Syst. Biol. 65(6):997–1008 2016

DOI:10.1093/sysbio/syw037
Advance Access publication April 26, 2016

Terrace Aware Data Structure for Phylogenomic Inference from Supermatrices

OLGA CHERNOMOR[1], ARNDT VON HAESELER[1,2], AND BUI QUANG MINH[1,*]

[1]*Center for Integrative Bioinformatics Vienna, Max F. Perutz Laboratories, University of Vienna, Medical University of Vienna, A-1030 Vienna, Austria and* [2]*Bioinformatics and Computational Biology, Faculty of Computer Science, University of Vienna, A-1090 Vienna, Austria;*
**Correspondence to be sent to: Center for Integrative Bioinformatics Vienna (CIBIV), Max F. Perutz Laboratories, Dr Bohrgasse 9, A-1030 Vienna, Austria; E-mail: minh.bui@univie.ac.at.*

Received 9 December 2015; reviews returned 18 April 2016; accepted 19 April 2016
Associate Editor: Edwards Susko

Abstract.—In phylogenomics the analysis of concatenated gene alignments, the so-called supermatrix, is commonly accompanied by the assumption of partition models. Under such models each gene, or more generally partition, is allowed to evolve under its own evolutionary model. Although partition models provide a more comprehensive analysis of supermatrices, missing data may hamper the tree search algorithms due to the existence of phylogenetic (partial) terraces. Here, we introduce the phylogenetic terrace aware (PTA) data structure for the efficient analysis under partition models. In the presence of missing data PTA exploits (partial) terraces and induced partition trees to save computation time. We show that an implementation of PTA in IQ-TREE leads to a substantial speedup of up to 4.5 and 8 times compared with the standard IQ-TREE and RAxML implementations, respectively. PTA is generally applicable to all types of partition models and common topological rearrangements thus can be employed by all phylogenomic inference software. [Maximum likelihood; partial terraces; partition models; phylogenetic terraces; phylogenomic inference.]

The gigantic amount of sequence data generated by next-generation sequencing technologies has spurred phylogenomics (Eisen 1998; Delsuc et al. 2005; Kumar et al. 2012). Here, one aims to infer the tree of life from multiple genes, loci, or even whole genomes, which provide enough phylogenetic information to resolve difficult branching orders (Bininda-Emonds et al. 1999; Rokas et al. 2003; Dunn et al. 2008; Meusemann et al. 2010).

Phylogenomic inference methods are categorized into supertree and supermatrix methods (De Queiroz et al. 1995; Sanderson et al. 1998; Bininda-Emonds et al. 2002; Delsuc et al. 2005; Kupczok et al. 2010). Supertree methods combine inferred (gene) trees into one "supertree". Supermatrix refers to the concatenation of multiple sequence alignments from different genes/sequences. Unavailable genes/sequences constitute the so-called *missing data* in the supermatrix. Standard phylogenetic methods are then used to infer the species tree from the concatenated alignment.

Complex evolutionary scenarios of multi-gene data sets raise additional difficulties for phylogenetic inferences from supermatrices. For example, failure to account for heterogeneous evolution caused by heterotachy (Lopez et al. 2002), that is, when evolutionary rates vary over time, leads to systematic errors in phylogenetic reconstruction (Kolaczkowski and Thornton 2004; Philippe et al. 2005).

To account for different evolutionary scenarios partition models were introduced (Yang 1996) that allow genes to evolve with different substitution models. Three types of partition models *Edge-Unlinked*, *Edge-Linked-equal*, and *Edge-Linked-proportional* are implemented in many maximum likelihood (ML) software packages (Table 1).

The EUL partition model, where each partition has its own set of branch lengths, is the most general. However, due to missing data an EUL model may lead to phylogenetic terraces (Sanderson et al. 2011), where different tree topologies have the same score (likelihood or parsimony). The more restrictive EL partition models could avoid terraces, but assuming these models in the presence of heterotachy can be misleading (Sanderson et al. 2015).

Large phylogenetic terraces may hamper a thorough exploration of tree space by current search algorithms. When encountering a large terrace during the tree search a lot of computation time is spent on the evaluation of trees with equal scores. Therefore, it is important to detect terraces and to reduce computations.

Recently, we generalized the concept of terraces to *partial terraces* (Chernomor et al. 2015) and provided mathematical conditions to quickly identify their occurrences for a species tree and a supermatrix. In order to detect partial terraces during the tree search it is enough to answer the question whether the topological rearrangement applied to a species tree T changes any of its induced partition trees. Here, an induced partition tree is the subtree restricted to species present in the corresponding partition. If some partition trees remained unchanged, then T and the newly obtained species tree T_{NEW} belong to one partial terrace and we only need to compute the score (likelihood or parsimony) for partition trees that were affected by the topological rearrangement. Chernomor et al. (2015) also predicted the potential computing time saving when accounting for (partial) terraces during the tree search. However, an efficient implementation of the theoretical results was not provided.

We note that before the terrace concept was introduced (Sanderson et al. 2011), its properties were implicitly exploited to speed up likelihood computations and the Subtree Pruning and Regrafting (SPR) tree search under the EUL model (Stamatakis and Ott 2008; Stamatakis and Alachiotis 2010). The authors introduced and implemented a data structure called *pointer meshes*,

TABLE 1. Availability of partition models in ML tree search software

Software	EL-equal	EL-proportional	EUL
MetaPIGA (Helaers and Milinkovitch 2010)	x	x	
PhyML (Guindon et al. 2010)			
GARLI (Zwickl 2006)	x	x	
RAxML (Stamatakis 2014)	x		x
TreeFinder (Jobb et al. 2004)	x	x	x
IQ-TREE (Nguyen et al. 2015)	x	x	x

EL, edge-linked; EUL, edge-unlinked.

which yielded a substantial computation time and memory saving. However, it is not clear how to apply the data structure to other topological rearrangements or EL partition models.

Here, we propose a more general *phylogenetic terrace aware* (PTA) data structure, which works for all common topological rearrangements and with EUL and EL partition models. In the following, we provide the theoretical background of phylogenetic partial terraces and a rule to quickly detect them. We also formally review the three partition models. Next, we describe the PTA data structure and provide a dynamic programming algorithm to build it. We reformulate conditions to quickly identify partial terraces and discuss additional timesaving features of different partition models in ML inference. We implemented PTA and the rule to detect partial terraces for EUL and EL partition models in IQ-TREE. Finally, we analyze the efficiency of PTA by examining 12 published alignments and compare the results with the standard IQ-TREE (Nguyen et al. 2015) implementation and with RAxML (Stamatakis 2014).

BACKGROUND

Partial and Full Phylogenetic Terraces

Let $k \geqslant 2$ denote the number of genes, loci, or codon positions in a supermatrix. In the following we use "partition" to generally refer to any subset of genomic positions. Denote by $Y_1, Y_2, \ldots, Y_k$ the species sets for the k partitions and $X = Y_1 \cup Y_2 \cup \ldots \cup Y_k$ the set of all species. The $S_1, S_2, \ldots, S_k$ denote the corresponding alignments and S is the concatenated alignment (supermatrix) of $S_1, S_2, \ldots, S_k$. Stretches of gaps are added to S if a species has no sequence for some partition (i.e., when $Y_i \subset X$). Only these gaps are referred to as missing data.

For a species tree T and a partition Y_i, the associated induced partition tree, denoted $T|Y_i$, is the tree obtained from T by pruning species with no sequence for partition Y_i (i.e., missing sequences). Hence, for every species tree there is a corresponding set of k induced partition trees.

If for two species trees T_1 and T_2 there exists a set of indices $J \subseteq \{1, \ldots, k\}$ such that $\forall j \in J$ the corresponding induced partition trees $T_1|Y_j$ and $T_2|Y_j$ are identical then T_1 and T_2 belong to one *partial terrace* (Chernomor et al. 2015). In this context, a phylogenetic terrace coined in Sanderson et al. (2011) is a special case when $J = \{1, \ldots, k\}$. For clarity we call this case a *full terrace*. $\forall j \in J$ scores (likelihood or parsimony) of $T_1|Y_j$ and $T_2|Y_j$ are equal. Therefore, if during the tree search we identify a full terrace, we need to compute the score of $T_1|Y_j$ or $T_2|Y_j$ only once $\forall j \in J$.

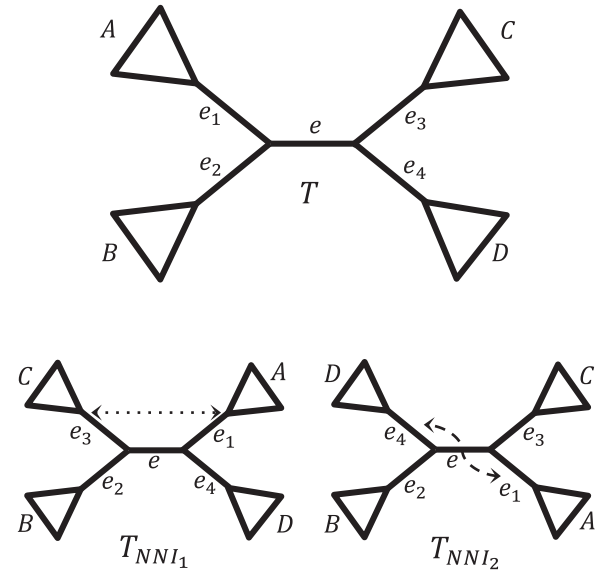

FIGURE 1. The species tree T and the two NNI neighboring trees T_{NNI_1} and T_{NNI_2}, obtained by NNIs around the central edge e. NNI, nearest neighbor interchange.

How to Identify Partial Terraces during the Tree Search

When searching for the optimal species tree we move from one species tree T to another T_{NEW} by means of some topological rearrangements. In phylogenetic software the most common topological rearrangements are Nearest Neighbor Interchange (NNI), SPR, and Tree Bisection and Reconnection (TBR) (Felsenstein 2004).

We now illustrate the necessary condition presented in Chernomor et al. (2015) for the NNI to change the topology of a given induced partition tree. Let T be a species tree and e an interior edge of T. We denote by e_1, e_2, e_3, e_4 edges adjacent to e and by A, B, C, D the taxon sets leading from them, respectively (Fig. 1). Now, let a new tree T_{NNI} be obtained from T via NNI. Then Proposition 1 (Chernomor et al. 2015) states that

> For a partition with a taxon set Y the topologies of $T|Y$ and $T_{\text{NNI}}|Y$ are different if and only if Y has at least one representative taxon in each subset A, B, C, D. (C.1)

Condition (C.1) simply means that for an NNI to change the topology of a partition tree $T|Y$ all $A \cap Y, B \cap Y, C \cap Y$, and $D \cap Y$ must be non-empty.

In the following we discuss the three partition models implemented in IQ-TREE and used to examine the performance of the terrace aware data structure proposed here.

Partition Models

Partition models allow for different evolutionary scenarios for each partition. More formally, each partition Y_i is assumed to evolve under its own substitution model M_i and the model parameters of each M_i are optimized separately on the corresponding partition tree.

The log-likelihood of a species tree T under a partition model M is the sum of partition tree log-likelihoods

$$\ell(T, M|S) = \sum_{i=1}^{k} \ell(T|Y_i, M_i|S_i). \quad (1)$$

Here, the log-likelihood ℓ of the partition trees depends on the topology and the edge lengths of each $T|Y_i$. The relation of edge lengths between species tree and partition trees is what distinguishes partition models.

The most general EUL model, denoted by M_{EUL}, allows each partition tree $T|Y_i$ to have its own set of edge lengths that are optimized separately per partition tree. We denote a length of e on $T|Y_i$ by $\lambda_i(e)$. The EUL model implies that the species tree T has no defined edge lengths. However, one can display the edge lengths for T, for example, as the weighted average of corresponding edge lengths on partition trees.

In contrast to EUL, the more restrictive EL models assume a relationship between edge lengths of the species tree and all partition trees. The *EL-proportional* model, denoted by $M_{\text{EL}_{\text{prop}}}$, assumes one set of edge lengths, $\lambda(.)$, for the species tree T and rescales the partition trees with specific positive rates $r_1, r_2, \ldots, r_k$ such that the weighted average rate is 1 (i.e., $\frac{\sum_{i=1}^{k} w_i r_i}{\sum_{i=1}^{k} w_i} = 1$, where w_i is the length of the partition alignment S_i). The partition rates r_i are optimized separately for each partition tree.

The second EL model, *EL-equal*, denoted by $M_{\text{EL}_{\text{equal}}}$, is a special case of $M_{\text{EL}_{\text{prop}}}$ with all the partition rates equal to 1 (i.e., $r_1 = r_2 = \cdots = r_k = 1$). This simply means that the edge lengths of partition trees are equal to the lengths of corresponding edges on the species tree. In contrast to M_{EUL}, which optimizes the edge lengths per partition tree, EL models optimize edge lengths, $\lambda(.)$, on the species tree.

PTA Data Structure

In this section, we introduce the PTA data structure, which facilitates detecting and handling partial terraces during tree search and provides an efficient analysis of supermatrices. The PTA consists of the species tree, the set of its induced partition trees, and the set of maps from edges of the species tree to each induced partition tree. In the following we introduce this map and an efficient algorithm to build it.

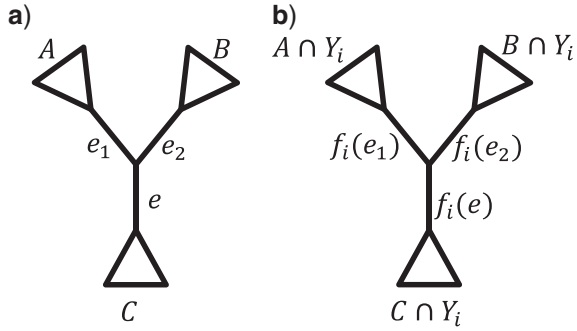

FIGURE 2. (a) Three adjacent edges on species tree T and (b) their corresponding edges on partition tree $T|Y_i$.

Map from the Species Tree onto Partition Trees

Let E denote the edge set of T and E_i the edge set of $T|Y_i$. We represent each edge $e \in E$ by its split $e = A|B$, where A and B are disjoint complementary non-empty subsets of the leaf set X with $|X| = n$. Let ε denote the empty edge or no edge, then for every partition Y_i we introduce the map

$$f_i : E \to E_i \cup \{\varepsilon\},$$
$$f_i(e) = \begin{cases} A \cap Y_i | B \cap Y_i, & \text{if } A \cap Y_i \neq \emptyset \text{ and } B \cap Y_i \neq \emptyset \\ \varepsilon, & \text{otherwise.} \end{cases} \quad (2)$$

Basically, $f_i(e)$, if not equal to ε, is an edge on $T|Y_i$ corresponding to e.

The collection of all maps $F = \{f_1, \ldots, f_k\}$ together with the trees $\{T, T|Y_1, \ldots, T|Y_k\}$ forms the PTA data structure for partition model analyses.

An Efficient Algorithm for Building F

We now describe a dynamic programming algorithm to build F for unrooted bifurcating trees. It first assigns f_i for external edges and then proceeds toward internal edges of the tree. More specifically, let $e = \{x\}|X \setminus \{x\} \in E$ be an external edge then

$$f_i(e) = \begin{cases} \varepsilon, & x \notin Y_i \\ \{x\}|Y_i \setminus \{x\}, & x \in Y_i \end{cases} \quad (3)$$

Obviously, Equation (3) follows directly from Equation (2). Now, let e be an internal edge with two adjacent edges e_1, e_2 (e.g., see Fig. 2a), where $f_i(e_1)$ and $f_i(e_2)$ were already computed, we assign $f_i(e)$ as follows:

$$f_i(e) = \begin{cases} \varepsilon, & f_i(e_1) = f_i(e_2) \\ f_i(e_1), & f_i(e_1) \neq \varepsilon \text{ and } f_i(e_2) = \varepsilon \\ f_i(e_2), & f_i(e_1) = \varepsilon \text{ and } f_i(e_2) \neq \varepsilon \\ e_i^*, & \text{otherwise,} \end{cases} \quad (4)$$

where e_i^* is an edge on $T|Y_i$ adjacent to $f_i(e_1)$ and $f_i(e_2)$ (Fig. 2b). Note that for an edge e where all four neighboring edges have their maps computed, assigning $f_i(e)$ from either side of e will have the same result.

We provide the proof of correctness of Equation (4) in Appendix 1.

The algorithm recursively computes f_i using Equations (3 and 4) by a single post-order tree traversal. Thus, the time complexity of computing f_i is linear in the number of taxa.

The described algorithm builds F in $O(nk)$ time, where n and k are the number of species and partitions, respectively. Note that F has to be recomputed each time the topology of the species tree changed. This sounds expensive at the first glance, but we will now show how to efficiently update the PTA when an NNI is applied to the species tree.

Identifying Unchanged Partition Trees with PTA

For a species tree T and a partition Y_i condition (C.1) provides a check whether an NNI applied to e changes the topology of $T|Y_i$. Using the map notation, (C.1) is equivalent to

> For a partition with a taxon set Y_i the topologies of $T|Y_i$ and $T_{NNI}|Y_i$ are different if and only if all $f_i(e_1), f_i(e_2), f_i(e_3), f_i(e_4) \neq \varepsilon$. (C.2)

Condition (C.2) follows directly from (C.1) and the definition of $f_i(.)$.

Therefore, when an NNI is applied to e, one updates each partition tree $T|Y_i$ using two rules:

1. If all $f_i(e_1), f_i(e_2), f_i(e_3), f_i(e_4) \neq \varepsilon$, then $T_{NNI}|Y_i$ will result from $T|Y_i$ by swapping the corresponding edges. For example, if e_1 and e_3 are swapped on T, then $f_i(e_1)$ and $f_i(e_3)$ are swapped on $T|Y_i$.

2. Otherwise, the topologies of $T_{NNI}|Y_i$ and $T|Y_i$ are identical, T_{NNI} and T belong to a partial terrace. Thus, we keep the tree topology of $T|Y_i$ and have to update $f_i(e)$ according to Equation (4).

Under the EUL partition model when computing the log-likelihood of T_{NNI} we only have to compute the log-likelihood of $T_{NNI}|Y_i$ when rule 1 applies. Otherwise the optimal log-likelihoods of $T|Y_i$ and $T_{NNI}|Y_i$ are equal. This advantage comes from the fact that under the EUL model the edge lengths of partition trees are optimized independently. Therefore, if the topologies of $T_{NNI}|Y_i$ and $T|Y_i$ are identical, there is no need to optimize edges. Thanks to this, the computing time for the EUL model greatly benefits from partial terraces. In fact, when T_{NNI} and T belong to one full terrace, that is, when for all partitions condition (C.2) is not satisfied, no likelihood recomputation is necessary.

More restrictive EL models require additional care. Since under EL models the edge lengths of the species tree and partition trees are *linked*, together with the topological changes of $T|Y_i$ we also have to account for changes in edge lengths. Using the map $f_i(.)$, the edge lengths of the partition tree $T|Y_i$ are computed as

$$\lambda_i(e') = r_i \times \sum_{e \in E: f_i(e) = e'} \lambda(e), \quad \forall e' \in E_i \tag{5}$$

where r_i is the rate of partition Y_i. Therefore, each time map $f_i(.)$ changes, the edge lengths on $T|Y_i$ will also change. If the topology of $T|Y_i$ is not changed by the NNI (i.e., condition (C.2) is not satisfied), four cases are possible, which lead to varying computational time savings (see Appendix 2).

For EL models PTA helps to save computation time during edge length optimization. To speed up the optimization of each edge $e \in E$, in Equation (1) one only has to sum the log-likelihoods over those partitions Y_i where $f_i(e) \neq \varepsilon$. This advantage is a result of using induced partition trees instead of complete partition trees (i.e., with a full set of species instead of Y_i).

Since NNI changes one split of the species tree, namely, the split associated with the edge NNI is applied to, F is recomputed in $O(k)$ time. Thus, the extra computations needed to maintain F are negligible compared with the expensive likelihood computations.

Although we only reformulated the condition for NNI, we note that a similar reformulation applies to SPR and TBR (see Appendix 3). Thus, the PTA can be employed with all common topological rearrangements.

TABLE 2. Benchmark alignments

Type ID	Taxa	Genes	Length	Missing data	Source
DNA1	128	34	29,198	30%	Stamatakis and Alachiotis (2010)
DNA2	180	15	14,912	60%	van der Linde et al. (2010)
DNA3	237	74	43,834	72%	Nyakatura and Bininda-Emonds (2012)
DNA4	279	27	42,666	79%	Fabre et al. (2009)
DNA5	298	3	5074	34%	Bouchenak-Khelladi et al. (2008)
DNA6	372	79	61,199	66%	Springer et al. (2012)
DNA7	404	11	13,158	60%	Stamatakis and Alachiotis (2010)
DNA8	435	18	16,016	73%	Hinchliff and Roalson (2013)
DNA9	767	5	5714	59%	Pyron et al. (2011)
AA10	69	31	8546	35%	Dell'Ampio et al. (2014)
AA11	70	35	11,789	34%	
AA12	72	51	12,548	35%	

PERFORMANCE ASSESSMENT ON REAL ALIGNMENTS

We denote by IQ-TREE$_{\text{PTA}}$, the IQ-TREE version 1.3.3 that implements the PTA data structure. The performance of IQ-TREE$_{\text{PTA}}$ is compared with the standard IQ-TREE implementation and with RAxML version 8.1.24. We also tested RAxML with option $-$U (denoted by RAxML$_{\text{optU}}$), which disregards missing data and results in memory and time saving for

TABLE 3. Comparison of average CPU runtimes between standard RAxML and IQ-TREE and their implementations accounting for missing data: $RAxML_{optU}$ (using −U option) and $IQ\text{-}TREE_{PTA}$

(a) EUL model

Alignments		Average CPU time (hh:mm:ss)				Average $IQ\text{-}TREE_{PTA}$ speedup compared with		
Type ID	Missing data	RAxML	$RAxML_{optU}$	IQ-TREE	$IQ\text{-}TREE_{PTA}$	RAxML	$RAxML_{optU}$	IQ-TREE
DNA1	30%	02:10:00	02:09:12	01:07:19	00:34:04	3.82	3.79	1.98
DNA2	60%	01:01:39	01:02:50	00:54:02	00:21:12	2.91	2.96	2.55
DNA3	72%	04:36:37	04:16:38	02:47:39	00:43:00	6.43	5.97	3.9
DNA4	79%	05:17:08	04:41:59	03:00:21	00:40:24	7.85	6.98	4.46
DNA5	34%	**00:46:31**	**00:48:38**	**02:18:17**	**00:52:03**	**0.89**	**0.93**	**2.66**
DNA6	66%	16:00:06	15:39:31	07:37:09	03:02:09	5.27	5.16	2.51
DNA7	60%	**02:51:22**	**02:50:35**	**05:55:22**	**01:23:02**	**2.06**	**2.05**	**4.28**
DNA8	73%	**02:38:09**	**02:38:15**	**02:43:44**	**01:08:53**	**2.3**	**2.3**	**2.38**
DNA9	59%	**03:47:55**	**03:50:47**	**04:03:38**	**01:41:37**	**2.24**	**2.27**	**2.4**
AA10	35%	02:01:50	02:01:58	01:32:23	00:33:21	3.65	3.66	2.77
AA11	34%	03:22:05	03:20:06	02:11:54	00:50:27	4.01	3.97	2.61
AA12	35%	03:54:33	03:55:09	03:00:38	00:57:09	4.1	4.11	3.16

(b) EL-equal model

DNA1	30%	02:09:55	02:07:04	00:52:32	00:41:36	3.12	3.05	1.26
DNA2	60%	01:07:19	01:08:41	00:56:53	00:26:46	2.51	2.57	2.13
DNA3	72%	07:08:35	06:32:35	06:53:42	02:24:24	2.97	2.72	2.86
DNA4	79%	07:44:45	06:38:53	04:03:20	01:27:09	5.33	4.58	2.79
DNA5	34%	00:38:00	00:39:38	01:16:04	00:47:59	0.79	0.83	1.59
DNA6	66%	26:27:37	25:51:58	15:00:14	07:00:22	3.78	3.69	2.14
DNA7	60%	02:49:49	02:46:45	07:33:26	03:23:30	0.83	0.82	2.23
DNA8	73%	**03:30:37**	**03:28:46**	**09:02:41**	**02:40:41**	**1.31**	**1.3**	**3.38**
DNA9	59%	**03:42:34**	**03:44:51**	**10:07:59**	**03:58:45**	**0.93**	**0.94**	**2.55**
AA10	35%	02:16:18	02:19:46	01:09:14	00:53:16	2.56	2.62	1.3
AA11	34%	03:42:14	03:41:14	01:48:42	01:23:53	2.65	2.64	1.3
AA12	35%	04:58:53	05:00:15	02:15:39	01:46:46	2.8	2.81	1.27

Notes: The speedup is computed as ratios of average CPU runtime of the corresponding implementation to $IQ\text{-}TREE_{PTA}$. The numbers in bold face correspond to alignments where standard IQ-TREE was slower than RAxML, while $IQ\text{-}TREE_{PTA}$ was faster than or as good as RAxML.

alignments with missing data or many gaps (Izquierdo-Carrasco et al. 2011). Note that RAxML implements EUL and EL-equal models, but not the EL-proportional model. Therefore, the last model was only examined with IQ-TREE and $IQ\text{-}TREE_{PTA}$.

We analyzed 12 (9 DNA and 3 AA) alignments (Table 2) with the percentages of missing data ranging from 30% to 79%. These were computed as the percentage of only those gaps in the taxon-by-character supermatrix, which were introduced due to unavailable sequences.

In our analysis each gene is treated as one partition (e.g., alignment DNA1 has 34 partitions). For all partition models we assumed the GTR+Γ (Lanave et al. 1984; Yang 1994) and the LG+Γ (Yang 1994; Le and Gascuel 2008) models for all genes in the DNA and the AA alignments, respectively. For each alignment and the three partition models we performed 10 tree reconstruction runs for each program. The 10 runs correspond to tree searches with different starting trees. For each run $IQ\text{-}TREE_{PTA}$ and IQ-TREE used the same set of 100 starting trees. For RAxML and $RAxML_{optU}$ we could only use one starting tree for each run. Therefore, the speedups between IQ-TREE and $IQ\text{-}TREE_{PTA}$ are explained by accounting for partial terraces, while the speedups between IQ-TREE and RAxML include differences in the inference, starting trees, and in accounting for partial terraces. All computations were carried out on the Vienna Scientific Cluster 3.

CPU Time Comparison

For each partition model and each alignment we computed: (i) the average CPU time for 10 runs for each program; and (ii) the speedup of $IQ\text{-}TREE_{PTA}$ compared with other implementations: the ratio between the average CPU time of each program and that of $IQ\text{-}TREE_{PTA}$.

Under the EUL model $IQ\text{-}TREE_{PTA}$ was the fastest program for 11 out of 12 alignments (Table 3a and Fig. 3a). Averaging over all alignments $IQ\text{-}TREE_{PTA}$ runs about three times faster than the standard IQ-TREE and 3.79 and 3.69 times faster than RAxML and $RAxML_{optU}$, respectively. For four alignments (DNA5, DNA7, DNA8, and DNA9), IQ-TREE was slower than RAxML and $RAxML_{optU}$ (Table 3a). However, $IQ\text{-}TREE_{PTA}$ improved the runtimes for all these alignments compared with IQ-TREE and $IQ\text{-}TREE_{PTA}$ was faster than RAxML except for DNA5.

Under the EL-equal partition model $IQ\text{-}TREE_{PTA}$ was on average 2.0 times faster than IQ-TREE (Table 3b

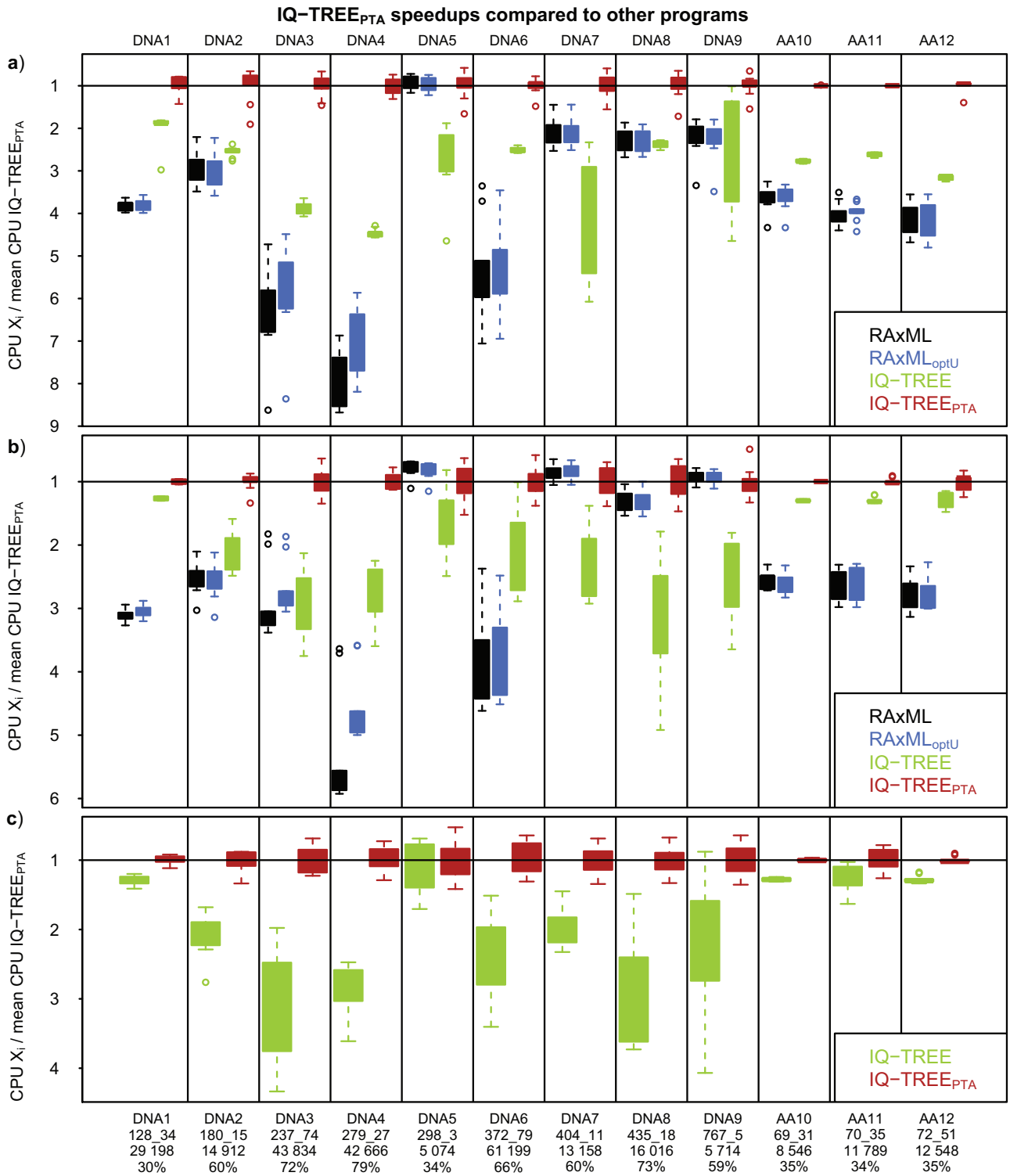

FIGURE 3. CPU time comparisons of different implementations under (a) EUL, (b) EL-equal, and (c) EL-proportional partition models. Each boxplot shows the distribution of the runtime ratios for 10 runs between a comparing program and the mean runtime of IQ-TREE$_{PTA}$. Boxes below the horizontal line indicate instances where corresponding program is slower than IQ-TREE$_{PTA}$. EUL, Edge-Unlinked and EL, Edge-Linked.

and Fig. 3b). The smallest speedups correspond to alignments with the lowest percentages of missing data (DNA1, DNA5, AA10, AA11, AA12). IQ-TREE$_{PTA}$ was slower than RAxML and RAxML$_{optU}$ for DNA5 and DNA7, equally fast for DNA9 and faster for the remaining nine alignments.

Finally, since the EL-proportional model is not available in RAxML it was only examined with IQ-TREE

TABLE 4. Comparison of average CPU runtimes between IQ-TREE standard implementation and IQ-TREE$_{PTA}$ for EL-proportional model

EL-proportional model				
Alignments		Average CPU time (hh:mm:ss)		Average IQ-TREE$_{PTA}$ speedup compared with
Type ID	Missing data	IQ-TREE	IQ-TREE$_{PTA}$	IQ-TREE
DNA1	30%	00:55:13	00:43:01	1.28
DNA2	60%	00:55:46	00:26:29	2.11
DNA3	72%	06:47:13	02:09:24	3.15
DNA4	79%	04:12:05	01:26:35	2.91
DNA5	34%	01:02:34	00:58:22	1.07
DNA6	66%	20:41:03	08:25:47	2.45
DNA7	60%	06:33:40	03:18:51	1.98
DNA8	73%	07:45:27	02:31:37	3.07
DNA9	59%	09:27:22	04:18:24	2.2
AA10	35%	01:10:04	00:54:46	1.28
AA11	34%	02:02:09	01:36:36	1.26
AA12	35%	02:02:02	01:35:31	1.28

Note: The speedup is computed as the ratio of average IQ-TREE runtimes to IQ-TREE$_{PTA}$.

and IQ-TREE$_{PTA}$. IQ-TREE$_{PTA}$ is on average 2.0 times faster than the standard IQ-TREE (Table 4 and Fig. 3c).

Figure 3 shows that IQ-TREE$_{PTA}$ is typically faster than the other three programs. We also computed the speedup between pairs of runs with the same starting trees (Online Appendix Fig. S1, available on Dryad at http://dx.doi.org/10.5061/dryad.v02t3). While IQ-TREE$_{PTA}$ found ML trees faster than IQ-TREE for all the cases, except runs for DNA5 under EL-proportional model, the running times of RAxML and RAxML$_{optU}$ were not significantly different.

Log-Likelihood Comparison

The results discussed in this section are based on the log-likelihoods reported by the respective programs. Note that after a major bug fix in RAxML included in version 8.1.24, the log-likelihoods reported by RAxML and IQ-TREE are directly comparable. That means that for the same tree topology with given branch lengths and same model parameters, the two programs return virtually identical log-likelihoods (Stamatakis A., personal communication).

Under the EUL model the log-likelihoods of the best-found trees obtained by different programs are quite similar with maximal difference of 50 (Fig. 4a) except for DNA1, DNA3, and DNA6, where the IQ-TREE$_{PTA}$ ML trees have log-likelihoods, which are 200, 190, and 100 units higher than RAxML trees, respectively. The log-likelihoods from 10 IQ-TREE$_{PTA}$ runs have consistently small variance, whereas the other programs sometimes show large variances (RAxML for DNA3 and DNA6, IQ-TREE for DNA9).

Under the EL-equal model the log-likelihoods of the best-found trees obtained by different programs are similar for most alignments (Fig. 4b) except for DNA3. The DNA3 alignment showed the largest variance of log-likelihoods over 10 runs: ±150 units for RAxML and ±100 units for IQ-TREE$_{PTA}$.

Under the EL-proportional model the average log-likelihoods for IQ-TREE$_{PTA}$ and IQ-TREE runs agree on nine alignments (Fig. 4c) while not for DNA3, DNA6, and DNA9. For DNA3 and DNA6 IQ-TREE$_{PTA}$ found trees that have on average 100 units higher log-likelihoods than IQ-TREE, whereas for DNA9 the IQ-TREE$_{PTA}$ trees showed 250 units smaller log-likelihoods. Finally, for DNA3 IQ-TREE showed large variance of ±100 units in log-likelihoods obtained from 10 runs.

The tree searches with RAxML and RAxML$_{optU}$ given the same starting tree resulted in identical ML trees. While the tree searches with IQ-TREE and IQ-TREE$_{PTA}$ with the same set of starting trees resulted in topologically different trees, but with sometimes identical log-likelihoods (Online Appendix Figs. S2–S4, available on Dryad). The search algorithm in both versions is the same, but re-optimization of some edge lengths can take place in IQ-TREE when an induced partition topology is not modified by the topological rearrangement, whereas IQ-TREE$_{PTA}$ leaves some or all edge lengths unchanged. This can create small numerical differences in likelihoods and therefore at some point different NNIs can be preferred by IQ-TREE and IQ-TREE$_{PTA}$, which results in different search paths. As a consequence this might lead to topologically different trees obtained by the two versions. These findings indicate that accounting for partial and full terraces influences the tree search and should not be ignored. The influence of partial/full terraces deserves further investigation and the development of new terrace-aware tree searches.

CONCLUSIONS

We introduced a PTA data structure for an efficient phylogenomic inference from supermatrices. PTA consists of the species tree, the set of its induced partition trees, and the set of maps, which map the edges of the species tree to the induced partition trees. This mapping is a key benefit of PTA compared with *pointer meshes* (Stamatakis and Ott 2008; Stamatakis and Alachiotis 2010). The mapping enables an easy topological synchronization between the species tree and partition trees after each topological rearrangement such as NNI presented here. We note that PTA can also be employed for SPR and TBR rearrangements following the conditions of Chernomor et al. (2015) (see Appendix 3 for more details). Thus, PTA is a general data structure that can be incorporated into existing ML software packages (cf. Table 1).

In the presence of missing data, to reduce computation time PTA exploits partial terraces and avoids unnecessary likelihood computation. The use of the induced partition trees in PTA saves time during edge length optimization, which is particularly helpful for the EL partition models. The overhead of maintaining

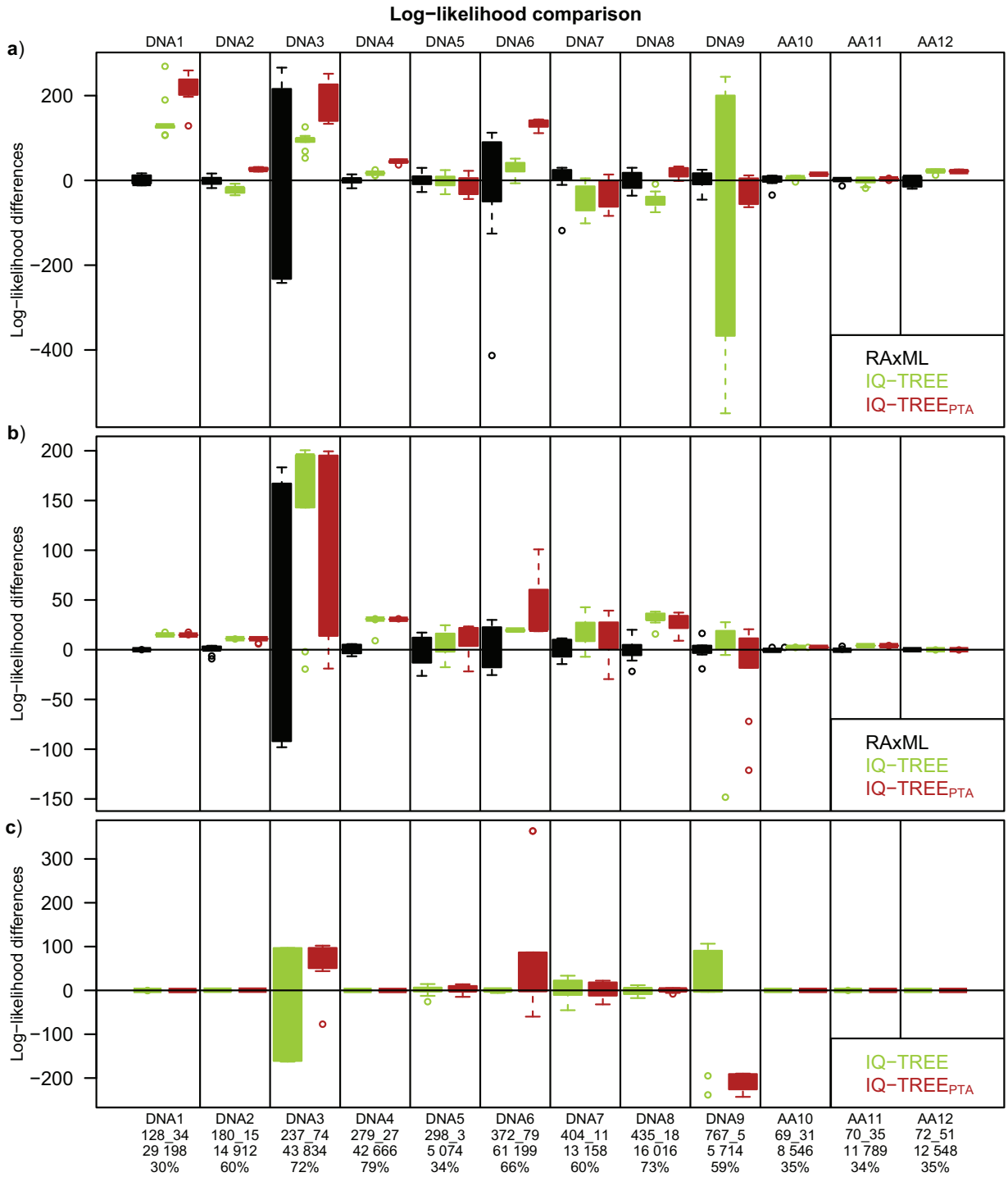

FIGURE 4. Log-likelihood comparisons of different implementations under (a) EUL, (b) EL-equal, and (c) EL-proportional partition models. Each boxplot shows the distribution of the log-likelihood differences for 10 runs between a comparing program and the mean log-likelihood of RAxML standard (panels a and b) or IQ-TREE standard (panel c). Boxes below the horizontal line indicate that the corresponding program has a smaller log-likelihood than RAxML mean log-likelihood (panels a and b) or IQ-TREE mean log-likelihood (panel c). RAxML and RAxML$_{optU}$ have identical log-likelihoods given the same starting tree. We therefore omit RAxML$_{optU}$ in the plot.

the PTA mapping is negligible compared with the time-consuming likelihood computation and the observed speedup is correlated with the amount of missing data.

We implemented PTA in IQ-TREE (Nguyen et al. 2015). Since IQ-TREE applies NNIs to search the tree space, we used the rule to identify partial terraces for NNI-based searches. Analysis on real alignments showed that

accounting for partial terraces, as expected from theory, substantially speeds up the tree search under partition models. Our analysis also revealed a high variability in log-likelihoods for different runs for both RAxML and IQ-TREE. This reinforces the observation (Nguyen et al. 2015) that one should run each program as often as possible to ensure more reliable results and one should also use different programs.

As a further step, we plan to implement the PTA data structure into the phylogenetic likelihood library (Flouri et al. 2015) and to perform the analysis also with SPR. Here, careful considerations are necessary to achieve a high parallel efficiency (Kobert et al. 2014).

While the PTA helps to speed up tree search, it would also be interesting to derive a tree search strategy that specifically exploits the special structure of large (partial) terraces. Here, it is desirable to direct the search to "escape" large partial terraces. Otherwise, a lot of computations might be unnecessarily spent evaluating less promising trees. Nevertheless, the PTA developed here can be useful. For example, one can choose topological rearrangements on the species tree such that all partition trees are changed. The resulting species tree will most likely belong to another partial terrace, thus providing the potential to explore another region of the tree space. To the best of our knowledge, since the introduction of the terrace concept (Sanderson et al. 2011) no terrace-aware search strategy has been introduced. Such a search strategy will be essential to adequately cope with gappy phylogenomic data.

Supplementary Material

Data available from the Dryad Digital Repository: http://dx.doi.org/10.5061/dryad.v02t3.

Funding

This work was supported by the Austrian Science Fund—FWF [grant numbers I-2805-B29 and I-1824-B22].

Acknowledgments

The computational results presented have been achieved using the Vienna Scientific Cluster. The authors thank Frank E. Anderson, Alexandros Stamatakis, Edward Susko, and two anonymous reviewers for helpful comments and suggestions on the article.

Appendix

The Correctness of Equation (4)

Proof.—Figure 2a in the main text illustrates T around e_1, e_2, e, where A, B, C are the three corresponding species sets. We note that $e = (A \cup B)|C$, $e_1 = A|(B \cup C)$, $e_2 = B|(A \cup C)$. From the definition of map $f_i(.)$ it follows that

$$f_i(e) = \begin{cases} (A \cup B) \cap Y_i | C \cap Y_i, & (A \cup B) \cap Y_i \neq \emptyset \text{ and } C \cap Y_i \neq \emptyset \\ \varepsilon, & \text{otherwise} \end{cases}, \tag{A.1}$$

$$f_i(e_1) = \begin{cases} A \cap Y_i | (B \cup C) \cap Y_i, & A \cap Y_i \neq \emptyset \text{ and } (B \cup C) \cap Y_i \neq \emptyset \\ \varepsilon, & \text{otherwise} \end{cases}, \tag{A.2}$$

$$f_i(e_2) = \begin{cases} B \cap Y_i | (A \cup C) \cap Y_i, & B \cap Y_i \neq \emptyset \text{ and } (A \cup C) \cap Y_i \neq \emptyset \\ \varepsilon, & \text{otherwise} \end{cases}. \tag{A.3}$$

We now consider the four cases from Equation (4):

1. If $f_i(e_1) = f_i(e_2) = \varepsilon$, then from Equations (A.2) and (A.3) it follows that at least two of the three intersections $A \cap Y_i$, $B \cap Y_i$, and $C \cap Y_i$ are empty. Therefore from Equation (A.1) we have $f_i(e) = \varepsilon$. Otherwise if $f_i(e_1) = f_i(e_2)$ and are different from ε, then we have $A \cap Y_i = (A \cup C) \cap Y_i$ and $B \cap Y_i = (B \cup C) \cap Y_i$, from which it follows that $C \cap Y_i = \emptyset$ and thus $f_i(e) = \varepsilon$.

2. $f_i(e_1) \neq \varepsilon$ and $f_i(e_2) = \varepsilon$. From $f_i(e_1) \neq \varepsilon$ it follows that $A \cap Y_i \neq \emptyset$ *and* $(B \cup C) \cap Y_i \neq \emptyset$, while from $f_i(e_2) = \varepsilon$, $B \cap Y_i = \emptyset$ *or* $(A \cup C) \cap Y_i = \emptyset$. Since $A \cap Y_i \neq \emptyset$ then $(A \cup C) \cap Y_i \neq \emptyset$, and therefore $B \cap Y_i = \emptyset$ and $C \cap Y_i \neq \emptyset$. Since sets $A \cap Y_i$ and $C \cap Y_i$ are not empty while $B \cap Y_i$ is, then
$$f_i(e) = (A \cup B) \cap Y_i | C \cap Y_i = A \cap Y_i | (B \cup C) \cap Y_i = f_i(e_1).$$

3. $f_i(e_1) = \varepsilon$ and $f_i(e_2) \neq \varepsilon$. Similar to condition (2), we have $f_i(e) = f_i(e_2)$.

4. If $f_i(e_1) \neq f_i(e_2)$ and are both not equal to ε. From $f_i(e_1) \neq \varepsilon$ we have that $A \cap Y_i \neq \emptyset$, from $f_i(e_2) \neq \varepsilon$ it follows that $B \cap Y_i \neq \emptyset$, and since $f_i(e_1) \neq f_i(e_2)$ then $C \cap Y_i \neq \emptyset$. Therefore, $f_i(e) = (A \cup B) \cap Y_i | C \cap Y_i \neq \varepsilon$ is an edge on subtree $T|Y_i$ incident to $f_i(e_1)$ and $f_i(e_2)$ (Fig. 2b).

Thus, Equation (4) is correct.

Applying NNIs for EL Partition Models

Under EL models together with the topological changes we also have to consider the changes of edge lengths on the partition tree after the NNI was applied to T.

If the central edge e has the same corresponding subsplit before and after NNI, the edge lengths on partition tree are not changed. Otherwise, from Equation (5) it follows that the corresponding edge $f_i(e)$ before NNI, if not equal to ε, should have its length decreased by $r_i \lambda(e)$, and the corresponding edge $f_i(e)$ after NNI, if

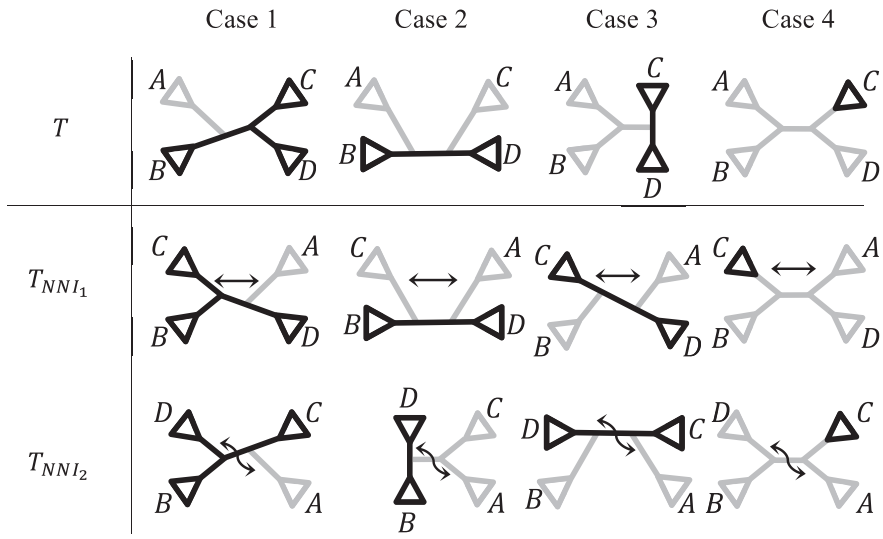

FIGURE A.1. Cases that do not change the topology of the partition tree under the EL models. Each tree is a species tree: before NNI (T, first row) and after NNI (T_{NNI}, second and third rows). The edges and the species sets leading to them are the same as in Figure 1 (e.g., on T the upper left edge is e_1 with the species set A and so on). Here, $f_i(.)$ of gray colored edges is equal to ε and gray triangles correspond to taxa sets, which are absent on the considered partition tree: $T|Y_i, T_{NNI_1}|Y_i$ or $T_{NNI_2}|Y_i$. The black colored parts correspond to the topologies of these induced partition trees. The arrows show edges that were swapped during NNI around the central branch e on T.

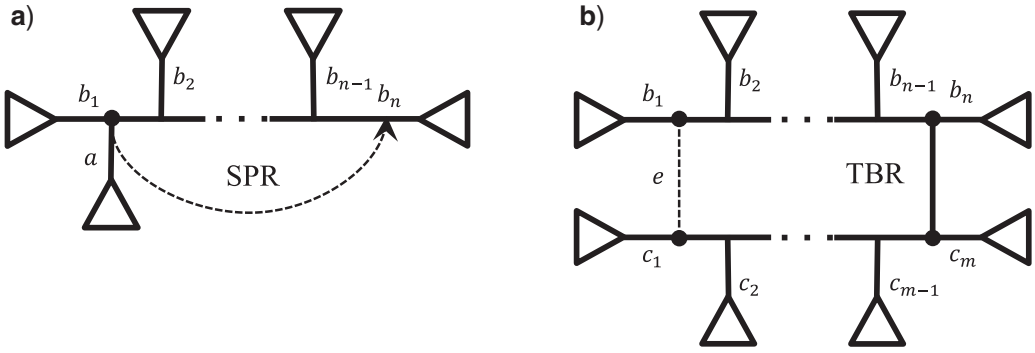

FIGURE A.2. General representations of (a) SPR and (b) TBR, where the triangles denote the subtrees below the corresponding edges. SPR, subtree pruning and regrafting, TBR, tree bisection and reconnection.

not equal to ε, should have its length increased by the same amount. Here, $\lambda(e)$ is the length of e on species tree T.

To save computing time one reoptimizes only edges in the vicinity of topological changes (Stamatakis et al. 2005; Guindon et al. 2010; Nguyen et al. 2015), in the following we assume that for the NNI one has to optimize five edges: e and its incident edges e_1, e_2, e_3, e_4.

There are four different cases depending on the number of edges e, e_1, e_2, e_3, e_4 on T that map to ε before NNI (Fig. A.1).

In case 1, when one out of $f_i(e_1), f_i(e_2), f_i(e_3), f_i(e_4)$ is equal to ε, the two NNIs change the corresponding edge lengths and the resulting partition trees, $T_{NNI_1}|Y_i$ and $T_{NNI_2}|Y_i$, have different edge lengths. In case 2, when the map of two non-incident edges is equal to ε, only one NNI changes the edge lengths. For example, in Figure A.1, where $f_i(e_1) = f_i(e_3) = \varepsilon$, NNI_1 does not change the edge lengths, because the $f_i(.)$ of e, e_1, e_2, e_3, e_4 is not changed, while after NNI_2 $f_i(e)$ is equal to ε. In case 3, when the map of two incident edges and as a result also $f_i(e)$ are equal to ε, both NNIs change the edge lengths, but the induced partition trees resulting from the NNIs have the same edge lengths.

In case 4, when at least any three out of $f_i(e_1), f_i(e_2), f_i(e_3), f_i(e_4)$ are equal to ε, the map $f_i(.)$ of all e, e_1, e_2, e_3, e_4 is not affected by the NNI and is equal to ε before and after an NNI is applied to the species tree T. Therefore, together with the topology also the edge lengths remain unchanged and as a result no recomputation is necessary for such a partition at all.

Although for cases 1–3 the edge lengths of partition trees are affected by the topological rearrangement and also have to undergo optimization, these computations are still less demanding than if the topology of the partition tree would be changed. Therefore, taking into account this information during the tree search still

leads to speedups as shown in the results section for EL models.

Details of Using PTA with SPR and TBR

Complexity of updating.—For an SPR (and similar for a TBR) the time needed to recompute F depends on the length of the move, that is, the number of edges between the cut edge and its reinsertion. Nevertheless, since phylogenetics software packages mainly apply short moves (e.g., RAxML, see Stamatakis et al. 2005), recomputation time of F is close to $O(k)$ also for SPR and TBR.

Detection of partial terraces with PTA in SPR- and TBR-based tree searches.—Here, we present reformulations of Propositions 2 and 3 (Chernomor et al. 2015) in terms of PTA maps. We follow the notations introduced in the original paper. Let T_{SPR} and T_{TBR} denote the species trees obtained from T by one SPR and one TBR, respectively. Any SPR and TBR can be represented in the forms shown in Figure A.2.

Condition for SPR: If an SPR is applied to a species tree T by pruning the subtree below edge a and regrafting it onto b_n (Fig. A.2a), then for a partition with a taxon set Y_i the following is true

(i) the topologies of $T|Y_i$ and $T_{\mathrm{SPR}}|Y_i$ are different, if $f_i(a)$ and at least three from $f_i(b_1), f_i(b_2), \ldots, f_i(b_n)$ are different from ε;

(ii) this SPR corresponds to an SPR on $T|Y_i$ obtained by pruning the subtree below edge $f_i(a)$ and regrafting it onto edge $f_i(b_k)$, where $k = \max_{1 \leqslant x \leqslant n}\{x | f_i(b_x) \neq \varepsilon\}$.

Condition for TBR: If a TBR is applied to a species tree T by cutting edge e and reconnecting b_n and c_m with a new edge (Fig. A.2b), then for a partition with a taxon set Y_i the following is true

(i) the topologies of $T|Y_i$ and $T_{\mathrm{TBR}}|Y_i$ are different, if at least one of the following conditions is satisfied:

- at least one from $f_i(b_1), f_i(b_2), \ldots, f_i(b_n)$ and at least another three from $f_i(c_1), f_i(c_2), \ldots, f_i(c_m)$ are different from ε;
- at least one from $f_i(c_1), f_i(c_2), \ldots, f_i(c_m)$ and at least another three from $f_i(b_1), f_i(b_2), \ldots, f_i(b_n)$ are different from ε;

(iii) this TBR corresponds to a TBR on $T|Y_i$ obtained by cutting the edge $f_i(e)$ and reconnecting edges $f_i(b_k)$ and $f_i(c_h)$, where $k = \max_{1 \leqslant x \leqslant n}\{x | f_i(b_x) \neq \varepsilon\}$ and $h = \max_{1 \leqslant y \leqslant n}\{y | f_i(c_y) \neq \varepsilon\}$.

References

Bininda-Emonds O.R., Gittleman J.L., Purvis A. 1999. Building large trees by combining phylogenetic information: a complete phylogeny of the extant Carnivora (Mammalia). Biol. Rev. Camb. Philos. Soc. 74(2):143–175.

Bininda-Emonds O.R.P., Gittleman J.L., Steel M.A. 2002. The (Super)tree of life: Procedures, problems, and prospects. Annu. Rev. Ecol. Syst. 33:265–289.

Bouchenak-Khelladi Y., Salamin N., Savolainen V., Forest F., Bank M.V., Chase M.W., Hodkinson T.R. 2008. Large multi-gene phylogenetic trees of the grasses (Poaceae): Progress towards complete tribal and generic level sampling. Mol. Phylogenet. Evol. 47(2):488–505.

Chernomor O., Minh B.Q., von Haeseler A. 2015. Consequences of common topological rearrangements for partition trees in phylogenomic inference. J. Comput. Biol. 22(12):1129–1142.

De Queiroz A., Donoghue M.J., Kim J. 1995. Separate versus combined analysis of phylogenetic evidence. Annu. Rev. Ecol. Syst. 26:657–681.

Dell'Ampio E., Meusemann K., Szucsich N.U., Peters R.S., Meyer B., Borner J., Petersen M., Aberer A.J., Stamatakis A., Walzl M.G., Minh B.Q., von Haeseler A., Ebersberger I., Pass G., Misof B. 2014. Decisive data sets in phylogenomics: Lessons from studies on the phylogenetic relationships of primarily wingless insects. Mol. Biol. Evol. 31(1):239–249.

Delsuc F., Brinkmann H., Philippe H. 2005. Phylogenomics and the reconstruction of the tree of life. Nat. Rev. Genet. 6(5):361–375.

Dunn C.W., Hejnol A., Matus D.Q., Pang K., Browne W.E., Smith S.A., Seaver E., Rouse G.W., Obst M., Edgecombe G.D., Sørensen M.V., Haddock S.H., Schmidt-Rhaesa A., Okusu A., Kristensen R.M., Wheeler W.C., Martindale M.Q., Giribet G. 2008. Broad phylogenomic sampling improves resolution of the animal tree of life. Nature 452(7188):745–749.

Eisen J.A. 1998. Phylogenomics: Improving functional predictions for uncharacterized genes by evolutionary analysis. Genome Res. 8(3):163–167.

Fabre P.H., Rodrigues A., Douzery E.J.P. 2009. Patterns of macroevolution among Primates inferred from a supermatrix of mitochondrial and nuclear DNA. Mol. Phylogenet. Evol. 53(3): 808–825.

Felsenstein J. 2004. Inferring phylogenies. Sunderland (MA): Sinauer Associates.

Flouri T., Izquierdo-Carrasco F., Darriba D., Aberer A.J., Nguyen L.T., Minh B.Q., Von Haeseler A., Stamatakis A. 2015. The phylogenetic likelihood library. Syst. Biol. 64(2):356–362.

Guindon S., Dufayard J.F., 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(3):307–321.

Helaers R., Milinkovitch M.C. 2010. MetaPIGA v2.0: Maximum likelihood large phylogeny estimation using the metapopulation genetic algorithm and other stochastic heuristics. BMC Bioinformatics 11:379.

Hinchliff C.E., Roalson E.H. 2013. Using supermatrices for phylogenetic inquiry: An example using the sedges. Syst. Biol. 62(2):205–219.

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

Jobb G., von Haeseler A., Strimmer K. 2004. TREEFINDER: A powerful graphical analysis environment for molecular phylogenetics. BMC Evol. Biol. 4:18.

Kobert K., Flouri T., Aberer A., Stamatakis A. 2014. The divisible load balance problem and its application to phylogenetic inference. Algorithms Bioinformatics 8701:204–216.

Kolaczkowski B., Thornton J.W. 2004. Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. Nature 431(7011):980–984.

Kumar S., Filipski A.J., Battistuzzi F.U., Kosakovsky Pond S.L., Tamura K. 2012. Statistics and truth in phylogenomics. Mol. Biol. Evol. 29(2):457–472.

Kupczok A., Schmidt H.A., von Haeseler A. 2010. Accuracy of phylogeny reconstruction methods combining overlapping gene data sets. Algorithms Mol. Biol. 5:37.

Lanave C., Preparata G., Saccone C., Serio G. 1984. A new method for calculating evolutionary substitution rates. J. Mol. Evol. 20(1):86–93.

Le S.Q., Gascuel O. 2008. An improved general amino acid replacement matrix. Mol. Biol. Evol. 25(7):1307–1320.

Lopez P., Casane D., Philippe H. 2002. Heterotachy, an important process of protein evolution. Mol. Biol. Evol. 19(1):1–7.

Meusemann K., von Reumont B.M., Simon S., Roeding F., Strauss S., Kück P., Ebersberger I., Walzl M., Pass G., Breuers S., Achter V., von Haeseler A., Burmester T., Hadrys H., Wägele J.W., Misof B. 2010. A phylogenomic approach to resolve the arthropod tree of life. Mol. Biol. Evol. 27(11):2451–2464.

Nguyen L.T., Schmidt H.A., von Haeseler A., Minh B.Q. 2015. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol. Biol. Evol. 32(1):268–274.

Nyakatura K., Bininda-Emonds O.R.P. 2012. Updating the evolutionary history of Carnivora (Mammalia): a new species-level supertree complete with divergence time estimates. BMC Biol. 10:12.

Philippe H., Zhou Y., Brinkmann H., Rodrigue N., Delsuc F. 2005. Heterotachy and long-branch attraction in phylogenetics. BMC Evol. Biol. 5:50.

Pyron R.A., Burbrink F.T., Colli G.R., de Oca A.N., Vitt L.J., Kuczynski C.A., Wiens J.J. 2011. The phylogeny of advanced snakes (Colubroidea), with discovery of a new subfamily and comparison of support methods for likelihood trees. Mol. Phylogenet. Evol. 58(2):329–342.

Rokas A., Williams B.L., King N., Carroll S.B. 2003. Genome-scale approaches to resolving incongruence in molecular phylogenies. Nature 425(6960):798–804.

Sanderson M.J., McMahon M.M., Stamatakis A., Zwickl D.J., Steel M. 2015. Impacts of terraces on phylogenetic inference. Syst. Biol. 64:709–726.

Sanderson M.J., McMahon M.M., Steel M. 2011. Terraces in phylogenetic tree space. Science 333(6041):448–450.

Sanderson M.J., Purvis A., Henze C. 1998. Phylogenetic supertrees: Assembling the trees of life. Trends Ecol. Evol. 13(3):105–109.

Springer M.S., Meredith R.W., Gatesy J., Emerling C.A., Park J., Rabosky D.L., Stadler T., Steiner C., Ryder O.A., Janečka J.E., Fisher C.A., Murphy W.J. 2012. Macroevolutionary dynamics and historical biogeography of primate diversification inferred from a species supermatrix. PLoS One 7(11):e49521.

Stamatakis A. 2014. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30(9): 1312–1313.

Stamatakis A., Alachiotis N. 2010. Time and memory efficient likelihood-based tree searches on phylogenomic alignments with missing data. Bioinformatics 26(12):i132–i139.

Stamatakis A., Ludwig T., Meier H. 2005. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics 21(4):456–463.

Stamatakis A., Ott M. 2008. Efficient computation of the phylogenetic likelihood function on multi-gene alignments and multi-core architectures. Phil. Trans. R. Soc. B Biol. Sci. 363(1512):3977–3984.

van der Linde K., Houle D., Spicer G.S., Steppan S.J. 2010. A supermatrix-based molecular phylogeny of the family Drosophilidae. Genet. Res. 92(1):25–38.

Yang Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. J. Mol. Evol. 39(3):306–314.

Yang Z. 1996. Maximum-likelihood models for combined analyses of multiple sequence data. J. Mol. Evol. 42(5):587–596.

Zwickl D.J. 2006. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. The University of Texas at Austin.