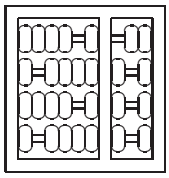

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A Fully Resolved Consensus Between Fully Resolved Phylogenetic Trees

J. A. A. Quitzau *J. Meidanis*

Technical Report - IC-05-027 - Relatório Técnico

October - 2005 - Outubro

A Fully Resolved Consensus Between Fully Resolved Phylogenetic Trees

José Augusto Amgarten Quitzau João Meidanis

October 12, 2005

Abstract

Nowadays, there is a considerable number of phylogeny reconstruction methods and many are able to produce reasonable trees, but none of the methods guarantees to reconstruct the true tree. On the other hand, there is a larger number of phylogenetic consensus methods, which usually put together the most common parts of a collection of phylogenetic trees. Unfortunately, there is also a taboo concerning the use of consensus methods to reconstruct phylogenetic trees, because most biologists see the phylogenetic consensus methods mainly as comparators and not as phylogenetic tree constructors.

In this work, we challenge this taboo by defining a consensus method which builds a fully resolved phylogenetic tree based on the most common parts of fully resolved trees in a given collection. We also present results of simple tests which show that this consensus is usually better than the trees used to build it.

1 Introduction

Walter Fitch said once that "*no one (classification) method or view has all the good points*" [3], which means that even the best phylogenetic reconstruction methods known make mistakes. This is certainly true, since simple methods usually make naive mistakes, while more elaborated ones involve the solution of NP-hard optimization problems and can be solved only by approximation methods, at least up to now. In addition, as Phillips and Warnow point out, "the true tree may only obtain a near-optimal score" [9]. Therefore, even if the optimization problems could be solved in polynomial time, the answer could not be the true tree.

But what makes a tree right or wrong? What exactly is a reconstruction method mistake? When we look at a (unrooted) phylogenetic tree, the information it represents is actually stored in its edges (or branches). Each edge in a phylogenetic tree represents a bipartition of its set of leaves. As a result, we have a pair of disjoint sets for each tree edge. If the edge is correct, at least one of the disjoint sets contains only and all the species that share a specific common ancestor. In other words, every edge in the correct phylogenetic tree separates a group of descendants of a common ancestor from the species that are not their descendants. The problem with all the reconstruction methods is that they are not able to infer all these separations correctly, mostly due to information loss occurred during the evolutionary process.

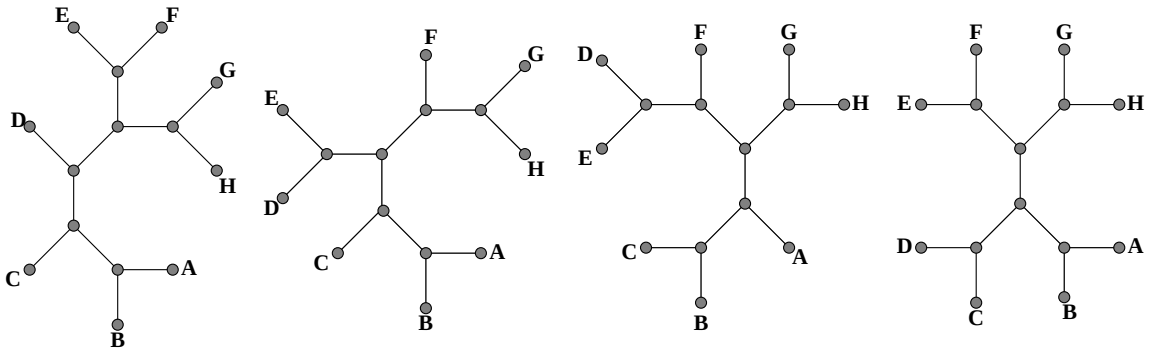

Figure 1: A collection of fully resolved phylogenetic trees.

The differences between trees built by distinct reconstruction methods show places where they may be not correct. Even thought they disagree in some parts, the great amount of similarity shared by these trees may be an indication that none of them is completely wrong. In other words, since reliable reconstruction methods must produce trees with a low number of mistakes, a tree built with the most common parts of the trees in a collection created by these methods has a better chance of being closer to the "true tree".

This work is dedicated to the definition and test of a phylogenetic consensus method that tries to built trees closer to the true tree than the trees used as input. The rest of this report is divided as follows: Section 2 presents the theoretical background for the definition of a consensus method and the description of an algorithm for the consensus calculation. Section 3 presents the result of tests that show the quality of the consensus in comparison to the trees used to build it. Finally, Section 4 presents a brief discussion about the results.

2 Materials and Methods

2.1 Definitions

A *phylogenetic tree* is an undirected acyclic and connected graph without vertices of degree 2. The vertices of degree 1 are called *leaves* and labeled with the elements of a set L. A vertex with degree greater than 3 is named a *polytomy* [8] and a phylogeny without politomies is called *fully resolved*; otherwise it is called *partially resolved*. Figure 1 presents four examples of fully resolved phylogenetic trees.

A *split* $S = (A, B)$ of an arbitrary set X is a bipartition of X in two non-empty subsets A and B. Let $\mathcal{S}(X)$ be the set of all possible splits defined on X, then every collection $\mathcal{S} \subseteq \mathcal{S}(X)$ is a *split system* (defined on X). Two splits S and S' are *compatible* if and only if there are subsets $A \in S$ and $A' \in S'$ such that $A \cap A' = \emptyset$; otherwise they are *incompatible* [2].

Every edge in a phylogenetic tree divides its leaf set into two subsets. In other words, every edge in a phylogenetic tree defines a split on its set of leaves. Therefore, every phylogenetic tree defines a split system on its set of leaves. Given a phylogenetic tree T, we denote the split system defined by the edges of T by $\mathcal{S}[T]$. Two splits in $\mathcal{S}[T]$ are always

compatible. In addition, every phylogenetic tree may be rebuilt based on its split system. We may define the *relative frequency* $p(S, \mathcal{T})$ of the split S in relation to a collection of phylogenetic trees $\mathcal{T}$ as follows:

$$p(S, \mathcal{T}) = \frac{|\{T \in \mathcal{T} \mid S \in \mathcal{S}[T]\}|}{|\mathcal{T}|}.$$

Let $\mathcal{P}[L]$ be the set of all subsets of L. A set $\Psi \subset \mathcal{P}[L]$ is an *n-tree* defined on L if it satisfies the following conditions [6, 10]:

1. $\emptyset \notin \Psi$
2. $L \in \Psi$
3. $\{i\} \in \Psi$ for all $i \in L$
4. $A \cap B \in \{A, B, \emptyset\}$ for all $A, B \in \Psi$

Every subset in an n-tree is called a *subgroup* or *cluster* and two clusters are called *compatible* if they satisfy condition 4; otherwise they are *incompatible*. An n-tree is *fully resolved* when it is maximal, that is, when every cluster that is not in the n-tree is incompatible with at least one of the clusters in the n-tree.

The following theorem characterizes a fully resolved n-tree:

Theorem 2.1 *An n-tree Ψ is fully resolved if and only if for every cluster $C \in \Psi$ with $|C| \geq 2$, there are two clusters $A, B \in \Psi$ such that $A \cup B = C$ and $A \cap B = \emptyset$.*

Proof *(scratch)*

$\Rightarrow$ Suppose that Ψ is fully resolved and there is a cluster $C \in \Psi$ with $|C| \geq 2$, and no clusters $A, B \in \Psi$ such that $A \cup B = C$ and $A \cap B = \emptyset$. Then there is a set of more than two disjoint clusters whose union is equal to C and the union of every pair of clusters in this set may be inserted in Ψ, which is a contradiction.

$\Leftarrow$ Suppose that Ψ is not fully resolved, but for every cluster $C \in \Psi$ with $|C| \geq 2$, there are two clusters $A, B \in \Psi$ such that $A \cup B = C$ and $A \cap B = \emptyset$. Then, there is at least one cluster C' that may be inserted into Ψ. There is a cluster $C \in \Psi$ such that $C' \subset C$ and C is minimal. By hypothesis, there are also $A, B \in \Psi$ such that $A \cup B = C$ and $A \cap B = \emptyset$. But, since $C' \notin \Psi$, if C' and A are compatible, then C' and B are not, and vice versa.

□

2.2 Phylogenetic trees and n-trees

Split systems are good ways to simplify the data represented by a phylogenetic tree, since all phylogenetic trees may be reconstructed from their split systems. However, even split systems may be simplified, as we show in this section.

First of all, let $\varphi \mapsto \mathbb{N}_L$, where $\mathbb{N}_L$ is the set of the first $|L|$ natural numbers, be an enumeration of the elements of L. Then we may define the set $\varphi(R)$, where R is a subset of L, as follows:

$$\varphi(R) = \{\varphi(r) \mid r \in R\}.$$

Given two different clusters R and S of L, we say that R is *smaller* than S, and denote this relation by $R < S$, if and only if

$$\begin{aligned} &|R| < |S| \quad \text{, or} \\ &|R| = |S| \quad \text{and} \quad \min(\varphi(R \setminus S)) < \min(\varphi(S \setminus R)). \end{aligned}$$

Quitzau [11, Theorem 2.3.2] proved that this is an order relation between clusters defined on L.

Let $S = \{A, B\}$ be a split on L such that $A < B$. Then A is the *small cluster* and B is the *big cluster* of S. If T is a phylogenetic tree, we denote by $\mathcal{F}[T]$ the set of small clusters of the splits in $\mathcal{S}[T]$.

Quitzau proved the following theorem:

Theorem 2.2 *Let T be a phylogenetic tree with more than two leaves. Then T is fully resolved if and only if $\mathcal{F}[T]$ has only three maximal n-trees and all these n-trees are fully resolved.*

2.3 The Most Probable Tree

Remembering the discussion in the introduction to this report, the tree built by reliable reconstruction methods are only slightly different, which means that they share a good deal of information. Because the reconstruction methods are trustworthy, the most common parts of these trees are very likely to be correct. It is also important to remember that the information presented by a phylogenetic tree consists basically of its split system. In the next section we present a consensus method that builds trees by creating maximal split systems which also maximize a certain weight function. By doing this, the method keeps only the most common splits found in the collection of trees given as input.

Definition Let the *weight* $p(T, \mathcal{T})$ of a phylogenetic tree T with respect to a collection of phylogenetic trees $\mathcal{T}$ be defined as follows:

$$p(T, \mathcal{T}) = \prod_{S \in \mathcal{S}[T]} p(S, \mathcal{T})$$

The tree T is a *most probable tree* of the collection $\mathcal{T}$ if the following conditions are satisfied:

1. T is a fully resolved phylogenetic tree.
2. There is no fully resolved phylogenetic tree T^* such that $p(T, \mathcal{T}) < p(T^*, \mathcal{T})$.

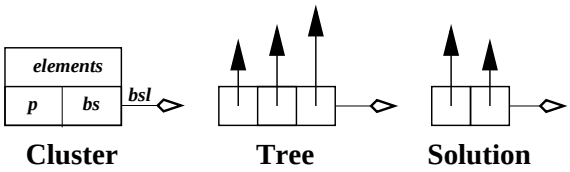

Figure 2: Graphic representation of the three basic types which are used to build the data structure that stores all the most probable trees. The `Cluster` type records the elements of a cluster (`elements`), its relative frequency (`p`), the weight of the stored n-trees (`bs`) and the pairs of clusters which can be combined to the actual cluster in order to build n-trees with the stored weight (`bsl`). The `Tree` type stores a triple of pointers to clusters. Finally, the type `Solution` stores a pair of disjoint clusters. Black arrows are pointers to `Clusters`, while white arrows are pointers to `Trees` and `Solutions`.

2.3.1 Algorithm

The algorithm presented on this section takes advantage of the correspondence between split systems and disjoint sets of n-trees and finds triples of n-trees instead of compatible split systems.

Firstly, we need to rewrite the tree weight function as a function of the small clusters of a phylogenetic tree and the collection of trees. The relative frequency of a cluster C may be written as:

$$p(C, \mathcal{T}) = \frac{|\{T \in \mathcal{T} \mid C \in \mathcal{F}[T]\}|}{|\mathcal{T}|},$$

which corresponds to the probability of finding C in $\mathcal{T}$, and the weight of an n-tree may be defined as

$$p(\Psi, \mathcal{T}) = \prod_{C \in \Psi} p(C, \mathcal{T}).$$

Therefore, the weight function used in the algorithm is:

$$p(T, \mathcal{T}) = \prod_{\substack{\Psi \subset \mathcal{F}[T] \\ \Psi \text{ is } n\text{-tree} \\ \Psi \text{ is maximal}}} p(\Psi, \mathcal{T}),$$

knowing that the maximal n-trees of $\mathcal{F}[T]$ are disjoint [11, Section 2.3].

Secondly, let us make some considerations about trivial n-trees and the combination of n-trees. If a cluster C has only a single element, we may trivially associate an n-tree to it, since the set $\Psi_C = \{C\}$ satisfies all the n-tree conditions. Note that, in this case, the trivial n-tree is maximal, therefore fully resolved. Furthermore, if A and B are two disjoint clusters, Ψ_A is an n-tree on A and Ψ_B is an n-tree on B, then the set $\Psi_C = \{A \cup B\} \cup \Psi_A \cup \Psi_B$ is an n-tree on $C = A \cup B$. In this case, if Ψ_A and Ψ_B are fully resolved, then Ψ_C is also a fully-resolved n-tree. We call a cluster C *solved* when there is at least one fully resolved n-tree Ψ_C associated to it and $p(\Psi_C, \mathcal{T})$ is maximal.

During an algorithm run, the cluster solutions and the phylogenetic trees with maximal weights are stored in a data structure constructed with the building blocks shown in Figure 2. The very basic type is the `Cluster`, which records the main information of a cluster. It has four fields: `elements`, which stores the cluster elements; `p`, which stores the cluster relative frequency, defined at the very beginning of this section; `bs`, which stores a list of pairs of clusters whose best solutions, together with the represented cluster, form an n-tree of maximal weight. The `Tree` type stores a triple of disjoint clusters that cover the set of leaves. The `Solution` type stores a tuple of disjoint clusters. The field `bsl` of the `Cluster` type is actually a linked list of `Solution`s.

Two important sub-routines are the one that stores the best solutions found for a cluster, called CLUSTER, and the one that stores the best phylogenetic trees, called FOREST. Both are described bellow:

```
CLUSTER(C, A, B)
 1  if (C.p * A.bs * B.bs > C.bs)
 2     then C.bsl ← {(A, B)}
 3          C.bs ← C.p * A.bs * B.bs
 4     else if (C.p * A.bs * B.bs = C.bs)
 5             then C.bsl ← C.bsl ∪ {(A, B)}
```

```
FOREST(A, B, C′, F)
 1  if (A.bs * B.bs * C′.bs > F.bs)
 2     then F.bsl ← {(A, B, C′)}
 3          F.bs ← A.bs * B.bs * C′.bs
 4     else if (A.bs * B.bs * C′.bs = F.bs)
 5             then F.bsl ← F.bsl ∪ {(A, B, C′)}
```

These two sub-routines are almost identical. In fact, the only differences between them are the formula of the calculated value and the type of the elements stored in the linked lists. In the case of CLUSTER, the calculated value is the weight of an n-tree, which is the product of the weights of two smaller n-trees and the relative frequency of a cluster. The element stored in this case is a `Solution`. Concerning the FOREST sub-routine, the calculated value is a phylogenetic tree weight and the stored type is a `Tree`.

Both sub-routines consist of two identical comparisons leading to three different actions. At line 1, the new calculated weight is tested and, if greater than the stored one, lines 2 and 3 discard the old list and initialize a new one, with the newly-calculated weight. Otherwise, the sub-routines test if the new weight is equal to the stored one. In an affirmative case, the new `Solution`/`Tree` is added to the best solution list (`bsl`); otherwise it is discarded. It is not difficult to see that both sub-routine running times are limited by a constant.

The core algorithm is presented in pseudo-code below. The procedure SMALL just extracts the small clusters of $\mathcal{T}$ and calculates their relative frequencies, returning a sorted array of clusters.

```
MOST-PROBABLE-TREE(T)
 1  Small ← SMALL(T)
 2  F.bsl ← ∅
 3  F.bs ← −∞
```

```
 4  for  each A in Small, in increasing order
 5     do for  each B in {B ∈ Small | B < A}
 6           do if (A ∩ B = ∅)
 7                 then C ← A ∪ B
 8                      if (|C| ≤ |L| /2)
 9                        then if (C ∈ Small)
10                                then Cluster(C, A, B)
11                                else  Discard (A, B)
12                        else  C' ← L \ C
13                              if (C' < B and C' ∈ Small)
14                                then Forest(A, B, C', F)
15                                else  Discard (A, B, C')
16                 else  Discard (A, B)
17  return F
```

The core of the algorithm is quite simple. It starts by creating an array of clusters at line 1. This array, called `Small`, constains all small clusters found in the collection $\mathcal{T}$ of fully resolved phylogenetic trees and is sorted in increasing order, using the order relation presented in Section 2.2. Lines 2 and 3 create an empty list of most probable trees and initialize the weight of the best tree found, respectively.

After that, the algorithm finds at least one maximal n-tree for each cluster in a bootstrap fashion. In other words, it solves trivially the unitary clusters and then starts to build solutions for larger clusters using the already solved ones. This is made by taking all the clusters $A \in$ Small in increasing order and then analyzing all pairs formed by A and the other clusters B such that $B < A$. Only pairs formed by disjoint clusters are analyzed further. These pairs fall in exactly one of three cases:

$A \cup B \in$ `Small`**:** In this case, for all n-trees Ψ_A such that $p(\Psi_A, \mathcal{T})$ is maximal and for all n-trees Ψ_B such that $p(\Psi_B, \mathcal{T})$ is maximal, $\{A \cup B\} \cup \Psi_A \cup \Psi_B$ may be an n-tree on $A \cup B$ with maximal weight. This case is treated by lines 9 and 10.

$L \setminus (A \cup B) \in$ `Small`**:** In this case, A, B and $L \setminus (A \cup B)$ are three disjoint sets that cover the set of leaves. Therefore, according to Theorem 2.2, the n-tree associated to these clusters together corresponds to a fully-resolved phylogenetic tree and the weight of this tree may be compared to the weight of the stored ones, what is made at lines 13 and 14.

Useless case: In this case, $A \cup B$ is not part of a split found in the collection $\mathcal{T}$ of phylogenetic trees. Such a pair must be discarded.

When all possible pairs have been analyzed, the clusters in `Small` are organized in a structure such as the one shown in Figure 3. Quitzau prooved that, after the execution of the algorithm, the returned structure represents all and only the most probable trees for the collection $\mathcal{T}$ given as input.

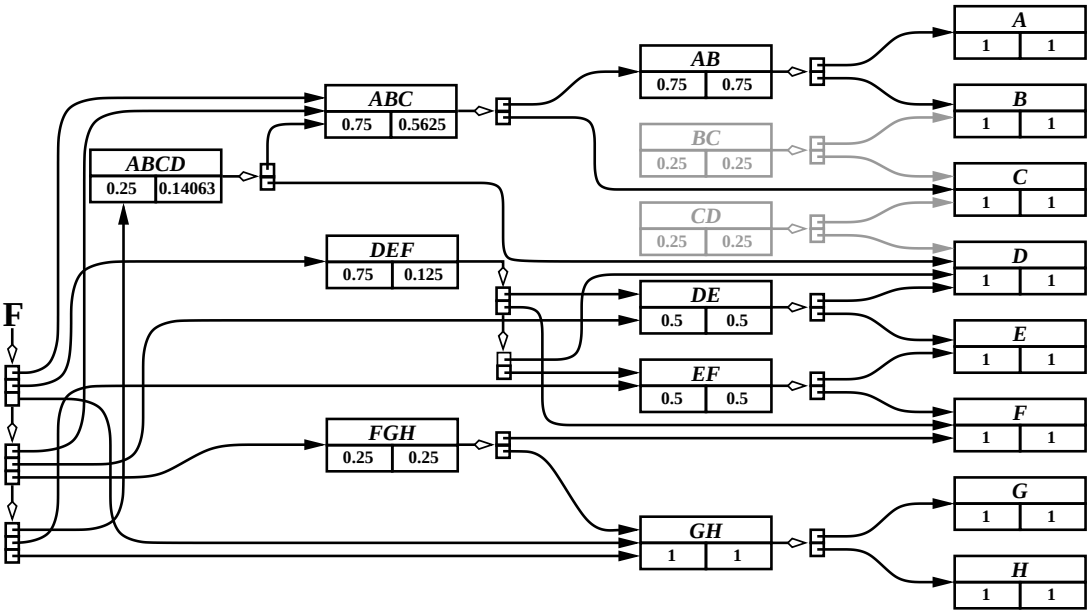

Figure 3: Example of the data structure returned by the algorithm. All the four most probable trees of the collection presented in Figure 2 are represented in this structure. The weight of these trees is 0.0703125. The clusters BC and CD are found in the collection, but no most probable tree T has any of these clusters in $\mathcal{F}[T]$.

2.3.2 The algorithm running time

Every fully resolved phylogenetic tree has $2l - 3$ edges, where l is the number of leaves. If the collection $\mathcal{T}$ has t fully resolved phylogenetic trees, then there are $2lt - 3t$ small clusters to be inserted into the array `Small`. We implemented the algorithm using an array of bits to represent the clusters. Therefore, the comparison between two clusters takes $O(l)$ time. To create the `Small` array, we need to find the position of $2lt - 3t$ clusters in the array by binary search, which takes $O(l^2 t \lg s - 3lt \lg s)$, where $s = |\texttt{Small}|$. In addition, s clusters must be inserted in the array, each one at the cost of $O(s)$. Summing up, the total time spent in the preparation of the array `Small` is

$$O(l^2 t \lg s + s^2).$$

However, since the number of distinct clusters in a collection is $2l - 3$, when all trees are equal, and $l + lt - 3t$ when all the trees are totally different, the time spent with the preparation of `Small` is

$$O(l^2 t \lg lt + l^2 t^2).$$

To find the cluster solutions, s^2 pairs of clusters are analyzed and, in the worst case, the analysis of a cluster requires a binary search in `Small`, which can be made in $O(l \lg s)$. Therefore, the total time used to find the solutions for all clusters is

$$O(s^2 l \lg s),$$

which is an upper bound for the running time of the algorithm and may be written as

$$O(l^3 t^2 \lg lt).$$

2.4 Testing the most probable tree

The tests with the most probable tree, like all reconstruction method tests, have a big problem: How can we measure the method's efficiency if the true tree is unknown? Usually, artificially created data are used in these cases. These sets are created by the application of an evolutionary model over an artificial, but well known, phylogenetic tree.

2.4.1 Datasets

We tested the efficiency of the most probable tree as a reconstruction method using four artificial sequence sets taken from the repository maintained by Gascuel [4]. We chose four artificial datasets simulating different hypotheses of the evolutionary mechanism:

K2P This sequence set was created using the Kimura 2-parameter evolutionary model together with a gamma distribution of transition/transversion rates across sites. The tree topology used was a phylogenetic tree with edge lengths that are not necessarily consistent with the molecular clock hypothesis.

K2Pm The conditions under which these sequences were created are similar to the first sequence set. The only difference is that the edge lengths are consistent with the molecular clock hypothesis.

COV The evolutionary model used for the creation of artificially evolved sequences in this set was the *covarion* model [7, Section 8.8.3]. Like the K2P dataset, the tree topology used does not take the molecular clock hypothesis into account.

COVm For the creation of this dataset, the covarion model was used over a topology that agrees with the molecular clock hypothesis.

A fifth dataset was created with sequences of small ribosomal subunit's RNA taken from the *Ribosomal Database Project* [1]. This dataset is called **REAL** and the reference tree used in this case was the tree also published at the Ribosomal Database Project.

2.4.2 Procedure

The datasets were used as input to the 19 phylogenetic tree constructors described in the Table 1. A phylogenetic constructor is a combination of a software, a tree reconstruction method and, when necessary, the evolutionary model used to estimate distances between sequences. All softwares were used with the default parameters.

As we can see in Table 2, most of the constructors produced one tree for each dataset. However, some of the methods produced hundreds, or even thousands, of different trees. In these cases, only the first output trees were considered for further analysis.

At this point, we had a reference tree and a set of reconstructed trees for each of the datasets. We were able to calculate a most probable tree for each of the datasets. Only the reconstructed trees were used in the construction of the most probable tree. The reference tree was only used to determine how far the rebuilt trees were from their target. To calculate distances between every pair of trees in all datasets, we used the *split distance* [13, Chapter 14].

3 Results

With the distances in hand, we performed two different analysis:

- Comparison of the average distances between reconstructed trees and the consensus tree.
- Identification of the trees that were closer to the reference tree.

The summary of each analysis is found in the next two sections.

Code	**Software**	**Methods**	
FMEJ	`fastMe`	*Minimum evolution*	`JC`
FMEK	`fastMe`	*Minimum evolution*	`K2P`
MMEJ	`Mega`	*Minimum evolution*	`JC`
MMEK	`Mega`	*Minimum evolution*	`K2P`
MMET	`Mega`	*Minimum evolution*	`TN`
MMP	`Mega`	*Maximum parsimony*	
MNJJ	`Mega`	*Neighbor joining*	`JC`
MNJK	`Mega`	*Neighbor joining*	`K2P`
MNJT	`Mega`	*Neighbor joining*	`TN`
PCO	`dnacomp`	*DNA compatibility*	
PML	`dnaml`	*Maximum likelihood*	
PMP	`dnapars`	*Maximum parsimony*	
PNJJ	`neighbor`	*Neighbor joining*	`JC`
PNJK	`neighbor`	*Neighbor joining*	`K2P`
PUPJ	`neighbor`	*UPGMA*	`JC`
PUPK	`neighbor`	*UPGMA*	`K2P`
WNWJ	`weighbor`	*Weighted neighbor joining*	`JC`
WNWK	`weighbor`	*Weighted neighbor joining*	`K2P`

Table 1: Phylogenetic tree constructors used in the tests. The column **Code** presents the codes given for each constructor. The column **Software** indicates the software used by the constructor. Finally, the column **Methods** gives the phylogenetic reconstruction method used to reconstruct the tree and the evolutionary models used to estimate distances, when necessary. The evolutionary models are represented by the abbreviations: `JC` for the Jukes and Cantor's one-parameter model; `K2P`, for Kimura's two-parameter model; and `TN`, for the Tamura-Nei model [12]. Descriptions of the `JC` and `K2P` models are easily found in the literature [5, 11].

Code	**K2P**	**K2Pm**	**COV**	**COVm**	**REAL**
FMEJ	1	1	1	1	1
FMEK	1	1	1	1	1
MMEJ	1	1	1	1	1
MMEK	1	1	1	1	1
MMET	1	1	1	1	1
MMP	27	29,625	403	4,479	8
MNJJ	1	1	1	1	1
MNJK	1	1	1	1	1
MNJT	1	1	1	1	1
PCO	100	100	100	100	100
PML	1	1	1	1	0
PMP	91	945	151	945	1
PNJJ	1	1	1	1	1
PNJK	1	1	1	1	1
PUPJ	1	1	1	1	1
PUPK	1	1	1	1	1
WNWJ	1	1	1	1	1
WNWK	1	1	1	1	1
TOTAL	223	30,685	669	5,539	123

Table 2: Number of trees produced by each constructor for each dataset. The number zero indicates that the constructor was not able to give an output after 24 hours.

Code	**K2P**	**K2Pm**	**COV**	**COVm**	**REAL**
CONS	43.44	77.78	52.67	69.11	60.71
FMEJ	52.78	89.56	64.33	80.11	68.59
FMEK	49.33	90.00	66.33	82.00	72.47
MMEJ	53.89	90.67	59.00	79.89	70.00
MMEK	57.56	90.00	66.11	78.33	70.47
MMET	57.11	92.67	60.11	78.11	74.59
MMP	61.33	115.89	86.00	101.33	90.59
MNJJ	52.33	90.44	59.00	76.33	71.29
MNJK	47.33	89.89	66.56	76.11	69.76
MNJT	52.00	93.11	57.56	77.11	71.41
PCO	147.78	124.78	173.89	117.11	125.29
PML	60.33	112.56	87.22	97.56	—
PMP	60.78	105.11	83.33	98.78	83.76
PNJJ	47.33	82.89	59.33	75.22	75.18
PNJK	47.33	82.33	58.89	78.67	76.82
PUPJ	103.44	129.44	119.56	116.22	76.59
PUPK	105.33	128.56	119.56	116.22	76.71
WNWJ	50.33	88.45	61.33	82.33	74.24
WNWK	51.56	86.78	59.00	83.44	75.53
TOTAL	43.44	77.78	52.67	69.11	60.71

Table 3: Average split distance between the trees in the collection composed by the reconstructed trees and the most probable tree. **CONS** gives the minimum values consistently

3.1 Average Distances

To perform the analysis based on average distances, we used only distances between the reconstructed trees and distances between consensus and the reconstructed trees. The values of the average distances are shown in Table 3.

We can see that the minimum average distance always belongs to the consensus tree. This is an indication that the definition of the most probable tree really reached its objective, which was to create trees that have the most common splits and, therefore, are centralized with respect to the collection of trees used to build them.

3.2 Distances to the reference tree

We calculated the split distance between each phylogenetic tree, including the consensus tree, and the reference tree. The distances are shown in Table 4. As we can see, for the vast majority of the datasets, the consensus tree is closer than more than 60% of the trees in the collection used to build it. The only case where the consensus tree had a bad performance was for the dataset **K2Pm**. Even so, for the dataset containing real sequences, the most probable tree is the tree that is closer to the reference.

4 Discussion

We presented a consensus method called *most probable tree*, that is able to produce fully resolved consensi for a collection of fully resolved phylogenetic trees. This consensus method is based on an optimization criterium. In spite of this, we presented a fast (polynomial) algorithm able to find all most probable trees for a given collection of fully resolved phylogenetic trees.

This consensus method was created with the aim of approximating the trees in a collection to the true tree. This characteristic of the most probable tree was tested by two simple testes repeated over five very different datasets. The behavior of the consensus tree was the same in all the datasets: the consensus tree was centralize in relation to the collection used to build it and it was, in most of the cases, one of the trees closer to the reference.

The use of default parameters when constructing the trees used in the tests may look like a weak point of the tests, since better chosen parameters for the reconstruction methods could have given rise to better trees. Actually, the lack of quality of the rebuilt trees is in fact the strongest point of the tests. Since the position of the tree in the space of phylogenetic trees depends on the position of the trees in the collection used to create it, better trees can only improve the quality of the consensus. What the results show is that the most probable tree is centralized an closer to the reference, regardless of the quality of the trees used to built it.

Of course, five datasets may not be enough to prove that consensus are the best phylogenetic reconstructors. We must keep in mind that consensus methods are not reconstruction methods at all, since they are not able to rebuild a tree from rough data. But phylogenetic consensus methods can be used to improve the quality of a collection of trees built over the same set of species.

Code	**K2P**	**K2Pm**	**COV**	**COVm**	**REAL**
CONS	48	118	88	108	88
FMEJ	58	116	86	110	88
FMEK	56	116	88	108	90
MMEJ	46	116	90	114	110
MMEK	58	116	94	114	110
MMET	54	114	92	114	106
MMP	44	114	88	88	106
MNJJ	52	122	90	114	106
MNJK	56	122	100	116	104
MNJT	52	120	92	114	106
PCO	148	124	174	128	130
PML	38	106	68	98	—
PMP	42	112	86	100	94
PNJJ	54	118	92	116	94
PNJK	56	116	96	118	90
PUPJ	112	136	124	118	98
PUPK	114	136	124	118	96
WNWJ	50	116	92	100	96
WNWK	42	114	84	102	92
$>$ (%)	72.22	33.33	66.67	66.67	94.12
$=$ (%)	0.00	5.56	11.11	5.55	5.88
$<$ (%)	27.78	61.11	22.22	27.78	0.00

Table 4: Distances to the reference tree. The three last rows indicate the percentage of distances that are bigger than the distance from consensus to the reference tree ($>$), the percentage of distances which are equal ($=$) ant the percentage of distances which are smaller ($<$) than the distance from the consensus to the reference tree.

References

[1] J. R. Cole, B. Chai, T. L. Marsh, R. J. Ferris, Q. Wang, S. A. Kulam, S. Chandra, D. M. McGarrell, T. M. Schmidt, G. M. Garrity, and J. M. Tiedje. The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucleic Acids Research*, 31(1):442–443, Jan 2003.

[2] Andreas W. M. Dress. Recent results and new problems in phylogenetic combinatorics. Technical report, University of Bielefeld, December 2001.

[3] Walter M. Fitch. Cladistic and other methods: Problems, pitfalls, and potentials. In T. Duncan and T. F. Stuessy, editors, *Cladistics: Perspectives on the Reconstruction of Evolutionary History*, chapter 12, pages 221–252. Columbia University Press, 1984.

[4] Olivier Gascuel. Methods and algorithms in bioinformatics. http://www.lirmm.fr/∼w3ifa/MAAS/US-MAAS.html, 2003.

[5] Dan Graur and Wen-Hsiung Li. *Fundamentals of Molecular Evolution*. Sinauer Associates, Inc., second edition, 1999.

[6] T. Margush and F. R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43(2):239–244, 1981.

[7] Masatoshi Nei. *Molecular Population Genetics and Evolution*, volume 40 of *Frontiers of Biology*. North-Holland Publishing Company, 1975.

[8] Roderic D. M. Page and Edward C. Holmes. *Molecular Evolution: A Phylogenetic Approach*. Blackwell Science Ltd, 1998.

[9] Cynthia Philips and Tandy J. Warnow. The asymmetric median tree – a new model for building consensus tree. *Discrete Applied Mathematics*, 71:311–335, 1996.

[10] R. C. Powers. Intersection rules for consensus n-trees. *Applied Mathematics Letters*, 8(4):51–55, 1995.

[11] José Augusto Amgarten Quitzau. Um consenso completamente resolvido entre árvores filogenéticas completamente resolvidas. Master's thesis, University of Campinas, February 2005. *Text in Portuguese*.

[12] Koichiro Tamura and Masatoshi Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular Biology and Evolution*, 10(3):512–526, 1993.

[13] Michael S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, 1995.