Uncovering Hidden Phylogenetic Consensus in Large Datasets

Nicholas D. Pattengale, Andre J. Aberer, Krister M. Swenson, Alexandros Stamatakis, Bernard M.E. Moret

Abstract—Many of the steps in phylogenetic reconstruction can be confounded by "rogue" taxa—taxa that cannot be placed with assurance anywhere within the tree, indeed, whose location within the tree varies with almost any choice of algorithm or parameters. Phylogenetic consensus methods, in particular, are known to suffer from this problem. In this paper we provide a novel framework in which to define and identify rogue taxa. In this framework, we formulate a bicriterion optimization problem, the relative information criterion, that models the net increase in useful information present in the consensus tree when certain taxa are removed from the input data. We also provide an effective greedy heuristic to identify a subset of rogue taxa and use this heuristic in a series of experiments, with both pathological examples from the literature and a collection of large biological datasets. As the presence of rogue taxa in a set of bootstrap replicates can lead to deceivingly poor support values, we propose a procedure to recompute support values in light of the rogue taxa identified by our algorithm; applying this procedure to our biological datasets caused a large number of edges to move from "unsupported" to "supported" status, indicating that many existing phylogenies should be recomputed and re-evaluated to reduce any inaccuracies introduced by rogue taxa. We also discuss the implementation issues encountered while integrating our algorithm into RAxML v7.2.7, particularly those dealing with scaling up the analyses. This integration enables practitioners to benefit from our algorithm in the analysis of very large datasets (up 2,500 taxa and 10,000 trees, although we present the results of even larger analyses).

Index Terms—phylogeny, consensus methods, bootstrapping, support values, MAST.

1 Introduction

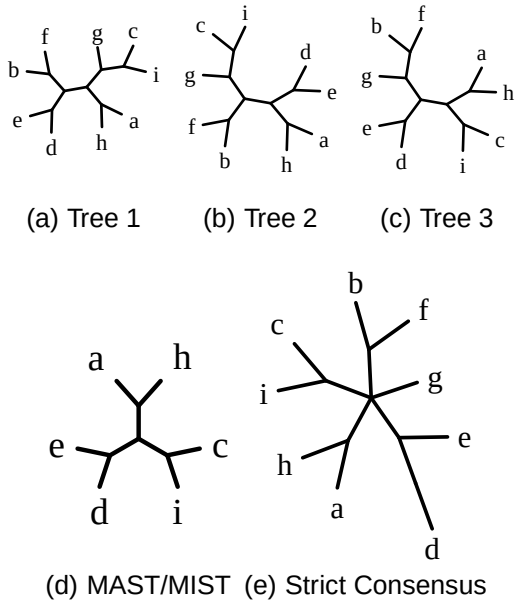
Phylogenetic consensus methods are used for combining a set of trees defined on the same set of leaves into a single tree, so as to summarize the information found in the set. By their very nature, these methods discard information, typically structural elements not prevalent in the set. However, the most popular consensus methods (strict and majority rule) are susceptible to the presence of so-called *rogue taxa* [21]. While the tree set may agree very strongly on the structure relating a large subset of the leaves, the remaining few leaves (the rogue taxa) can effectively prevent this underlying structure from appearing in the strict or majority consensus tree. In other words, these methods end up discarding structural elements that are, in fact, prevalent in the set.

Much work has been done on the problem of summarizing a set of trees, including some work on the issue of rogue taxa. The pioneering work of Wilkinson [21], [22], [23] addresses the summarization problem by returning *sets* of trees, some of which are missing leaves, with the aim of conveying the prevalent structural elements in at least one of the returned trees. While theoretically satisfying, this approach suffers from computational complexity problems and, more importantly, from difficulties in interpretation.

A problem closely related to both consensus and rogue taxa is the *Maximum Agreement Subtree (MAST)*. A MAST on a set of input trees is the largest (induced) subtree common to all input trees—one can think of choosing a largest subset of leaves such that the induced subtrees are the same in all input trees. While the general problem of finding the MAST of three or more trees is $\mathcal{NP}$-hard [1], it can be solved efficiently when at least one of the input trees has bounded degree [7]. Another agreement subtree optimization problem, *Maximum Information Subtree (MIST)*, was proposed by Bryant[3] to overcome a crucial deficiency of MAST, namely that the maximization of leaf-set cardinality can entirely obscure important internal structure revealed by a smaller leaf subset. Bryant's algorithm for solving MIST, whose complexity mirrors that of MAST algorithms, actually affords the practitioner an option to weight the importance placed on leaf-set cardinality versus internal structure in the solution. As such, the optimization function for MIST resembles the MISC optimization problem we propose below. Unfortunately, *all* approaches based on agreement subtrees can be overly conservative; in particular, there exist instances where the strict consensus tree (without dropping any leaves) has more internal edges than any MAST or MIST—Fig. 1 depicts such an instance.

- *Nicholas Pattengale is with Sandia National Laboratories, Albuquerque New Mexico, USA*
- *Andre Aberer and Alexandros Stamatakis are with The Exelixis Lab, Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Heidelberg, Germany*
- *Krister Swenson is with The Laboratory for Innovation in Bioinformatics at the University of Ottawa and The Laboratoire de combinatoire et d'informatique mathématique (LaCIM) at the Université du Québec à Montréal (UQAM)*
- *Bernard Moret is with The Laboratory for Computational Biology and Bioinformatics at Ecole Polytechnique Fédérale (EPFL) de Lausanne, Switzerland and the Swiss Institute for Bioinformatics*

- *A preliminary version of this paper appeared at ISBRA 2010[15]*



(a) Tree 1 (b) Tree 2 (c) Tree 3

(d) MAST/MIST (e) Strict Consensus

Fig. 1. An instance where the strict consensus tree has more internal edges than any agreement subtree.

Cranston and Rannala recently presented a Markov Chain Monte Carlo (MCMC) method for identifying a version of rogue taxa in the context of Bayesian phylogenetic reconstruction [6]. Their approach identifies subsets of leaves for which the posterior distribution strongly supports the structure of the induced subtree—leaves left out can be viewed as rogue taxa, albeit in the narrow context of a sampling of trees in a Bayesian search, rather than in the general context of a consensus of trees.

All of the approaches mentioned thus far fall into the category of "leaf-dropping methods," in the terminology of Redelings [16]. In contrast, Redelings presents, again in the context of Bayesian phylogenetics, a method that returns a "multi-connected tree," which includes all leaves, but does not summarize the information through a single tree and thus again raises issues of interpretation—an issue plaguing all approaches producing non-trees [2], [5], [10], [11]. This is not to say that consensus methods are without question preferable to methods that return non-trees. Gauthier and Lapointe [10] present examples where very informative results are returned in non-trees, and there is certainly future work to be done in deciding the optimal manner in which consensus information can be best utilized by practitioners. However, we argue that the staying power and overwhelming popularity of the consensus methods (particularly the majority rules consensus method) lie primarily it their ability to return a single tree.

In this paper we contribute a new framework and optimization criterion, based on the tradeoff between dropping leaves and uncovering additional consensus structure. Most methods based on leaf-dropping will freely discard leaves in order to uncover *any* underlying structure; in contrast, our approach sets up a bicriterion problem, in which leaves are discarded only if the gain in uncovered internal edges outweighs the loss incurred by discarding the leaves. We define a formal measure of *relative information content* (unrelated to classical information theory, in contrast to the measures proposed by Thorley[20] or Gauthier and Lapointe[10] which are rooted in information theory) to capture the tradeoff and formulate a bicriterion optimization problem based on this measure. We are not the first researchers to define some notion of relative information content for consensus trees [10], [20], but our definition is the first to take into account *explicitly* the loss incurred by dropping taxa, as well as to generalize beyond the setting of agreement subtrees.

Finally, we describe an effective greedy heuristic to compute a good (if not necessarily optimal) set of rogue taxa, provide an implementation within the popular RAxML phylogenetic reconstruction package, and report on its application to both pathological examples from the literature and a collection of large biological datasets that we used in a prior study of bootstrapping. As the presence of rogue taxa in a set of bootstrap replicates can lead to deceivingly poor support values, we propose a procedure to recompute support values in light of the rogue taxa identified by our algorithm; applying this procedure to our biological datasets caused a large number of edges to change from "unsupported" to "supported" status, indicating that many existing phylogenies should be recomputed and reevaluated to reduce any inaccuracies introduced by rogue taxa.

The conference paper on which this work is based [15] introduced the relative information content, the bicriterion optimization, and the greedy heuristic. This extended version again covers these topics, and then discusses the issues of implementation with RAxML, with particular emphasis on speed of execution, and presents much more extensive experimentation, including the analysis of some very large datasets. The rest of the paper is organized as follows. In Section 2 we define concepts and terminology. In Section 3 we define our measure of relative information content, formalize the bicriterion optimization problem for consensus and rogue taxa, and present some theoretical results that underlie our approach. In Section 4 we present an efficient greedy heuristic for our bicriterion problem, as well as detail the implementation effort required to incorporate our algorithm into RAxML. In Section 5 we present the results of our experiments. In Section 6 we detail the performance of our algorithm on large datasets.

Before moving into the main body of the paper, we note that consensus tree building is only one of many phases in phylogenetic reconstruction where rogue elimination/removal may prove fruitful. Ultimately, it would be ideal to eliminate rogues from the leaf set before reconstruction begins. To this end, we discuss briefly in the conclusion a few ways that rogues may arise, and leave a systematic treatment of this deep issue as

future work. In other words, eliminating rogues at the source appears more difficult for now than detecting and eliminating them at the reconstruction stage.

2 Preliminaries

We use standard set and graph terminology and notation; in particular, $\cup$ refers to union, $\cap$ to intersection, $\setminus$ to set difference, and Δ to symmetric difference—i.e., $S \Delta T = (S \cup T) \setminus (S \cap T)$.

A *phylogenetic tree* represents the evolutionary relationships among a collection of living organisms. Homologous data (typically molecular sequences), with one datum for each organism, are placed at the tips of the tree—hereafter called the *leaves*; the internal structure of the tree—its *edges* (sometimes also called branches)—represents the evolutionary relationships. The removal of an edge disconnects the tree and partitions the set of leaves into two subsets; thus each edge corresponds to a *bipartition* of the set of leaves. Every tree includes the same *trivial bipartitions*, which separate one leaf from all others; the other bipartitions are called *nontrivial* and correspond to an *internal* edge of a tree, that is, an edge not incident on a leaf. We can thus view a phylogenetic tree as a leaf-labeled tree $T = (L, B)$, where L is the set of leaves and B is its set of nontrivial bipartitions. To describe a bipartition, we list the two sets of leaves, separated by a | symbol. To ensure an equivalence between nontrivial bipartitions and internal edges, we require that every internal node in a phylogeny have degree at least 3. The number $|B|$ of nontrivial bipartitions in a phylogeny is at most $|L| - 3$; when the upper bound is met we say that the tree is *fully resolved* or binary; otherwise, there must exist an internal node of degree at least 4 and any such node is known as a *polytomy*.

The *consensus* problem is given by a set $\mathcal{T}$ of m trees defined on a common set L of n taxa (leaves). The *bipartition profile* of $\mathcal{T}$ is the pair

$$\mathcal{P} = (B_{\mathcal{T}},\, \nu \colon B_{\mathcal{T}} \to 2^{\mathcal{T}})$$

where $B_{\mathcal{T}}$ is the set of all nontrivial bipartitions found across all m trees (equiv. the union of the bipartitions of the m trees in T) in the set and ν is a function mapping bipartitions to the trees in which they appear.

We denote the removal of leaves from trees through the *restriction* operator—which also uses the | symbol. For example, $\mathcal{T}|L'$ refers to restricting each tree in the set $\mathcal{T}$ to the leaf subset $L' \subseteq L$, which corresponds to removing each leaf in $L \setminus L'$ from each tree, as well as removing any nodes of degree 2 created in the process. Individual trees, tree sets, and bipartition profiles can appear on the left-hand side of the restriction operator.

We focus on consensus methods based on bipartition *frequency*—see the excellent survey of Bryant [4] for a comprehensive treatment of consensus methods. Given a threshold parameter $\frac{m}{2} < t < m$, the t-consensus tree is composed of all of the bipartitions that occur in more than t trees. The *majority rule* consensus [12] is obtained by setting t to $\frac{m}{2}$, while the *strict* consensus is obtained by setting t to $m - 1$. We denote t-consensus methods by $\mathcal{C}_t$. Thus $\mathcal{C}_{m-1}(\mathcal{T})$ corresponds to taking the strict consensus tree of the set $\mathcal{T}$.

3 Consensus, Relative Information Content, and Rogue Taxa

3.1 The measure and the problem

The general problem we study can be phrased as follows: given a set $\mathcal{T}$ of trees on a common leaf set L and given a frequency-based consensus method $\mathcal{C}_t$, we want to find a leaf subset L' that optimizes the relative information content of the consensus returned by $\mathcal{C}_t$ on the set of subtrees induced by L'. The crucial notion here is that of relative information content. Formally, if $\mathcal{C}_t(\mathcal{T}|L')$ yields $T' = (L', B')$, then the *relative information content* is given by

$$I(T', L, \mathcal{C}_t) = \frac{|L'| + |B'|}{|L| + (|L| - 3)} \tag{1}$$

This measure is the ratio of the total number of bipartitions (trivial and nontrivial) in the consensus tree derived on the reduced leaf set to the total number of bipartitions in an ideal, fully resolved tree on the original leaf set. By taking trivial bipartitions into account, we automatically penalize a method for removing many leaves, since the number of trivial bipartitions is simply the number of leaves. By adding the number of nontrivial bipartitions, we reward a method for preserving more internal edges, since the denominator is fixed to the number of such edges in an ideal tree. Note that the use of the word 'information' in our definition does not imply information-theoretic foundations. (A definition of relative information content that does not explicitly reference a consensus method may be of independent interest. In Eqn. 1, if $T' = (L', B')$ is provided independently, then $\mathcal{C}_t$ can be dropped from the arguments of I.)

A curious property of our definition for RIC is how it performs in scoring poorly resolved consensus trees. Specifically, using the strict consensus method as an example, it is the case that a dataset whose consensus tree is the star tree (i.e. contains zero nontrivial bipartitions) will have RIC tending toward $\frac{1}{2}$. That this value is nonzero is seemingly at odds with a common perception that a star tree conveys no information. Again, this is precisely the point of explicitly including leaves in our formulation of RIC, as we contend that the leaves themselves convey information. Finally, notice that it is possible to have an RIC of value less than $\frac{1}{2}$ (in fact, an RIC of zero is possible) when we consider dropping leaves in a way that the consensus of the restricted set is still poorly resolved.

We now formulate our main problem, which we call *MISC*, for *Maximum-Information Subtree Consensus*.

Problem Given a set $\mathcal{T}$ of trees defined on a common leaf set L and a frequency-based consensus method $\mathcal{C}_t$, find

a leaf subset L' that maximizes the relative information content $I(\mathcal{C}_t(\mathcal{T}|L'), L, \mathcal{C}_t)$.

The MAST solution typically maximizes the $|B'|$ term at the expense of the $|L'|$ term—it has no direct penalty for dropping leaves. In contrast, consensus methods typically maximize $|L'|$ (in the case of majority and strict consensus, by forcing $L' = L$) at the expense of $|B'|$. MISC, on the other hand, combines the two aspects into a single formulation.

Before moving on, we propose an extension to RIC that may warrant further exploration. Specifically, it may make sense to redefine RIC more generally as

$$I(T', L, \mathcal{C}_t) = \frac{\alpha|L'| + \beta|B'|}{|L| + (|L| - 3)} \quad (2)$$

subject to α and β being nonnegative. This formulation affords control of the emphasis placed on leaves versus nontrivial bipartitions as being informative in a phylogeny. This is similar in spirit to the parameters defined in MIST by Bryant[3] in the agreement subtree setting. For the remainder of this paper, though, we assume that $\alpha = \beta = 1$.

3.2 How bipartitions change under leaf deletion

We begin by studying the effect that dropping leaves has on a bipartition profile. For any bipartition in the original profile, there are three cases. We illustrate these cases through a simple example, with an original leaf set of a, b, c, d, e, f and with leaves b and e dropped.

1) **merge:** If two bipartitions differ solely in (a subset of) the leaves being dropped, then those bipartitions get merged in the new profile. For example $ac|bdef$ and $abc|def$ merge into $ac|df$ and the ν set for the merged bipartition consists of the union of ν sets of the two original bipartitions.
2) **disappear:** If dropping the leaves creates a bipartition with an empty side or makes the bipartition trivial, then the bipartition disappears. For example, both $acdf|be$ and $acd|bef$ disappear.
3) **no change:** Otherwise, the restricted bipartition remains in the bipartition profile with an unchanged ν.

An important observation is that, for all $L'' \subseteq L' \subseteq L$, every nontrivial bipartition in $\mathcal{P}|L''$ and in $\mathcal{C}_t(\mathcal{T}|L'')$ arises as a result of a "no change" of a single bipartition or a "merge" of two or more bipartitions in $\mathcal{P}|L'$. Unfortunately this observation has not led to an efficient exact algorithm for the MISC problem.

3.3 Finding subsets of leaves to drop

Given two bipartitions b_1 and b_2 of L, we can easily identify all leaf subsets L' of minimum cardinality such that dropping L' from L merges b_1 and b_2. If we have $b_1 = A|B$ and $b_2 = C|D$, then the *dropset* L' is the smaller of the two following sets (or either set in case they have the same size):

$$(A\Delta C) \cup (B\Delta D) \text{ or } (A\Delta D) \cup (B\Delta C) \quad (3)$$

The dropset computation can be simplified to

$$(A\Delta C) \text{ or } (A\Delta D) \quad (4)$$

by observing that $(A\Delta C) = (B\Delta D)$ and $(A\Delta D) = (B\Delta C)$ due to A and B (resp. C and D) being complementary.

This concept is exploited in Algorithm 1 (see the pseudocode below). Observe that, in the terminology of [17], removing the dropset of b_1 and b_2 yields the largest partial X-split such that b_1 and b_2 both extend it.

Theorem 1. *Algorithm 1 computes the minimum cardinality dropsets (of which there will be at most two such dropsets) for any pair of bipartitions of L.*

Proof: That the dropset causes the two partitions to merge is evident. We establish that the dropset has minimum cardinality by contradiction. Consider that there exists a smaller dropset merging the two bipartitions. Then there is at least one leaf ℓ in the dropset returned by our algorithm that is not in the smaller dropset. This leaf must be on the same side of the partition in both b_1 and b_2, since otherwise our dropset would not merge the two. But our algorithm uses the symmetric difference of these two sides in computing the dropset, so it could not have chosen ℓ, a contradiction. □

Theorem 2. *The cardinalities of the dropsets returned by Algorithm 1 define a metric on the space of bipartitions of L.*

Proof: Three properties characterize a metric: it must be positive definite and symmetric, and it must obey the triangle inequality. The first two properties are trivial in this case. Suppose we have bipartitions b_1, b_2, and b_3; we want to show that the cardinality of the dropset of b_1 and b_3 cannot exceed the sum of the cardinalities of the dropsets of b_1 and b_2 and of b_2 and b_3. Note that removing both of these dropsets from both b_1 and b_3 merges the two bipartitions, thereby establishing an upper bound on the distance between these two bipartitions in our space;

Algorithm 1 Find minimum cardinality leaf-dropset that renders $b_1 = b_2$

Input: two bipartitions on the same leaf set
Output: a set of dropsets (of cardinality one or two)

```
function BIPARTITION-PAIR-DROPSET(b1 = A|B, b2 = C|D)
    S0 ← AΔC
    S1 ← AΔD
    if |S0| < |S1| then
        return {S0}
    else if |S1| < |S0 then
        return {S1}
    else
        return {S0,S1}
    end if
end function
```

but the distance is the size of the dropset of b_1 and b_3, so that the triangle inequality holds. □

We note that Theorem 1 is not used in the remainder of this paper, but has spurred other research investigations which are currently in submission.

4 The Algorithm

4.1 Description and Pseudocode

We describe the algorithm at a conceptual level, leaving a more formal specification to inset pseudocode. First, we build the bipartition profile for the given tree set. Next, we compute the dropset for each pair of bipartitions in the profile such that neither bipartition in the pair appears in the consensus tree, but the pair would appear if merged. For each unique dropset we accumulate the list of bipartition pairs yielding that dropset. These last two steps are formalized in Algorithm 2. We then compute the *impact* of each dropset as the number of bipartition pairs giving rise to that dropset minus the size of the dropset itself. This score corresponds roughly to the difference between the number of edges that will be created and the number of leaves that will be lost should that dropset be used. The dropset of largest impact is then used, the profile updated, the impacts updated, and the process repeated until there does not remain any dropset with a nonnegative impact. This greedy framework is formalized in Algorithm 3. The impact measure ignores disappearing edges and dropsets that are subsets of another—the latter because a superset with deceivingly poor score is likely to get chosen in a subsequent round. The overall algorithm is a greedy heuristic, but does well in practice and on hard instances, as we demonstrate in the next two sections.

There remains the issue, as with all leaf-dropping methods, of what to do with the dropped leaves. The staying power of consensus methods argues for producing a single tree and our method does that. For rogue taxa, we provide a strategy in Section 5.3 that is applicable in some settings.

Algorithm 2 Find potential dropsets by examining all pairs in a profile

Input: A bipartition profile $\mathcal{P} = (B_\mathcal{T}, \nu : B_\mathcal{T} \to 2^\mathcal{T})$
Input: A frequency-only consensus method $\mathcal{C}_t$ with threshold t
Output: An object mapping dropsets to lists of bipartition pairs

function POTENTIAL-PROFILE-DROPSETS($\mathcal{P}$, $\mathcal{C}_t$)
  $\Gamma \leftarrow \{b \mid b \in B_\mathcal{T} \text{ and } |\nu(b)| \leq t\}$
  for all pairs of bipartitions b_1,b_2 in Γ **do**
    if $|\nu(b_1) \cup \nu(b_2)| > t$ **then**
      $L \leftarrow$ BIPARTITION-PAIR-DROPSET(b_1, b_2)
      for $d \in L$ **do**
        $\delta[d] \leftarrow \delta[d] \cup \{(b_1, b_2)\}$
      end for
    end if
  end for
  return δ
end function

Algorithm 3 Our top level iterative heuristic for finding dropsets

Input: A tree set $\mathcal{T}$
Input: A frequency-only consensus method C with threshold t
Output: A set of leaves to drop, composed of the union of dropsets

function SELECT-AND-REMOVE-DROPSETS($\mathcal{T}$)
  $d^* \leftarrow d_{greedy} \leftarrow \emptyset$
  repeat
    $\mathcal{P} \leftarrow$ BUILD-BIPARTITION-PROFILE$(\mathcal{T}|(L - d^*))$
    $\delta \leftarrow$ POTENTIAL-PROFILE-DROPSETS$(\mathcal{P}, \mathcal{C}_t)$
    $maximpact = 0$
    $d_{greedy} = \emptyset$
    for all $d \in \delta$'s domain **do**
      if $|\delta[d]| - |d| \geq maximpact$ **then**
        $d_{greedy} = d$
        $maximpact = |\delta[d]| - |d|$
      end if
    end for
    $d^* = d^* \cup d_{greedy}$
  until $d_{greedy} = \emptyset$
  return d^*
end function

4.2 Implementation and Optimization

Since the preliminary version of this paper, we have implemented and optimized our algorithm as part of RAxML [18]. This was a logical choice as the RAxML codebase provides efficient data structures for operations on bipartitions. RAxML v7.2.7, which implements the leaf-dropping algorithm, is available for download at http://wwwkramer.in.tum.de/exelixis/software/

4.2.1 Implementation

In the RAxML implementation, the bipartition profile of the input tree collection and the set comprising potential dropsets are stored as hash-tables. In addition to the quick indexing afforded by the bipartition profile hash-table, it also provides a mapping to the set of trees that support a particular bipartition. The dropset hash-table provides a mapping of the taxa in a dropset to the number of pairs of bipartitions that will be merged when applying the dropset.

The sets in the bipartition profile, that is, the bipartitions induced by the tree collection, are hashed as bit vectors. To save memory in the dropset hash-table, dropsets are hashed as list of (taxon) indices. This is because they are typically very small sets, and a bit-vector representation would prove sparse (thereby requiring an

excessive amount of memory). A representation as bitvector is advantageous if, on average, more than 1 bit (per 32 bits) is set, that is, more than $\#taxa/32$ taxa would be dropped. Thus, to compute entries for the dropset hash-table we initially apply a XOR operation on bitvectors. The bits that are set in the result are then converted into an index list.

4.2.2 Optimization

The operation for computing candidate dropsets (this operation has complexity $O(|B_{\mathcal{T}}|^2)$, where $B_{\mathcal{T}}$ is the number of bipartitions in the profile) dominates the execution time of the unoptimized algorithm and accounts for approximately 97% of overall execution time.

To improve the performance of this operation we use sorting along with the (consensus specific) threshold t (see Sect. 2 for the definition of $\mathcal{C}_t$ and t) value to avoid unnecessary computations. Specifically, we begin by sorting the list of bipartitions by their frequency of occurrence. Then we check pairs of bipartitions (b_i, b_j), where $i \neq j$, by order of decreasing frequencies. When the sum of the frequencies of b_i and b_j is below the frequency threshold t (see Section 2), no other (less frequently occurring) bipartition b_k, where $k < j$, can be merged with b_i to form a bipartition with a higher overall frequency. Hence, we can omit comparisons to all successive bipartitions that are stored in the sorted list because they have a lower frequency.

In the optimized algorithm, the computation of candidate dropsets requires 60% of overall execution time.

5 Algorithm Performance

In the following, we present results on artificial datasets constructed to cause difficulties to various consensus methods, followed by results on biological datasets that we used in previous work on bootstrapping. We then discuss implications of our results on the interpretation of phylogenetic reconstruction. We conclude by a smaller study on biological datasets using a slight modification of our algorithm to maximize the number of nontrivial bipartitions in the result.

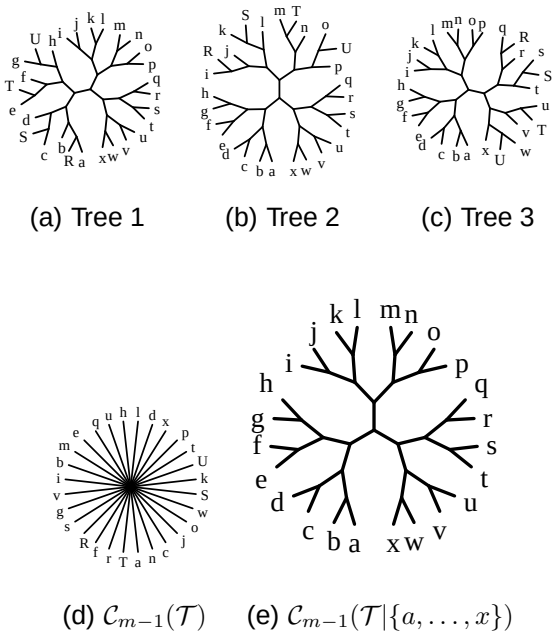
5.1 Difficult instances

Our algorithm is particularly well suited to the so-called "pathological" instances used in the literature to critique the strict or majority consensus. In this section we cover a number of specific instances and instance families which exhibit the inherent limitations of frequency-based consensus methods, the effectiveness of our approach, as well as limitations with our approach.

5.1.1 A First Example

A classic example is an instance where the trees share a common subtree of $n-k$ leaves, but where the remaining k leaves destroy resolution in the consensus.

This example uses the strict consensus. An instance consists of just three trees, defined on the 28-leaf set



(a) Tree 1 (b) Tree 2 (c) Tree 3

(d) $\mathcal{C}_{m-1}(\mathcal{T})$ (e) $\mathcal{C}_{m-1}(\mathcal{T}|\{a,\ldots,x\})$

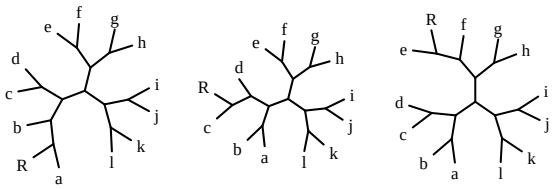
Fig. 2. A simple, yet starkly contrasting, example (top) for which the strict consensus returns a star tree, but for which our algorithm correctly identifies the rogue taxa and produces a fully resolved tree (bottom).

$\{a, b, \ldots, x, R, S, T, U\}$. The common backbone consists of the 24 taxa $\{a, b, \ldots, x\}$, as illustrated in Figure 2(e)). The rogue taxa form the set $\{R, S, T, U\}$; they vary in position on the backbone as indicated in Figures 2(a), 2(b), and 2(c). The strict consensus tree of the three trees is shown in Figure 2(d): it is a star, with no nontrivial bipartition (no internal tree edge) and its relative information content is $I(\mathcal{T}, L, \mathcal{C}_{m-1}) = \frac{28+0}{28+25} = \frac{28}{53} \approx 0.53$. Our algorithm correctly identifies the rogue set, however, so that its strict consensus tree on the remaining set of leaves is the backbone, with a relative information content of $I(\mathcal{T}|\{a,\ldots,x\}, L, \mathcal{C}_{m-1}) = \frac{24+21}{28+25} = \frac{45}{53} \approx 0.85$.

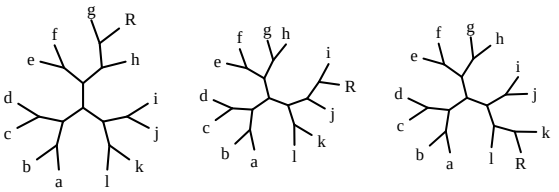
5.1.2 The 1-Cherry Trees

The behavior exhibited in the previous example is not limited to small trees. In this section we introduce a simple, but infinitely large, family of instances exhibiting similar behavior. An instance in this family is fully specified by a parameter k. An instance has $m = 3 \cdot 2^k$ trees, and $n = 3 \cdot 2^{k+1} + 1$ leaves. The common backbone that exists in each tree consists of all but one of the leaves and structurally is a fully balanced binary tree. The remaining leaf, name it R wanders and occurs together with every second leaf X in a bipartition of the form $RX|rest$ (which is colloquially referred to as a *cherry* in phylogenetics, hence the name of the family). See Figure 3 for the cherry tree with $k = 1$.

Our algorithm correctly identifies the rogue taxon in all cherry trees. The relative information content of



(a) Tree 1 (b) Tree 2 (c) Tree 3



(d) Tree 4 (e) Tree 5 (f) Tree 6

Fig. 3. The second instance (the first is $k = 0$) of our family of cherry trees. There are $3 \cdot 2^k = 6$ trees on $3 \cdot 2^{k+1} + 1 = 13$ leaves.

cherry tree k is

$$\frac{3 \cdot 2^{k+1} + 1}{(3 \cdot 2^{k+1} + 1) + (3 \cdot 2^{k+1} + 1 - 3)} = \frac{3 \cdot 2^{k+1} + 1}{2 \cdot 3 \cdot 2^{k+1} - 1}$$

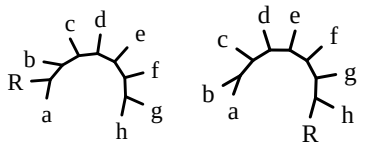
which in the limit $k, n, m \to \infty$ tends toward $\frac{1}{2}$. After rogue identification and elimination by our algorithm, which in this case optimally solves MISC-$\mathcal{C}_{m-1}$, the relative information content is

$$\frac{3 \cdot 2^{k+1} + 3 \cdot 2^{k+1} - 3}{(3 \cdot 2^{k+1} + 1) + (3 \cdot 2^{k+1} + 1 - 3)} = \frac{2 \cdot 3 \cdot 2^{k+1} - 3}{2 \cdot 3 \cdot 2^{k+1} - 1}$$

which in the limit $k, n, m \to \infty$ tends toward 1.

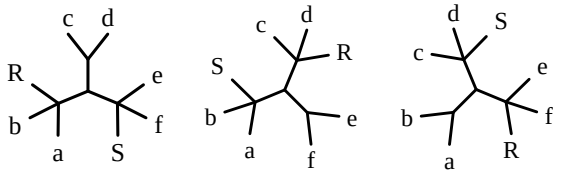
5.1.3 The r-Cherry Trees

The 1-cherry tree instances are somewhat unsatisfying as pathological instances because a much simpler strategy than our algorithm is sufficient for finding the single rogue taxon (e.g., the polynomial time procedure of dropping each leaf in turn and applying the strict consensus method to assess relative information content of the result). However, it is straightforward to adapt the basic principle behind 1-cherry trees to induce an effectively arbitrary number of rogue taxa. Given k and a desired number of rogue taxa r (where r divides $\frac{n}{2}$ evenly), we take $n = 3 \cdot 2^{k+1} + r$ and $m = \frac{3 \cdot 2^k}{r}$. In the first tree, the rogue taxa are attached as cherries to every second taxon, for a total of $\frac{n}{r}$ taxa. In the second tree, the rogue taxa are attached to the next $\frac{n}{r}$ (while attaching as a cherry to every second) taxa. This pattern continues in each subsequent tree. An instance of this family is actually what we showed in Sect. 5.1.1 as a motivational example for our algorithm (Figure 2), which is the $k = 2$'th 4-cherry tree (i.e. $k = 2$ and $r = 4$).



(a) Tree 1 (b) Tree 2

Fig. 4. An instance where all of the internal structure of the strict consensus tree is destroyed by a single rogue taxon, even with only two trees in the set.



(a) Tree 1 (b) Tree 2 (c) Tree 3

Fig. 5. This instance will not be solved correctly by our algorithm. This is because identifying R and S as rogue requires merging three bipartitions.

5.1.4 The Comb/Caterpillar

Another instance (family) that exhibits extremal behavior with respect to relative information content arises when subjecting the *comb* or *caterpillar* tree (so named by their appearance when drawn) to rogue taxa. The simplest case consists of two trees, on an arbitrary number of taxa, where the rogue taxon occurs at each end of the backbone tree. See Figure 4 for the instance of this family with nine taxa. This family of caterpillar instances is also often used to demonstrate a brittleness of the Robinson-Foulds metric. Namely, the RF distance between the two trees is maximum (for the given number of taxa), whereas the trees are clearly nearly identical. Our algorithm performs particularly well on caterpillar (and related) instances because hidden edges in such instances are almost always revealed by merging *pairs* of bipartitions.

5.1.5 Instance Requiring a 3-way Merge

As was discussed earlier, bipartitions in a restricted consensus tree can be uncovered as the result of merging three or more bipartitions. Since our algorithm only considers bipartition pairs as merging candidates, an instance which only admits improvement via a 3 (or more) way merge will fail to be solved by our algorithm. Figure. 5 illustrates such an instance. The relative information content of the non-restricted instance is $\frac{8}{13}$ whereas removing leaves R and S yields a situation where the relative information content is $\frac{9}{13}$. However, our algorithm will recommend to not drop any leaves.

5.1.6 Some notes about C_t where $t < m$

All of the instances presented in this section considered relative information content in light of the strict consensus method ($\mathcal{C}_{m-1}$). However, it is trivial to adapt these instance families into easy/difficult cases for our algorithm when operating under the majority rule ($\mathcal{C}_{\frac{m}{2}}$) consensus method. Namely, for any instance discussed so far with m trees, add an additional $m-1$ *star* trees (recall that the star tree contains zero nontrivial bipartitions). This simple transformation yields instances that perform identically with respect to relative information content (with $\mathcal{C}_{\frac{m}{2}}$ substituted for $\mathcal{C}_{m-1}$).

It is also interesting to note that all of the examples presented in this section can be optimally solved by MAST/MIST. This is evidence of the logic behind conceiving these instances. That is, start with a common backbone tree (which in the end turns out to be the MAST/MIST), and add rogues in various ways so as to confound the consensus method. However, as we showed in Fig. 1, MAST/MIST will not always optimally solve MISC even when the consensus method is specified as strict consensus. And, clearly, the usefulness of MAST/MIST diminishes in relation to solving MISC when we consider consensus methods where the threshold is less than strict.

Refer to Table. 1 for a comparison of dropset sizes for MAST/MIST and those that our algorithm returns, along with the corresponding relative information content of each solution.

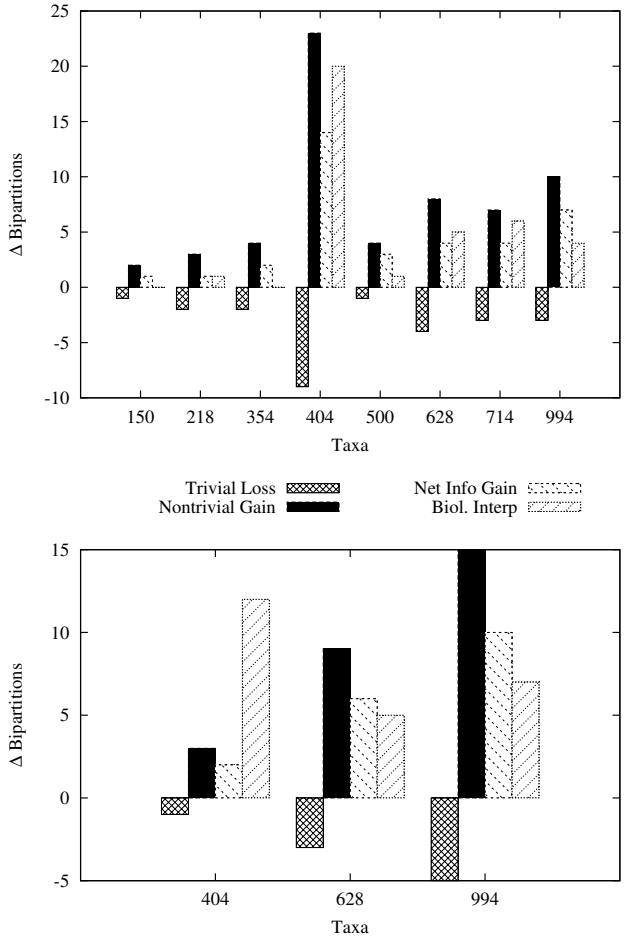


Fig. 6. The performance of Algorithm 3 in terms of how much "hidden" consensus is uncovered in biological data sets. The top plot is for majority consensus, the bottom for strict consensus. The tree sets each consist of 1,000 bootstrap replicates generated by the RAxML 7.2.5 Rapid Bootstrap Algorithm.

5.2 Results on biological data

We applied our method to the datasets we used in an earlier study of bootstrapping methods [13] and available at http://lcbb.epfl.ch/BS.tar.bz2. There are 10 datasets of single-gene and multi-gene DNA sequences, with anywhere from 125 to 994 taxa. For each dataset we generated 1,000 bootstrap replicates and applied our algorithm to the resulting trees using both $\mathcal{C}_{\frac{m}{2}}$ and $\mathcal{C}_{m-1}$. Our algorithm found rather diverse dropset sizes across the 10 datasets. The results are depicted in Figure 6, where a quartet of histogram bars are shown for each dataset with a nonempty dropset. The first histogram bar (a negative quantity) denotes how many leaves were dropped, while the second bar (a positive quantity) denotes how many nontrivial bipartitions were uncovered. The third bar is the sum of the first two, simply depicting the net (non-normalized) contribution to relative information content. The final bar is discussed in Section 5.3.

For comparison, we have included in Table 1 the size of the MAST/MIST (which are equivalent in this case as all of the input trees are binary) for our datasets.

5.3 Biological interpretation

Maximum likelihood phylogenetic analyses are typically conducted in two steps. First the reconstruction proper is performed, yielding a "best tree." Then a number of bootstrap replicate trees are generated, say 500 of them; for each bipartition b in the best tree, its support value is calculated as a normalized count of the number of replicates in which b appears. Researchers tend to consider edges with support lower than 75% as unreliable [8].

If rogue taxa are present in the replicate set, the support values for certain bipartitions can be deceivingly depressed. To remedy this problem, we propose that Algorithm 3 be applied to the replicate set in order to identify rogue taxa. If a dropset of nonzero size is found, this dropset is then removed from each tree in the replicate set. Finally, the support value is calculated as a normalized count of the replicates in which b' appears such that, if we have $b = A|B$, then, without loss of generality, we have $b' = A'|B'$ such that $A' \subseteq A$ and $B' \subseteq B$. In this way, support values in the "best tree" are less susceptible to the deceiving influence of rogue taxa. This approach offers one possible solution to

taxa	125	150	218	354	404	500	628	714	994
RIC for full dataset (MR)	0.992	0.848	0.790	0.711	0.806	0.853	0.816	0.841	0.889
RIC for MISC (MR)	0.992	0.852	0.790	0.713	0.822	0.855	0.820	0.844	0.893
RIC for full dataset (SC)	0.895	0.572	0.550	0.517	0.524	0.555	0.547	0.561	0.589
RIC for MISC (SC)	0.895	0.572	0.550	0.517	0.527	0.555	0.551	0.561	0.594
RIC for MAST	0.700	0.118	0.081	0.021	0.125	0.061	0.042	0.061	0.045
taxaInMISC (MR)	125	149	214	352	393	496	624	708	991
taxaInMISC (SC)	125	150	218	354	403	500	625	714	989
taxaInMAST	88	19	19	9	52	32	28	45	46

TABLE 1
A comparison of our dropset sizes versus the number of leaves dropped in a MAST/MIST. Note that MISC refers to the output of our (greedy) algorithm, and not necessarily the optimal solution to MISC. SC refers to the Strict Consensus and MR refers to Majority Rules Consensus.

the data display problem of leaf-dropping methods. We still return a single tree on the original leaf set (the "best tree" as reconstructed by an ML method), but support values for individual bipartitions more accurately reflect the underlying replicate data. In our datasets, recomputing support values as suggested above yields very intriguing and promising results. All but two of the identified dropsets succeeded in pushing at least one previously hidden edge in the "best tree" over the 75% threshold. The number of edges uncovered by this application of our technique is displayed in the fourth histogram bar in Figures 6(a) and 6(b). In the dataset with 404 taxa, 20 edges were uncovered in this manner, pointing to a need for reevaluation of the phylogeny.

5.4 Increasing resolution

Our algorithm can easily be modified to maximize non-trivial bipartitions, that is, to remove taxa so as to increase resolution. With such a setting, our algorithm loosely matches the goal of Cranston and Rannala [6], so we analyzed the same dataset with our technique to compare our results to theirs. The data set consists of 85 species of Canformia Carnivora [9]. We obtained the sequence data from TreeBASE (http://www.treebase.org, Study Accession # S1532) and reconstructed a tree using RAxML-7.2.5 [18] under the GTRCAT approximation. Additionally, RAxML was used to generate 350 bootstrap replicates (the number chosen by RAxML's bootstopping algorithm). Analyzing these 350 trees with our modified Algorithm 3 and using majority consensus generated fully resolved trees with 50 to 55 taxa, a value consistent with the size of the agreement subtrees observed by Cranston and Rannala [6].

6 Runtime Performance

Execution times for the optimized RAxML implementation were measured on collections of 10,000 bootstrap replicate trees on 16 real biological datasets containing between 150 and 2,554 taxa [13]. The test runs were conducted on a Sun x4440 server (24 AMD cores, 128GB RAM) and on a Sun x4600 server (32 AMD cores, 64GB RAM).

In Table 2 we provide the overall execution times in seconds as well as the execution times of the dropset phase (see Section 4.2) for the unoptimized and optimized RAxML implementations. Note that, in the unoptimized case, dropset computation typically accounts for more than 90% of total runtime. In row # *bipartitions* we also indicate the total number of unique bipartitions in each tree collection.

As discussed in Sect. 4.2.2, the optimization of the dropset calculation phase avoids unnecessary computation on pairs of bipartitions that cannot be merged into a frequently (specifically, with frequency exceeding t, where t was defined in Sect. 2) occurring bipartition. This simple shortcut yields overall runtime improvements between one to two orders of magnitude. This improvement is achieved because a large fraction of bipartitions stored in the profile occur in few trees, so that a substantial number of pairwise operations on bipartitions can be avoided (and the effect is more pronounced with higher values of t).

The run-time contribution of dropset computations decreases in the optimized implementation because other operations, such as parsing input trees, building a bipartition profile, and reconstructing an output tree from a set of bipartitions, now require a larger fraction of runtime. The optimized RAxML implementation is two to three orders of magnitude (dataset specific) faster than the original python script.

Table 2 also shows that execution time and number of taxa are not strongly correlated. The run time of the most computationally intensive phase, the dropset calculation, depends on (i) the size of the bipartition profile which grows with the number of taxa and also grows, but saturates, with the number of trees, (ii) the number of iterations required until no additional dropset with positive impact can be found, and (iii) the size of the bit-vectors, that increases linearly with the number of taxa *and* number of trees.

To illustrate factor (ii), see Table 2 which shows that 40 dropset identification iterations were required for the 2,000 taxon dataset, whereas only 8 iterations were required for the 2,308 taxon dataset. To illustrate factor (i), our tree collection containing 2,304-taxon trees contains significantly fewer unique bipartitions than the 2,000-

# taxa	150	218	354	404	500	628	714	994
run-time	2.9	23.1	1,908.9	837.5	106.2	267.4	413.9	254.9
dropset time	2.1	21.3	1,902.7	830.9	101.1	260.9	403.6	243.9
opt. run-time	0.8	1.9	5.7	9.1	6.1	8.1	14.3	14.2
opt. dropset time	0.1	0.2	0.3	2.1	1.1	1.4	5.3	3.1
# bipartitions	11,336	23,276	170,397	73,434	42,630	75,156	61,280	64,497
# taxa	1,288	1,481	1,512	1,604	1,908	2,000	2,308	2,554
run-time	6,673.9	634,188.4	225,587.1	141,573.5	61,942.2	1,704,656.0	5,759.9	423,813.2
dropset time	6,640.9	634,025.2	225,451.7	141,406.6	61,854.0	1,704,244.0	5,693.1	423,522.0
opt. run-time	49.9	238.6	166.5	223.7	117.5	727.4	114.6	743.8
opt. dropset time	18.7	75.4	54.2	106.2	43.0	315.4	42.5	452.6
# bipartitions	178,242	718,719	589,985	352,229	439,279	761,654	156,812	470,434

TABLE 2
Overall execution times (run-time) and dropset phase execution times (dropset time) for the unoptimized and optimized (opt.) RAxML implementations in seconds. Data for 10,000 BS replicate trees on 16 real-world datasets with 150 up to 2,554 taxa.

taxon and 2,554-taxon tree collections.

These confounding factors make it difficult to forecast runtimes for our implementation on larger datasets. Despite these challenges, we are currently able to handle datasets of 10,000 trees on 2,554 taxa, with more than 760,000 unique bipartitions—a size that should cover most datasets analyzed today.

Finally, we explored the scalability limits of the RAxML implementation by using 672 bootstrap replicate trees generated for a phylogenetic analysis of angiosperms comprising 37,831 taxa (Stephen A. Smith, Jeremy M. Beualieu, A. Stamatakis, submitted). We found that, despite using a multi-core system with 128GB of main memory for testing, memory consumption represents the main limiting factor (in the sense that paging is required), when using a non-strict consensus threshold. When using a strict consensus threshold, our implementation successfully completed and revealed 208 new strict consensus edges by dropping 13 taxa (which is more than an 8% gain in consensus bipartitions). Memory issues can be handled by moving to larger workstations (high-end workstations today can be configured with 1TB of main memory) or, if necessary, to supercomputers; the latter can also help with running times if one uses parallel code.

7 Conclusions and Future Work

We have presented a novel framework to define rogue taxa so as to maximize the relative information present in a consensus tree computed after removing these rogue taxa. This framework defines a bicriterion problem, MISC, that is the first to explicitly balance loss of taxa with gain in resolution in a setting other than agreement subtrees. We have also provided an effective greedy heuristic to find a good set of such rogue taxa. This algorithm was tested on both pathological cases from the literature and a variety of biological data. The changes in the consensus tree can be parlayed into more accurate bootstrap scores, which in turn can lead to the reevaluation of phylogenetic trees, as we showed on our biological datasets.

Rogue taxa do not have a unique source. Some are difficult to place reliably because of poor taxon sampling: no matter where they are placed, they connect to the rest of the tree through a very long edge, so that moving them around a tree hardly alters the score of that tree. For other rogues, poor sequence alignment is the most likely culprit, but that in turn could be due to higher than expected divergence, poor data quality, or poor choice of genomic sequences. Yet others could be created through an unexpected interaction between the reconstruction algorithm and the specific values associated with the rogue and its neighbors. In general, eliminating rogues at the source appears more difficult for now than detecting and eliminating them at the reconstruction stage.

Further work includes a characterization of the computational complexity of the MISC problem, as well as improved algorithms for it, including approximation algorithms with known performance guarantees. Generalizing our approach to support consensus methods other than frequency-based methods is another algorithmic problem worth investigating. Finally, there is certainly room to extend and apply our techniques in different domains, most notably in Bayesian phylogenetics (as suggested in Section 5.4) and for the subtree mergers used in the Disk-Covering Methods (as suggested in [14]). On the bioinformatics side, our preliminary findings indicate that existing phylogenies can be significantly refined by applying our approach to the recomputation of bootstrap support.

Finally, we have followed through on a suggestion made in the preliminary version of this paper, and implemented our approach in the open-source phylogenetics package RAxML. Whereas our original implementation was only suitable for 1,000 taxon, 1,000 tree datasets, our RAxML-based implementation scales to 2,500 taxon, 10,000 tree datasets. We also, remarkably, succeeded in applying our RAxML-based algorithm (using the strict consensus method) to a 38,000 taxon dataset. Our implementation has been fully integrated into RAxML v7.2.7 which is freely available from http://wwwkramer.in.tum.de/exelixis/software/

REFERENCES

[1] A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees. *SIAM J. Comput.* 26:758–769, 1994.

[2] H. Bandelt and A. Dress. Split decomposition: A new and useful approach to phylogenetic analysis of distance data. *Mol. Phyl. Evol.* 1(3):242–252, 1992.

[3] D. Bryant. *Hunting for trees, building trees and comparing trees: Theory and method in phylogenetic analysis*. PhD thesis, U. Canterbury, 1997.

[4] D. Bryant. A classification of consensus methods for phylogenetics. In *Bioconsensus*, vol. 61 of *DIMACS Series in Discrete Math. & Theor. Comput. Sci.*, pages 163–184. AMS Press, 2002.

[5] D. Bryant and V. Moulton. Neighbor-Net: An agglomerative method for the construction of phylogenetic networks. *Mol. Bio. Evol.* 21(2):255–265, 2004.

[6] K.A. Cranston and B. Rannala. Summarizing a posterior distribution of trees using agreement subtrees. *Sys. Bio.* 56(4):578–590, 2007.

[7] M. Farach, T.M. Przytycka, and M. Thorup. On the agreement of many trees. *Inf. Proc. Letters* 55(6):297–301, 1995.

[8] J. Felsenstein. *Inferring Phylogenies*. Sinauer Assoc. Inc., Boston, 2004.

[9] T.L. Fulton and C. Strobeck. Molecular phylogeny of the arctoidea (carnivora): Effect of missing data on supertree and supermatrix analyses of multiple gene data sets. *Mol. Phyl. Evol,* 41(1):165–181, 2006.

[10] O. Gauthier and F.J. Lapointe. Seeing the trees for the network: Consensus, information content, and superphylogenies. *Sys. Bio.* 56(2):345–355, 2007.

[11] D. Huson. SplitsTree: Analyzing and visualizing evolutionary data. *Bioinformatics* 14(1):68–73, 1998.

[12] T. Margush and F.R. McMorris. Consensus n-trees. *Bull. Math. Bio.* 43:239–244, 1981.

[13] N.D. Pattengale, M. Alipour, O.R.P. Bininda-Emonds, B.M.E. Moret, and A. Stamatakis. How many bootstrap replicates are necessary? In *Proc. RECOMB'09*, vol. 5541 of *LNCS*, pages 184–200. Springer, 2009.

[14] N.D. Pattengale, K.M. Swenson, M.M. Morin, and B.M.E. Moret. Higher fidelity subtree merging for disk-covering methods. Poster, Algorithmic Biology, 2006. http://www.calit2.net/events/algorithmicbio/files/PattengaleAlgoBio2006.pdf.

[15] N.D. Pattengale, K.M. Swenson, and B.M.E. Moret. Uncovering Hidden Phylogenetic Consensus In *Proc. ISBRA'10*, vol. 6053 of *LNBI*, pages 128–139. Springer, 2010.

[16] B. Redelings. Bayesian phylogenies unplugged: Majority consensus trees with wandering taxa. http://www4.ncsu.edu/∼bdredeli/wandering.pdf.

[17] C. Semple and M. Steel. Tree reconstruction via a closure operation on partial splits. In *Proc. JOBIM'00*, vol. 2066 of LNCS, pages 126–134. Springer, 2001.

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

[19] A. Stamatakis and M. Ott. Efficient Computation of the Phylogenetic Likelihood Function on Multi-Gene Alignments and Multi-Core Architectures. *Philosophical Transactions of the Royal Society B* 363:3977–3984, 2008.

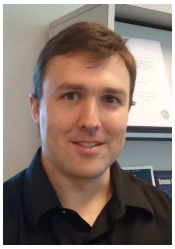
[20] J.L. Thorley, M. Wilkinson, and M. Charleston. The information content of consensus trees. In *Studies in Classification, Data Analysis, and Knowledge Organization*, Adv. in Data Science and Classif., pages 91–98. Springer, 1998.

[21] M. Wilkinson. Common cladistic information and its consensus representation: Reduced Adams and reduced cladistic consensus trees and profiles. *Sys. Bio.* 43(3):343–368, 1994.

[22] M. Wilkinson. More on reduced consensus methods. *Sys. Bio.* 44:435–439, 1995.

[23] M. Wilkinson. Majority-rule reduced consensus trees and their use in bootstrapping. *Mol. Bio. Evol.* 13(3):437–444, 1996.

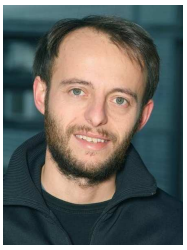
[24] A. J. Aberer, N. D. Pattengale, and A. Stamatakis. Parallelized phylogenetic post-analysis on multi-core architectures. *Journal of Computational Science* 1: 107-114, 2010.



Nicholas D. Pattengale Nicholas (Nick) Pattengale is an algorithms researcher and software engineer at Sandia National Laboratories in Albuquerque, New Mexico, USA. He received a Ph.D. in Computer Science from the University of New Mexico in 2010. His research focus in Computational Molecular Biology (primarily Phylogenetics) has been in deriving efficient algorithms for phylogenetic post-analysis.

Andre J. Aberer Andre J. Aberer is a Masters student in the joint Bioinformatics program of the LMU and TU Munich. His main research interests are statistical modelling, discrete algorithms, and parallel computing in Bioinformatics. In 2011 he will start working on his PhD thesis at the Scientific Computing Group at the Heidelberg Institute for Theoretical Studies.

Krister M. Swenson Krister Swenson is currently a postdoctoral researcher in the Laboratory for Innovation in Bioinformatics at the University of Ottawa and in the Laboratoire de combinatoire et d'informatique mathématique (LaCIM) at the Université du Québec à Montréal (UQAM). He received his Bachelors in Computer Science at University of New Hampshire and his Ph.D. in Computer Science at the Ecole Polytechnique Fédérale de Lausanne, Switzerland.



Alexandros Stamatakis Alexandros (Alexis) Stamatakis is currently the group leader of the Scientific Computing Group at the Heidelberg Institute for Theoretical Studies. Previously he was a junior group leader at the CS department of the Technical University of Munich, Germany. He received a Ph.D. in Computer Science from the Technical University of Munich in 2004. His research focus is on tools, algorithms, emerging parallel architectures, and high performance computing for evolutionary biology. Alexis is the main developer of RAxML, a code for large-scale phylogeny reconstruction under Maximum Likelihood.

Bernard M.E. Moret Bernard Moret is Professor of Computer Science, holding the chair of Bioinformatics at the EPFL, the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his PhD in 1980 from the U. of Tennessee and was on the faculty of the Dept. of Computer Science at the U. of New Mexico from 1980 until 2006, serving as chairman from 1991 till 1993. His research interests are in the area of algorithms and applications. He founded the ACM Journal of Experimental Algorithmics in 1995, serving as its editor-in-chief for 7 years. Since 2000, he has focused on the development of models and algorithms for evolutionary genomics, publishing around 80 peer-reviewed articles in the area and founding, in 2001, the annual Workshop on Algorithms in Bioinformatics (WABI).