Properties of Compatibility and Consensus Sets of Phylogenetic Trees

Tanya Y. Berger-Wolf

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

Abstract. One of the fundamental problems in phylogeny reconstruction is combining a set of trees into one “representative” tree. There are exist numerous methods for that purpose that combine trees over the same taxa (consensus) or different taxa (supertree) sets. However, many limitations and shortcomings of these methods have been pointed out. In this paper, we state various desirable properties of “representative” trees and the impact of considering sets of trees to represent the input. We analyze the sets of compatibility trees and several consensus sets and show which of the properties are satisfied. Specifically, one of the desired properties is the output-polynomial enumeration of the set itself.

1 Introduction

One of the fundamental problems in biology is reconstructing the evolutionary history of a set of organisms (or taxa). Relationships among the taxa are modeled as a phylogenetic tree, which was constructed from present-day organisms. Often, in the course of phylogenetic reconstruction, scientists are faced with numerous (sometimes thousands) of competing phylogenetic trees as possible answers. This may happen either as a result of a computational phylogenetic reconstruction heuristic or different types of data being used for tree reconstruction (e.g. DNA vs. genomic or morphological vs. genetic data) or different taxa representatives being chosen for analysis. Such a result is quite unsatisfying since the objective is to obtain a single “representative” tree that best describes the relationships among the organisms of interest. Therefore, a consensus method is applied to combine all of the trees into a final tree. The number of proposed consensus tree methods has grown substantially since the first introduction of the technique by Edward N. Adams III in 1972 [1]. See the two recent surveys [8, 11] of the existing supertree and consensus methods for more details.

Recently Steel et al. [31] have published a paper on the limitations of consensus and supertree methods, painting a pessimistic picture. The authors list a set of desired properties of consensus and supertree methods and show that no such methods can satisfy all of them. One of the fundamental assumptions used is the fact that any consensus or a supertree method has to return one tree. While desirable, this is not a necessary requirement in phylogenetic analysis and reconstruction. Rather than using automated computational methods to process a large collection of trees and combine them into a single tree, it may be preferable to represent this collection by a more compact and meaningful set of trees, leaving the task of deciding on the final tree up to the biologists. In fact, the consensus and supertree approaches have been criticized for losing valuable phylogenetic information [5, 14, 22, 28]. In this paper we explore what happens when presented with a set of possible phylogenetic trees we return a representative consensus or supertree set rather than a single tree.

The formal mathematical study of biological consensus takes its roots in the theory of group choice formalized by Kenneth Arrow [4]. There is an extensive literature on the formal treatment of the phylogenetic consensus and the supertree methods [19, 7]. The standard approaches in the field are either to list a set of desired mathematical properties and design a consensus method that satisfies them or to determine a set of properties satisfied by a given consensus method [25]. Here we combine the two approaches as we use the augmented list of properties proposed in [31] and test which of those are satisfied by some common consensus sets. We focus on a few consensus methods (the most popular strict and majority rule consensus and the computational optimization asymmetric median tree method) and their relaxations that allow multiple trees as an answer. While going down the list of properties, we also give efficient set enumeration algorithms where necessary and possible.
2 Preliminaries

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 *clade* is a subset of taxa defined uniquely by a subtree rooted at a node of a rooted tree. We denote the set of all the clades of a tree \( T \) by \( C(T) \). For a tree defined over the set of leaves \( S \) a clade \( A \) is also a bipartition \( A|S-A \).

A tree \( T_2 \) *resolves or refines* a tree \( T_1 \) on the same set of taxa if all the bipartitions (clades) of \( T_1 \) are also bipartitions (clades) of \( T_2 \). That is, \( C(T_1) \subseteq C(T_2) \).

A tree \( T_1 \) *displays* a tree \( T_2 \) if when the set of taxa of \( T_1 \) is limited to that of \( T_2 \) and nodes of degree 2 are eliminated, the tree \( T_1 \) equals to \( T_2 \). That is, if \( S_1 \) is the set of taxa of \( T_1 \) and \( S_2 \) is the set of taxa of \( T_2 \), then the set of bipartitions \( C(T_1) \) limited to \( S_2 \) equals to \( C(T_2) \).

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 [15, 16]. 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 [12, 24]. 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 [18, 34]. Alternatively, a collection of clades or trees is compatible if there exists a tree that displays each tree in the collection. This tree is called *compatibility (super)tree* or *parent tree* for the collection.

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

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* [26] 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 \( O(kn) \) [13].

A *majority rule* [6, 23, 26, 32], 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 \( O(kn) \) [27].

A *greedy* resolution algorithm of the majority consensus tree sorts the bipartitions by the frequency of their appearance in the input trees. The algorithm then proceeds by adding each bipartition in order to the tree if the bipartition is compatible with the bipartitions added so far. Note that all the bipartitions with the frequency greater than \( k/2 \) can always be added to the consensus tree, since those form the uniquely defined majority consensus tree.

The *Asymmetric Median Tree* (AMT) [29] is a tree defined by the maximum weight collection of compatible bipartitions, where the weight of collection is the sum of the input frequencies of the bipartitions in the collection. Computing the AMT of an arbitrary collection of \( k \) trees is NP-hard (it is equivalent to the Maximum Weighted Independent Set problem). However, an AMT of two trees can be computed in \( O(n^{2.5}) \).
3 Properties

First, we state the list of desirable properties that any consensus method should satisfy. The properties (P1)–(P7) are generalizations of the corresponding properties from [31]. Steel et al. also provides a more detailed motivation and description of the properties. We add the properties (P8) and (P9) which are the analogs of the properties (P6) and (P7) for bipartitions (clades).

(P1): The method can be applied to any unordered set of input trees.

(P2): If we re-name all the species, and then apply the method to the new input trees, the output set of trees is simply the old output set, but with the species renamed accordingly.

(P3)a: If there exists at least one parent tree for the given collection of input trees, then the output set contains at least one of those parent trees (that is, the output trees display the input trees whenever the latter are compatible).

b: If there exists at least one parent tree for the given collection of input trees, then the output set contains all of those parent trees (that is, the output trees display the input trees whenever the latter are compatible).

(P4)a: Each leaf (taxon) that occurs in at least one input tree occurs in at least one output tree.

b: Each leaf (taxon) that occurs in at least one input tree occurs in every output tree.

(P5): The time required to compute the output trees grows polynomially with the size of the input and the output.

(P5'): The output set can be enumerated by a polynomial delay algorithm.

(P6): If all input trees display (IJ)(KL) or (IJ)K, then all the output trees display (IJ)(KL) or (IJ)K.

(P7)a: If at least one input tree displays (IJ)K and no input tree displays (IK)J or (JK)I, then at least one output tree displays (IJ)K. Alternatively, if at least one input tree displays (IJ)(KL) and no input tree contradicts it, then at least one output tree displays (IJ)(KL).

b: If at least one input tree displays (IJ)K and no input tree displays (IK)J or (JK)I, then all the output trees display (IJ)K. Alternatively, if at least one input tree displays (IJ)(KL) and no input tree contradicts it, then all output trees display (IJ)(KL).

(P8): If all input trees have bipartition (clade) \( \pi \) then all output trees have \( \pi \).

(P9)a: If at least one input tree has bipartition (clade) \( \pi \) and no input tree contradicts it then at least one output tree has \( \pi \).

b: If at least one input tree has bipartition (clade) \( \pi \) and no input tree contradicts it then all output trees have \( \pi \).

Note that the (a) versions of the properties are weaker than the (b) versions. Hence if a collection of output trees satisfies the (b) version it necessarily satisfies the (a) version of a property.

Remark 1. Any output set of trees that is defined uniquely over a set of input trees, by definition, is independent of the order in which the input trees appear and, hence, satisfies (P1).

Lemma 1. A consensus or a supertree method that depends only on the number of input trees in which a given clade or bipartition appears, satisfies (P2).

Proof. If a clade appeared in a particular tree before relabeling then the renamed version of the clade will appear in that tree after relabeling. Conversely, if a clade did not appear in a tree then the renamed version of it will not appear in that tree. Thus the total number of trees in which a clade appears in remains the same. Therefore a method that relies only on the tree count for each clade makes the same decisions about it before and after relabeling.

Remark 2. If the input trees are not compatible then property (P3) is satisfied trivially.

Remark 3. Since many of the output sets under the consideration may contain the number of trees exponential in the number of leaves and input sets, we cannot use the input-polynomial time complexity restriction from [31]. Instead, we use the the output-polynomial time complexity measure in (P5). The property (P5') is a stronger version of the property (P5) since it specifies in what manner should the computational time be polynomial in the output. That is, it takes time polynomial in the input to produce the first output instance, and the delay between any consecutive output instances is polynomial in the input size. Thus, any set that satisfies (P5') clearly satisfies (P5).
Remark 4. If a tree \( T_1 \) displays either \((IJ)K\) or \((IJ)(KL)\), and \( T_2 \) is a refinement of \( T_1 \), then \( T_2 \) displays either \((IJ)K\) or \((IJ)(KL)\). Thus if \( T_1 \) satisfies (P6) or (P7) with regard to a triple or a quartet then a set that contains \( T_1 \) and any of its resolutions still satisfies (P6) or (P7). Similarly, if a tree \( T_1 \) contains a bipartition \( \pi \) then, by definition, all of its refinements contain \( \pi \). Hence, if \( T_1 \) satisfies (P8) or (P9) then a set that contains \( T_1 \) and any of its refinements satisfies (P8) or (P9).

Remark 5. If a tree has a bipartition \( \pi = A|B \) such that \( I, J \in A \) and \( K, L \in B \) then it displays the quartet \((IJ)(KL)\). If the trees are over the same leaf set \( S \) then if a tree has a clade \( A \subseteq S \), such that \( I, J \in A \) and \( K \in S - A \), then the tree displays the triple \((IJ)K\). Thus, the properties (P8) and (P9) for bipartitions are special cases of the properties (P6) and (P7) for quartets. However, this does not hold for clades and triples if the trees are not over the same leaf set.

Remark 6. If all trees display a bipartition \( \pi = A|B \) then all these trees must have the same leaf set \( S = A \cup B \).

4 Compatible trees

We first discuss the case when the input trees are compatible. As has been shown by Semple and Steel [30], a modification of Aho et al. method [3] results in a supertree method that satisfies properties (P1)–(P5). We now show that the set of all the compatibility trees satisfies the remaining properties with a caveat for the time complexity.

Theorem 1. If the input trees are compatible then the set of all the compatibility (super)trees satisfies properties (P1)–(P4) and (P6)–(P8). In addition, if the compatibility tree is unique then it satisfies property (P5) and if the input trees are over the same leaf set, then the set of all the compatibility consensus trees satisfies properties (P5′) and (P9).

Proof. Let \( T = \{T_1, \ldots, T_k\} \) be the set of input trees and \( S = \{S_1, \ldots, S_m\} \) be the set of output trees as specified in the statement of the theorem. First we will show that the set of compatibility supertrees (we will call it “compatibility set”) for compatible input trees satisfies the required properties.

The set of all the compatibility supertrees for a collection of compatible trees is uniquely defined. Hence, by Remark 1, it satisfies (P1).

The compatibility set is the set of all trees that display each input tree (when the latter are compatible). The compatibility relationship is independent of the labeling of the taxa. Therefore, if the input trees are compatible, they remain so after relabeling. Moreover, if a compatibility tree displays all the input trees, then the appropriately relabeled compatibility tree displays the input trees after renaming of taxa. Since the taxa renaming procedure is a bijection, the set of compatibility trees remains the same cardinality and hence, is the appropriately relabeled version of the output set of trees before renaming of taxa. Hence (P2) is satisfied.

By definition, the compatibility set contains every parent tree that displays the input trees, hence it satisfies (P3).

Again, by the definition of the compatibility trees, and since the input trees are compatible, every tree in the output set contains each leaf that occurs in at least one input tree. Hence (P4) is satisfied.

The number of compatibility trees can be exponential in the number of leaves [9], even when all of them are binary. Böcker et al. [10] give an algorithm that returns the compatibility supertree if it is unique in polynomial time. When the trees are all over the same set of taxa, the problem becomes much easier, the compatibility tree is unique, and Warnow’s algorithm [34] returns the compatibility tree in linear time. Thus, in these cases property (P5) is satisfied. All compatibility trees can be enumerated using a polynomial delay algorithm by encoding the bipartitions or clades into string characters and using character compatibility tree enumeration algorithm of Kannan and Warnow [21]. Thus, the set of the compatibility trees in the consensus setting satisfies property (P5′).

Compatibility trees, by definition, display each of the input trees (if they are compatible). Therefore, if each input tree displays a particular phylogenetic relationship, either quartet or triple, then each compatibility tree displays it as well. Hence property (P6) is satisfied.
Again, since the input trees are compatible and the compatibility trees display each of the input trees, and one of the input trees displays a relationship \((IJ)K\) or \((IJ)(KL)\), then all of the output trees will display \((IJ)K\) or \((IJ)(KL)\). Hence, \((P7)b\) is satisfied.

By remarks 6 and 5, compatibility trees over the same leaf set satisfy property \((P8)\). The only case we need to consider is that of rooted trees \(T_1, \ldots, T_k\) over different leaf sets \(S_1, \ldots, S_k\). Suppose that all the input trees contain a particular clade \(A \subseteq \bigcap_i S_i\). There exists a tree \(T_i\) such that \(S_i - A \neq \emptyset\) (otherwise all the tree have the same leaf set \(A\)). Suppose there exists a compatibility tree \(T\) that does not have the clade \(A\). By property \((P4)b\), every compatibility tree contains all the leaves \(\cup_i S_i\). Thus, it must contain a clade that contains \(A\) (it may be the trivial root clade). Let \(B\) be the smallest such clade, defined by the least common ancestor of \(A\) in \(T\). Let \(a\) denote the internal node which is the least common ancestor of \(A\) in \(T\) (and the root of the clade \(B\)). There must be at least two leaves in \(I, J \in A\) such that \(a\) is the least common ancestor of \(I\) and \(J\). For every leaf \(K \in B - A\) there exists an input tree that displays the triple \((IJ)K\) (since every input tree contains \(A\)). However, the least common ancestor of \(I\) and \(K\) in \(T\) is either \(a\) or a descendent of \(a\), therefore \(T\) does not display \((IJ)K\). This is a contradiction to the definition of the compatibility tree. Thus, the set of compatibility trees satisfies property \((P8)\).

Again, by the remark 5, compatibility trees over the same leaf set satisfy property \((P9)b\). However, if the input trees are defined over different leaf sets, then even \((P9)a\) does not necessarily hold. Consider the following set of input trees:

\[
T_1 = (IJ)(KL), \quad T_2 = (IX)(KY), \quad T_3 = (XJ)(YL)
\]

While the input trees are compatible, no compatibility supertree has any of the partitions of the input trees. This example holds for the rooted trees as well with the addition of a root at the one internal edge.

Thus, all the properties \((P1)–(P9)\) are satisfied in case of compatible input trees in the consensus setting and properties \((P1)–(P6), (P6)–(P8)\) are satisfied for compatible trees in the supertree setting and \((P5)\) is satisfied if the compatibility supertree is unique.

### 5 Consensus of incompatible trees

We now turn our attention to the incompatible input trees. In this paper, we consider the case of the consensus setting, that is, all the input trees have the same leaf set. We consider the following output sets

\begin{enumerate}
  \item[(S1):] Strict consensus tree
  \item[(S2):] All refinements of the strict consensus
  \item[(S3):] Majority consensus tree
  \item[(S4):] All refinements of the majority consensus
  \item[(S5):] All greedy refinements of the majority consensus
  \item[(S6):] All asymmetric median trees
  \item[(S7):] The set of input trees
\end{enumerate}

Before we set out to prove the main results, we point out that Steel et al. [31] mention that strict and majority consensus trees (sets (S1) and (S3)) satisfy properties \((P1), (P2), (P4),\) and \((P5)\) and can be modified to satisfy property \((P3)\). It has also been shown that Adams consensus tree satisfies properties \((P1)–(P6)\) for rooted trees [2].

**Theorem 2.** All the output sets \((S1)–(S7)\) satisfy properties \((P1)–(P4)\) and \((P8)\).

**Proof.** All the output sets of trees are uniquely defined for a set of input trees and hence, by Remark 1, satisfy property \((P1)\).

Strict, majority, greedy, and asymmetric median consensus trees depend only on the number of trees a particular clade or bipartition appear and hence, by Lemma 1, satisfy \((P2)\). The set of all possible resolutions of a given tree is independent of the leaf labels and, hence, the sets \((S2)\) and \((S4)\) satisfy \((P2)\). The set of input trees as the output set is, clearly, invariant under leaf renaming. Thus, all the output sets satisfy property \((P2)\).
For an incompatible set of input trees the property \((P3)\) is trivially satisfied, by Remark 2.

Since we are discussing the full consensus methods, by definition, each leaf appears in all the input trees and all the output trees. Hence, property \((P4)b\) is satisfied.

If all the input trees have a particular bipartition \(\pi\) then, by definition, strict \((S1)\) and majority \((S3)\) consensus trees contain \(\pi\). By remark 4, all the trees in \((S2), (S4),\) and \((S5)\) contain \(\pi\) as well. Trivially, all the trees in \((S7)\) contain \(\pi\). Since all the input trees contain \(\pi\), it is compatible with every bipartition in the set of all the bipartitions of the input trees and, clearly, with any subset of those bipartitions. Therefore, by definition, it is contained in every asymmetric median tree. Thus all the sets \((S1)-(S7)\) satisfy property \((P8)\).

**Theorem 3.** The sets \((S1), (S3), (S5),\) and \((S7)\) satisfy property \((P5)\). The sets \((S2)\) and \((S4)\) satisfy property \((P5')\).

**Proof.** Day’s strict consensus algorithm [13] runs in \(O(kn)\) time for \(k\) trees on \(n\) leaves. This is also the running time of the best known majority rule consensus algorithm [27]. Clearly, the input set can be listed in trivially in linear time as well. Thus, the sets \((S1), (S3),\) and \((S7)\) satisfy property \((P5)\).

There are \((2l - 5)!!\) binary (fully resolved) unrooted phylogenetic trees on \(l\) leaves and \((2l - 3)!!\) rooted binary trees. Foulds and Robinson [17] essentially give a polynomial delay algorithm for enumerating binary trees on \(l\) leaves. Given a tree which is not fully resolved, we can enumerate all its resolutions by recursively considering each node of degree \(l > 3\) as the set of all binary trees on \(l\) leaves. Thus, the sets \((S2)\) and \((S4)\) can be enumerated with polynomial delay and, hence, satisfy property \((P5')\).

Recall that a greedy resolution of the majority consensus tree is obtained by iteratively adding bipartitions that are compatible with the accumulated consensus tree so far. The bipartitions are added in the descending order of the frequency of their appearance in the input trees. Multiple greedy resolutions of the majority consensus tree arise only when there are bipartitions compatible with the current consensus tree that appear in the same number of input trees. Any permutation of the order of the addition of the same frequency bipartitions may potentially give rise to a different greedy tree. Below is an algorithm that enumerates all greedy resolutions of the majority consensus tree given the sorted set of input bipartitions with their frequency \(P = \{(\pi_i, f_i)\}\) (represented by a linked list). We assume that the starting compatible set of bipartitions \(M\) are those with frequency greater than \(k/2\), i.e. the ones that form the majority consensus tree.

\[
\text{Algorithm } \text{EnumerateGreedy}(M, P)
\]

1. IF \(P = \emptyset\) THEN RETURN tree \(T\) defined by \(M\)
2. Let \(F \subseteq P\) be a set of bipartitions with the same highest frequency
3. Construct a graph \(G = (V, E)\)
   \(V = F \cup \{M\}\) (one supernode for \(M\))
   \((u, v) \in E\) if \(u\) and \(v\) are compatible
   \((u, M) \in E\) if \(u\) is compatible with each bipartition in \(M\)
4. FOR EACH maximal clique \(Q\) in \(G\) that contains \(M\) DO
5. RETURN \(\text{EnumerateGreedy}(M \cup Q, P - F)\)

First, we remark that the algorithm always returns a valid tree. It terminates when the set of remaining unadded bipartitions is empty. Starting with the set of bipartitions of the majority consensus tree (which is by definition compatible and defines a valid tree), any bipartitions added to \(M\) are always pairwise compatible and are compatible with every bipartitions in \(M\). Thus, the set \(M\) is always the set of pairwise compatible bipartitions and, by definition, defines a valid tree. Gusfield’s or Warnow’s algorithms [18, 34] can be then used to build a tree defined by those bipartitions in \(O(kn)\) time.

We now need to verify the following four facts about the \(\text{EnumerateGreedy}\) algorithm:

1. The algorithm returns all possible greedy resolutions of the majority consensus tree.
2. Each solution returned by the algorithm is a greedy resolution of the majority consensus tree.
3. The algorithm returns each tree only once.
4. The time it takes to output all the greedy resolutions is polynomial in $gn$ where $g$ is the number of greedy consensus trees.

Before we prove these statements, we prove the following subsidiary lemma.

**Lemma 2.** A collection of bipartitions with the same input frequency forms a maximal clique that contains $M$ in the EnumerateGreedy algorithm graph $G$ if and only if it is the set of all the bipartitions of the same input frequency in some greedy resolution of a majority consensus tree.

**Proof.** Assume, by induction, that in the process of greedy resolution of the majority consensus tree all the bipartitions in $M$ have been added and only those. This holds true for the initial $M$, which are the bipartitions of the majority consensus tree itself. All the bipartitions that have not been added have an input frequency lower than those in $M$. Let $Q$ be a collection of bipartitions with frequency $f$ that form a maximal clique that contains $M$ in the graph $G$. By definition of $G$, this means that all bipartitions in $Q$ are pairwise compatible and each one is compatible with $M$. Recall that a collection of bipartitions is compatible if and only if it is pairwise compatible [15, 16]. Thus, each of the bipartitions in $Q$ can be added to the tree defined by $M$ greedily and for any other bipartition with frequency $f$ it is incompatible with some bipartition already added. Hence only the bipartitions in $Q$ are the ones of frequency $f$ added to that greedy resolution tree. Conversely, any set of same frequency $f$ bipartitions that can be greedily added to $M$ must be pairwise compatible and compatible with $M$. By definition, this set forms a clique in $G$ that contains $M$. If the clique is not maximal, there must be another bipartition of frequency $f$ that is compatible with every bipartition in the set and with $M$. But then it could be greedily added to the tree, which it was not. Thus, by induction, a set of bipartitions of frequency $f$ forms a maximal clique containing $M$ in $G$ if and only if it is the set of all the bipartitions of frequency $f$ in some greedy resolution of a majority consensus tree.

We now continue with the proof of Theorem 3. Specifically, we show that the algorithm EnumerateGreedy fulfills the four requirements stated above.

**All greedy solutions are enumerated by the algorithm.** Suppose there exists a greedy resolution $T_g$ of the majority consensus tree which is not returned by the EnumerateGreedy algorithm. Consider the bipartitions of $T_g$ sorted by the frequency of their appearance in the input trees. Let $f$ be the largest frequency such that the set of bipartitions of frequency $f$ in $T_g$ is different from any tree returned by the algorithm. By Lemma 2 this set forms some maximal clique that contains $M$ in $G$. But then it would have been added to a tree returned by the algorithm.

**Each algorithm tree is a greedy tree.** Suppose to the contrary that the algorithm returns a tree $T_h$ which cannot be obtained by any greedy resolution of the majority consensus tree. Similarly to the previous case, consider all the bipartitions of $T_h$ sorted by the frequency of their appearance in the input trees. Let $f$ be the largest frequency such that the set of bipartitions of frequency $f$ in $T_h$ is different from any greedy tree. This set forms a maximal clique in $G$ that contains $M$. By Lemma 2 this is the set of bipartitions of frequency $f$ in some greedy tree. This is a contradiction.

**The algorithm returns each tree once.** For any given input frequency $f$ the set of bipartitions of frequency $f$ in any tree returned by the algorithm forms a maximal clique that contains $M$, the set of bipartitions with higher frequency. Once that set is added to $M$, it is never considered by the algorithm again, since the entire set of bipartitions of frequency $f$ is removed from consideration. Thus, each possible set of same frequency bipartitions is considered only once and each possible tree is returned only once.

**The algorithm runs in time polynomial in $gn$ where $g$ is the number of greedy resolutions.**

Line 1 of the algorithm uses the $O(kn)$ time algorithms by Gusfield or Warnow [18, 34] to build a tree from the set of compatible bipartitions. Lines 2 and 3 collect all the bipartitions of the same frequency from the head of a sorted linked list and build a compatibility graph. This graph can be built in $O(|F|n)$ time. Generating all $c$ maximal cliques (or independent sets) can be done in $O(|V|^3c)$ time [20, 33]. In the specific case of our algorithm, the output parameters are the number of bipartitions of frequency $f$ and the number of maximal cliques that contain $M$. In practice, these are very small, so the number of the iterations of the
loop on line 4 is close to a constant for a fixed $f$. There is a unique recursive call for each maximal clique. By Lemma 2, this means there is a unique recursive call for each possible greedy set of bipartitions of frequency $f$. Thus, overall, the number of calls is exactly the number of greedy resolutions $g$ and the algorithm runs in time polynomial in that number and $n$. That is, the algorithm is polynomial in $gn$ and satisfies property $(P5)$.

Thus, we have shown that all the sets except for $(S6)$ satisfy the property $(P5)$.

Remark 7. When the number of input trees is at most 2, as we have mentioned in the preliminaries, there exists a polynomial time algorithm to produce an AMT [29]. In the same paper the author give an $O(n^{3.5}p)$ polynomial delay algorithm (where $p$ is the number of AMTs) to enumerate all AMTs of two input trees. Thus, the set $(S6)$ satisfies property $(P5)$ when the input set is at most 2.

While computing an AMT of $K > 2$ trees is NP-hard, it is open (the best of the authors’ knowledge) whether all the AMTs can be enumerated in output-polynomial time. To prove the statement in the negative version one would need to find a class of graphs for which the number of maximum independent sets is polynomial in the number of vertices (such as line graphs, for example) yet it is still NP-hard to compute a maximum independent set. In this case, the size of the output would still be polynomial in the size of the input so computing all maximum independent sets should be polynomial in the input.

**Theorem 4.** None of the output sets $(S1)$–$(S5)$ satisfy property $(P6)$.

**Proof.** Consider the four input trees based on the quartet $(IJ)(KL)$ and a leaf $X$ attached to every leaf in the quartet (see Figure 1). Both the strict and the majority consensus trees are stars and therefore the sets $(S1)$ and $(S3)$ do not satisfy $(P6)$. Every possible refinement of a star include every possible tree on the leaf set $L,J,K,L,X$, including, for example, the tree $(IK)(JLX)$, which does not display $(IJ)(KL)$. Thus, the sets $(S2)$ and $(S4)$ do not satisfy $(P6)$. A similar counterexample exists for the case of displayed triples. Figure 2 shows an example of the input set for which the output set $(S5)$ does not satisfy property $(P6)$.

![Fig. 1. An example showing that the strict and majority consensus output trees do not satisfy property (P6).](image1)

![Fig. 2. All the input trees display a triple $(IJ)K$. However, one of the greedy consensus trees for this input set is $((IX),(JY),(KZ))$, which does not display this triple.](image2)
Theorem 5. The output set of asymmetric median trees (S6) satisfies property (P6).

Proof. We will prove the statement for triples, the proof for the case of quartets is similar. Suppose that all the input trees display a triple (IJ)K. There exists a clade that appears in all the input trees that contains all I, J, and K (may be the root). Therefore that clade appears in every maximal asymmetric median tree. Furthermore, since every input tree displays (IJ)K, there is no clade that contains K and only one of I or J. Thus, in any maximal collection of compatible clades there exists a clade that contains I and J but not K. Therefore, every maximal asymmetric median tree displays (IJ)K. Hence (P6) is satisfied.

Remark 8. If all the input trees display a triple or quartet, then each tree in (S7), obviously, displays it. Thus, (S7) satisfies (P6).

Since none of the output sets (S1)–(S5) satisfy property (P6), none of those sets can satisfy property (P7)b. Since the sets (S1) and (S3) each contain only one tree, they, clearly, cannot satisfy property (P7)a. Thus, we can only hope that the sets (S2), (S4), and (S5) satisfy property (P7)a. This, however, is not the case as the following theorems show it.

Theorem 6. The output sets (S2), (S4), and (S7) satisfy property (P7)a.

Proof. We will prove the theorem for the case of triples. The proof for the case of quartets is similar. If there exists an input tree that displays (IJ)K and no input tree displays (IK)J or (JK)I, then for any input tree it either displays (IJ)K or (IJK). Since none of the input trees display a triple (IK)J or (JK)I, then none of them has a clade that contains IK but not J or JK but not I. Thus, the strict and majority consensus trees can only have clades that either contain all three IJK or only IJ but not K. In the latter case such a tree displays the triple (IJ)K and the property (P7)a is satisfied. In the former case, there exists a refinement among all possible refinements that displays the triple. Therefore, the sets (S2) and (S4) satisfy (P7)a. The input set trivially satisfies the property (P7)a and, therefore, (S7) satisfies it.

Theorem 7. The output sets (S5) and (S6) do not satisfy property (P7)a.

Proof. Consider the following three input trees: (IJX)K, and two copies of (IJK)X. There is a single asymmetric median tree, which is also a majority tree (IJK)X and it does not display (IJ)K. There is no additional refinement possible for the majority using the clades that appear in the input trees and therefore the set of greedy refinements is the majority tree itself. Thus, the sets (S5) and (S6) do not satisfy the property (P7).

Theorem 8. None of the output sets (S1)–(S4) and (S7) satisfy (P9)b.

Proof. If there exists an input tree that does not have \(\pi\) then, by definition the strict consensus tree does not have \(\pi\) and therefore the sets (S1) and (S2) do not satisfy (P9)b. If majority of the input trees do not have \(\pi\) then, by definition, the majority consensus tree does not have \(\pi\) and the sets (S3)–(S5) do not satisfy (P9)b. If at least one of the input trees does not have \(\pi\) then the set (S7), trivially, does not satisfy the property.

Theorem 9. The sets (S5) and (S6) satisfy the property (P9)b.

Proof. In the consensus setting, if there exists an input tree that contains a bipartition \(\pi\) and none of the trees contradict it, then \(\pi\) is compatible with every bipartition in the set of the bipartitions of the input trees and, clearly, with every subset of those bipartitions. Thus, by definition, it is contained in every asymmetric median tree and therefore the set (S6) satisfies (P9)b. Since \(\pi\) is compatible with every subset of bipartitions of the input trees, it can be added to any refinement of the majority tree and therefore the set (S5) satisfies (P9)b.

Theorem 10. The sets (S2), (S4), and (S7) satisfy the property (P9)a.
Proof. Once again, in the consensus setting, if there exists an input tree with a bipartition $\pi$ with no input trees contradicting it, then $\pi$ is compatible with all the partitions of the input trees and any subset of those. Therefore, $\pi$ is compatible with strict and majority consensus tree bipartitions. All possible refinements of a tree include additions of all the compatible bipartitions to the bipartitions of the tree, specifically, $\pi$. Thus, the output sets (S2) and (S4) satisfy the property (P9)a. The input set (S7), trivially satisfies (P9)a.

Table 1 summarizes our results for the consensus setting of incompatible input trees.

<table>
<thead>
<tr>
<th>Compatibility</th>
<th>S1:Strict</th>
<th>S2:Strict+refs</th>
<th>S3:Majority</th>
<th>S4:Maj+refs</th>
<th>S5:Maj+greedy</th>
<th>S6:AMT</th>
<th>S7:Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P3</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P4</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P5</td>
<td>unique, consensus</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P6</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>P7</td>
<td>Y</td>
<td>N</td>
<td>a</td>
<td>N</td>
<td>a</td>
<td>N</td>
<td>a</td>
</tr>
<tr>
<td>P8</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>P9</td>
<td>consensus</td>
<td>N</td>
<td>a</td>
<td>N</td>
<td>a</td>
<td>Y</td>
<td>a</td>
</tr>
</tbody>
</table>

Table 1. The summary of the results presented in this paper. “Y” means a set satisfies a property. If the property has both (a) and (b) versions, then “Y” means the set satisfies the (b) version of the property. “N” means a set does not satisfy a property, even the (a) version. “a” means a set satisfies only the (a) version of a property.

6 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 many deep insights.

References


