Online journal ISSN - 1676-5680
GMR Genetics and Molecular Research
EVOLUTION AND TECHNOLOGY

A fully resolved consensus between fully resolved phylogenetic trees

José Augusto Amgarten Quitzau and João Meidanis

Instituto de Computação, Unicamp, Campinas, SP, Brasil
Scylla Bioinformática, Campinas, SP, Brasil
Corresponding author: J. Meidanis
E-mail: meidanis@scylla.com.br

Genet. Mol. Res. 5 (1): 269-283 (2006)
Received January 10, 2006
Accepted February 17, 2006
Published March 31, 2006

ABSTRACT. Nowadays, there are many phylogeny reconstruction methods, each with advantages and disadvantages. We explored the advantages of each method, putting together the common parts of trees constructed by several methods, by means of a consensus computation. A number of phylogenetic consensus methods are already known. Unfortunately, there is also a taboo concerning consensus methods, because most biologists see them mainly as comparators and not as phylogenetic tree constructors. We challenged this taboo by defining a consensus method that builds a fully resolved phylogenetic tree based on the most common parts of fully resolved trees in a given collection. We also generated results showing that this consensus is in a way a kind of "median" of the input trees; as such it can be closer to the correct tree in many situations.

Key words: Phylogeny, Phylogenetic consensus, Splits

INTRODUCTION

There are many cases in which, having a collection of similar elements, one desires to compare these elements or combine the information of the whole collection into a single, consensus element. Probably, the most widely known consensus in bioinformatics is the consensus between sequences. Sequence consensi may be used both for comparison, for instance, to point out probable SNPs in a collection of very similar sequences, and for the reconstruction of a larger or more accurate sequence, in the case of EST clustering or genome assembling.

In the field of phylogenetics, a consensus is mostly used to summarize similarities between trees. As a result, we have many consensus methods designed to point out the common parts of phylogenetic trees. Examples include strict and semi-strict consensus (Kitching, 1998), Nelson-Page consensus (Bryant, 2003), majority rule consensus (Margush and McMorris, 1981) and Adams consensus (Bryant, 2003). Only a few methods, such as the asymmetric median tree (AMT) proposed by Phillips and Warnow (1996), are designed to be used as a final step in tree reconstruction. The main difference between AMT and the other consensus methods is that the former is designed to produce a tree that is a substitute for the collection of trees used to build it; whereas the trees of the other consensus methods are just an image, or a summary, of the whole collection.

One problem with the AMT is that it relies on an optimization problem that is NP-hard in general. We define a new kind of consensus tree, which we call the most probable tree. Its definition relies on a maximization problem that can be solved efficiently by dynamic programming, yielding a polynomial time algorithm for its computation.

In addition, we tested the properties of our new method with several experiments. The first batch of experiments, designed to determine to what extent our tree can be said to be "in between" the trees used to construct it, used artificially generated data. A final experiment, which shows a particular case where our tree came closer to the "original" or "true" tree than most reconstruction methods, is based on the ribosomal RNA sequences used as a standard reference of the phylogeny of living organisms.

BASIC CONCEPTS

Phylogenetic trees

In the present study, we only consider unrooted phylogenetic trees. Therefore, every time that the term phylogenetic tree, or simply tree, appears in this text, it refers to an unrooted phylogenetic tree. A phylogenetic tree is basically a set of nodes connected by branches, as in Figure 1. The nodes connected to a certain node are its neighbors. If a node has only one neighbor, it is called a leaf; otherwise it is an internal node. There are no nodes with exactly two neighbors in the trees, since they are unrooted. Nodes with more than three neighbors are called polytomies. In our work, both the trees in the given collection and the consensus trees have no polytomy, and therefore they are called fully resolved.

Clusters

We denoted the set of leaves of a phylogenetic tree with the letter *L*. Any non-empty

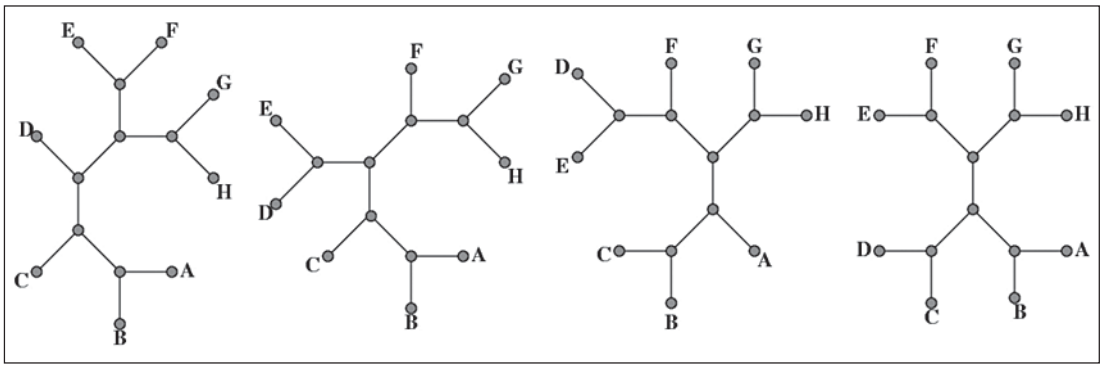

Figure 1. A collection of fully resolved phylogenetic trees.

subset of *L* is called a cluster defined on *L*. When comparing two different clusters, if their sizes differ, we indicated that the cluster with less elements is smaller than the other. For instance, if we have two different clusters *R* and *S*, with *R* having 2 elements and *S* having 3, then *R* is smaller than *S* and we can write $R < S$. If the two clusters have the same number of elements, we have the situation shown in Figure 2. In this case, we use an enumeration of the elements of *L* as a tiebreaker as follows:

1. Consider the elements that are in exactly one of the clusters only.
2. Choose the element numbered with the smallest number.
3. The cluster that has the chosen element is the smallest one.

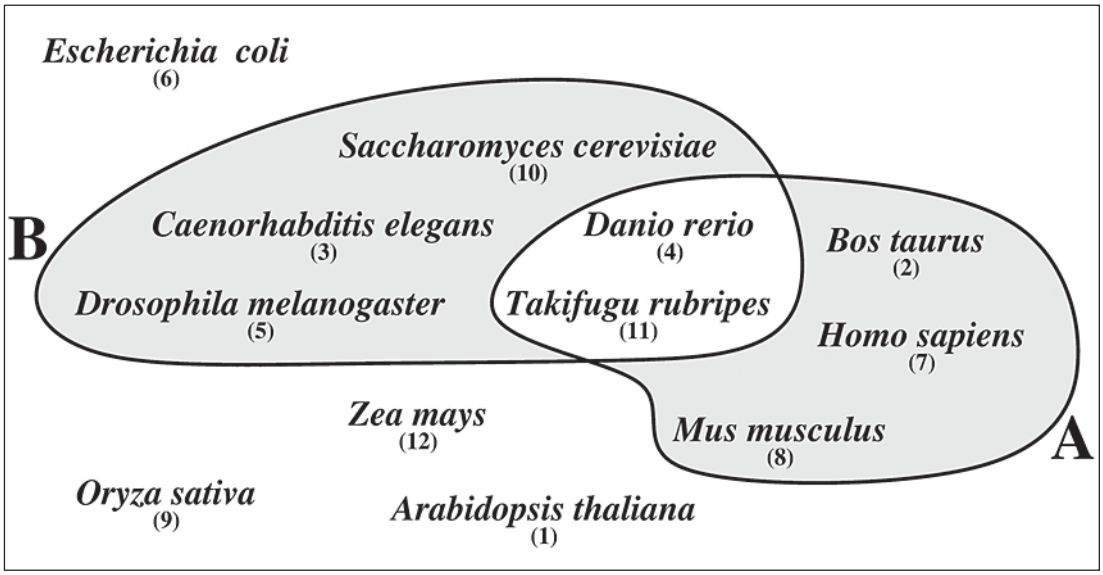

Figure 2. Example of two different clusters of the same size. In these cases, we use an enumeration of the elements of *L* to determine which cluster is the smaller one. The enumeration chosen in the example is the alphabetical order of species names. Note that, among the six species considered (shaded region), the one that has the smallest number is *Bos taurus*. Therefore, we can say that $A < B$.

Quitzau and Meidanis (2005) proved that, when using a fixed enumeration, if one has three different clusters *A*, *B* and *C*, with $A < B$ and $B < C$, then $A < C$. This is an important result because it allows us to sort a collection of clusters and find a cluster very quickly by binary search.

Two clusters *A* and *B* such that $A \cap B = \varnothing$ or $A \cup B = A$ or $A \cup B = B$ are called compatible; otherwise they are incompatible. Clusters can be combined to form trees only when they are compatible.

Splits

It is easy to see that the removal of any branch in a phylogenetic tree transforms it into a pair of rooted phylogenetic trees with smaller sets of leaves, as Figure 3 exemplifies. The sets of leaves of the new trees are clearly clusters of *L* and have the property of being disjoint; that is, they have no common element. At the same time, their union is the set of leaves of the original phylogenetic tree. A pair of clusters with these two properties is called a split of the set of leaves. Note that, since a split is a pair of different clusters, it is always possible to compare them and define which one is the small cluster and which is the large cluster of a split. In addition, two splits are compatible if their small clusters are compatible; otherwise they are incompatible (Quitzau and Meidanis, 2005).

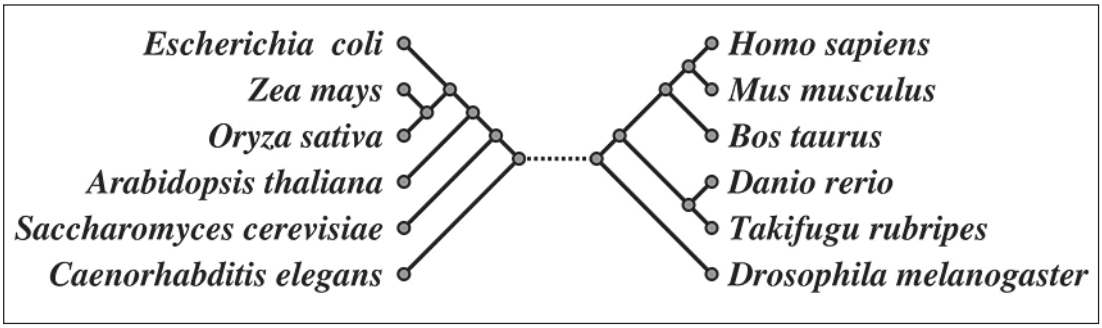

Figure 3. A phylogenetic tree and one of its splits. Note that the sets A = {*Escherichia coli*, *Zea mays*, *Oryza sativa*, *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, *Caenorhabditis elegans*} and B = {*Drosophila melanogaster*, *Takifugu rubripes*, *Danio rerio*, *Bos taurus*, *Mus musculus*, *Homo sapiens*} are such that $A \cap B = \varnothing$, and $A \cup B$ is the whole set of leaves.

Every single branch in a phylogenetic tree corresponds to a different split on its set of leaves. The set of all splits of a phylogenetic tree corresponds to its split system. We denote the split system of a phylogenetic tree *T* by **S**[*T*]. Finally, two different splits found in the split system of the same phylogenetic tree are always compatible.

Split distances

It is possible to measure the distance between two trees by counting the minimum number of changes that must be made in the split system of the first tree in order to obtain the split system of the second one. There are only two possible changes that can be made in a split

system: the addition and the removal of a split. The minimum number of changes is called split distance (Waterman, 1995; Quitzau and Meidanis, 2005) and may be calculated using the formula:

$$\rho(\mathbf{S}[T_1], \mathbf{S}[T_2]) = |\mathbf{S}[T_1]| + |\mathbf{S}[T_2]| - 2|\mathbf{S}[T_1] \cap \mathbf{S}[T_2]| ,$$

where T_1 and T_2 are the first and the second trees, respectively, and $|\mathbf{S}[T]|$ denotes the size, or the number of elements, of the split system $\mathbf{S}[T]$. Figure 4 shows an example of two trees and one of the minimal sequences of operations that transforms the first tree into the second.

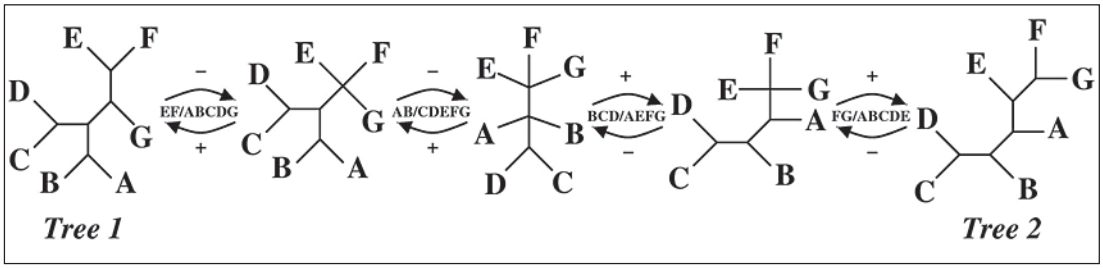

Figure 4. Two fully resolved phylogenetic trees with split distance four, and one of the possible sequences of changes in the split system that transforms one into the other in four steps. The sign + represents the addition and the sign - the removal of a split. The upper arrows show a path from Tree 1 to Tree 2, whereas the lower arrows show the reverse path.

Sets of small clusters

A set of clusters that are all subsets of the same set X is an n-tree if and only if it has the four properties below:

1. It contains the cluster X.
2. It contains a cluster $\{i\}$ for all elements $i \in X$.
3. It does not contain the empty set, since it is not a cluster.
4. All its pairs of clusters are compatible.

If T is a phylogenetic tree, then the set of all small clusters of $\mathbf{S}[T]$, denoted by $\mathbf{F}[T]$, is a set of disjoint n-trees. In particular, if T is fully resolved, then $\mathbf{F}[T]$ has exactly three disjoint maximal n-trees.

Weights

Let $\mathbf{T}$ be a collection of phylogenetic trees, S be a split, and S_s the small cluster of S. Then the weight of S with respect to $\mathbf{T}$, $p(S, \mathbf{T})$, is the relative frequency of S in $\mathbf{T}$, given by the formula

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

which corresponds to the probability of finding S in **T**. The weight can also be defined for clusters, as follows

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

In fact, recalling that S_s denotes the small cluster of S, we have

$$p(S, \mathbf{T}) = p(S_s, \mathbf{T}).$$

The most probable tree

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

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

Note that $p(T, \mathbf{T})$ should correspond to the probability of finding T in **T**, if the choice of the splits of **S**[T] could be made independently; therefore, we call T a most probable tree of a collection **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 $p(T, \mathbf{T}) < p(T^*, \mathbf{T})$.

Algorithm

The algorithm presented in this section takes advantage of the correspondence between split systems and disjoint sets of n-trees and finds trios of n-trees instead of compatible split systems. To begin with, we rewrite the tree weight function as a function of the small clusters of a phylogenetic tree and the collection of trees. The weight of an n-tree can be defined as

$$p(\psi, \mathbf{T}) = \prod_{\mathbf{C} \in \psi} p(C, \mathbf{T}).$$

Therefore, the weight function used in the algorithm is:

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

knowing that the maximal n-trees of **F**[T] are disjoint (Quitzau and Meidanis, 2005).

We now 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 with it and $p(\psi_C, \mathbf{T})$ is maximal.

The algorithm uses the dynamic programming paradigm, using solutions to smaller problem instances to build solutions for larger instances. 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 5. The most basic type is the "Cluster", which records the main information of a cluster. It has four fields: elements, which store the cluster elements; p, which stores the cluster relative frequency; 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 trio of disjoint clusters that cover the set of leaves. The "Solution" type stores a pair of disjoint clusters. The field bsl of the "Cluster" type is actually a linked list of "Solutions".

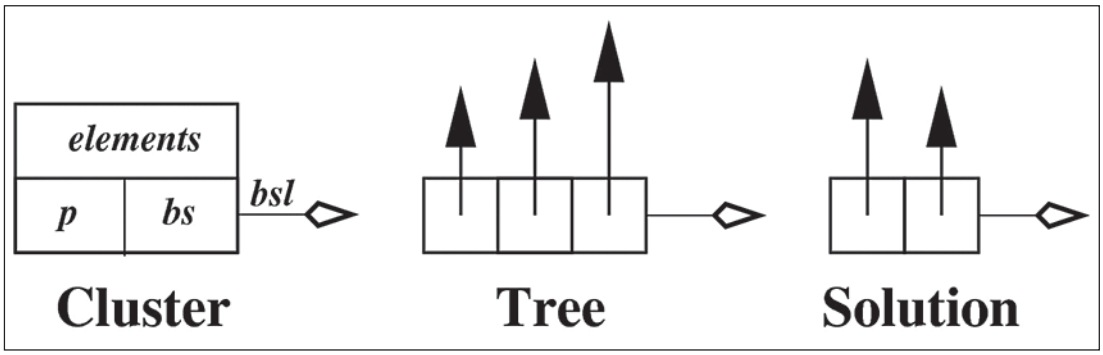

Figure 5. Graphic representation of the three basic types used to build the data structure that stores all the most probable trees. The "Cluster" type records the elements of a cluster (elements), its relative frequency (p), the weight of the stored *n*-trees (bs) and the pairs of clusters that can be combined with the actual cluster in order to build *n*-trees with the stored weight (bsl). The "Tree" type stores a trio 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" or "Solutions".

Two important sub-routines are the one that store the best solutions found for a cluster, called Cluster, and the one that stores the best phylogenetic trees, called Forest. Both are described below:

```
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 analogous comparisons leading to one of three different actions. At line 1, the new calculated weight is tested, and if it is greater than the stored weight, 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. If the answer is affirmative, the new "Solution"/"Tree" is added to the best solution list (bsl); otherwise it is discarded. It is not difficult to see that the running time of each sub-routine is bound by a constant.

The core algorithm is presented in pseudo-code below. The procedure Small just extracts the small clusters of **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, contains all small clusters found in the collection **T** of fully resolved

phylogenetic trees and is sorted in increasing order. 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 trivially solves the singleton clusters and then starts to build solutions for larger clusters using the already-solved clusters. This is achieved by processing 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 further analyzed. These pairs fall in exactly one of three cases:

***A* ∪ *B* ∈ *Small*:** In this case, for all *n*-trees ψ_A such that $p(\psi_A, T)$ is maximal and for all *n*-trees ψ_B such that $p(\psi_B, T)$ 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 in lines 9 and 10.

***L* \ (*A* ∪ *B*) ∈ *Small*:** In this case, A, B and $L \setminus (A \cup B)$ are three disjoint sets that cover the set of leaves. Therefore, the *n*-trees associated with these clusters together correspond to a fully resolved phylogenetic tree, and the weight of this tree is compared to the best weight stored so far, in lines 13 and 14.

Useless case: In this case, $A \cup B$ is not part of a split found in the collection **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 6. Quitzau and Meidanis (2005) proved that, after the execution of the algorithm, the returned structure represents all and only the most probable trees for the collection **T** given as input.

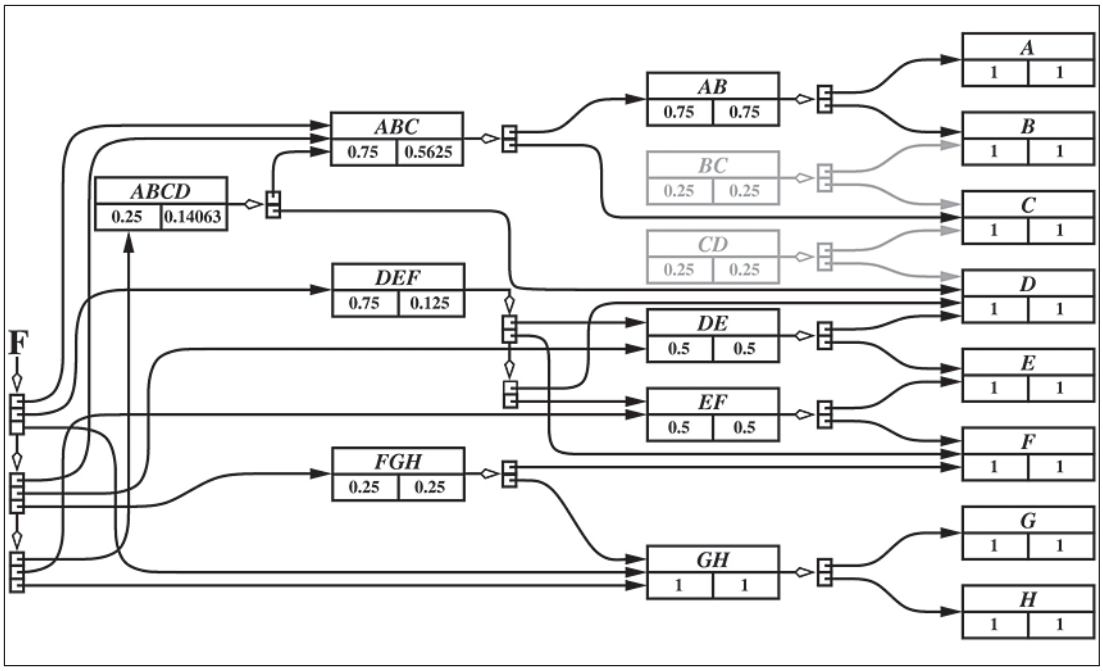

Figure 6. Example of the data structure returned by the algorithm. All four most probable trees of the collection presented in Figure 5 are represented in this structure. The weight of these trees is 0.0703125. The clusters *BC* and *CD* are present in the collection, but none of the most probable trees *T* has any of these clusters in **F**[*T*].

The algorithm running time

Every fully resolved phylogenetic tree has $2l - 3$ edges, where l is the number of leaves. If collection **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 $O(l^2t \lg s - 3lt \lg s)$, where $s = |\text{Small}|$. 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

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

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

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

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

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

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

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

MATERIAL AND METHODS

Data sets

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 (http://www.lirmm.fr/~w3ifa/MAAS/US-MAAS.html). We chose four artificial data sets simulating different hypotheses of the evolutionary mechanism:

K2P - This sequence set was created using Kimura's two-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 (Nei, 1975). As in the K2P data set, the tree topology used does not take the molecular clock hypothesis into account.

COVm - For the creation of this data set, the covarion model was used over a topology compatible with the molecular clock hypothesis.

A fifth data set was created with sequences of small ribosomal subunit RNAs taken from the Ribosomal Database Project (Cole et al., 2003). This data set is called REAL and the reference tree used in this case was the tree published by the Ribosomal Database Project.

Creation of the collections of trees

We chose eight publicly available softwares to create our collections of phylogenetic trees: *fastME* (Desper and Gascuel, 2002); *Mega* (version 3 for Windows) (Kumar et al., 2004); the softwares *dnacomp*, *dnaml*, *dnamlk*, *dnapars*, and *neighbor* from the PHYLIP (Felsenstein, 1989) package (version 3.6), and *weighbor* (Bruno et al., 2000). These softwares allowed us to work with seven phylogenetic reconstruction methods: DNA compatibility, maximum likelihood, maximum parsimony, minimum evolution, neighbor joining, UPGMA, and weighted version of the neighbor joining. In cases where it was necessary to estimate the distance between sequences, three evolutionary models where used: Jukes and Cantor's model, Kimura's two-parameter model (Graur and Li, 1999; Quitzau and Meidanis, 2005), and the Tamura-Nei model (Tamura and Nei, 1993). We call a constructor a combination of a software, a tree reconstruction method and, when necessary, the evolutionary model used to estimate distances between sequences, and we used the 18 constructors shown in Table 1 to build collections of trees. All softwares were used with the default parameters. In the case of the methods DNA compatibility and maximum parsimony, whenever more than one tree was produced, only the first tree in the output file was considered for further analysis. The constructor PML was unable to produce an output for the data set REAL in less than 24 h. It was therefore stopped, and its result not considered.

After creating the trees with the constructors, we used them to create a most probable tree for each collection. In the end, each data set included a reference tree, a set of reconstructed trees and a most probable tree. The distances between each pair of trees were calculated and used for further analysis.

RESULTS

Average distances

To perform the analysis based on average distances, we used only distances between the reconstructed trees and distances between most probable and reconstructed trees (Table 2). The code CONS corresponds to our most probable tree.

We can see that the minimum average distance always belongs to the consensus tree. This means that the most probable tree is centralized with respect to the collection of input trees. Since we used the split distance to make the comparisons, the distance is closely related to the number of splits that the trees have in common. As a result, we note that the definition of the most probable tree actually reached its objective, which was to create trees that have the most common splits found in a set.

Table 1. Phylogenetic tree constructors used in the tests.

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	
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

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 following 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.

Distances to the reference tree

We calculated the split distance between each phylogenetic tree, including the consensus tree, and the reference tree (Table 3). As we can see, for the vast majority of the data sets, the consensus tree is closer to the reference tree 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 data set K2Pm. Even so, for the data set containing real sequences, the most probable tree is the tree that is closest to the reference.

DISCUSSION

We suggest a very simple strategy for a better approximation of reconstructed trees. The main idea was to determine if phylogenetic consensus makes any sense in the context of phylogenetic reconstruction. For verification, we chose reference trees as "true trees" and used sets of DNA sequences to try to reconstruct these trees. In spite of the fact that we knew the parameters used to create the data sets, we tried to simulate a real case of phylogenetic reconstruction. Therefore, because no one really knows which model rules natural evolution, we decided to treat every data set as if we had no idea of the way the sequences evolved. In many cases, this decision led us to create trees that were quite bad, such as the trees built for the

Table 2. Average split distance between the trees in the collection composed by the reconstructed trees and the most probable tree.

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
MINIMUM	*43.44*	*77.78*	*52.67*	*69.11*	*60.71*

CONS, which is our most probable tree, consistently gives the minimum values (in italic).

K2Pm and COVm models. However, our purpose was not to determine whether a reconstruction method based on a certain model is able to reconstruct a tree based on sequences created using the same model; they are supposed to do it successfully. The scenario we tried to analyze was the situation where different, trustworthy methods generate different trees using real data.

In this scenario, we have a tree that is assumed to be correct and unknown. For this reason we chose artificially created data to represent DNA extracted from nature. Since we cannot be sure about which evolutionary model best describes actual DNA evolution, we created four data sets using two different models (covarions and rate across sites) in two opposite variations (molecular clock rules × molecular clock does not rule). In addition, we also used a real data set, based on a well-studied phylogeny: the phylogeny of ribosomal small subunit RNAs.

With the models, the sequences, and the trees in hand, another problem appeared: how to measure distance among trees? We decided to use the split distance to compare the trees, since this definition of distance is directly related to the number of groups (clusters) that the reconstructed tree has in common with the reference cluster. It is unfair to compare partially resolved trees by split distance, because every cluster in a fully resolved phylogenetic tree that is not in the reference is counted at least twice, because one has at least to take a split out of the reconstructed tree and to put the right one in place. This can be seen in Figure 4. In this figure, if we consider that tree number 2 is the reference tree, then both tree 1 and the tree in the middle of the figure make the same mistakes: they were not able to group the clusters {*B*, *C*, *D*} and {*F*, *G*} properly; but the tree in the middle has distance 2, while tree 1 has distance 4. It

Table 3. Distances to the reference tree.

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

The three last rows indicate the percentage of distances that are larger than the distance from consensus to the reference tree (>), the percentage of distances that are equal (=) and the percentage of distances that are smaller (<) than the distance from the consensus to the reference tree (in italic).

could get worse. Note that the tree that has only seven leaves and one internal node has also distance 4 to tree 2, despite the fact that such a tree represents no information at all.

To overcome this obstacle, we presented a consensus method called most probable tree, which is able to produce fully resolved consensi for a collection of fully resolved phylogenetic trees. This consensus method is based on an optimization criterion. 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.

The new 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 checked by two simple tests repeated over five very different data sets. The behavior of the consensus tree was the same in all the data sets: the consensus tree was centralized with respect to the collection used to build it and it was, in most cases, one of the trees closest to the reference.

The use of default parameters when constructing the trees used in the tests may appear to be 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 consensus 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 with respect to the input trees. If those are distributed around the reference tree, the consensus will be closer to the reference, regardless of the quality of the trees used to build it.

Of course, five data sets may not be enough to prove that consensus techniques 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 raw data. But phylogenetic consensus methods can be used to improve the quality of a collection of trees built with the same set of species.

REFERENCES

Bruno WJ, Socci ND and Halpern AL (2000). Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. *Mol. Biol. Evol.* 17: 189-197.

Bryant D (2003). A classification of consensus methods for phylogenetics. In: Bioconsensus (Janowitz M, Lapointe FJ, McMorris F, Roberts FS, eds.). DIMACS-AMS Series, Providence, RI, USA, Vol. 61, pp. 163-184.

Cole JR, Chai B, Marsh TL, Farris RJ, et al. (2003). The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucleic Acids Res.* 31: 442-443.

Desper R and Gascuel O (2002). Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *J. Comput. Biol.* 9: 687-705.

Felsenstein J (1989). PHYLIP - phylogeny inference package (version 3.2). *Cladistics* 5: 164-166.

Graur D and Li WH (1999). Fundamentals of molecular evolution. 2nd edn. Sinauer Associates Inc., Sunderland, MA, USA.

Kitching IJ, Forey PL, Humpries CJ and Williams DM (1998). Cladistics: The theory and practice of parsimony analysis. In: The systematics association publication. 2nd edn. Oxford Publication Press, New York, NY, USA.

Kumar S, Tamura K and Nei M (2004). MEGA3: Integrated software for molecular evolutionary genetics analysis and sequence alignment. *Briefings Bioinformatics* 5: 150-163.

Margush T and McMorris FR (1981). Consensus *n*-trees. *Bull. Math. Biol.* 43: 239-244.

Nei M (1975). Molecular population genetics and evolution. In: Frontiers of biology (Neuberger A and Tatum EL, eds.). Vol. 40. North-Holland Publishing Co., Amsterdam, Netherlands.

Phillips C and Warnow TJ (1996). The asymmetric median tree - a new model for building consensus tree. *Discrete Applied Math.* 71: 311-335.

Quitzau JAA and Meidanis J (2005). A fully resolved consensus between fully resolved phylogenetic trees. Technical Report IC-05-027, Unicamp, Campinas, SP, Brazil.

Tamura K and Nei M (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.* 10: 512-526.

Waterman MS (1995). Introduction to computational biology. Chapman and Hall Ltd., London, England.