Online Consensus and Agreement of Phylogenetic Trees.

Tanya Y. Berger-Wolf[1]

Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA.
tanyabw@cs.unm.edu

Abstract. Computational heuristics are the primary methods for reconstruction of phylogenetic trees on large datasets. Most large-scale phylogenetic analyses produce numerous trees that are equivalent for some optimization criteria. Even using the best heuristics, it takes significant amount of time to obtain optimal trees in simulation experiments. When biological data are used, the score of the optimal tree is not known. As a result, the heuristics are either run for a fixed (long) period of time, or until some measure of a lack of improvement is achieved. It is unclear, though, what is a good criterion for measuring this lack of improvement. However, often it is useful to represent the collection of best trees so far in a compact way to allow scientists to monitor the reconstruction progress. Consensus and agreement trees are common such representations. Using existing static algorithms to produce these trees increases an already lengthy computational time substantially. In this paper we present efficient online algorithms for computing strict and majority consensi and the maximum agreement subtree.

1 Introduction

Reconstruction of the evolutionary history (phylogeny) of a set of organisms is one of the fundamental problems in biology. Computational heuristics are the primary methods of phylogeny reconstruction on large datasets (for example [3, 6, 7, 16, 22, 23, 26]). Most large-scale phylogenetic analyses (including Bayesian methods) produce numerous trees that are equivalent for some optimization criteria (such as maximum parsimony or maximum likelihood). Even using the best heuristics, it takes significant amount of time to obtain optimal trees in simulation experiments. As the number of taxa increases, the running time of various heuristics increases substantially. When biological data are used, the score of the optimal tree is not known. Therefore, at any given point in running the heuristic, we do not know whether the current best score can be improved if the program is run for longer time. As a result, the heuristics are either run for a fixed (long) period of time, or "long enough" until some measure of a lack of improvement is achieved (*e.g.* tree scores do not improve). It is unclear, though, what is a good criterion for measuring this lack of improvement (one such recently proposed criterion is the small topological difference between the majority consensus of the best and the second best trees so far [28]). However, to allow scientists to monitor the reconstruction progress, often it is useful to represent the collection of the best trees so far in a compact way. Consensus and agreement trees are common such representations.

The existing static consensus and agreement methods take time polynomial in the number of input trees, multiplied by the number of taxa, to compute a single tree. Repeating this computation at every iteration of a phylogeny reconstruction heuristic, when a new tree is added to the set of best-scoring trees, is impractical. Such approach would significantly slow down an already lengthy computation. The only way to avoid this repetitive computational penalty is to update the consensus tree iteratively, using an on-line algorithm. This paper introduces on-line algorithms for computing the two most common types of consensus trees, strict and majority, and the maximum agreement subtree. The consensus and binary tree agreement algorithms are efficient, robust, and are simple to implement. To the best of our knowledge, this is the first paper explicitly addressing the issue of designing on-line algorithms for computing consensus and agreement of phylogenetic trees.

The rest of this paper is organized as follows. Section 2 provides basic definitions and a description of consensus and agreement techniques. The algorithms for on-line strict and majority consensus are given in Sections 3 and 4, respectively. The algorithm for online maximum agreement subtree is given in Section 5. Conclusions and directions of future work are discussed in Section 6.

2 Definitions

Taxon is the representation of the biological entity for which a phylogeny is desired. We denote the set of taxa by $S = \{s_1, ..., s_n\}$ and let n denote the number of taxa.

A *(rooted) phylogenetic or evolutionary tree* is a (rooted) tree with every internal (non-leaf and non-root) node of degree at least three and the leaves labeled by taxa. We denote a particular tree by T. Unless otherwise specified, a "tree" refers to a "phylogenetic tree". A tree is *binary* or *fully resolved* if every internal node has degree exactly three.

A *bipartition* is a pair of subsets of taxa defined uniquely by the deletion of an edge in a tree. We denote a bipartition by $A|B$ where $A, B \subseteq S, B = S - A$, and the set of all the bipartitions of a tree T by $C(T)$.

A collection of bipartitions is *compatible* if there exists a tree T such that the set of its bipartitions, $C(T)$, is exactly the given collection. A set of bipartitions is compatible if and only if it is pairwise compatible [10, 11]. A pair of bipartitions $A_1|B_1$ and $A_2|B_2$ is compatible if and only if at least one of the intersections $A_1 \cap A_2$, $A_1 \cap B_2$, $B_1 \cap A_2$, or $B_1 \cap B_2$ is empty [5, 19]. A pair of clades A_1 and A_2 is compatible if and only if $A_1 \cap A_2 \in \{A_1, A_2, \emptyset\}$. Determining whether a collection of m bipartitions over a set of n taxa is compatible can be done in $O(mn)$ time [15, 27].

A *consensus method* is a technique that combines a collection of trees (called a profile) on the same set of taxa into a single tree representative of the profile for some criteria. We denote the number of trees in a profile by k.

Strict consensus [20] is the most conservative of the consensus methods and produces a tree with only those bipartitions that are common to all the trees in the profile. That is, given a profile $T_1, ..., T_k$ over a set of taxa S, the strict consensus tree $SC(T_1, ..., T_k)$ is the tree uniquely defined by the set of bipartitions $C(SC) = \cap_{i=1}^{k} C(T_i)$. The strict consensus tree of k trees can be computed in time $O(kn)$ [9].

A *majority rule* [2, 18, 20, 25], consensus tree is defined by the set of bipartitions that appear in more than half of the trees in the profile. That is, given a profile $T_1, ..., T_k$ over a set of taxa S, the majority rule consensus tree $MRC(T_1, ..., T_k)$ is the tree uniquely defined by the set of bipartitions $C(MRC) = \{\pi,\ s.t.\ |\{\pi \in C(T_i), 1 \leq i \leq k\}| > k/2\}$. The majority consensus tree always exists and is unique. The majority consensus tree of k trees can be computed in time $O(kn)$ [21].

A subtree of T *induced* by a subset of taxa $R \subseteq S$ is a phylogenetic tree on the leaves labeled by R that contains only the paths in T between the leaves in R and the degree 2 nodes removed. We denote such a subtree $T|_R$.

Given a collection of k trees $T_1, T_2, ..., T_k$ with the leaves labeled by $S_1, S_2, ..., S_k$ respectively, an *agreement subtree* is a subtree induced by a set $L \subseteq S = \cap_{i=1}^{k} S_i$ such that $T_1|_L = T_2|_L = ... = T_k|_L$. *Maximum agreement subtree*, denoted $MAST(T_1, ..., T_k)$ is an agreement subtree with the maximum size of the set L [13]. There can be exponentially many (in the number of leaves) MAST for a given collection of trees [17].

A *rooted triple* is a binary subtree of a rooted phylogenetic tree induced by three leaves. If the leaves are labeled by a, b, c and a and b have a common ancestor which is not the root of the subtree, it is denoted by $ab|c$. The set of all the rooted triples of a tree T is denoted by $r(T)$. A subtree induced by three leaves in which all the leaves have the subtree root as their least common ancestor is called a *fan* and is denoted by (abc). The set of all the fans of a tree T is denoted by $f(T)$.

A *quartet* is a binary subtree of an unrooted phylogenetic tree induced by four leaves. We denote the quartet by its (unique) bipartition. The set of all the quartets of a tree T is denoted by $q(T)$. A subtree induced by four leaves with a single internal node is called a *star* and is denoted by $(abcd)$, The set of all the stars of a tree T is denoted by $s(T)$.

3 Online Strict Consensus

First, for the sake of completeness, we present the simple online algorithms for the strict consensus. The strict consensus tree contains bipartitions common to all the source trees. That is, if SC is the strict consensus tree of the set of source trees $T_1, T_2, ..., T_k$, then

$$C(SC) = \cap_{i=1}^{k} C(T_i).$$

We formulate the on-line strict consensus problem as follows.

Input: A set of evolutionary trees $T_1, T_2, ..., T_i, ..., T_k$ arriving online one at a time. All the trees are over the same set of leaves $S = \{s_1, ..., s_n\}$.

Output: At each step i we wish to maintain the strict consensus tree SC_i of the trees $T_1, ..., T_i$.

Solution: The strict consensus tree contains only those bipartitions that appear in all the source trees. Hence, given a strict consensus tree of the first $i-1$ trees, SC_{i-1}, the strict consensus tree of the first i trees is the strict consensus of SC_{i-1} and T_i. That is,

$$C(SC_i) = \cap_{j=1}^{i} C(T_j) = \cap_{j=1}^{i-1} C(T_j) \cap C(T_i) = C(SC_{i-1}) \cap C(T_i).$$

The intersection of the sets of bipartitions of SC_{i-1} and T_i can be computed in $\Theta(n)$ time.

Since it takes $O(n)$ time to process the tree T_i, this solution is time-optimal. A tree is uniquely defined by a set of its (compatible) bipartitions and can be computed in linear time [15, 27]. Thus, there is no additional space requirements beyond storing the set of $O(n)$ bipartitions of the current consensus tree.

This algorithm is essentially Day's strict consensus algorithm [9] (assuming that the number of bits used to store each bipartition is $O(\log n)$. It can also be viewed as a repetitive application of Day's algorithm, which takes $O(kn)$ time to compute the strict consensus of k trees. Thus, we have shown that the following statement is true.

Proposition 1. *The time it takes to compute the strict consensus tree SC_i with the arrival of each new tree T_i is $O(n)$ and the total time to compute the strict consensus tree of k trees online is $O(kn)$ and is optimal.*

4 Online Majority Consensus

The majority rule tree contains those bipartitions that appear in more than half of the source trees. That is, if M is the majority consensus tree of the set of source trees $T_1, T_2, ..., T_k$, then

$$A|B \in C(M) \text{ if and only if } |\{C(T_i) \text{ s.t. } A|B \in C(T_i)\}| > \frac{k}{2}.$$

The solution for the on-line majority consensus is slightly more complicated than that for the strict consensus.

Input: A set of evolutionary trees $T_1, T_2, ..., T_i, ..., T_k$ arriving online one at a time. All the trees are over the same set of leaves $S = \{s_1, ..., s_n\}$.

Output: At each step i we wish to maintain the majority consensus tree M_i of the trees $T_1, ..., T_i$.

Solution: We maintain a set of bipartitions that have appeared in all the trees seen so far. For each bipartition we keep a count of the number of trees it has appeared in up to this point. The majority tree at any point, by definition, is the collection of bipartitions that have appeared in the majority of the trees. When a new tree T_i arrives, we update the count on all the bipartitions that appear in that tree. If any of these now make the majority, we add them to the majority tree M_i. We check the counts on the bipartitions that were in the previous majority tree M_{i-1}. If any of the bipartitions now drop below the majority, then we remove them from the majority tree. Below is the formal description of the algorithm.

Algorithm Online Majority

1 $C = \emptyset$
2 **FOR** each new tree T_i **DO**
3   **FOR** each bipartition $c \in C(T_i)$ **DO**
4   **IF** $c \notin C$ **THEN**
5             $C = C \cup \{c\}$
6             $count(c) = 0$
7         $count(c) + +$
8   **IF** $count(c) > i/2$ **THEN**
9             $C(M_i) = C(M_{i-1}) \cup \{c\}$
10   **FOR** each bipartition $c \in C(M_{i-1})$ **DO**
11   **IF** $count(c) \leq i/2$ **THEN**
12             $C(M_i) = C(M_{i-1}) - \{c\}$
13   Build the tree M_i from $C(M_i)$
14   **RETURN** M_i

Line 3 is a Depth First Search (DFS) traversal of the tree T_i. The FOR loop is executed in the order of the finish times of the DFS for the nodes associated with the bipartitions. The set of bipartitions is maintained using any efficient implementation of a set with a SEARCH operation (a dictionary). The correctness and the time complexity of the algorithm are discussed below.

4.1 Correctness of the Online Majority Algorithm

Proposition 2. *Given a majority tree M_{i-1} and a new tree T_i, only the bipartitions inM_{i-1} or in T_i can be in the new majority tree M_i:*

$$C(M_i) \subseteq C(M_{i-1}) \cup C(T_i)$$

Proof. The count of any bipartition which is not in M_{i-1} was at most $(i-1)/2 < i/2$. If this bipartition is not in T_i then the number of trees it appears in has not increased with the arrival of T_i. Thus, it cannot be in M_i.

With the arrival of a new tree $T - i$ the algorithm checks the count of every bipartition in $C(M_{i-1}) \cup C(T_i)$ and retains only those whose count is greater than $i/2$. Thus, at every step i the tree T contains only the bipartitions that appear in the majority of trees, therefore, by definition, T is the majority consensus tree of the trees so far.

4.2 Running Time of the Online Majority Algorithm

Lemma 1. *The time it takes to compute the majority consensus tree M_i with the arrival of each new tree T_i is $O(n \times f(n))$, where $f(n)$ is the time of the SEARCH operation of a dictionary data structure implementation.*

Proof. We use several data structures to store the information. The underlying dictionary data structure is discussed below. The bipartitions of the current majority tree in addition are stored in a linked list (although not duplicated) with a pointer to the root (or the head of the linked list) which is null if the bipartition is not in the current tree.

As we mention above, each new arriving tree T_i is traversed using DFS and the bipartitions are processed in the order of their completion times. When each new bipartition is processed, the time it takes to check whether it is in the current set of bipartitions C (line 4) depends on the implementation of the dictionary structure. Since the bipartitions in the majority tree have a pointer to the non-null root, it takes constant additional time to check whether that bipartition is in the current majority tree as well. If the bipartition is not in the current majority tree, we add it to the linked list (in constant time, after the head) and update its root pointer. After we finish processing the T_i tree, we traverse the linked list of the majority tree, discarding the bipartitions whose count is less than the majority. Again, we use a linear time algorithm [15, 27] to build the tree in line 13. Thus, the total time for lines 7–14 is $O(n)$. Therefore, the total time *for each new tree* is $O(n \times f(n))$, where $f(n)$ is the time of the SEARCH operation for a dictionary data structure and depends on the implementation.

Theorem 1. *The time it takes to compute the majority consensus tree M_i with the arrival of each new tree T_i is at most $O(n)$ and the total time to compute the majority consensus tree of k trees online is $O(kn)$ and is optimal.*

Proof. From Lemma 1 the time needed to compute the majority consensus with the arrival of each new tree is $O(n \times f(n))$, where $f(n)$ is the time of the SEARCH operation of a dictionary data structure. A standard implementation of a dictionary data structure is a *hash table*. Given a uniform universal hash function with h keys, the expected SEARCH time to access a table with s items is at most $O(1+s/h)$. In our case, $s = |C|$ is the total number of bipartitions in k trees, which is at most kn. Making h a constant fraction of s gives a constant expected time SEARCH operation, albeit with a possibly high constant. Thus, the total running time for recomputing majority consensus using a hash table is $O(n)$ with $O(|C|)$ space requirements. The $O(n)$ running time is optimal since it takes $O(n)$ time process the input of a new tree and the $O(|C|)$ space is necessary. For a detailed implementation of a hashing table for bipartitions see Amenta *et al.* [21]. Notice that for k trees the running time of our algorithm and Amenta et al.'s algorithm is the same $O(kn)$. Thus we provide yet another optimal linear time algorithm for majority consensus, either online or off line.

5 Online Maximum Agreement Subtree

Maximum agreement subtree (MAST) represents yet another valuable piece of information about a collection of trees. It shows how much of the phylogenetic information is common to all the trees in the input. It looks at all the phylogenetic relationships, not only the bipartitions. We formulate the online version of the MAST problem as follows.

Input: A set of evolutionary trees $T_1, T_2, ..., T_i, ..., T_k$ arriving online one at a time. All the trees are over the same set of leaves $S = \{s_1, ..., s_n\}$.

Output: At each step i we wish to maintain a maximum agreement subtree $MAST_i$ of the trees $T_1, ..., T_i$.

MAST of two arbitrary trees is polynomial [14, 24] and for two binary trees Cole *et al.* [8] present an $O(n \lg n)$ algorithm. For an arbitrary collection of $k \geq 3$ trees the offline MAST problem is NP-hard [1]. However, it is fixed parameter tractable. When the degree of even one tree in the input is bounded by d, both Bryant [4] and Farach *et al.* [12] present an $O(kn^3 + n^d)$ offline algorithm.

5.1 Greedy Online MAST

The simplest online algorithm is, of course, a greedy one:

$$MAST_i = MAST(MAST_{i-1}, T_i).$$

This algorithm is polynomial and for k trees its running time is $O(kf(n))$, where $f(n)$ is the running time for the MAST of two trees. However, when $MAST_i$ is not unique we must make a decision which tree to retain. We cannot retain all the trees since there may be exponentially many of them. We show that retaining the wrong tree may have a dramatic effect on the size of subsequent agreement subtrees.

Proposition 3. *The greedy algorithm for the online MAST problem can produce trees with unbounded size ratio with respect to the optimal tree.*

Proof. Consider the following example described in Figure 1. The input trees T_1, T_2, T_3 arrive in order. There are three $MAST(T_1, T_2)$ trees, all with 3 leaves. However, if the algorithm chooses $MAST_2 = ((13)4)$ then $MAST_3$ has three leaves (and is optimal), while if $MAST_2 = ((12)4)$ or $MAST_2 = ((23)4)$ then $MAST_3$ is empty. Now consider the case when instead of the leaves there are subtrees of size t each. In this case, all the choices for $MAST_2$ are of size $3t$. However, one of them produces a $MAST_3$ of size $3t$, while the other

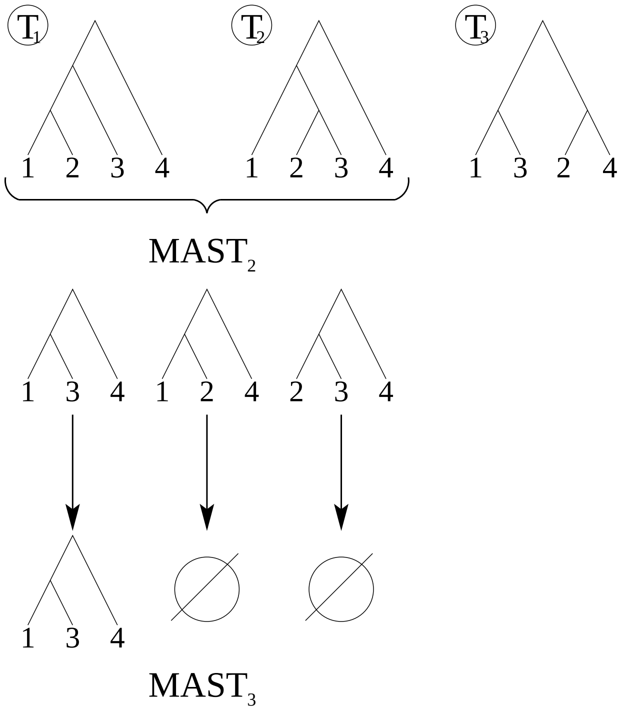

Fig. 1. An example of a choice of MAST in earlier stages of the greedy algorithm affecting the outcome of the later stages.

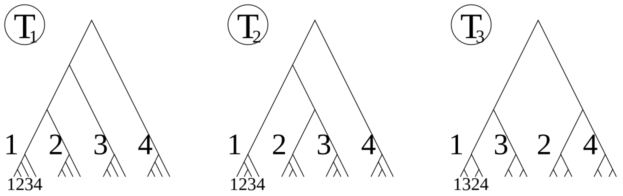

Fig. 2. The input trees T_1, T_2, T_3 with the structure of the tree recursively repeated at every leaf.

two result in a $MAST_3$ of size $2t$. We can now recursively build the subtrees 1, 2, 3, and 4, each a copy of the structure of the larger tree. Figure 2 shows the first level of the recurrence. After j levels of the recursive structure being repeated, there are 3^{3^j+1} possible $MAST_2$ trees, each with $3^{j+1}t$ leaves. However, only one of them gives rise to the $MAST_3$ with $3^{j+1}t$ leaves, while others produce $MAST_3$ with only $2^{j+1}t$ leaves. Thus the wrong choice of a $MAST_i$ by the algorithm at an earlier stage can arbitrarily badly affect the size of $MAST_k$.

5.2 Online MAST Algorithm

As we have mentioned, we cannot retain all the MAST trees at every stage of the algorithm. Recomputing MAST in a straight forward way when every new tree T_i arrives is too expensive: it increases the computational time by a factor of k. Instead, we modify a MAST algorithm to maintain a structure that allows to compute $MAST_i$ at every stage. Specifically, we will follow Bryant's algorithm [4, page 180] for computing MAST. The algorithm is a dynamic programming algorithm that relies on the following fact ([4], Lemma 6.6), which we restate here.

Lemma 2. *A tree T is an agreement subtree of $T_1, ..., T_i$ if and only if*

$$r(T) \subseteq R_i = \bigcap_{j=1}^{i} r(T_j) \text{ and } f(T) \subseteq F_i = \bigcap_{j=1}^{i} f(T_j).$$

Note that using the property of intersection for any collection of sets $A_1, ..., A_i$

$$\bigcap_{j=1}^{i} A_j = \bigcap_{j=1}^{i-1} A_j \cap A_i,$$

it is easy to maintain the sets R_i and F_i online greedily.

For the simplicity of the discussion we also restate Bryant's algorithm here.

Algorithm BRYANTMAST(a, b)

1 **IF** $a = b$ **THEN RETURN** 1
2 Construct:
  $A \leftarrow \{x : ax|b \in R\} \cup \{a\}$
  $B \leftarrow \{x : by|a \in R\} \cup \{b\}$
  $C \leftarrow \{z : (azb) \in F\}$
3 Choose $x^* \in A$ that maximizes $MAST(a, x^*)$
4 Choose $y^* \in B$ that maximizes $MAST(b, y^*)$
5 **FOR** (each $z \in C$) **DO**
  Choose $z^* \in \{z' \in C : zz'|a \in R\} \cup \{z\}$ that maximizes $MAST(z, z^*)$
6 Construct a weighted graph $G = (V, E)$
  $V = C$, $(v, w) \in E \Leftrightarrow (avw) \in F$, $w(v) = MAST(v, v^*)$
7 Choose maximum weight clique Q in G
8 **RETURN** $MAST(a, x^*) + MAST(b, y^*) + \sum_{z \in Q} MAST(z, z^*)$

The algorithm fills in an $n \times n$ matrix of all leaf pairs with the sizes of the respective MASTs. The actual MAST can be reconstructed from this matrix using the standard dynamic programming trace of computation. The running time of Bryant's algorithm is $O(kn^3 + n^d)$, where all trees have the maximum degree at least d. When even one tree is binary (a common case when the input trees are a result of a tree reconstruction heuristic), then there is no set F and the algorithm reduces to steps 1, 2 (sets A and B only), 3, 4, and 8. The running time in this case is $O(kn^3)$, which is the time it takes to construct the set R.

We are now ready to state and analyze the online MAST algorithm.

Theorem 2. *Let the input trees $T_1, ..., T_i, ..., T_k$ have the corresponding maximum degrees $d_1, ..., d_i, ..., d_k$ and let $min_i = \min_{1 \le i \le k} d_i$. The MAST of these k trees can be maintained online in $O(kn^3 + \sum_{i=1}^{k} n^{min_i})$ time.*

Before we prove Theorem 2, we state an important corollary.

Corollary 1. *If the tth input tree is the first binary tree in the input then the MAST of these k trees can be maintained online in $O(kn^3 + \sum_{i=1}^{t} n^{min_i})$. Specifically, if the first tree of the input is binary, the MAST of k trees can be maintained online in $O(kn^3)$.*

Proof. To maintain $MAST_i$ online, we use Bryant's algorithm stated earlier. When a new tree T_i with the maximum degree d_i arrives we update the sets R_i and F_i of the common triples and fans using the intersection property:

$$R \leftarrow R \cap r(T_i),\ F \leftarrow F \cap f(T_i)$$

Note, that the vertices of T_i of degree greater than min_{i-1} do not contribute fans to the intersection and hence the set F is limited by the smallest d_i so far. We then update the MAST matrix using Bryant's algorithm. To maintain the structure needed to compute the MAST, we need to maintain the sets A, B, and C *for every pair of vertices*. Let us examine what can happen to those sets as a new tree T_i arrives. As we are computing the new R, some triples may be eliminated from the set. This would possibly eliminate elements from the sets A and B. If this happens and those elements were the elements x^* and y^* (lines 3, 4) that maximize the corresponding MASTs, new maxima need to be found. However, recalculating the sets A and B and choosing new maxima takes $O(n)$. Thus, redoing the procedure for each for each vertex pair would take $O(n^3)$ time and, hence, would not increase the (asymptotic) running time over k input trees.

When the minimum maximum degree in first i trees is $min_i > 2$, the most computationally expensive step of the offline algorithm is line 7. The lines 5 and 6 still take $O(n)$ time for each pair, $O(n^3)$ total for a new tree. However, any time a fan is deleted from the set F, both a vertex and an edge can potentially be removed from the graph G *for every vertex pair*. Moreover, the vertex weights need to be recomputed. Thus, we need to recompute the maximum weight clique S, possibly from scratch, for every pair, for every new tree. The maximum degree of the graph G is no greater than the maximum size of a fan in the set F, that is at most min_i at the time $MAST_i$ is calculated. Thus, the line 7 adds $O(n^{min_i-2})$ running time to the algorithm per vertex pair, $O(n^{min_i})$ total for each new tree T_i.

Thus, the total time to compute $MAST_i$ online using Bryant's algorithm is $O(n^3 + n^{min_i})$ adding up to the total of $O(kn^3 + \sum_{i=1}^{k} n^{min_i})$ for k trees.

The arrival of a binary tree T_t sets $F = \emptyset$, therefore all the calculations related to the set C from then on are eliminated and only the lines 1,2,3,4, and 8 are executed. Thus, the computational time from that point on for each new tree is $O(n^3)$. Therefore, the total computational time for k trees in this case is $O(kn^3 + \sum_{i=1}^{t} n^{min_i})$. When $t = 1$ this, of course, becomes $O(kn^3)$.

6 Conclusions

Phylogeny reconstruction heuristics on biological data cannot recognize when an optimal or a "true" tree is produced. Thus, they need to use some criteria for termination. Initial experiments show [28] that the lack of difference between the consensus of the trees with the top score and the trees with the second best score may be a good criterion. To use these criteria, we need to have algorithms that maintain consensus of a sequence of trees on-line, as the new trees are generated by a heuristic. Another or additional option is to rely on expert knowledge to determine when the reconstruction process has been run sufficiently long. For this, scientists need to monitor the progress of the reconstruction presented in a meaningful and compact way. Consensus and agreement trees are common such representations. We have proposed and analyzed algorithms for the on-line computation of strict and majority consensus trees and maximum agreement subtree. We have shown that the on-line strict and majority consensus algorithms are time and space-optimal. Thus these easy to implement algorithms can be used to maintain stopping criteria or monitor the reconstruction progress without substantially increasing the overall running time of the heuristic search. Finally, we have also proposed an online maximum agreement subtree algorithm which does not increase the overall computational time with respect to the best known offline algorithms.

Clearly, we need experimental results to verify that the algorithms are efficient in practice. The ultimate goal is to develop a suite of tools that allow compact and efficient online representation of the various aspects of the collections of trees arising in the phylogenetic reconstruction process.

7 Acknowledgments

This work is supported by the National Science Foundation Postdoctoral Fellowship grant EIA 02-03584. The author is deeply grateful to Tandy Warnow for suggesting the problem and for many insights.

References

1. A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees - metrics and efficient algorithms. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 758–769, 1994.

2. J. P. Barthélemy and F. R. McMorris. The median procedure for n-trees. *Journal of Classification*, 3:329–334, 1986.
3. M. L. Berbee. The phylogeny of plant and animal pathogens in the Ascomycota. *Physiological and Molecular Plant Pathology*, 2001.
4. D. Bryant. *Building trees, hunting for trees, and comparing trees: Theory and methods in phylogenetic analysis*. PhD thesis, University of Canterbury, 1997.
5. P. Buneman. The recovery of trees from measures of dissimilarity. In F.R.Hodson, D.G.Kendall, and P.Tautu, editors, *Mathematics in the Archeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.
6. R. M. Bush, W. M. Fitch, C. A. Bender, and N. J. Co. Positive selection on the H3 hemagglutinin gene of human influenza virus A. *Molecular Biology and Evolution*, 16:1457–1465, 1999.
7. M. W. Chase, D. E. Soltis, R. G. Olmstead, D. Morgan, D. H. Les, B. D. Mishler, M. R. Duvall, R. A. Price, H. G. Hills, Y. L. Qiu, K. A. Kron, J. H. Rettig, E. Conti, J. D. Palmer, J. R. Manhart, K. J. Sytsma, H. J. Michaels, W. J. Kress, K. G. Karol, W. D. Clark, M. Hedren, B. S. Gaut, R. K. Jansen, K. J. Kim, C. F. Wimpee, J. F. Smith, G. R. Furnier, S. H. Strauss, Q. Y. Xiang, G. M. Plunkett, P. S. Soltis, S. M. Swensen, S. E. Williams, P. A. Gadek, C. J. Quinn, L. E. Eguiarte, E. Golenberg, G. H. Learn, Jr., S. W. Graham, S. C. H. Barrett, S. Dayanandan, and V. A. Albert. Phylogenetics of seed plants: an analysis of nucleotide sequences from the plastid gene rbcL. *Annals of the Missouri Botanical Garden*, 80:528–580, 1993.
8. R. Cole, M. Farach-Colton, R. Hariharan, T. M. Przytycka, and M. Thorup. An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM Journal of Computing*, 30(5):1385–1404, 2000.
9. W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2:7–28, 1985.
10. G. F. Estabrook, Jr. C. S. Johnson, and F. R. McMorris. An algebraic analysis of cladistic characters. *Discrete Mathematics*, 16:141–147, 1976.
11. G. F. Estabrook and F. R. McMorris. When is one estimate of evolutionary history a refinement of another? *Mathematical Biology*, 10:367–373, 1980.
12. M. Farach, T. Przytycka, and M. Thorup. On the agreement of many trees. *Information Processing Letters*, 55:297–301, 1995.
13. C. R Finden and A. D. Gordon. Obtaining common pruned trees. *Journal of Classification*, 2:255–276, 1985.
14. W Goddard, E. Kubicka, G. Kubicki, and F. R. McMorris. The agreement metric for labelled binary trees. *Mathematical Biosciences*, 123:215–226, 1994.
15. D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:12–28, 1991.
16. M. Källersjö, J. S. Farris, M. W. Chase, B. Bremer, M. F. Fay, C. J. Humphries, G. Pedersen, O. Seberg, and K. Bremer. Simultaneous parsimony jackknife analysis of 2538 rbcl DNA sequences reveals support for major clades of green plants, land plants, seed plants and flowering plants. *Plant Systematics and Evolution*, 213:259–287, 1998.
17. E. Kubicka, G. Kubicki, and F. R. McMorris. On agreement subtrees of two binary trees. *Congressus Numerantium*, 88:217–224, 1992.
18. T. Margush and F. R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43(2):239–244, 1981.
19. F. R. McMorris. On the compatibility of binary qualitive taxonomic characters. *Bulletin of Mathematical Biology*, 39:133–138, 1977.
20. F. R. McMorris, D. B. Meronik, and D. A. Neumann. A view of some consensus methods for trees. In J. Felsenstein, editor, *Numerical Taxonomy*, pages 122–125. Springer-Verlag, 1983.
21. Katherine St. John Nina Amenta, Frederick Clarke. A linear-time majority tree algorithm. In Gary Benson and Roderic D. M. Page, editors, *Algorithms in Bioinformatics, Third International Workshop, WABI 2003, Budapest, Hungary, September 15-20, 2003, Proceedings*, volume 2812 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2003.
22. V. Savolainen, M. W. Chase, S. B. Hoot, C. M. Morton, D. E. Soltis, C. Bayer, M. F. Fay, A. Y. De Bruijn, S. Sullivan, and Y. L. Qiu. Phylogenetics of flowering plants based on combined analysis of plastid atpB and rbcL gene sequences. *Systematic Biology*, 49:306–362, 2000.
23. P. S. Soltis, D. E. Soltis, and M. W. Chase. Angiosperm phylogeny inferred from multiple genes as a tool for comparative biology. *Nature*, 402:402–404, 1999.
24. M. Steel and T. Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Information Processing Letters*, 48(2):77–82, 1993.
25. D. L. Swofford. When are phylogeny estimates from molecular and morphological data incongruent? In M. M. Miyamoto and J. Cracraft, editors, *Phylogenetic Analysis of DNA Sequences*, pages 295–333. Oxford University Press, 1996.

26. Y. Van de Peer and R. De Wachter. Evolutionary relationships among the eukaryotic crown taxa taking into account site-to-site rate variation in 18S rRNA. *Journal of molecular evolution*, 45:619–630, 1997.
27. T. J. Warnow. Three compatibility and inferring evolutionary history. *Journal of Algorithms*, 16:388–407, 1991.
28. T. L. Williams, T. Y. Berger-Wolf, B. M. E. Moret, U. Roshan, and T. J. Warnow. The relationship between maximum parsimony scores and phylogenetic tree topologies. Technical Report TR-CS-2004-04, University of New Mexico, 2004.