Syst. Biol. 61(1):90–106, 2011
© The Author(s) 2011. Published by Oxford University Press, on behalf of the Society of Systematic Biologists. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
DOI:10.1093/sysbio/syr095
Advance Access publication on December 1, 2011

SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees

KEVIN LIU[1], TANDY J. WARNOW[1], MARK T. HOLDER[2], SERITA M. NELESEN[3], JIAYE YU[2], ALEXANDROS P. STAMATAKIS[4], AND C. RANDAL LINDER[5,*]

[1]*Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA;* [2]*Department of Ecology and Evolutionary Biology, University of Kansas at Lawrence, Lawrence, KS 66045, USA;* [3]*Department of Computer Science, Calvin College, Grand Rapids, MI 49546, USA; and* [4]*Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, D-69118 Heidelberg, Germany;* [5]*Section of Integrative Biology, School of Biological Sciences, University of Texas at Austin, Austin, TX, USA;*
**Correspondence to be sent to: The University of Texas at Austin, Austin, TX 78712, USA; E-mail: rlinder@mail.utexas.edu.*

Received 20 August 2010; reviews returned 9 November 2010; accepted 3 June 2011
Associate Editor: Laura Kubatko

Abstract.—Highly accurate estimation of phylogenetic trees for large data sets is difficult, in part because multiple sequence alignments must be accurate for phylogeny estimation methods to be accurate. Coestimation of alignments and trees has been attempted but currently only SATé estimates reasonably accurate trees and alignments for large data sets in practical time frames (Liu K., Raghavan S., Nelesen S., Linder C.R., Warnow T. 2009b. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science. 324:1561–1564). Here, we present a modification to the original SATé algorithm that improves upon SATé (which we now call SATé-I) in terms of speed and of phylogenetic and alignment accuracy. SATé-II uses a different divide-and-conquer strategy than SATé-I and so produces smaller more closely related subsets than SATé-I; as a result, SATé-II produces more accurate alignments and trees, can analyze larger data sets, and runs more efficiently than SATé-I. Generally, SATé is a metamethod that takes an existing multiple sequence alignment method as an input parameter and boosts the quality of that alignment method. SATé-II-boosted alignment methods are significantly more accurate than their unboosted versions, and trees based upon these improved alignments are more accurate than trees based upon the original alignments. Because SATé-I used maximum likelihood (ML) methods that treat gaps as missing data to estimate trees and because we found a correlation between the quality of tree/alignment pairs and ML scores, we explored the degree to which SATé's performance depends on using ML with gaps treated as missing data to determine the best tree/alignment pair. We present two lines of evidence that using ML with gaps treated as missing data to optimize the alignment and tree produces very poor results. First, we show that the optimization problem where a set of unaligned DNA sequences is given and the output is the tree and alignment of those sequences that maximize likelihood under the Jukes–Cantor model is uninformative in the worst possible sense. For all inputs, all trees optimize the likelihood score. Second, we show that a greedy heuristic that uses GTR+Gamma ML to optimize the alignment and the tree can produce very poor alignments and trees. Therefore, the excellent performance of SATé-II and SATé-I is not because ML is used as an optimization criterion for choosing the best tree/alignment pair but rather due to the particular divide-and-conquer realignment techniques employed. [Alignment; maximum likelihood; phylogenetics; SATé.]

Phylogenetic estimation using nucleotide and amino acid sequences typically proceeds in two phases: a multiple sequence alignment is produced for a set of unaligned sequences and then a tree is estimated using the multiple sequence alignment. Given sufficient computational resources, these "two-phase" methods can analyze data sets of thousands of taxa and aligned sites but can have poor accuracy on data sets that have evolved under high substitution and indel (insertion and deletion) rates (Liu, Raghavan et al. 2009; Liu et al. 2010).

Methods that simultaneously estimate a multiple sequence alignment and a phylogenetic tree from unaligned sequences have been developed over the course of two decades, starting in the 1990s (Hein 1990; Wheeler and Gladstein 1994; Wheeler 1995, 1996.) Today's methods are broadly classified into two categories: methods based upon parametric statistical models of sequence evolution that incorporate substitution and indel events (e.g., TKF1 (Thorne et al. 1991) and TKF2 (Thorne et al. 1992)) and nonparametric methods that resemble maximum parsimony. Current methods that fall into the parametric category include BAli-Phy (Redelings and Suchard 2005; Suchard and Redelings 2006), StatAlign (Novák et al. 2008), the method publishedby Lunter et al. (2005), and ALIFRITZ (Fleissner et al. 2005). Methods in the second category, by contrast, typically attempt to minimize the total number of evolutionary events or the total cost of these events. The most frequently used method of this type is POY (Varón et al. 2010) but the basic approach was introduced by Sankoff (1975), and there are other methods of this type (Sankoff and Cedergren 1983; Lancia and Ravi 1999; Liu, Nelesen et al. 2009). Of all these methods for simultaneous estimation of alignments and trees, only SATé (Liu, Raghavan et al., 2009) has been demonstrated to produce alignments and trees that are more accurate than two-phase methods on large data sets. The Megaphylogeny method (Smith et al. 2009) can analyze very large data sets and is potentially promising; however, it has not yet been compared with other methods.

SATé uses an iterative procedure to compute a series of alignment/tree pairs. The first tree is produced by running RAxML (Stamatakis 2006) on a MAFFT (Katoh et al. 2005; Katoh and Toh 2008) alignment, running each method in its default (and most accurate) setting. To compute a new alignment, the current tree is used to produce a division of the sequence data set into disjoint subsets; MAFFT alignments are computed on each subset, and these alignments are merged into an alignment on the entire data set using Muscle (Edgar 2004a,b);

finally, a GTR+Gamma (Rodriguez et al. 1990) ML tree is computed using RAxML on the new alignment. If the ML score improves, the new alignment/tree pair is accepted; otherwise, SATé realigns the sequences using a somewhat different technique. This process repeats until a stopping criterion is satisfied, and the alignment/tree pair is returned that had the best GTR+Gamma ML score. On moderate to difficult data sets (sequences with high levels of substitution and/or indels), SATé produces much more accurate alignments and trees than existing two-phase techniques and does so fairly rapidly (Liu, Raghavan et al. 2009). However, the topological accuracy of trees estimated using SATé on data simulated under difficult model conditions differs significantly from the accuracy of trees produced using maximum likelihood (ML) estimation on the true alignment, suggesting that further improvement might still be obtained through a re-design of the algorithm.

Because the sequence evolution model for ML phylogeny estimation used in SATé is GTR+Gamma with gaps treated as missing data, SATé is not attempting to solve ML under a sequence evolution model that includes indels. This suggests that SATé is not likely to be statistically consistent under a model in which sequences evolve with indels. Even so, SATé has very good performance on simulated data under a wide range of model conditions (Liu, Raghavan et al. 2009). Therefore, although SATé uses ML estimation to select among its generated alignment/tree pairs, it remains an open question whether the good performance of SATé is due to its ability to optimize alignments and trees under GTR+Gamma or to some other aspect of its algorithm design.

In this paper, we make progress on understanding why SATé produces excellent results. We show definitively that the reason SATé is highly accurate is *not* due to the use of the ML criterion to select among the alignment/tree pairs it generates. We show empirically that whether SATé uses ML to choose among tree/alignment pairs or simply takes the last tree alignment pair generated after at least three to five iterations has little or no effect on the topological accuracy of the tree produced. In addition, we provide a mathematical proof that *all trees are optimal solutions for all inputs* when optimizing the Jukes–Cantor (JC) (Jukes and Cantor 1969) ML score—treating gaps as missing data—while allowing the alignment and tree to change arbitrarily (Appendix 1). Finally, we show that a greedy heuristic that seeks to optimize the GTR+Gamma ML score (again treating gaps as missing data) can produce very poor alignments and trees (Appendix 1).

These insights led us to explore the algorithmic techniques in SATé and to the design of a new method, SATé-II. We focused on the decomposition technique used in SATé and observed that (i) on some large data sets, it produced very large subsets and (ii) these subsets sometimes contained highly dissimilar sequences. We therefore changed the decomposition technique so that all subsets were small (at most 200 sequences), which improved both the algorithm's speed and the accuracy of the alignments. We also explored other modifications (see below) that improved the method but to a lesser degree.

We show that SATé-II produces more accurate alignments and trees than SATé (which we now call SATé-I) and two-phase methods. Furthermore, SATé-II is designed so it can be used with any alignment method—not just MAFFT—so it is now a metamethod that takes an existing multiple sequence alignment method as an input parameter. We show that alignment methods boosted by SATé-II are significantly more accurate than their unboosted versions and that trees based upon these improved alignments are more accurate than trees based upon the unboosted alignments. Importantly, SATé-II can be used to analyze much larger data sets than SATé-I, and we demonstrate its scalability and accuracy on biological data sets with up to almost 28,000 sequences. Thus, SATé-II provides, for the first time, an automated method that is capable of producing highly accurate sequence alignments and trees for very large molecular data sets.

Materials and Methods

Algorithm and Study Design

We had two objectives in this study. The first objective was to understand which aspects of SATé-I's algorithm design contributed the most to its tree and alignment accuracy and, in particular, to understand the extent to which using ML to select the best alignment/tree pair contributed to SATé-I's accuracy. The second objective was to use what we learned about the impact of the algorithmic design on phylogenetic accuracy to produce a more accurate coestimation method.

Designing SATé-II.—In designing SATé-II, we sought to improve the quality of its alignments, the topological accuracy of its trees, and its speed over SATé-I. To make the changes clear, we first describe SATé-I and then show how SATé-II differs.

SATé-I algorithm.—SATé-I produces a starting tree by estimating a set of alignments on a data set, computing GTR+Gamma ML trees using RAxML on each alignment and then keeping the tree with the best GTR+Gamma ML score. SATé-I then iterates, where each iteration begins using a divide-and-conquer strategy, guided by the current tree, to produce nonoverlapping subsets of taxa. These subsets are realigned using MAFFT (Katoh et al. 2005; Katoh and Toh 2008) and then merged using Muscle (Edgar 2004a, 2004b) to produce a new alignment on the full set of taxa. The iteration finishes by computing a GTR+Gamma ML tree on the new alignment using RAxML.

The divide-and-conquer strategy used in SATé-I is the CT-5 (center tree-5) decomposition (Liu, Raghavan et al. 2009). The CT-5 decomposition takes a branch *e* inthe input tree T and extends outwards from this branch to define a binary tree t within T with up to 32 (i.e., 2^5) leaves by including all nodes in T that are at most 4 branches distant from e. (More generally, a CT-k decomposition takes an internal branch and extends outwards, creating a subtree that contains all nodes at most $k-1$ branches distant from e.) Once the subtree based upon a CT-5 decomposition is defined, removing it from the original tree separates the leaf set into at most 32 subsets. SATé-I then computes alignments on each subset using MAFFT and merges these alignments into a single alignment on the entire data set using Muscle. Finally, SATé-I computes a GTR+Gamma ML tree on this alignment using RAxML. This method of producing subsets of taxa does not constrain the maximum number of taxa in a subset.

SATé-II algorithm.—SATé-II differs from SATé-I in three ways, the first two of which have the greatest effect on alignment and tree accuracy. Most importantly, SATé-II uses a different divide-and-conquer strategy. The CT-5 decomposition used in SATé-I sometimes produces very large subsets, and these subsets sometimes contain very dissimilar sequences, often negatively impacting alignment and tree accuracy. In contrast, SATé-II divides the sequences into two sets by removing the longest branch in the current tree (Fig. 1), where branch lengths are given by the ML tree on the previously computed alignment. Then SATé-II recurses, selecting the longest branch in each subset to further subdivide the taxa, until each subset has at most 200 taxa. (Note that this strategy always performs at least one realignment on all data sets because the first decomposition is always performed, no matter how small the tree is.) This decomposition technique is guaranteed to produce small subsets and ones that are less likely to contain very dissimilar sequences. Second, for merging subalignments, we determined that Opal (Wheeler and Kececioglu 2007) produced more accurate merges than Muscle, leading to more accurate alignments and trees. Finally, SATé-II uses only the RAxML tree on the MAFFT alignment as its starting tree, rather than computing several possible starting trees. This improves running time without affecting the final accuracy of the final alignment and tree.

Using this basic algorithm as a foundation, we designed three variants of SATé-II (SATé-II simple, SATé-II ML, and SATé-II fast) to determine the fastest and most accurate versions, as well as to directly test the impact of using the ML score under GTR+Gamma (treating gaps as missing data) to select among the tree/alignment pairs generated during the SATé-II iterations. The variants differ in the number of iterations performed and the criterion used to select the tree/alignment pair to be returned from the set of generated tree/alignment pairs. However, each method uses the same iterative strategy, with each iteration producing a new alignment and then an ML tree on that new alignment. SATé-II simple runs until a time-limited or iteration-limited stopping rule of the user's choosing is satisfied, returning the final

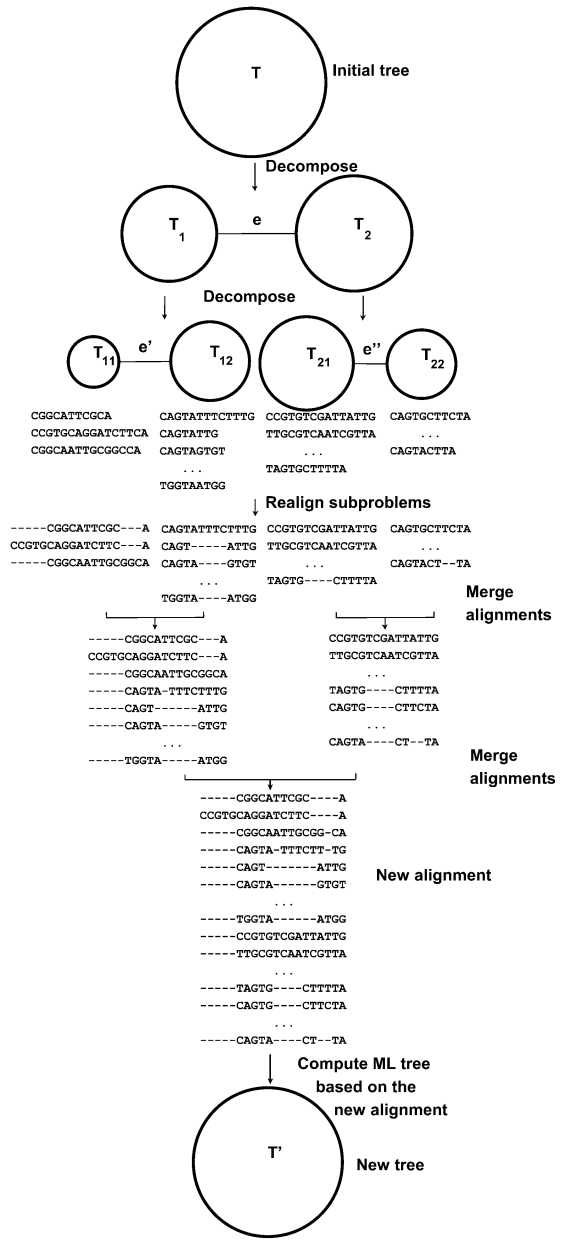

FIGURE 1. A cartoon of a SATé-II iteration. The tree T is divided into two subtrees T_1 and T_2 using a longest branch e in T, and this process is repeated until each subtree has no more than the maximum allowable number of leaves; here, we show this process producing four subtrees T_{11}, T_{12}, T_{21}, and T_{22}. Then the sequences at the leaves in each subtree are aligned, and the resultant alignments on subsets are merged, pairwise, in the reverse order in which they were created. Finally, a new ML tree T' is estimated on the resultant alignment.

tree/alignment pair produced by the iterative strategy. The use of ML within SATé-II simple is limited to the generation of trees from estimated alignments and is not used as a criterion to determine the final tree/alignment pair. SATé-II ML performs the same iterations as SATé-II simple but compares all tree/alignment pairs produced and returns the tree with the best GTR+Gamma ML score and the alignment from which it was estimated. The third variant, SATé-II fast, runs only until an iteration produces a worse ML score than the previous iteration; it then stops and returns the penultimate tree/alignment pair.

Data Sets for the Study

We used simulated data sets to explore the performance of SATé-II's variants relative to one another and in comparison to SATé-I and two-phase methods. We also used simulated data sets to evaluate the heuristic we developed to directly optimize ML on both the alignment and the tree. We used biological data sets to evaluate the final SATé-II algorithm in comparison to SATé-I and two-phase methods.

Simulated data sets.—Our simulated data sets included the data sets originally generated to evaluate SATé-I (Liu, Raghavan et al., 2009), and ranged from 100 to 1000 taxa. We varied rates of substitutions and indels, and included different gap length distributions, to produce a diverse set of data sets of varying levels of difficulty. Since the 100-taxon data sets in Liu, Raghavan et al. (2009) were relatively easy to align, we added some more difficult 100-taxon models that were produced using the same methodology as in Liu, Raghavan et al. (2009) (Table S1 in online Appendix 1, available from http://www.sysbio.oxfordjournals.org/). We used r8s (Sanderson 2003) version 1.7 to generate nonultrametric birth–death model trees and ROSE (Stoye et al. 1998) version 1.3 to simulate evolution of sequences down the model trees (see Appendix 2 for specific commands).

All simulated data sets were produced by simulating evolution with indels and substitutions, under the GTR+Gamma+Gap model (see Liu, Raghavan et al. (2009) and Appendix 2). Therefore, substitutions were drawn from the GTR+Gamma model, and gap probabilities were scaled with substitution probabilities. We simulated under three different gap length distributions (short, medium, and long), three different numbers of taxa (100, 500, and 1000), and five different combinations of tree height and gap probability. In total, we simulated under 45 model conditions. For each model condition, we generated 20 model trees, and we then generated one data set for each model tree. Thus, we produced 900 data sets overall. Some of these model conditions are "easy" because SATé-I and all the two-phase methods produced trees with missing branch rates (i.e., the false-negative rate, which is the percentage of the branches in the true tree that are not present in the estimated tree) essentially identical to that of RAxML on the true alignment (Liu, Raghavan et al. 2009). For the other model conditions, the two-phase methods differed in their abilities to produce accurate trees.

For each simulated data set, we reported various values, as follows. We compared each estimated alignment to the true alignment using the alignment SP-FN error rate, which is the percentage of the number of homologous pairs of nucleotides in the true alignment that are missing in the estimated alignment (Thompson et al. 1999). The topology of each estimated tree was compared to that of the *potentially inferable model tree* (PIMT), which consisted of the model tree with all zero-event branches (i.e., branches on which no sequence change occurred) contracted; using PIMTs prevents methods from being penalized for failing to recover branches that cannot be inferred from the data. We report the missing branch rates (the percentage of branches in the true tree that do not appear in the estimated tree) and false-positive rates (the percentage of branches in the estimated tree that do not appear in the true tree) of the estimated trees with respect to their PIMTs. Finally, we report the likelihood scores returned by RAxML, normalized by the likelihood score produced by RAxML on the MAFFT alignment for that data set. Normalized likelihood scores less than 1.0 indicate alignments for which RAxML produced a tree with a better likelihood than it produced using an MAFFT alignment, whereas normalized likelihood scores greater than 1.0 indicate alignments for which RAxML produced a tree with a likelihood score worse than that it produced using a MAFFT alignment.

Biological data sets.—We explored performance on the six data sets studied in Liu, Raghavan et al. (2009) and two much larger data sets from Cannone et al. (2002): the 16S.T data set (rRNA sequences from the three domains of life and two organelles) and the 16S.B.ALL data set (rRNA sequences from the Bacteria domain). All these data sets are from Robin Gutell's rRNA Web site (http://www.rna.ccbb.utexas.edu/DAT/3C/Alignment/) and have the virtue of having their curated alignments inferred using sequence covariation, secondary structure, and higher level structural features (Gutell et al. 1994; Cannone et al. 2002; Gutell et al. 2002). While no alignment of biological data can be guaranteed to be completely accurate, because of the approaches employed by Gutell, the biological data sets we used are considered some of the most reliable curated nucleic acid sequence alignments.

We "cleaned" the 16S.T and 16S.B.ALL data sets using the same techniques employed in Liu, Raghavan et al. (2009) to remove sites occupied entirely by gaps and taxa containing more than 50% unsequenced characters. Cleaning removed 3% of the 16S.T sites and 11.8% of the 16S.B.ALL sites. No taxa were removed from the 16S.T data set, but 23.2% of the taxa in the 16S.B.ALL data set were removed. Table S2 lists cleaning statistics. The cleaned 16S.T curated alignment had 7350 sequences, 11,856 aligned sites, 34.5% average p-distance, 90.1% maximum p-distance, 87.4% indels, and average gap length of 12.1. The cleaned 16S.B.ALL data set had 27,643 sequences, 6857 aligned sites, 21.0% average p-distance, 76.9% maximum p-distance, 80.0% indels, and average gap length of 4.9. Thus, both the 16S.T and 16S.B.ALL data sets present substantial challenges to alignment estimators, due to the curated alignments' substitution saturation and high gappiness.

We computed curation-based trees on the biological data sets as follows. First, we ran RAxML on each curated alignment to produce an initial topology. We then determined the support on each edge using a bootstrap analysis (RAxML) on the curated alignments. The number of bootstrap replicates depended upon the data set: 346 replicates on the 16S.T data set, 444 replicates on the 16S.B.ALL data set, and 500 bootstrap replicates on the other data sets. Branches with at least 75% bootstrap support were retained in the curation-based tree (Hillis and Bull 1993). Table S3 in the online Appendix 1 gives these statistics for all the biological data sets.

To evaluate estimated alignments on biological data sets, we compared them to the curated alignments using the alignment SP-FN error rate. To evaluate estimated trees, we used the missing branch rate to compare their topologies to the curation-based trees. We recognize that comparison of estimated trees with the curation-based trees is not a measure of the true accuracy of the estimated trees; however, given the high quality of the rRNA-curated alignments we used, the comparison is likely to be informative about the quality of the estimated trees, particularly because low-support branches are excluded from the comparison.

Two-Phase Methods

We aligned all the simulated and biological data sets—with the exception of the two largest biological data sets, 16S.T and 16S.B.ALL—using ClustalW (Thompson et al. 1994) version 2.0.4, MAFFT version 6.240, Muscle version 3.7, Prank+GT (as described in Liu, Raghavan et al. (2009)), and Opal version 1.0.2. Each alignment method was run in its default setting. For each resulting alignment, we ran RAxML to estimate an ML tree.

Due to the high number of taxa in the 16S.T and 16S.B.ALL data sets, we used the PartTree (Katoh and Toh 2007) algorithm in MAFFT and the Quicktree (Thompson et al. 1994) option in ClustalW to estimate multiple sequence alignments and RAxML to estimate ML trees on each alignment. On the 16S.B.ALL data set, we also used FastTree (Price et al. 2010) to estimate an ML tree on the PartTree alignment. Appendix 3 provides complete details, including program versions and commands, for our experiments with these methods.

Variations of SATé-I and SATé-II Studied

Except for the two largest data sets (16S.T and 16S.B.ALL), we ran SATé-I and SATé-II in their default settings. The default settings for SATé-I follow those given in Liu, Raghavan et al. (2009), running SATé-I until 24 h had passed, at which point we returned the last tree.

Both SATé-I and SATé-II analyses of the two largest data sets were modified to enable them to complete. First, instead of the starting tree being based upon an MAFFT alignment, we based it on the PartTree (Katoh and Toh 2007) alignment because MAFFT cannot run on data sets of these sizes. Of the alignment methods that can handle very large numbers of taxa, the PartTree method yields among the most accurate alignments in our experience (Liu et al. 2010).

To compute the starting tree using SATé-I, we used RAxML on the PartTree alignment. For the SATé-II analysis of the 16S.T data set, we used RAxML to compute the ML tree on the PartTree alignment. For the SATé-II analysis of the 16S.B.ALL data set, we used FastTree (a less accurate but faster ML method) (Price et al. 2010) to estimate an ML tree on the PartTree alignment. Thus, for the 16S.T data set, SATé-I and SATé-II started with the same starting tree, whereas for the 16S.B.ALL data set, SATé-II started with a less accurate (but more quickly computed) starting tree. We made additional modifications to SATé-I and SATé-II on the 16S.B.ALL data set, due to its large size. We used Muscle to merge subalignments produced by both SATé-I and SATé-II since Opal failed to run for such large data sets. We also used FastTree instead of RAxML inside each iteration of SATé-II to compute trees on each newly estimated alignment.

The modifications to SATé-II for the largest data sets had the effect of making SATé-II run faster and use less memory than SATé-I but might also have reduced topological and alignment accuracy. For example, on the largest data set (16S.B.ALL), SATé-II started with a less accurate starting tree and used a less accurate ML estimation technique within its iterative strategy than SATé-I. Since our objective was to compare SATé-II to SATé-I primarily with respect to tree and alignment error, the effect of these modifications likely biases the comparison in favor of SATé-I rather than in favor of SATé-II.

Understanding the Role of ML in SATé

ML has two roles in SATé-I: It is used in each iteration to compute a tree on each new alignment and also used to select an alignment/tree pair from the sequence of such pairs produced during the iterative strategy. Within the iterative procedure where the tree is inferred, ML is used rather than other phylogenetic method estimators—for example, maximum parsimony and neighbor joining—because the alternatives would likely produce less accurate trees, particularly on data sets with high rates of substitution. RAxML, in particular, is used because it is able to analyze large data sets and is known to do a good job of optimizing likelihood scores.

With respect to the second use of ML—selecting a single alignment/tree pair from the sequence of such pairs—it is less clear that it is helping to find the best alignment/tree pair. We therefore, undertook two studies to test whether optimizing likelihood on a tree/alignment pair will produce better alignments and trees. We provide a mathematical proof that seeking to maximize the JC likelihood score (treating gaps as missing data) and allowing the alignment to change is a poor optimality criterion because all trees are optimal solutions (see Appendix 1). It is still unknown whether this statement is true or false for the GTR model (see Appendix 1).

We also explored whether directly seeking to produce an alignment and tree that optimized the GTR+Gamma ML score, treating gaps as missing data, was a poor optimization approach for phylogeny estimation. We designed a simple, greedy heuristic to optimize GTR+Gamma ML scores, and tested it on some of our 1000 taxon simulated data sets (see online Appendix 2).

Computational Resources

To perform simulations and run the two-phase methods on simulated and biological data sets—other than the 16S.T and 16S.B.ALL data sets—we used a heterogeneous Condor cluster (Litzkow 1987) at the University of Texas at Austin, consisting of computers generally comparable to a 3.0 Ghz Intel 32-bit CPU with between 512 MB and 4 GB of main memory. For the SATé-I and SATé-II runs on simulated and biological data sets—other than the 16S.T and 16S.B.ALL data sets—we used the Mastodon computing cluster at the University of Texas at Austin, consisting of Intel 32-bit CPUs running at 2.6, 2.66, or 3.06 Ghz with 4 GB main memory per CPU.

To run SATé-II on the 16S.T and 16S.B.ALL data sets and to run SATé-I on the 16S.T data set, we used a 64-bit computing cluster at the University of Texas at Austin, consisting of machines with 8-core 2.83 Ghz Intel Xeon 64-bit CPUs with 32 GB main memory per CPU. Due to the memory requirements of SATé-I and the two-phase methods on the 16S.B.ALL data sets, we used two machines with very large shared memory, each having a 16-core 64-bit AMD Opteron CPU running at 2.5 Ghz and with either 128 GB or 256 GB main memory. This machine was also used to construct the curation-based tree for the 16S.T and 16S.B.ALL data set.

For the experiment with the greedy heuristic, we used the SSE3-based SIMD sequential version of RAxML to conduct all computational experiments on a Sun x4600 system with 32 cores and 64 GB of main memory. Finally, to assess support on the curation-based tree on the curated alignment for the 16S.B.ALL data set, we performed a bootstrap analysis using the Ranger supercomputer at the Texas Advanced Computing Center (TACC) at the University of Texas at Austin. The Ranger supercomputer has 3936 SunBlade x6420 16-core 2.3 Ghz compute nodes for a total of 62,976 processing cores and 123 TB of main memory.

Results

Simulated Data

Role of ML.—We began by assessing SATé-II simple and SATé-II ML to determine whether using ML to choose the best tree/alignment pair was the cause of SATé's excellent performance. SATé-II ML and SATé-II simple were run for 1 week on the 1000-taxon data sets and 3 days on the 500- and 100-taxon data sets. We tested for statistically significant differences (false-discovery rate corrected (Benjamini and Hochberg 1995) pairwise *t*-tests) in the final trees and alignments on the moderate to difficult data sets. Although SATé-II ML had statistically better ML scores than SATé-II simple (as would be expected since SATé-II ML is explicitly optimizing the ML score), the differences in their final trees' topological accuracy was not statistically significant (Fig. 2 for the 1000-taxon moderate-to-difficult data sets and Fig. S1 in online Appendix 1 for the 100- and 500-taxon data sets of similar difficulty). This result shows clearly that using ML to choose the best tree/alignment pair is not responsible for the quality of the tree produced by SATé. Interestingly, SATé-II ML did produce alignments that had lower SP-FN errors than SATé-II simple, and although these improvements were statistically significant, they were quite small. At this time, it is not clear why choosing the best ML score affects alignment accuracy, but evidently, the small improvement in alignment accuracy is insufficient to produce an improvement in the tree topology.

To further understand the relationship between ML score and tree and alignment accuracy, we examined the

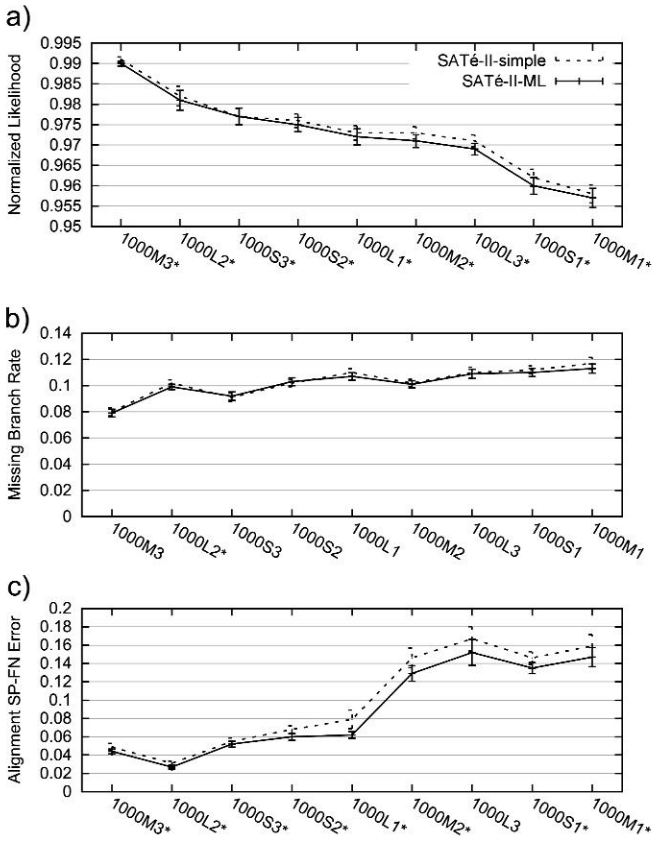

FIGURE 2. Comparison of 1-week runs of SATé-II simple and SATé-II ML on simulated 1000-taxon moderate-to-difficult data sets. Graphs from top to bottom are (a) normalized ML scores, (b) missing branch rates, and (c) alignment SP-FN errors. For each data point, $n = 20$; bars are standard errors. For normalized ML score, missing branch rate and alignment SP-FN error charts, model conditions marked with an asterisk (*) indicate that SATé-II ML was significantly better than SATé-II simple by Benjamini and Hochberg (1995) corrected pairwise *t*-tests ($n = 40$, $\alpha = 0.05$).

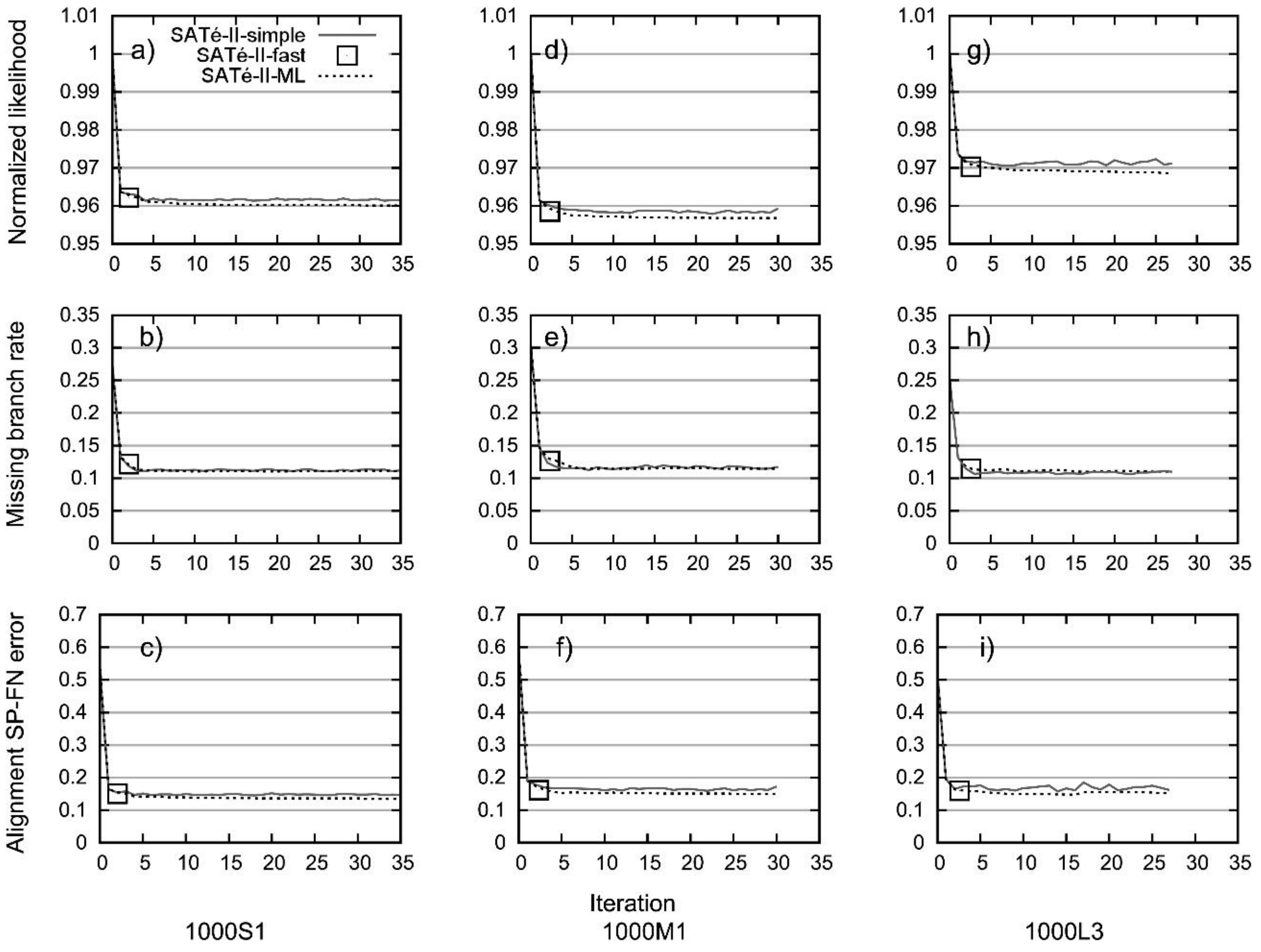

FIGURE 3. Timeplots are shown for the three SATé-II variants on the three most difficult 1000-taxon simulated model conditions. From top to bottom, normalized ML scores, missing branch rates, and alignment SP-FN errors are shown. The zero iteration is the starting alignment and tree. For each model condition, only iterations that were finished by all SATé-II runs on all 20 data sets are included. $n = 20$ for each datapoint.

improvement over time of ML score, tree accuracy, and alignment accuracy of SATé-II simple and SATé-II ML (Fig. 3 for the 1000 taxon data sets, online Appendix 1 Fig. S2 for the 500-taxon data sets and online Appendix 1 Fig. S3 for the 100-taxon data sets). ML scores and alignment and topological accuracy for SATé-II simple and SATé-II ML tended to improve in parallel, and the rate of improvement dramatically slowed or stopped within a small number of iterations. Thus, whether ML is used to guide the search for the best tree/alignment pair or not, the initial iterations have a large impact on tree and alignment accuracy, and the magnitude of improvement decreases rapidly.

We also explored the performance of a heuristic (see online Appendix 2) that builds an alignment greedily, one sequence at a time while explicitly optimizing the GTR+Gamma ML score during each sequence addition. We tested this heuristic on a collection of 1000 taxon data sets generated in this study. Very poor trees and alignments were obtained for the moderate and difficult model conditions as a result of optimizing GTR+Gamma ML, treating gaps as missing data (online Appendix 2). Although it is possible that this result is due in part to the operation of our heuristic, the fact that it consistently produced worse alignments and trees by optimizing ML on data sets of moderate and difficult model conditions suggests that the problem lay in using ML treating gaps as missing data.

Impact of additional iterations on SATé-II's performance.— Having shown that using ML scores to pick the best tree produced by SATé-II is unnecessary, we next explored how to make SATé-II as fast as possible, without sacrificing too much accuracy. We compared the starting tree and alignment to the tree and alignment produced by a single iteration of SATé-II (Fig. S4). Under hard 1000-taxon model conditions, very substantial improvements in both alignment and tree accuracy arose from a single iteration through the SATé-II loop. In fact, the trees resulting from the first iteration of SATé-II were more accurate than the result of a 24-h analysis using SATé-I. In particular, SATé-II fast, which stops searching for a better tree/alignment pair once the ML score gets worse, does as well as the other SATé-II variants for easy models (data not shown) and results in only a small decrease in tree accuracy for the harder models (Fig. S4 for the 1000-taxon moderate-to-difficult data sets) data sets.

Other aspects of algorithm design.—We explored other changes to SATé-II's algorithm design, including the choice of starting tree, using different alignment methods to align subsets, the use of Muscle instead of Opal as the alignment merger, the maximum data set size for the decomposition strategy, and the edge around which the decomposition occurs. These experiments (Table S4) showed that SATé-II fast is robust to the choice of starting tree and that three changes to the algorithm relative to SATé-I yielded improvements in alignment and/or tree accuracy: SATé-II was improved by using Opal as the alignment merger instead of Muscle, by decomposing the data into smaller subsets, and by decomposing around the longest branch instead of other branches.

Of these algorithm design issues, the technique used to merge subalignments (i.e., Muscle or Opal) and the size of the decomposition tended to have very large impacts, whereas the choice of decomposition edge had a relatively small impact (e.g., Table S4). At present, we do not know what algorithmic differences between Muscle and Opal cause Opal to produce more accurate merges. Both Muscle and Opal use profile–profile alignment to merge subalignments but they differ in the details. One possible significant difference is their scoring functions for estimating the number of gaps: Muscle uses a log-expectation scoring function (Edgar 2004a, 2004b) with optimistic gap counts (Kececioglu and Zhang 1998), whereas Opal uses a sum-of-pairs scoring function with pessimistic gap counts (Kececioglu and Zhang 1998; Wheeler and Kececioglu 2007). With respect to decomposition size, decomposing several times produced much bigger improvements than decomposing only once, and decomposing to data sets of size at most 200 produced better results than decomposing to larger data sets (e.g., Table S4 which includes comparison of decomposition variants of SATé-II on one difficult 1000-taxon model condition). The choice of method for aligning subsets also had a large impact on the alignment and tree accuracy and is discussed below.

Comparison of SATé-II fast with SATé-I and two-phase methods.—For the simulated data sets, SATé-II fast matched or improved upon SATé-I and all the two-phase methods with respect to ML scores, false-negative rates, false-positive rates, and alignment errors. With the exception of false-positive topological error rates, Figure 4 shows these results for the 1000-taxon data sets, and Figure S5 in online Appendix 1 shows these results for smaller data sets. The false-positive topological error rates are shown in Figure S6. For the difficult 500- and 1000-taxon data sets, the accuracy of the SATé-II fast trees was substantially greater than that of the next most accurate method, which was usually SATé-I and almost as accurate as RAxML trees produced on the true alignments. Finally, although SATé-I was reasonably efficient (even on the 1000-taxon data sets), SATé-II fast was *much*

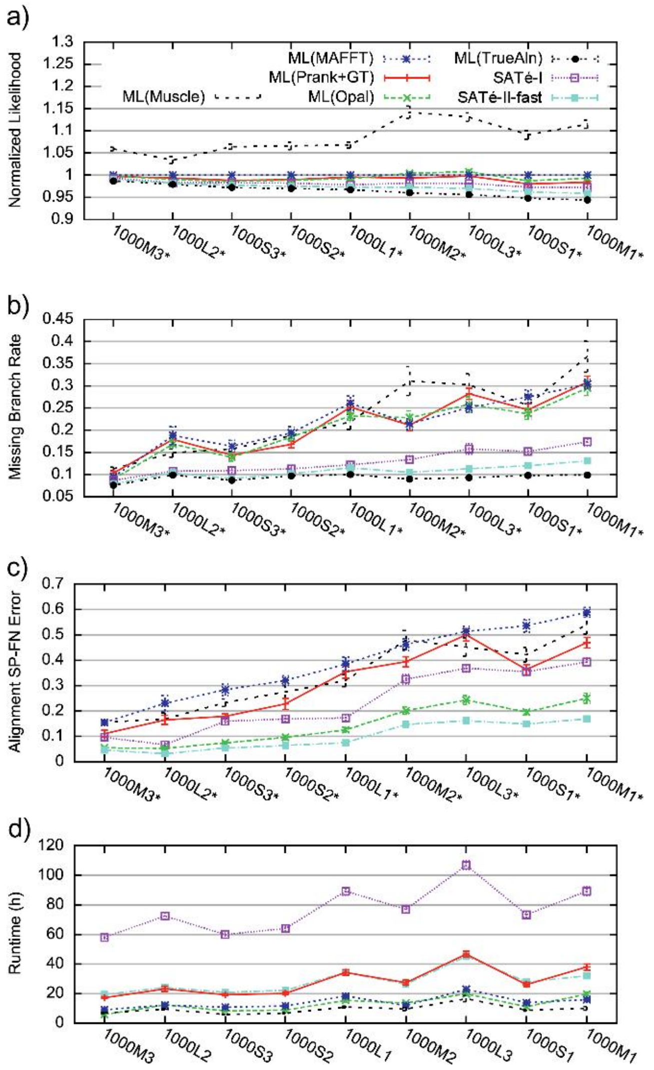

FIGURE 4. Comparisons of SATé-II fast to SATé-I and two-phase methods on "moderate-to-difficult" simulated 1000-taxon data sets. Panels from top to bottom are (a) ML scores (normalized by scores obtained by ML [MAFFT]), (b) missing branch rates (%), (c) alignment SP-FN error (%), and (d) runtime (h); $n = 20$ for each model condition. SATé runtimes include the time to compute starting trees. Model conditions marked with an asterisk (*) indicate significant improvement by SATé-II fast over the next best estimated method, as measured by Benjamini and Hochberg (1995) corrected pairwise t-tests ($n = 40$, $\alpha = 0.05$). Q values for these statistical tests are reported in Table S6.

faster than SATé-I (which by default runs for 24 h after the starting tree is computed).

SATé-II as an alignment method booster.—We explored using SATé-II fast to "boost" the performance of alignment methods, by substituting a given alignment method (e.g., Muscle) for MAFFT to estimate the initial tree and to align the subsets. These "SATé-II-boosted" methods produced substantially more accurate trees and alignments than their base methods on large difficult to align data sets (Fig. 5). Particularly, dramatic improvements were obtained when the base method was ClustalW or Muscle. Furthermore, SATé-II-boosted versions of Opal and Prank+GT were close to SATé-II fast in terms of topological accuracy. Thus, SATé-II can be considered a "metamethod" that can improve the accuracy of alignment methods, enabling them to produce more accurate alignments and enabling RAxML to produce more accurate trees. The magnitude of the improvement in alignment accuracy depends on the accuracy of the original alignment method on the data set, with improvements greatest on the harder-to-align data sets.

Biological Data

For the six smallest data sets, RAxML on a MAFFT alignment, SATé-I, and SATé-II had essentially the same accuracy (Table S5), and all the other two-phase methods did quite poorly. On the two larger biological data sets, 16S.T and 16S.B.ALL, we observed substantial differences between SATé-I, SATé-II and all the two-phase methods. First, on the largest of these data sets, 16S.B.ALL, the only two-phase methods that completed were RAxML and FastTree on the PartTree alignment. SATé-I failed to complete even one alignment because the decomposition produced several very large subsets of sequences (the largest having 8666 sequences), and MAFFT failed when analyzing these subsets, even given 256 GB of memory. In contrast, SATé-II ran well on a 32 GB machine and completed several iterations.

On the 16S.B.ALL data set, using a FastTree ML tree estimated on the PartTree alignment as its starting tree, SATé-II produced a more accurate tree than any two-phase methods after just one iteration (428.9 h, about 17.9 days), and additional iterations continued to improve the topological accuracy (Fig. 6). By the eighth iteration, SATé-II's missing branch rate was less than half of that of RAxML on the PartTree alignment. SATé-II's trees were calculated reasonably efficiently. For example, RAxML on the PartTree required 1418 h runtime (almost 59 days) to compute, whereas, including the time to compute the starting tree, four iterations of SATé-II required 2067.9 h (about 86 days), and eight iterations required 4083.6 h (about 170 days).

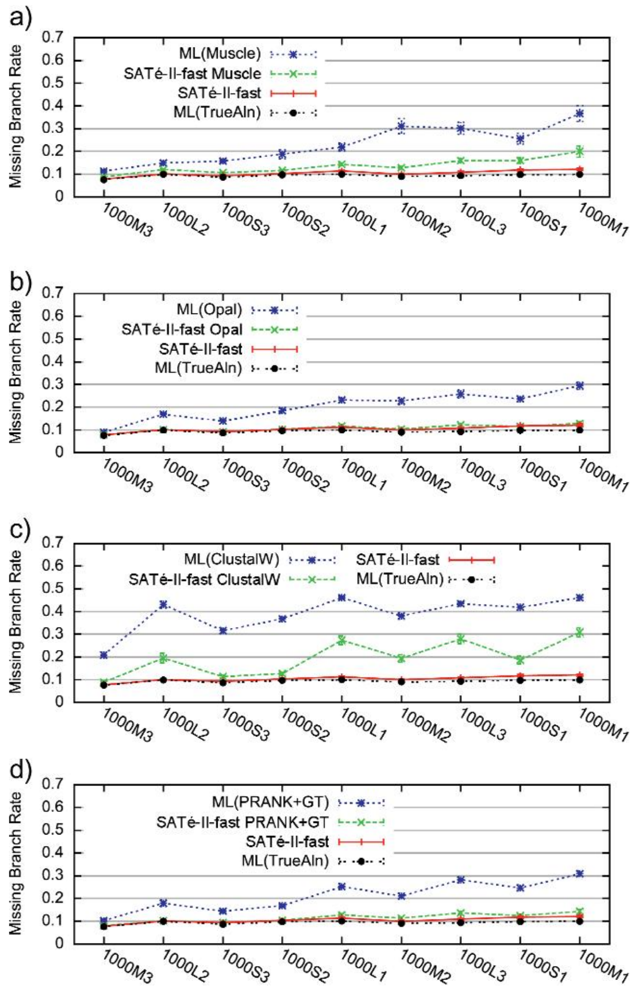

FIGURE 5. Comparisons of SATé-II-boosted MSA methods to the unboosted MSA methods with respect to missing branch rates on simulated moderate-to-difficult 1000-taxon data sets. Panels from top to bottom display results for alignment methods (a) Muscle, (b) Opal, (c) ClustalW, and (d) Prank+GT. For each panel, the unboosted MSA methods are shown in blue, whereas the boosted SATé-II fast variants are shown in green. SATé-II fast in its default mode is shown in red, and RAxML on the true alignment is shown in black.

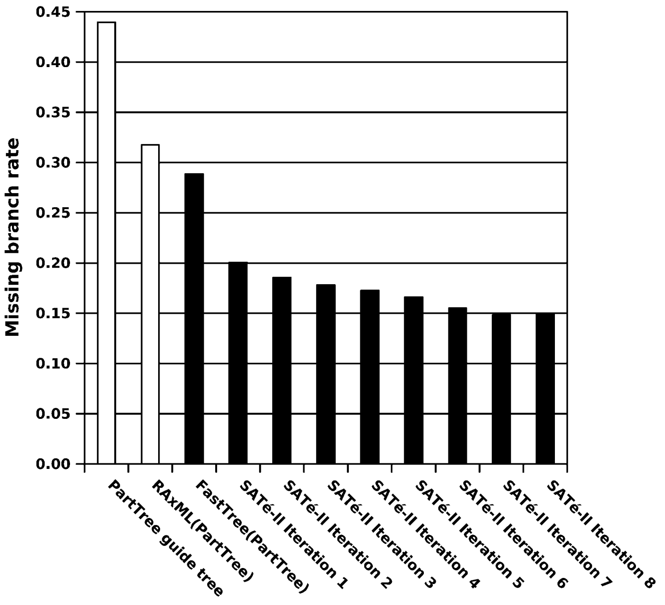

FIGURE 6. Missing branch rates of two-phase methods on the PartTree alignment and SATé-II on the 16S.B.ALL data set. The two-phase methods are shown in white, and the SATé-II iterations are shown in black. The SATé-II analyses start from FastTree on the PartTree alignment. The missing branch rates are computed with respect to the curation-based tree, which is an ML tree estimated using RAxML on the curated alignment after contraction of branches with less than 75% support from a RAxML bootstrapping analysis with 444 replicates. Each iteration of SATé-II computes subalignments using MAFFT, merges subalignments using Muscle, and computes a tree using FastTree; these iterations take between 7 to 35 days each and run on a 32 GB machine. By comparison, SATé-I failed to run on this data set. $n = 1$ for all reported values.

Performance on the 16S.T data set, which has 7350 taxa, showed similar advantages. SATé-I and SATé-II gave dramatic improvements over the best two-phase methods, RAxML on the PartTree alignment, which had 16.5% missing branch rate (Fig. S7). One iteration of SATé-II reduced the error to 11.2% error, and subsequent iterations reduced the error to 8.2% error.

Relative Scalability of SATé-I and SATé-II

A significant advantage of SATé-II over SATé-I was running time and memory, and this advantage increases with the data set size. On the smaller data sets (i.e., the simulated data sets with up to 1000 sequences and the six biological data sets in Liu, Raghavan et al. (2009)), SATé-II finished in a fraction of the time of a SATé-I analysis (Fig. 4 and Fig. S5 for the simulated data sets, and Table S5 for the biological data sets). Improvements in speed and memory usage were even larger on the largest data sets we examined (the 7350 taxon 16S.T data set, and the 27,643 taxon 16S.B.ALL data set). On the 16S.T data set (Fig. S7), the realignment step within each iteration of SATé-I took almost 20 times as long as the realignment step of SATé-II (150.5 h, on average for SATé-I compared to 8.3 h for each realignment done by SATé-II). The difference on the 16S.B.ALL data set was especially significant, since SATé-II finished each iteration in approximately 1 week, but SATé-I failed to complete its first realignment of the 16S.B.ALL data set.

The memory requirements of SATé-I and SATé-II are also worth noting. On the simulated data sets and all biological data sets except for the two largest, 4 GB of main memory was sufficient to run both SATé-I and SATé-II. On 16S.T, the second largest biological data set, SATé-I and SATé-II required machines with 32 GB of main memory to run to completion. For the 16S.B.ALL data set, with 27,643 taxa, although SATé-I failed to complete its first alignment, SATé-II finished several iterations on a machine with 32 GB of main memory.

Methods that Optimize ML, Treating Gaps as Missing Data

Our theoretical (Appendix 1) and empirical results (Fig. S8 in online Appendix 1 and online Appendix 2) regarding approaches that aggressively seek the alignment/tree pair that optimizes ML scores while treating gaps as missing data, suggest that this is a very poor approach for some models of evolution and may be generally inadvisable.

However, the way SATé-II utilizes ML is not subject to this problem. SATé-II uses ML in two ways—to compute trees on estimated alignments and to select a tree/alignment pair from among a set of alignment/tree pairs generated by SATé-II's iterative procedure. The alignments estimated by SATé-II are produced by a combination of MAFFT and Opal or Muscle, methods that use algorithmic criteria that are independent of ML. Thus, while ML is an integral component of SATé-II's algorithmic strategy, the technique SATé-II uses to produce alignments is independent of ML. This may be why the use of ML within SATé-II is not detrimental.

Interestingly, our experiments showed that SATé-II ML produced alignments that were slightly more accurate than SATé-II simple alignments, and trees that were statistically of equal accuracy to SATé-II simple trees. Why this use of ML (to select from a set of tree/alignment pairs) provides this small advantage is still unclear.

Discussion

SATé-II is a novel and highly accurate method for simultaneously estimating phylogenetic alignments and trees. Furthermore, its computationally most efficient method, SATé-II fast, offers dramatic improvements over SATé-I and two-phase methods on large difficult-to-align data sets. Also, SATé-II is a metamethod that can be used to improve the accuracy of a given alignment method and in turn improve tree estimations based upon these alignments.

The evidence presented here strongly suggests that the most important part of the algorithm design for SATé-II is its iterative use of a divide-and-conquer realignment strategy. The intuition behind this algorithm design is that even the best alignment methods have difficulty aligning sequence data sets when they are large and have evolved with many substitutions and indels. The SATé-II decomposition technique seeks to break the large data set into smaller subsets, so that each of the smaller subsets has a smaller maximum evolutionary distance between its taxa and is fairly densely sampled within that subset. This enables the subset alignments produced by MAFFT (or another alignment method) to be more accurate. Furthermore, since the merger technique does not undo the alignments on subsets, the result of merging the improved subset alignments is able to maintain the improved accuracy. Therefore, for imperfect initial alignments, performing the SATé-II realignment technique on an ML tree estimated on the initial alignment will produce a more accurate alignment, and RAxML on the new alignment will produce a more accurate tree. Furthermore, subsequent iterations will tend to provide additional improvements, until the alignment/tree pairs it produces have become fairly accurate.

SATé-II provides a very substantial improvement in scalability and accuracy over SATé-I because of the specific divide-and-conquer design used in SATé-II, which ensures that the subsets it creates for realignment are quite small (at most 200 sequences); by comparison, the subsets produced by the SATé-I decomposition can be unboundedly large. More generally, this means that very large data sets can be analyzed using SATé-II that cannot be analyzed by SATé-I in the same timeframes.

As we have demonstrated, SATé-II trees approach the accuracy of GTR+Gamma ML trees (estimated using RAxML) on the true alignment. However, RAxML and other standard ML methods (e.g., PhyML (Guindon and

Gascuel 2003), PAUP* (Swofford 2003), FastTree (Price et al. 2010), etc.) treat gaps as missing data and are based upon statistical models of evolution that do not include indel events. There are two consequences of this observation. The first is that SATé-II is not likely to be statistically consistent under models that include indel events, because standard ML methods are themselves not statistically consistent, even if given the true alignment. That is, treating gaps as missing data is not a statistically consistent way of estimating phylogenies from alignments that have gaps. The second consequence is that while SATé-II is a substantial advance, it seems likely that even better results could be obtained when *scalable* algorithms and software are developed that estimate phylogenies under statistical models that include indels as well as substitutions. Unfortunately, existing methods that are based upon statistical models that include indel events (e.g., StatAlign (Novák et al. 2008), BAli-Phy (Suchard and Redelings 2006), and ALIFRITZ (Fleissner et al. 2005)) are not yet able to analyze even moderately large data sets.

Supplementary Material

Supplementary material, including data files and/or online-only appendices, can be found at http://www.sysbio.oxfordjournals.org/.

Funding

A.S. was supported under the auspices of the Emmy Noether program by the German Science Foundation (Deutsche Forschungsgemeinschaft). T.J.W., K.L., and S.N. were supported by the US National Science Foundation under ITR-0331453 (CIPRES) and T.J.W., K.L., and C.R.L. were supported by the US National Science Foundation grant ATOL-0733029. M.T.H. and J.Y. were supported by the US National Science Foundation through grant ATOL-0732920. T.J.W. also received support from the Guggenheim Foundation and from Microsoft Research New England. This research was supported in part by the US National Science Foundation Research Infrastructure Grant EIA-0303609.

Acknowledgments

The authors would like to acknowledge the Texas Advanced Computing Center (TACC) (http://www.tacc.utexas.edu) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. We also wish to thank John Cazes and Dan Stanzione for their assistance with the analyses on the TACC HPC resources. The authors also thank Steve Evans for helpful discussions regarding the theoretical results, as well as Ron DeBry, Laura Kubatko, and two anonymous reviewers for useful suggestions.

References

Benjamini Y., Hochberg Y. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Stat. Soc. Series B Met. 57:289–300.

Cannone J., Subramanian S., Schnare M., Collett J., D'Souza L., Du Y., Feng B., Lin N., Madabusi L., Muller K., Pande N., Shang Z., Yu N., Gutell R. 2002. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron and other RNAs. BMC Bioinformatics. 3:2. Available from: http://www.rna.ccbb.utexas.edu. Accessed 4 October 2011.

Edgar R.C. 2004a. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 5:113.

Edgar R.C. 2004b. MUSCLE: a multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 32:1792–1797.

Fleissner R., Metzler D., von Haeseler A. 2005. Simultaneous statistical multiple alignment and phylogeny reconstruction. Syst. Biol. 54:548–561.

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

Gutell R.R., Larsen N., Woese C.R. 1994. Lessons from an evolving ribosomal-RNA—16S and 23S ribosomal RNA structures from a comparative perspective. Microbiol. Rev. 58:10–26.

Gutell R.R., Lee J.C., Cannone J.J. 2002. The accuracy of ribosomal RNA comparative structure models. Curr. Opin. Struct. Biol. 12:301–310.

Hein J. 1990. Unified approach to alignment and phylogenies. Methods Enzymol. 183:626–645.

Hillis D., Bull J. 1993. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. Syst. Biol. 42: 182–192.

Jukes T., Cantor C. 1969. Evolution of protein molecules. In: Munro H.N. and Allison J.B., editors. Mammalian protein metabolism. New York: Academic Press. p. 21–132.

Katoh K., Kuma K., Toh H., Miyata T. 2005. MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic Acids Res. 33:511–518.

Katoh K., Toh H. 2007. PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. Bioinformatics. 23:372–374.

Katoh K., Toh H. 2008. Recent developments in the MAFFT multiple sequence alignment program. Brief. Bioinform. 9:286–298.

Kececioglu J.D., Zhang W. 1998. Aligning alignments. In: Farach M., editor. Proceedings of CPM 1998 Lecture Notes in Computer Science. Berlin (Germany): Springer. p. 189–208.

Kimura M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J. Mol. Evol. 16:111–120.

Lancia G., Ravi R. 1999. GESTALT: Genomic Steiner alignments. Lect. Notes Comput. Sci. 1645:101–114.

Litzkow M. 1987. Remote Unix—turning idle workstations into cycle servers. Usenix summer conference. USENIX Association. p. 381–384.

Liu K., Linder C.R., Warnow T. 2010. Multiple sequence alignment: a major challenge to large-scale phylogenetics. PLoS Curr. Tree of Life. Available from: http://knol.google.com/k/kevin-liu/multiple-sequence-alignment-a-major/ectabesw3uba/9. Accessed 4 October 2011.

Liu K., Nelesen S., Raghavan S., Linder C.R., Warnow T. 2009. Barking up the wrong treelength: the impact of gap penalty on alignment and tree accuracy. IEEE/ACM Trans. Comput. Biol. Bioinform. 6:7–21.

Liu K., Raghavan S., Nelesen S., Linder C.R., Warnow T. 2009. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science. 324:1561–1564.

Lunter G., Miklós I., Drummond A., Jensen J.L., Hein J. 2005. Bayesian coestimation of phylogeny and sequence alignment. BMC Bioinformatics. 6:83.

Nakhleh L., Moret B.M.E., Roshan U., St. John K. Sun J., Warnow T. 2002. The accuracy of fast phylogenetic methods for large datasets.

Proceedings of the 7th Pacific Symposium on BioComputing (PSB02). 2002 January 3–7; Lihue, Hawaii. p. 211–222.
Novák Á., Miklós I., Lyngsoe R., Hein J. 2008. StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics. 24:2403–2404.
Price M., Dehal P., Arkin A. 2010. FastTree 2—approximately maximum-likelihood trees for large alignments. PLoS One. 5:e9490.
Redelings B.D., Suchard M.A. 2005. Joint Bayesian estimation of alignment and phylogeny. Syst. Biol. 54:401–418.
Rodriguez F., Oliver J., Marin A., Medina J. 1990. The general stochastic model of nucleotide substitution. J. Theor. Biol. 142:485–501.
Sanderson M.J. 2003. r8s: inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock. Bioinformatics. 19:301–302.
Sankoff D. 1975. Minimal mutation trees of sequences. SIAM J. Appl. Math. 28:35–42.
Sankoff D., Cedergren R. 1983. Simultaneous comparison of three or more sequences related by a tree. In: Sankoff D. and Kruskal J.B., editors. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. Chapter 2. Reading (MA): Addison-Wesley. p. 253–264.
Smith S., Beaulieu J.M., Donoghue M.J. 2009. Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. BMC Evol. Biol. 9:37.
Stamatakis A. 2006. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics. 22:2688–2690.
Stoye J., Evers D., Meyer F. 1998. Rose: generating sequence families. Bioinformatics. 14:157–163.
Suchard M.A., Redelings B.D. 2006. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics. 22:2047–2048.
Swofford D.L. 2003. PAUP*: phylogenetic analysis using parsimony (*and other methods). version 4. Sunderland (MA): Sinauer Associates.
Thompson J., Higgins D., Gibson T. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22:4673–4680.
Thompson J.D., Plewniak F., Poch O. 1999. A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Res. 27:2682–2690.
Thorne J.L., Kishino H., Felsenstein J. 1991. An evolutionary model for maximum likelihood alignment of DNA sequences. J. Mol. Evol. 33:114–124.
Thorne J.L., Kishino H., Felsenstein J. 1992. Inching toward reality: an improved likelihood model of sequence evolution. J. Mol. Evol. 34:3–16.
Tuffley C., Steel M. 1997. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. Bull. Math. Biol. 59:581–607.
Varón A., Vinh L.S., Wheeler W.C. 2010. POY version 4: phylogenetic analysis using dynamic homologies. Cladistics. 26:72–85.
Wheeler W.C. 1995. Sequence alignment, parameter sensitivity, and the phylogenetic analysis of molecular data. Syst. Biol. 44:321–331.
Wheeler W.C. 1996. Optimization alignment: the end of multiple sequence alignment in phylogenetics? Cladistics. 12:1–9.
Wheeler W.C., Gladstein D.S. 1994. MALIGN: a multiple sequence alignment program. J. Hered. 85:417–418.
Wheeler T., Kececioglu J. 2007. Multiple alignment by aligning alignments. Bioinformatics. 23:i559–i568.

Appendix 1

Role of Likelihood Optimization, Treating Gaps as Missing Data

In this section, we provide the results regarding the use of optimizing likelihood while treating gaps as missing data. We present a theoretical result for an extended version of the JC model (a special case of the GTR model), in which we allow branches of the model tree to have substitution probabilities set to 0. We call this extension the EJC model. We show that optimizing likelihood under the EJC model, allowing the alignment to change arbitrarily, is uninformative in that there is some alignment for which all trees are optimal under this criterion. Then, by continuity, it follows that the same result is true for the JC model, which requires that all branches have nonzero substitution probability. We show as well that the proof technique does not apply to the Kimura 2-parameter (K2P) model (a submodel of GTR in which transitions and transversions each have a fixed probability, but these can differ). Finally, online Appendix 2 provides the results of a simple heuristic we designed that produces a greedy alignment and tree and which shows that improving the GTR+Gamma ML score can lead to very poor alignments and trees.

Lemma 1 *Let A be a DNA alignment in which no site is entirely gapped, and let r_i denote the number of states present in site i. Let A' be the alignment obtained by splitting every site i in A into r_i sites, so that each of these r_i sites has only one state. For example, if the site is given by*

$$s_1 = A$$
$$s_2 = C$$
$$s_3 = A$$
$$s_4 = -$$
$$s_5 = -$$
$$s_6 = T$$
$$s_7 = A$$

then we would obtain three sites, as follows:

$$s_1 = A - -$$
$$s_2 = -C-$$
$$s_3 = A - -$$
$$s_4 = - - -$$
$$s_5 = - - -$$
$$s_6 = - - T$$
$$s_7 = A - -$$

Then, for all trees T, if gaps are treated as missing data, $ML_{EJC}(A,T) \leq ML_{EJC}(A',T) = 1/4^R$, where R is the number of sites in alignment A'.

Proof. We begin by noting that $\mathrm{ML}_{\mathrm{EJC}}(A,T) \leq \mathrm{ML}_{\mathrm{NCM}}(A,T)$, where NCM denotes the No Common Mechanism model (Tuffley and Steel 1997). Under the NCM model, sites evolve under a JC model down a common tree, and for every site i and edge e combination, there is a probability of change $p(e,i)$ such that $0 \leq p(e,i) < 3/4$. Thus, site i changes on edge e with probability $p(e,i)$. Clearly, the NCM model contains the EJC model. However, the NCM model does not contain more general models such as K2P (Kimura 1980). Note also that treating gaps as missing data within the NCM model yields $\mathrm{ML}_{\mathrm{NCM}}(A,T) = 1/4^{k+MP(T,A)}$, where k is the number of sites in A, and $MP(T,A)$ denotes the parsimony score of alignment A on tree T.

We also observe that $\mathrm{ML}_{\mathrm{EJC}}(T,A')$ is obtained by setting all substitution probabilities to 0 and so $\mathrm{ML}_{\mathrm{EJC}}(T,A') = 1/4^R$, where R is the number of sites in the alignment A'. Note that $R = \sum_i r_i$ (where r_i is the number of states that appear in site i in alignment A), and that $r_i \leq MP(T,i)+1$, where $MP(T,i)$ is the number of times site i changes on tree T. Therefore $R \leq MP(T,A)+k$. Putting all this together, we obtain

$$\begin{aligned}\mathrm{ML}_{\mathrm{EJC}}(T,A) &\leq \mathrm{ML}_{\mathrm{NCM}}(T,A)\\ &= 1/4^{MP(T,A)+k}\\ &\leq 1/4^R\\ &= \mathrm{ML}_{\mathrm{EJC}}(T,A')\end{aligned}$$

□

Theorem 1 *Let $\Phi(S)$ be a method that takes as input a set S of unaligned sequences and returns all pairs (T^{opt},A^{opt}), where T^{opt} is a phylogenetic tree and A^{opt} is an alignment on S, such that for some set θ^{opt} of EJC parameters, $Pr[A^{\mathrm{opt}} \mid T^{\mathrm{opt}},\theta^{\mathrm{opt}}]$ is maximum among all possible alignments of S, trees on S, and EJC parameter values. (Recall that gaps are treated as missing data under this calculation.) Then for any set S of unaligned sequences, there is an alignment A^{opt} such that $\Phi(S) = \{(T \in A^{\mathrm{opt}}) : T \in T_S\}$, where T_S is the set of all trees with leaf set S. In particular, the tree with all substitution probabilities set to zero is an optimal solution.*

Proof. Let S be an arbitrary set of n DNA sequences. Let (T,A,θ) be an optimal solution to EJC ML, so that A is an alignment on S, (T,θ) is an EJC model tree, and $Pr[A|(T,\theta)]$ is maximized among all possible alignments A of S and model trees (T,θ). Without loss of generality, we will assume that A has no entirely gapped sites. Then by Lemma 1, we can assume that A has only one type of nucleotide appearing in any site, that θ sets all branch substitution probabilities to zero, and that $Pr[A|(T,\theta)] = 1/4^R$, where R is the number of sites in alignment A. Note that $Pr[A|(T,\theta)]$ therefore does not depend upon the tree T, so that $\phi(S)$ contains all trees T on the leaf set S. □

We now show that we can extend this theorem to the JC model, in which all branch substitution probabilities are nonzero. We will work with a modification of the ML problem, in order to be able to address the fact that optimum ML solutions cannot be achieved without permitting substitution probabilities to be zero.
Supremum Likelihood:

- Input: unaligned sequences S
- Output: Tree T and alignment A for S such that $sup_\theta\{Pr[A|(T,\theta)]\}$ is maximized.

Theorem 2 *Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of unaligned sequences. For T a tree on $\{s_1, s_2, \ldots, s_n\}$, let $f(S,T) = sup_{\theta,A}\{Pr[A|T,\theta]\}$, where A is an alignment on S and θ the JC parameters (edge substitution probabilities). Then for all sets S of unaligned sequences and all trees T leaf labeled by the taxon set of S, $f(S,T) = f(S,T')$. Hence, all trees are optimal solutions to supremum likelihood for JC, if the alignment is allowed to change arbitrarily.*

Proof. Let S be an arbitrary set of sequences. By Theorem 1, there is an alignment A^* on S optimizing the EJC ML score, such that for all bifurcating trees T, by setting all branch substitution probabilities on the branches of T to zero, we obtain an optimal EJC ML score. Let $M = Pr[A^*|T,\theta]$, where θ sets all substitution probabilities to zero. Note therefore that the optimal solution to Supremum Likelihood under the JC model will be *at most* M. We now show that all trees can achieve M (i.e., for all trees T, $sup_{\theta,A}\{Pr[A|T,\theta]\} = M$); the result then follows.

Let $\epsilon > 0$ be given. By continuity of likelihood scores, for small enough (but all positive) branch substitution probabilities, θ', on T, $Pr[A^*|T,\theta'] \geq M - \epsilon$. Since $\epsilon > 0$ was arbitrary, this shows that $sup_{\theta,A}\{Pr[A|T,\theta]\} = M$. Since the tree T was arbitrary, this shows that all trees achieve the optimal supremum likelihood score and the result follows. □

Interestingly, the conclusion of Theorem 1 does not apply to models in which the relative rates associated with different substitution classes vary. For example, even under EK2P (the "extended" K2P model), it is *not* the case that $ML_{EK2P}(T,A) \leq ML_{EK2P}(T,A')$ for any tree T and alignment A.

Consider the following alignment:

$$s_1 = A$$
$$s_2 = A$$
$$s_3 = G$$
$$s_4 = G$$

Because the data can be explained without any transversions, the ML estimate of the transition/transversion rate ratio (κ) is infinity. Table 1 shows the tree topologies in Newick notation and the maximum (really "supremum", as discussed above) likelihood score for each tree. We note that the ML score for these trees is obtained in the limit, by allowing $\nu \to \infty$ (where ν denotes the lengths of certain branches in the trees below). Note that on the tree topology that requires only one change, the likelihood can approach 1/8, whereas on the other two tree topologies, the ML score cannot exceed 1/16. The root state frequency accounts for a factor of 1/4

TABLE 1. Example trees and ML scores

Tree[a]	K2P ML score[b]
(1:0,2:0,(3:0,4:0):ν)	1/8
(1:0,3:ν,(2:0,4:ν):0)	1/16
(1:ν,3:0,(2:ν,4:0):0)	1/16
(1:0,4:ν,(2:0,3:ν):0)	1/16
(1:ν,4:0,(2:ν,3:0):0)	1/16

[a] ν denotes the lengths of certain branches in the trees. All other branch lengths are zero.
[b] The ML score is attained as ν approaches infinity.

in the K2P model. Transversions are disallowed when $\kappa = \infty$, so

$$\lim_{\nu\to\infty} \Pr(A \to G|\nu) = \lim_{\nu\to\infty} \Pr(G \to A|\nu) = 1/2.$$

Thus, the ML score can be obtained for this data set by multiplying 1/4 by $1/2^{MP(T,A)}$, where $MP(T,A)$ is the parsimony score of tree T.

The trivial alignment, A', for this data set would be,

$$s_1 = A-$$
$$s_2 = A-$$
$$s_3 = -G$$
$$s_4 = -G$$

and the ML score (attainable on any tree with all branch lengths set to zero) would be 1/16 (the product of the state frequencies at each column).

This demonstrates that, when relative rates of substitution are not all equal, it is possible for selection of the optimal tree and alignment pair to favor one tree topology over another. Thus, for some models, tree inference is technically possible under ML while jointly estimating the alignment and treating gaps as missing data. Despite this, we emphasize that it will almost never be advisable to optimize the alignment without penalizing gaps.

APPENDIX 2

The following methodology and program commands are identical to the simulation study performed in Liu, Raghavan et al. (2009).

We simulated data sets for our simulation study in two steps. In Step 1, we generated model trees using r8s (Sanderson 2003), and in Step 2, we simulated sequences on the model trees using ROSE (Stoye et al. 1998).

Step 1: Generation of model trees using r8s. We used r8s version 1.7 to generate birth–death model trees with 100, 500, or 1000 taxa. r8s used the following script as input to perform this step:

```
begin rates;
simulate diversemodel=bdback seed=<random seed>
ntaxa=<100 or 500 or 1000> T=0;
describe tree=0 plot=tree_description;
end;
```

For each model tree, we deviated the branch lengths away from ultrametricity according to the calculation from Nakhleh et al. (2002) with deviation factor $c = 2.0$. To perform this calculation, we used the custom program from Liu, Raghavan et al. (2009), obtained from http://www.cs.utexas.edu/users/tandy/science-paper.html. Next, for each model tree, we multiplied the branch lengths by a factor which we varied as the "tree height" parameter of simulation. Finally, due to ROSE's inability to handle fractional branch lengths in its model of evolution, we multiplied all branch lengths by a factor of 100 to ensure that branch lengths were generally not less than 1.

Step 2: Simulation of sequences using ROSE. We used ROSE version 1.3 to simulate the evolution of DNA sequences on each model tree under a GTR+Gamma+Gap model.

- GTR+Gamma model of substitution—the GTR model (Rodriguez et al. 1990) is parameterized by the frequencies of nucleotides in the root sequence and the instantaneous rates of change between nucleotides. For these parameters, we used the estimates produced by PAUP* on the NemATOL alignment (obtained from the NemATOL website at http://nematol.unh.edu/tree/tree1/vIch20ct682_c1w1.aln) of 682 species of nematodes. We used the instantaneous rate matrix to compute the transitional probability matrix $P(t) = e^{At}$ which is required as input to ROSE, where A is the instantaneous rate matrix and $t = 0.001$.

  We used a sequence length of 1000 nucleotides for the root sequence. The frequencies of nucleotides in the root sequence f_X, where X is a nucleotide, were

  $$[f_A, f_C, f_G, f_T] = [.300414, .191363, .196748, .311475].$$

  The instantaneous rates of change between nucleotides were

A-C	1.24284
A-G	3.47484
A-T	0.48667
C-G	1.07118
C-T	4.38510
G-T	1.0

  We set the shape parameter α of the Gamma distribution, which controls the rate of variation across sites, to 1.0.

- Probability of a gap event—we set the probability of insertions equal to the probability of deletions for each model condition and each model condition used a different setting for this parameter.

- Gap length distribution—once an insertion or deletion event occurs, the length of that event was drawn from the gap length distribution. We used three geometric gap length distributions with finite tails: long, medium, and short. The gap length distributions used in our study are given by the following lists, where the first element of a list is the probability of a gap of length one given that a gap event occurs, the second element is the probability of a gap of length two given that a gap event occurs, and so forth.

  Long gap length distribution:

  [0.1028, 0.0899, 0.0792, 0.0702, 0.0627, 0.0565, 0.0514, 0.0470, 0.0433, 0.0400, 0.0369, 0.0341, 0.0314, 0.0289, 0.0266, 0.0245, 0.0225, 0.0206, 0.0188, 0.0171, 0.0155, 0.0141, 0.0127, 0.0114, 0.0100, 0.0087, 0.0075, 0.0063, 0.0052, 0.0042]

  Medium gap length distribution:

  [0.2012, 0.1600, 0.1280, 0.1024, 0.0819, 0.0655, 0.0524, 0.0419, 0.0336, 0.0268, 0.0215, 0.0172, 0.0137, 0.0110, 0.0088, 0.0070, 0.0056, 0.0045, 0.0036, 0.0029, 0.0023, 0.0018, 0.0015, 0.0012, 0.0009, 0.0008, 0.0006, 0.0005, 0.0004, 0.0003, 0.0002]

  Short gap length distribution:

  [0.4613, 0.2527, 0.1545, 0.0896, 0.0419]

  The long gap length distribution had expected gap length of 9.2 and median gap length of 7; the medium gap distribution had expected gap length of 5.0 and median gap length of 4, and the short gap distribution had expected gap length of 2.0 and median gap length of 2.

In summary, there are a total of four parameters that we varied in our simulation: the number of taxa, the tree height, the probability of a gap event, and the gap length distribution. Table S7 provides the parameter settings that constitute each of the fifteen 1000-taxon model conditions, and similarly for Table S8 for the fifteen 500-taxon model conditions and Table S1 for the fifteen 100-taxon model conditions.

For each model condition, we repeated Step 1 to obtain 20 different model trees, and for each model tree, we performed Step 2 once to simulate one data set. This produced 20 replicate data sets per model condition.

The empirical statistics of the true alignments and true trees from each model condition are listed in Table S7 for the 1000-taxon model conditions, Table S8 for the 500-taxon model conditions, and Table S1 for the 100-taxon model conditions.

The complete script used as input to ROSE for Step 2 was

```
TheInsFunc = <Insert event gap length distribution -
long, medium or short>

TheDelFunc = <Delete event gap length distribution -
long, medium or short>
InputType = 4

TheAlphabet = “ACGT”

TheFreq = [.300414, .191363, .196748, .311475]

ThePAMMatrix =

[[0.9948, 0.0012, 0.0035, 0.0005],
[0.0012, 0.9933, 0.0011, 0.0044],
[0.0035, 0.0011, 0.9944, 0.0010],
[0.0005, 0.0044, 0.0010, 0.9941]]
TheInsertThreshold = <Insertion event probability>

TheDeleteThreshold = <Deletion event probability>

SequenceLen = 1000

TheTree =<birth-death model tree in Newick format with branch lengths, deviated from ultrametricity>

ChooseFromLeaves = False

AlignmentWithAncestors = True

TreeWithAncestors = True

SequenceNum = <99 or 499 or 999>

SeedVal = <random seed integer>

TheMutationProbability=<site-by-site vector listing rate multipliers for that site>
```

Appendix 3

Several programs were invoked during the course of our work to estimate multiple sequence alignments. These programs were ClustalW version 2.0.4, MAFFT version 6.240, Muscle version 3.7, Prank+GT using Prank version 080904 and using RAxML (MAFFT) as a guide tree, and Opal version 1.0.2, which required Java version 1.6.0_02. Below are the commands used for each program, where <input> is a FASTA-formatted input file containing unaligned sequences and <output> is the resulting FASTA-formatted output file. Note that we ran MAFFT and ClustalW each in two ways: one uses the default, and more accurate, command, and the other uses the command that can run on large data sets.

- ClustalW (default):
  ```
  clustalw2 -align -infile=<input>
  -outfile=<output> -output=fasta
  ```
- ClustalW-Quicktree:
  ```
  clustalw2 -align -infile=<input>
  -outfile=<output> -output=fasta
  -quicktree
  ```
- MAFFT (default)
  ```
  mafft –localpair –maxiterate 1000
  –quiet <input> > <output>
  ```
- MAFFT-PartTree:
  ```
  mafft –parttree –retree 2
  –partsize 1000 <input> > <output>
  ```
- Muscle:
  ```
  muscle -in <input> -out <output> -quiet
  ```
- Prank+GT:
  ```
  prank -d=<input> -o=<output>
  -t=<RAxML(MAFFT) guide tree> -noxml -notree
  -nopost +F -matinitsize=5 -uselogs
  ```
- Opal:
  ```
  java -jar Opal.1.0.2.jar –in <input>
  –out <output>
  ```

We also used Muscle version 3.7 and Opal version 1.0.2 to merge two input alignments into an output alignment. We list the commands used for each program below, where <input 1> and <input 2> are FASTA-formatted input files each containing a multiple sequence alignment and <output> is the resulting FASTA-formatted output file:

- Opal merge:
  ```
  java -jar Opal.1.0.2.jar –in <input 1>
  –in2 <input 2> –out <output>
  –align_method profile
  ```
- Muscle merge:
  ```
  muscle -profile -in1 <input 1>
  -in2 <input 2> -out <output> -quiet
  ```

ML trees for all data sets other than the two largest biological data sets (16S.T and 16S.B.ALL) were estimated using RAxML version 7.0.4. All two-phase tree estimations on alignments generated by the above MSA methods used the default RAxML command below. All two-phase RAxML estimations were run to completion. <input> is a PHYLIP-formatted input alignment file.

- RAxML default:
  ```
  raxmlHPC -m GTRMIX -w <work dir>
  -n <identifying suffix> -s <input>
  ```

On the 16S.T data set, we used several different versions and commands for RAxML, as listed below. We ran an alpha-version 7.2.4_sse3 of RAxML which makes use of a stopping rule that permits faster runs than the default RAxML version 7.0.4.

- RAxML default version 7.0.4, used for two-phase ML tree estimation:
  ```
  raxmlHPC -m GTRMIX -w <work dir>
  -n <identifying suffix> -s <input>
  ```
- RAxML alpha-version 7.2.4_sse3 fast search, used for SATé-I and SATé-II:
  ```
  raxmlHPC-SSE3 -F -D -m GTRCAT
  -n <identifying suffix> -s <input>
  ```
- RAxML alpha-version 7.2.4_sse3 score and report branch lengths, used for SATé-I and SATé-II:
  ```
  raxmlHPC-SSE3 -m GTRGAMMA
  -n <identifying suffix> -s <input alignment>
  -t <input tree> -f e
  ```

On the 16S.B.ALL data set, we used several different versions and commands for ML tree estimation, as listed below. Due to the time and space requirements for RAxML on this data set, SATé-II used FastTree (Price et al. 2010) in place of RAxML as an ML tree estimator on a fixed alignment. Our attempts to run SATé-I on this data set failed prior to any ML tree estimation.

- RAxML default version 7.0.4, used for two-phase ML tree estimation:
  ```
  raxmlHPC -m GTRMIX -w <work dir>
  -n <identifying suffix> -s <input>
  ```
- FastTree version 2.1.3, used for SATé-II's starting tree and during search by SATé-II:
  ```
  FastTree -nt -gtr -nosupport
  -log <log file> <input alignment> > <output tree>
  ```

To construct a curation-based tree on the biological data sets other than 16S.T and 16S.B.ALL, we used the following command in RAxML version 7.0.4 to estimate an ML tree on the curated alignment and assign support values to the branches of the ML tree using a rapid bootstrapping analysis with 500 bootstrap replicates:

- RAxML rapid bootstrap:
  ```
  raxmlHPC -f a -m GTRGAMMA -s <input>
  -n <identifying suffix> -x <random number>
  -p <random number> -N <number of replicates>
  ```

To construct a curation-based tree on the 16S.T data set, we estimated an ML tree on the curated alignment using RAxML version 7.2.6 with the RAxML default command above. Then, we performed a rapid bootstrap analysis with 346 bootstrap replicates using RAxML version 7.0.4 and the RAxML rapid bootstrap command above to assign support values to the branches of the ML tree.

To construct a curation-based tree on the 16S.B.ALL data set, we estimated an ML tree on the curated alignment using RAxML version 7.2.6 with the RAxML default command listed above. Next, a standard bootstrap analysis with 444 bootstrap replicates was performed using RAxML version 7.2.5 on TACC's Ranger supercomputing cluster to assign support values to the branches of the ML tree. The RAxML analysis of each bootstrap replicate was parallelized using the 4 cores of a single compute node on Ranger via the following command.

- RAxML analysis of a single standard bootstrap replicate:

  raxmlHPC-PTHREADS-SSE3 -m GTRCAT -s <input>

  -n <identifying suffix> -b <random number>

  -p <random number> -N 1 -T 4'

When necessary, we enabled multithreading for the RAxML runs either by recompiling with PTHREADS and running with an additional flag -T <number of threads> or by recompiling with MPI for the rapid bootstrapping analysis; this parallelization does not otherwise change the RAxML commands. The RAxML outputs are unaffected by parallelization, and all reported runtimes are for serialized execution.