 21:3794–3796.
Money D, Whelan S. 2012. Characterizing the phylogenetic tree-search problem. *Syst Biol.* 61:228–239.
Nguyen MAT, Gesell T, von Haeseler A. 2012. ImOSM: intermittent evolution and robustness of phylogenetic methods. *Mol Biol Evol.* 29:663–673.
Nguyen MAT, Klaere S, von Haeseler A. 2011. MISFITS: evaluating the goodness of fit between a phylogenetic model and an alignment. *Mol Biol Evol.* 28:143–152.
Pattengale ND, Alipour M, Bininda-Emonds ORP, Moret BME, Stamatakis A. 2010. How many bootstrap replicates are necessary? *J Comput Biol.* 17:337–354.
Posada D, Crandall KA. 1998. MODELTEST: testing the model of DNA substitution. *Bioinformatics* 14:817–818.
Rambaut A, Grassly NC. 1997. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput Appl Biosci.* 13:235–238.
Ronquist F, Teslenko M, van der Mark P, Ayres DL, Darling A, Hohna S, Larget B, Liu L, Suchard MA, Huelsenbeck JP. 2012. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. *Syst Biol.* 61:539–542.
Schmidt HA, Strimmer K, Vingron M, von Haeseler A. 2002. TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* 18:502–504.
Shimodaira H, Hasegawa M. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol Biol Evol.* 16:1114–1116.
Stamatakis A. 2006. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* 22:2688–2690.
Stamatakis A, Hoover P, Rougemont J. 2008. A rapid bootstrap algorithm for the RAxML web servers. *Syst Biol.* 57:758–771.
Strimmer K, von Haeseler A. 1996. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Mol Biol Evol.* 13:964–969.
Strimmer K, von Haeseler A. 1997. Likelihood-mapping: a simple method to visualize phylogenetic content of a sequence alignment. *Proc Natl Acad Sci U S A.* 94:6815–6819.
Sullivan J, Swofford DL. 2001. Should we use model-based methods for phylogenetic inference when we know that assumptions about among-site rate variation and nucleotide substitution pattern are violated? *Syst Biol.* 50:723–729.
Suzuki Y, Glazko GV, Nei M. 2002. Overcredibility of molecular phylogenies obtained by Bayesian phylogenetics. *Proc Natl Acad Sci U S A.* 99:16138–16143.
Van de Peer Y, Frickey T, Taylor JS, Meyer A. 2002. Dealing with saturation at the amino acid level: a case study based on anciently duplicated zebrafish genes. *Gene* 295:205–211.
Vinh LS, von Haeseler A. 2004. IQPNNI: moving fast through tree space and stopping in time. *Mol Biol Evol.* 21:1565–1571.
Waddell PJ, Kishino H, Ota R. 2002. Very fast algorithms for evaluating the stability of ML and Bayesian phylogenetic trees from sequence data. *Genome Inform.* 13:82–92.
Weiss G, von Haeseler A. 2003. Testing substitution models within a phylogenetic tree. *Mol Biol Evol.* 20:572–578.
Whelan S, de Bakker PIW, Quevillon E, Rodriguez N, Goldman N. 2006. PANDIT: an evolution-centric database of protein and associated nucleotide domains with inferred trees. *Nucleic Acids Res.* 34: D327–D331.
Whelan S, Goldman N. 2001. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Mol Biol Evol.* 18:691–699.
Whelan S, Money D. 2010. The prevalence of multifurcations in tree-space and their implications for tree-search. *Mol Biol Evol.* 27: 2674–2677.
Xia XH, Xie Z, Salemi M, Chen L, Wang Y. 2003. An index of substitution saturation and its application. *Mol Phylogenet Evol.* 26:1–7.
Yang Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol.* 39:306–314.
Yang ZH, Rannala B. 1997. Bayesian phylogenetic inference using DNA sequences: a Markov Chain Monte Carlo method. *Mol Biol Evol.* 14: 717–724.
Zwickl DJ. 2006. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. Austin (TX): The University of Texas.