Building species trees from larger parts of phylogenomic databases

Celine Scornavacca[1,2,*], Vincent Berry[2], and Vincent Ranwez[1]

[1]Institut des Sciences de l'Evolution (ISEM, UMR 5554 CNRS), Université Montpellier II, Place E. Bataillon - CC 064 - 34095
[2] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM,UMR 5506, CNRS), Université Montpellier II 161, rue Ada, 34392 Montpellier Cedex 5, France,
{scornava, vberry}@lirmm.fr, Vincent.Ranwez@univ-montp2.fr

*Corresponding author

Abstract. Gene trees are leaf-labeled trees inferred from molecular sequences. Due to duplication events arising in genome evolution, gene trees usually have multiple copies of some labels, *i.e.*, species. Inferring a species tree from a set of multi-labeled gene trees (MUL trees) is a well-known problem in computational biology. We propose a novel approach to tackle this problem, mainly to transform a collection of MUL trees into a collection of evolutionary trees, each containing single copies of labels. To that aim, we provide several algorithmic building stones and describe how they fit within a general species tree inference process. First of all, we propose to separately preprocess MUL trees in order to remove their redundant parts with respect to speciation events. For this purpose, we present a tree isomorphism algorithm for MUL trees that can be applied to the pairs of subtrees hanging from duplication nodes. This preprocess lowers the number of duplication nodes in gene trees. For the gene trees that still have duplication nodes, we define the topological information of a MUL tree that can be thought of as being unambiguously related to speciation events. When the MUL tree contains a coherent speciation signal, we show that we can replace the MUL tree with a single-labeled tree representing its speciation information. Otherwise, we propose to extract a maximum subtree that is free of duplication events. Most algorithms have a linear-time complexity, except for an FPT algorithm proposed for a problem that we show to be intractable. The algorithms described in this paper are used to analyse the HOGENOM database, a database of homologous genes from fully sequenced genomes .

1 Introduction

An evolutionary tree (or *phylogeny*), is a tree displaying the evolutionary history of a set of sequences or organisms. A *gene tree* is an evolutionary tree built by analyzing a gene family, *i.e.*, homologous molecular sequences appearing in the genome of different organisms. Gene trees are primarily used to estimate *species*

trees, *i.e.*, trees displaying the evolutionary relationships among studied species. Unfortunately, most gene trees can significantly differ from the species tree for methodological or biological reasons, such as long branch attraction, lateral gene transfers, deep gene coalescence and, principally, gene duplications and losses [1]. For this reason, species trees are usually estimated from a large number of gene trees.

Inferring a species tree from gene trees is mostly done in a two-step approach. First, a micro-evolutionary model that takes into account events affecting individual sites is used to infer the gene trees. The species tree is then inferred on the basis of a macro-evolutionary model, *i.e.*, minimizing the number of transfer, duplication and loss events [2–6]. To produce more biologically meaningful trees, unified models have been proposed in which the micro and macro-evolutionary dimensions are entangled [7–9]. However, it is difficult to determine how to incorporate events occurring on different spatial and temporal scales, as well as belonging to neutral and non-neutral processes, in a single model [9]. Lately, a hybrid approach has been proposed, where a first draft of a species tree is inferred with a micro-evolutionary model, the most uncertain parts of which are then corrected according to a macro-evolutionary model [9].

In this paper, we propose instead to take advantage of the very large number of gene trees present in recent phylogenomic projects to avoid entering into the detail of all possible macro-evolutionary scenarios (*e.g.* is a parsimony approach always justified? Should only the most parsimonious scenario be retained?). We propose to extract the non-ambiguous part of the topological information contained in the gene trees, *i.e.*, that resulting from speciation events as opposed to duplication events, and then apply a traditional supertree method letting the weight of evidence decide in favor of one candidate species tree [10–12].

This approach is only possible when the number of gene trees is very large, and indeed this is now the case in projects such as the HOMOLENS database (`http://pbil.univ-lyon1.fr/databases/homolens.php`) and the HOGENOM database (`http://pbil.univ-lyon1.fr/databases/hogenom.php`) storing several thousands of gene trees. In the release 04 of these databases, respectively 51% and 71% of gene families have paralogous sequences, *i.e.*, sequences where duplications and losses have actually taken place. Currently, these gene families are discarded when inferring a supertree of the concerned species. Disentangling information derived from speciation events from that resulting from duplication events would thus provide more information for species tree inference.

Supertree methods combine source trees whose leaves are labeled with individual species into a larger species tree. The source trees are *single-labeled*, *i.e.*, each species labels at most one leaf. Note that, by definition, the inferred supertree is also single-labeled. In contrast, gene trees are usually *multi-labeled*, *i.e.*, a single species can label more than one leaf, since duplication events almost always resulted in the presence of several copies of the genes in the species genomes. The task we therefore have to solve is to extract the largest amount of unambiguous topological information from the multi-labeled gene trees under

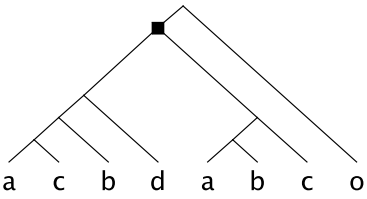

Fig. 1. An example of MUL tree with one odn indicated by a black square

the form of single-labeled trees. This paper presents a number of results in this direction, that all play a role in the general scheme that is fully described below.

First of all, we propose to separately preprocess MUL trees in order to remove their redundant parts with respect to speciation events. For this purpose, we extend the tree isomorphism algorithm of [13] making it applicable to MUL trees while preserving a linear running time (section 3). This algorithm is then applied to the pairs of subtrees hanging from duplication nodes in MUL trees. This preprocess lowers the number of duplication nodes in gene trees. We also give in passing a linear time algorithm to identify duplication nodes in MUL trees (section 2.1). For the gene trees that still have duplication nodes, we define a set $\mathcal{R}$ of triplets (binary rooted trees on three leaves [14, 12]) containing the topological information of a MUL tree that can be thought of as being unambiguously related to speciation events. We show that this set of triplets can be computed in $O(|\mathcal{R}|)$ time (section 4). When this set is compatible, the MUL tree contributes a coherent topological signal to build the species tree. In such a case, we can replace the MUL tree with a single-labeled tree representing its associated set of triplets by using the AncestralBuild algorithm [17] (section 4) or the PhySIC algorithm [24] (section 7). In section 4 we also show that it is possible to check the auto-coherency of a binary MUL tree M by using a triplet set whose size is at most equal to the number of speciation nodes of M. When a MUL tree is not auto-coherent, we propose to extract a maximum subtree that is both auto-coherent and free of duplication events. Surprisingly, this optimization problem can be solved in linear time (section 5). When extracting largest single-labeled subtrees from MUL trees it is possible to obtain an incompatible collection, when a compatible collection could have been obtained by choosing subtrees of MUL trees in a coordinated way. However, solving this problem is computationally harder, as we show by providing an NP-completeness proof (section 6). The algorithms described in this paper are used to analyse the HOGENOM database, a database of homologous genes from fully sequenced genomes (section 7).

2 Preliminaries

In this paper we focus on rooted binary multi-labeled (MUL) trees. Let M be a

MUL tree and v a node of M. If v is a leaf node, we denote by $\boldsymbol{l_v}$ its label. The set of leaf nodes of M is denoted by $\mathcal{L}(M)$. If v is an internal node, denote by $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ the two sons of v and by $\mathbf{sons(v)}$ the set $\{v_1, v_2\}$. We define by $\boldsymbol{M_v}$ the subtree with v as root and by $\boldsymbol{L(v)}$ the multiset of labels of M_v. We denote by $L(M)$ the multiset $L(root(M))$.

Definition 1. *A node v of M is called an **observed duplication node (odn)** if the intersection of $L(v_1)$ and $L(v_2)$ is not empty.*

Note that, for an odn v, $L(v)$ will always contain some label more than once. We denote by $\mathcal{D}(M)$ the set of odn. A label $l \in L(M)$ is a *repeated label* for M iff the label l occurs more than once in $L(M)$. We say that f is a *repeated leaf* for M iff $L(f)$ is a repeated label.

2.1 Computing $\mathcal{D}(M)$ in Linear Time

The easiest way to compute $\mathcal{D}(M)$ is checking for each node v if the sets $L(v_1)$ and $L(v_2)$ have at least one label in common; in the case of a positive answer, v is inserted in $\mathcal{D}(M)$. The complexity of this approach is $O(n^2)$, since it requires computing $O(n)$ intersections of two lists of $O(n)$ elements. The algorithm 1 uses the lca to find the set of odn $\mathcal{D}(M)$ and requires only linear time. To demonstrate the correctness of algorithm 1, we need to determine some relationships between the lca and the odn.

Lemma 1. *A node is an odn if and only if it is the lca of at least two repeated leaves m and p.*

Proof. Indeed, from the definition 1, v is an odn iff $\mathrm{L}(v_1) \cap \mathrm{L}(v_2) \neq \emptyset$. Therefore, there exist two leaf nodes m and p with $m \in M_{v_1}$ and $p \in M_{v_2}$ such that $l_m = l_p$. Thus v is a common ancestor of the two leaves m and p with the same label. Now, m and p belong to two different subtrees having v as father ($m \in M_{v_1}$ and $p \in M_{v_2}$), hence v is their lowest common ancestor in M. Reciprocally, if v is the lca of two leaves m and p with the same label, this means that $\mathrm{L}(v_1) \cap \mathrm{L}(v_2) \neq \emptyset$, then v is an odn according to definition 1. □

According to Lemma 1, we can search for the lca of any two leaves m and p with the same label. To determine the lca between multiple pairs of nodes, one can use an algorithm in [15] which preprocesses a data structure in O(n) time, where n is the number of nodes and returns the lca of any two specific nodes from the data structure in O(1). We still have $\mathrm{O}(n^2)$ of these couples, and even constant time for each gives an $\mathrm{O}(n^2)$ total complexity. However, since there are only O(n) odn, checking the lca of any pair of leaves computes the same lca several times. A smarter approach is used in algorithm 1: first of all, the subtrees of M are ordered from the left to the right in an arbitrary way. Then, each repeated leaf, starting from the left of the tree and moving to the right, is tagged with the repeated label followed by its occurrence number. Then, for each repeated label e, the lca of any two successive occurrences of e, e_i and e_{i+1} is inserted in $\mathcal{D}(M)$. This leads to a linear time complexity. Indeed, we have $O(n)$

of these couples since each leaf of M is involved in at most two pairs (e_i, e_{i+1}).

Algorithm 1: CompDuplicationNodes(r)

Data: A MUL tree M.
Result: A set of odn $\mathcal{D}(M)$.
Order M in an arbitrary way. In this order, tag each duplicated leaf with the repeated label followed by its occurrence number. Compute the lca for each couple of leaves.
$\mathcal{D}(M) \leftarrow \emptyset$;
foreach (*repeated label* e) **do**
  foreach $(\{e_j, e_{j+1}\})$ **do** $\mathcal{D}(M) \leftarrow lca(e_j, e_{j+1})$;
return $\mathcal{D}(M)$;

The correctness of algorithm 1 is justified by Lemma 2 showing that algorithm 1 retrieves all the odn of M.

Lemma 2. *Let M be a MUL tree. For each odn v, $\exists$ two successive occurrences of a label e denoted by e_i and e_{i+1} s.t. $v = lca(e_i, e_{i+1})$.*

Proof. Given an odn v, there exists at least one label e present in both subtrees M_{v_1} and M_{v_2}, where v_1 (resp v_2) is the left (resp right) son of v. We denote by A the set of leaves a_i *s.t.* $a_i \in M_{v_1}$ and $l_{a_i} = e$ and by B the set of leaves b_j *s.t.* $b_j \in M_{v_2}$ and $l_{b_j} = e$. If we take the last element of A ($a_{|A|}$) and the first one of b (b_1), we know that v is their lca. Additionally, due to the way we tagged M, we know that there is no other occurrence of the label e between $a_{|A|}$ and b_1. Indeed, if there was another leaf x labeled with e, it would be either in v_1 (and then $x = a_{|A|}$) or in v_2 (and then $x = b_1$). Then $a_{|A|}$ and b_1 are two successive occurrences of the same label and their lca is the node v. □

3 Isomorphic Subtrees

Definition 2. *Two rooted trees T_1 , T_2 are isomorphic (denoted by T_1=T_2) iff there exists a one-to-one mapping from the nodes of T_1 onto the nodes of T_2 preserving leaf labels and descendancy.*

We are interested in testing if, for each odn v, the two subtrees v_1 and v_2 are isomorphic or not. In the positive, we can prune one of the two isomorphic subtrees without losing any topological information related to speciation events (see Proposition 1 in section 4) and eliminate the odn v, as in the example

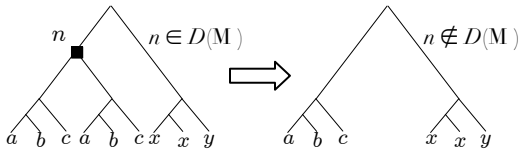

Fig. 2. An example of MUL tree where the sons of the duplication node are isomorphic

of Fig. 2. For detecting isomorphism of MUL trees, we propose Algorithm 2, an extension of the CHECK-ISOMORPHISM-OR-FIND-CONFLICT algorithm [22]. Indeed, the latter does not deal with MUL trees. Alternatively, we could have proposed an appropriate variant of the tree isomorphism algorithm detailed in [23]. However, such an algorithm would likely have been less space efficient than the one we present here due to a number of string sorting steps using a number of queues and lists to ensure linear running time.

A node that has only two leaves as children is called **cherry**. In the case of single-labeled trees we have the following lemma:

Lemma 3. *[13] Let T_1, T_2 be two isomorphic trees and let c_1 be a cherry in T_1. Then, there is a cherry $c_2 \in T_2$ s.t. $L(c_1) = L(c_2)$.*

In the case of MUL trees, we can have several copies of the same cherry. We call a **multiple cherry** the list of cherries on the same two labels. For a multiple cherry mc, we note $|mc|$ the number of occurences of the cherry in its tree.

Lemma 4. *Let M_1, M_2 be two isomorphic MUL trees and let mc_1 be a multiple cherry in M_1. Then, there is a multiple cherry $mc_2 \in M_2$ s.t. $L(mc_1) = L(mc_2)$ and $|mc_1| = |mc_2|$.*

The proof is inspired from that of Lemma 3 in [13] and left to the reader.

3.1 Outline of the Algorithm

First of all, we find all the multiple cherries for the MUL trees M_1 and M_2. We store them in the list L_{mc} using a simple linked list. Additionally, we use a hashtable H where each $mc \in L_{mc}$ is a key. H associates to each multiple cherry mc two linked lists, $O_1(mc)$ and $O_2(mc)$, storing pointers to nodes of M_1 and M_2 respectively that correspond to the occurrences of mc. The multiple cherries of a MUL tree are then examined in a bottom-up process. Given a multiple cherry mc in L_{mc} we check if the size of $O_1(mc)$ is the same as that of $O_2(mc)$. If this is not the case, we have found a multiple cherry for which we do not have the same number of occurrences in M_1 and M_2. In this instance, M_1 and M_2 are not isomorphic (Lemma 4) and the algorithm returns FALSE. Otherwise we turn all the cherries in $O_1(mc)$ and $O_2(mc)$ into leaves to which a same new label, different from all other labels in M_1 and M_2, is assigned. This modification of M_1 and M_2 can turn the fathers of some cherries in $O_1(mc)$ and $O_2(mc)$ into new cherries. Then L_{mc} is updated and the processing of cherries in M_1 is iterated until both MUL trees are reduced to a single leaf with the same label if M_1 and M_2 are isomorphic, or a FALSE statement is returned.

Theorem 1. *Let M_1 and M_2 be two rooted MUL trees with $L(M_1) = L(M_2)$ of cardinality n. In time $O(n)$, algorithm 4 returns TRUE if M_1 and M_2 are isomorphic, FALSE otherwise.*

Proof. This algorithm is an extension of the CHECK-ISOMORPHISM-OR-FIND-CONFLICT algorithm [22] applicable to MUL trees. We show here that we can

Algorithm 2: CheckIsomorphismMULTree(M_1,M_2)

Data: Two MUL tree M_1 and M_2.
Result: TRUE if M_1 and M_2 are isomorphic, FALSE otherwise.
Let L_{mc} be the list of multiple cherries in M_1 and M_2. Let H be the hashtable where each $mc \in L_{mc}$ is a key. To each mc, H associates two lists $O_1(mc)$ and $O_2(mc)$, respectively of the occurrences of mc in M_1 and M_2;
while $(L_{mc} \neq \emptyset)$ **do**
  $mc \leftarrow$ removeFirst(L_{mc});
  if $(O_1(mc) = O_2(mc))$ **then**
    Turn all cherries in $O_1(mc)$ and $O_2(mc)$ into leaves to which a same new label is assigned;
    add the new multiple cherries in L_{mc} and H;
  else return FALSE;
return TRUE;

keep a linear time execution, using supplementary data structures.
A simple depth-first search of trees M_1 and M_2 initializes L_{mc} and H in $O(n)$ time. At each iteration of the algorithm, choosing a multiple cherry mc to process is done in $O(1)$ by removing the first element mc of L_{mc}. H then provides in $O(1)$ the lists $O_1(mc)$ and $O_2(mc)$ of its occurrences in the trees. Checking that these lists have the same number of elements is proportional to the number of nodes they contain, hence costs $O(n)$ amortized time, as each node is only once in such a list, and the list is processed once during the whole algorithm. Replacing all occurrences of mc by a new label is done in $O(n)$ amortized time, since each replacement is a local operation replacing three nodes by one in a tree and at most $O(n)$ such replacements can take place in a tree to reduce it down to a single node (the algorithm stops when this situation is reached). Reducing a cherry can create a new occurrence $o_{mc'}$ of a cherry mc'. Checking in $O(1)$ time if mc' is a key in H allows to know whether occurrences of mc' have already been encountered or not. In the positive, we simply add $o_{mc'}$ to the beginning of the list $O_1(mc)$ (if $o_{mc'} \in M_1$) or $O_2(mc)$ (if $o_{mc'} \in M_2$), requiring $O(1)$ time. In the negative, we add mc' to the beginning of L_{mc}, create a new entry in H for mc', and initialize the associated lists $O_1(mc)$ and $O_2(mc)$ so that one contains $o_{mc'}$ and the other is the empty list. Again, this requires only $O(1)$ time. Thus, performing all operations required by the algorithm globally costs $O(n)$ time. □

Apply algorithm 2 to M_{v_1} and M_{v_2} for each odn v of a MUL tree M in a bottom-up approach requires $O(dn)$ time, where d is the number of duplication nodes in M.

4 Auto-coherency of a MUL Tree

The algorithm 2 can be used to lower the number of duplication nodes in gene trees. Let M be a gene tree M that still have duplication nodes after having

removed isomorphic sibling subtrees in a bottom-up approach as described in section 3. Since M contains several copies of the same gene, we can wonder whether the evolutionary signal of each copy is coherent or not. Let introduce some notations to formalize this concept.

Given a (single/multi)labeled evolutionary tree M. For every three leaves in $L(M)$, we can have three different rooted tree binary shapes, called *triplets*. We denote by $ab|c$ the rooted tree that connects the pair of taxa (a, b) to c via the root and by $\mathcal{R}(M)$ the set of triplets of M *i.e.*, $\mathcal{R}(M) = \{ab|c$ *s.t.* there exist three leaf nodes $x, y, z \in M : l_x = a$, $l_y = b$, $l_z = c$ and $lca(x, y) \neq (lca(x, z) = lca(y, z))\}$[1].

Definition 3. *Let M be a MUL tree. We define by $\boldsymbol{\mathcal{R}_{wd}(M)}$ ($\mathcal{R}(M)$ without duplications) the set of triplets $ab|c$ of $\mathcal{R}(M)$* s.t. *there exist three leaf nodes $x, y, z \in M : l_x = a$, $l_y = b$, $l_z = c$ and $lca(x, y) \notin \mathcal{D}(M)$, $lca(x, y, z) \notin \mathcal{D}(M)$, $lca(x, y) \neq (lca(x, z) = lca(y, z))$.*

For example, for the MUL tree in Fig. 1, $\mathcal{R}_{wd}(M)$=$\{ac|b$, $ac|d$,$ab|d$,$bc|d$,$ac|o$,$ab|o$, $ad|o$,$bc|o$,$cd|o$,$bd|o$,$ab|c\}$. Hence, not all the triplets of $\mathcal{R}(M)$ are kept. This is due to the fact that, once a duplication event occurred in a gene's history, the two copies of the gene evolved independently. The history of each copy is influenced by the species history but, considering them simultaneously may produce information unrelated to the species evolution. Therefore, it is more appropriate to discard the triplets mixing the histories of distinct copies of a gene.

$\mathcal{R}_{wd}(M)$ has size $O(n^3)$ and can be computed in $O(n^3)$ time, where n is the number of leaf nodes of M. Indeed, once the lca of all pairs of nodes in M are computed in $O(n)$ time (see [15, 16]), checking for three leaf nodes x, y, z of M if they satisfy Definition 3 can be done in $O(1)$ time, thus in $O(n^3)$ for all triplets of leaves in M.

Proposition 1. *Let M be a MUL tree and M' the MUL tree obtain by applying algorithm 2 to eliminate isomorphic sibling subtrees. Then $\mathcal{R}_{wd}(M) = \mathcal{R}_{wd}(M')$.*

Definition 4. *A triplet set R is said to be **compatible** if there exists a single-labeled tree T such that $R \subseteq \mathcal{R}(T)$.*

Definition 5. *A MUL tree M is said to be **auto-coherent** if the triplet set $\mathcal{R}_{wd}(M)$ is compatible,* i.e.*, if there exists a single-labeled tree T such that $\mathcal{R}_{wd}(M) \subseteq \mathcal{R}(T)$.*

In the case of an auto-coherent MUL tree, we know that there exists at least one tree T containing all the information in $\mathcal{R}_{wd}(M)$, *i.e.*, the information of M that is considered reliable. To check if a MUL tree is auto-coherent, we use the AncestralBuild algorithm of [17]. For a set of triplets R, this algorithm indicates in $O(|R| \cdot \log^2(|L(R)|))$ time whether R is compatible, where $L(R)$ is the set of

[1] The root of the tree is considered to be up and its leaves down. $lca(x, y)$ denotes the least common ancestor of nodes x and y, *i.e.*, the lowest node in the tree that has both x and y as descendants.

leaf labels of the elements of R. Moreover, in case of a positive answer it returns a tree T *s.t.* $R \subseteq \mathcal{R}(T)$.

It has been proved in [18] that a binary single-labeled rooted tree T can be encoded using a triplet set $\mathcal{R}^l(T)$ whose size is the number of inner nodes of T. In this section we show that it is possible to check the auto-coherency of a binary MUL tree M by using as representation of $\mathcal{R}_{wd}(M)$ a triplet set $\mathcal{R}^l_{wd}(M)$ whose size is at most equal to the number of speciation nodes of M. To univocally define the set $\mathcal{R}^l_{wd}(M)$, let $<$ be a total order on the leaf set $L(M)$. For each node v of M, we denote by $\mathbf{sm}(\mathbf{v})$ the smallest element of $L(M_v)$ according to $<$ and by $\mathbf{anc}(\mathbf{v})$ the set of nodes belonging to the path from v to the root of M. Let $\mathbf{lsa}(\mathbf{v})$ be the least speciation ancestor of v *i.e.* the speciation node in $anc(v)$ closest to v and $\mathbf{v'}$ the son of $lsa(v)$ such that $v \notin M_{v'}$. Note that, if the father of v is not in $\mathcal{D}(M)$, it coincides with $lsa(v)$ while v' is the sibling node of v.

Definition 6. *Let M be a binary MUL tree and $<$ a total order on $L(M)$. We define by $\mathcal{R}^l_{wd}(M)$ the set of triplets $ab|c$ such that $ab|c \in \mathcal{R}_{wd}(M)$ and there exists a speciation node v in M such that $sm(v_1) = a$, $sm(v_2) = b$ and $sm(v') = c$.*

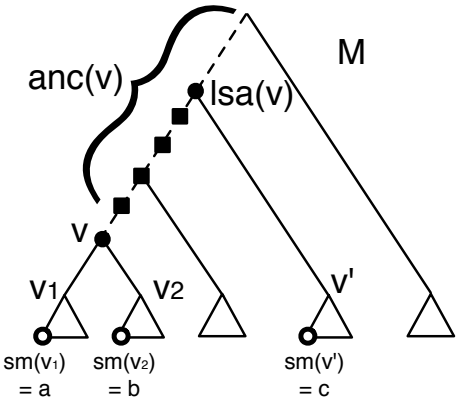

Fig. 3. The only triplet of $\mathcal{R}^l_{wd}(M)$ associated to the speciation node v is $ab|c$, while the triplet set associated to v in $\mathcal{R}_{wd}(M)$ is composed by the triplets $l_x l_y | l_z$ of $\mathcal{R}(M)$, where $x \in \mathcal{L}(M_{v_1})$, $y \in \mathcal{L}(M_{v_2})$ and $z \in \mathcal{L}(M)$ such that $lca(x, y, z) \notin \mathcal{D}(M)$ and $lca(x, y) \neq (lca(x, z) = lca(y, z))$.

Note that, for each speciation node v, the set $\mathcal{R}^l_{wd}(M)$ contains at most one triplet while $\mathcal{R}_{wd}(M)$ tipically contains many more triplets (see figure 3).

Once the set of duplication nodes $\mathcal{D}(M)$ is calculated, Algorithm 3 computes $\mathcal{R}^l_{wd}(M)$ in linear time (see Theorem 2). We now need to show that checking the auto-coherency of $\mathcal{R}_{wd}(M)$ and $\mathcal{R}^l_{wd}(M)$ is equivalent. To do that, we need

Algorithm 3: linearRepresentation$(M, \mathcal{D}(M), v)$

Data: A binary MUL tree M, the set of duplication nodes $\mathcal{D}(M)$ of M, a node v in M.
Result: A set of triplets $\mathcal{R}^l$.
$\mathcal{R}^l \leftarrow \emptyset$;
if (v *is not a leaf* ***and*** v *is not the root node*) **then**
  $f \leftarrow$ the father of v;
  if ($f \notin \mathcal{D}(M)$) **then**
    if ($f_1 = v$) **then** $v' \leftarrow f_2$;
    else $v' \leftarrow f_1$;
  else
    $v' \leftarrow f'$;
$\mathcal{R}^l \leftarrow \mathcal{R}^l \cup$ linearRepresentation$(M, \mathcal{D}(M), v_1)$;
$\mathcal{R}^l \leftarrow \mathcal{R}^l \cup$ linearRepresentation$(M, \mathcal{D}(M), v_2)$;
if ($v \notin \mathcal{D}(M)$) *and* ($v' \neq \emptyset$) **then**
  $\mathcal{R}^l \leftarrow sm(v_1)sm(v_2)|sm(v')$;
return $\mathcal{R}^l$;

to introduce some notations. Given a node v of a MUL tree M, we define the height of v, denoted by $\mathbf{h(v)}$, as the longest path between v and its descendants. More formally, the height of a leaf is fixed to zero and that of a internal node v is max($h(v_1)$,$h(v_2)$)+1. We denote by $\boldsymbol{\mathcal{G}(R)}$ the Aho graph, or clustering graph built from R (see [14, 19, 18]). The set of vertices of this graph is $L(R)$. There is an edge in $\mathcal{G}(R)$ connecting two vertices a and b iff there exists $ab|c \in R$. This graph is the traditional way to check the compatibility of a set of rooted trees (see [17] for another approach to this problem).

The proof that the auto-coherency of $\mathcal{R}_{wd}(M)$ can be tested by checking that of $\mathcal{R}^l_{wd}(M)$ relies on the following Lemma.

Lemma 5. *Let M be a binary MUL tree and v a node of M. Then, if $anc(v)$ contains at least one speciation node, then $\mathcal{G}(\mathcal{R}^l_{wd}(M))|L(M_v)$ is connected.*

Proof. We prove the Lemma by induction on the height of the node v. Note that we need to consider only those nodes having at least one speciation node as ancestor since Lemma 5 does not apply for nodes not having this property.

Let start showing that Lemma 5 is valid for all nodes with height 0. In this case $L(M_v)$ contains a single label, hence $\mathcal{G}(\mathcal{R}^l_{wd}(M))|L(M_v)$ contains only one vertex *i.e.*, it is trivially connected.

Now suppose that Lemma 5 is valid for all nodes v such that $h(v) < \bar{h}$. We want to prove that this implies that it is true for all nodes v: $h(v) \leqslant \bar{h}$. Let v be a node for which $anc(v)$ contains at least one speciation node and $h(v) = \bar{h}$. Since $h(v_1) = h(v) - 1$ and $anc(v_1) \supseteq lsa(v)$ we know that $\mathcal{G}(\mathcal{R}^l_{wd}(M))|L(M_{v_1})$ is connected and the same holds for $\mathcal{G}(\mathcal{R}^l_{wd}(M))|L(M_{v_2})$. It remains to prove that there exists an edge connecting the two connected components C_1 and C_2. Either v is a speciation node or a duplication node. If v is a speciation node,

then for the definition of $\mathcal{R}^l_{wd}(M)$ there exists a triplet $t \in \mathcal{R}^l_{wd}(M)$ such that $t = sm(v_1)sm(v_2)|sm(v')$ and thus t induces an edge between C_1 and C_2. If v is a duplication node, there exists at least a label d such that $d \in L(M_{v_1}) \cap L(M_{v_2})$ and this label is represented in $\mathcal{G}(\mathcal{R}^l_{wd}(M))$ by a single vertex present in both C_1 and C_2. Then $\mathcal{G}(\mathcal{R}^l_{wd}(M))|L(M_v)$ is connected in both cases. This concludes the proof. □

Lemma 5 will be useful while proving Lemma 6. Let introduce the notion of closure of a compatible triplet set. Given a compatible triplet set R, we say that a triplet $ab|c$ is the closure of R, denoted by $\boldsymbol{cl(R)}$, iff $ab|c \in \mathcal{R}(T), \forall T : R \subseteq \mathcal{R}(T)$. This is equivalent to requiring that both sets $\{R \cup \{ac|b\}\}$ and $\{R \cup \{bc|a\}\}$ are incompatible [20]. We introduce a result on the closure of a triplet set that will be useful later on.

Proposition 2. *Given a compatible triplet set R, then $cl(R)$ is compatible.*

Proof. For the definition of compatibility, a triplet set R is compatible if there exists a tree T such that $R \subseteq R(T)$. From proposition 4(6)[2] of [21] we know that if such a tree exists, this tree has also the property $cl(R) \subseteq R(T)$. It follows that $cl(R)$ is compatible. □

We prove the following result.

Lemma 6. *Let M be a binary MUL tree. If the triplet set $\mathcal{R}^l_{wd}(M)$ is compatible, then $\mathcal{R}_{wd}(M) \subseteq cl(\mathcal{R}^l_{wd}(M))$.*

Proof. We prove this statement for all subtrees M_v of M by induction on the height of the node v in M. As $M = M_{root(M)}$ this shows the statement.
If $h(v) = 0$ then $\mathcal{R}_{wd}(M_v) = cl(\mathcal{R}^l_{wd}(M_v)) = \emptyset$.

Now suppose that $\mathcal{R}_{wd}(M_v) \subseteq cl(\mathcal{R}^l_{wd}(M_v))$ for all nodes v such that $h(v) < \bar{h}$ and let v be a node such that $h(v) = \bar{h}$. If v is a duplication node, then $lca(x, y, z) \in \mathcal{D}(M)$, for $x, y \in \mathcal{L}(M_{v_1})$ and $z \in \mathcal{L}(M_{v_2})$ (and the symmetric case). This implies that $\mathcal{R}_{wd}(M_v) = \mathcal{R}_{wd}(M_{v_1}) \cup \mathcal{R}_{wd}(M_{v_2})$ and $\mathcal{R}^l_{wd}(M_v) = \mathcal{R}^l_{wd}(M_{v_1}) \cup \mathcal{R}^l_{wd}(M_{v_2})$. It follows that $\mathcal{R}_{wd}(M_v) \subseteq cl(\mathcal{R}^l_{wd}(M_{v_1})) \cup cl(\mathcal{R}^l_{wd}(M_{v_2}))$ $\subseteq cl(\mathcal{R}^l_{wd}(M_{v_1}) \cup \mathcal{R}^l_{wd}(M_{v_2})) = cl(\mathcal{R}^l_{wd}(M_v))$. Note that, if $|\mathcal{L}(M_{v_1})| < 3$ (resp $|\mathcal{L}(M_{v_2})| < 3$) then $\mathcal{R}_{wd}(M_{v_1}) = \emptyset$ (resp $\mathcal{R}_{wd}(M_{v_2}) = \emptyset$) and Lemma 6 still holds.
If v is a speciation node, then by induction all triplets $l_x l_y | l_z \in \mathcal{R}_{wd}(M_v)$ with $x, y, z \in \mathcal{L}(M_{v_1})$ or $x, y, z \in \mathcal{L}(M_{v_2})$ are in $cl(\mathcal{R}^l_{wd}(M_v))$. Let t be a triplet $l_x l_y | l_z$ of $\mathcal{R}_{wd}(M_v)$ with $x, y \in \mathcal{L}(M_{v_1})$ and $z \in \mathcal{L}(M_{v_2})$. We prove that t is in $cl(\mathcal{R}^l_{wd}(M_v))$ by proving that $(\mathcal{R}^l_{wd}(M_v) \cup l_x l_z | l_y)$ and $(\mathcal{R}^l_{wd}(M_v) \cup l_y l_z | l_x)$ are both incompatible. From Lemma 5 we know that $\mathcal{G}(\mathcal{R}^l_{wd}(M_v))|L(M_{v_1})$ and $\mathcal{G}(\mathcal{R}^l_{wd}(M_v))|L(M_{v_2})$ are two connected components C_1 and C_2, since v is a speciation node above v_1 (resp v_2). As $L(M_v) = L(M_{v_1}) \cup L(M_{v_2})$, the graph

[2] Proposition 4 of [21] is defined for quartets but it remains valid for rooted triplets (see page 441).

$\mathcal{G}(\mathcal{R}^l_{wd}(M_v))$ has at most two connected components. Since $\mathcal{R}^l_{wd}(M)$ is compatible, $\mathcal{R}^l_{wd}(M_v) \subseteq \mathcal{R}^l_{wd}(M)$ is also compatible then $\mathcal{G}(\mathcal{R}^l_{wd}(M_v))$ is composed by exactly two connected components (see Theorem 2 in [21]) *i.e.*, C_1 and C_2. Since $l_x l_y | l_z \in \mathcal{R}_{wd}(M)$ then $l_x \neq l_y \neq l_z$: this means that $l_x, l_y \in C_1$, $l_x, l_y \notin C_2$ and $l_z \in C_2$, $l_z \notin C_1$. Then both triplets $l_x l_z \big| l_y$ and $l_y l_z \big| l_x$ would connect the two connected components. This implies that $(\mathcal{R}^l_{wd}(M_v) \cup l_x l_z \big| l_y)$ and $(\mathcal{R}^l_{wd}(M_v) \cup l_y l_z \big| l_x)$ are both incompatible and then t is in $cl(\mathcal{R}^l_{wd}(M_v))$. The same result holds for the symmetric case $x, y \in \mathcal{L}(M_{v_2})$ and $z \in \mathcal{L}(M_{v_1})$. Note that Lemma 6 works also if $|\mathcal{L}(M_{v_1})| = 1$ and/or $|\mathcal{L}(M_{v_2})| = 1$. This proves that $\mathcal{R}_{wd}(M) \subseteq cl(\mathcal{R}^l_{wd}(M))$. □

Lemma 7. *Let M be a binary MUL tree. If the triplet set $\mathcal{R}^l_{wd}(M)$ is compatible, then $cl(\mathcal{R}^l_{wd}(M)) = cl(\mathcal{R}_{wd}(M))$.*

Proof. Since if $\mathcal{R}^l_{wd}(M)$ is compatible then it follows from Proposition 2 that $cl(\mathcal{R}^l_{wd}(M))$ is compatible. Lemma 6 implies that $\mathcal{R}_{wd}(M)$ is also compatible. Then the closure of $\mathcal{R}_{wd}(M)$ is well defined and we can then deduce from Lemma 6 that $cl(\mathcal{R}_{wd}(M)) \subseteq cl(\mathcal{R}^l_{wd}(M))$, since $cl(cl(\mathcal{R}^l_{wd}(M))) = cl(\mathcal{R}^l_{wd}(M))$. By construction $\mathcal{R}_{wd}(M) \supseteq \mathcal{R}^l_{wd}(M)$. This implies that $cl(\mathcal{R}_{wd}(M)) \supseteq cl(\mathcal{R}^l_{wd}(M))$. This concludes the proof. □

Corollary 1. *The triplet set $\mathcal{R}^l_{wd}(M)$ is compatible iff the triplet set $\mathcal{R}_{wd}(M)$ is compatible.*

Proof. The fact that $\mathcal{R}_{wd}(M) \supseteq \mathcal{R}^l_{wd}(M)$ implies that if $\mathcal{R}_{wd}(M)$ is compatible then $\mathcal{R}^l_{wd}(M)$ is also compatible while if $\mathcal{R}^l_{wd}(M)$ is not then $\mathcal{R}_{wd}(M)$ is not compatible. While proving Lemma 7 we proved that if $\mathcal{R}^l_{wd}(M)$ is compatible then $\mathcal{R}_{wd}(M)$ is compatible. This implies that if $\mathcal{R}_{wd}(M)$ is not compatible then $\mathcal{R}^l_{wd}(M)$ is also not compatible, otherwise we would have $\mathcal{R}^l_{wd}(M)$ compatible and $\mathcal{R}_{wd}(M)$ incompatible and this would violate Lemma 7. This proves the corollary. □

Theorem 2. *Checking the auto-coherency of a binary MUL tree M can be done in $O(n \cdot \log^2 n)$ time.*

Proof. From Lemma 7 and Corollary 1 follows that checking the auto-coherency of a binary MUL tree M can be done using the triplet set $\mathcal{R}^l_{wd}(M)$. This set can be computed in linear time by algorithm 3. Given the set $\mathcal{D}(M)$ and having previously calculated $sm(v)$ for each node v, algorithm 3 computes v' for each node v in M in a top-down approach. If $v \notin \mathcal{D}(M)$ and $v' \neq \emptyset$, algorithm 3 inserts in $\mathcal{R}_{wd}(M)$ the triplet $sm(v_1)sm(v_2)|sm(v')$: this is exactly the definition of $\mathcal{R}^l_{wd}(M)$. This proves that algorithm 3 computes $\mathcal{R}^l_{wd}(M)$.

Let us demonstrate that algorithm 3 computes $\mathcal{R}^l_{wd}(M)$ in linear time. The value of $sm(v)$ for each node can be computed in a bottom-up approach requiring linear time. The set of duplication nodes $\mathcal{D}(M)$ can be also computed in linear time (see section 2.1). Algorithm LINEARREPRESENTATION$(M, \mathcal{D}(M), root(M), \mathcal{R}^l)$ consists in a postorder walk on the MUL tree M and takes a linear time. Since

AncentralBuild checks the compatibility of a triplet set R on a label set of size n in $O(|R| \cdot \log^2 n)$ time, this concludes the proof. □

Remark 1. The trees constructed by AncentralBuild from $\mathcal{R}^l_{wd}(M)$ and $\mathcal{R}_{wd}(M)$ are isomorphic.

It follows from Theorem 2 and Remark 1 that, given a MUL tree M, we can check in $O(n \cdot \log^2 n)$ time whether there exists a single-label tree T such that $\mathcal{R}^l_{wd}(M) \subseteq \mathcal{R}(T)$ and, in case of positive answer, obtain such a tree. In section 7, for building T we rely on the PhySIC heuristic algorithm [24] since this algorithm returns a tree T that represents as much as possible $\mathcal{R}^l_{wd}(M)$ while not containing at all additional, hence arbitrary, triplets.

5 Computing a Largest Duplication-free Subtree of a MUL Tree

If a MUL tree is not auto-coherent, identifying duplication nodes allows for the discrimination of leaves representing orthologous and paralogous sequences. Since only orthologous sequence history reflects the species history, a natural question is to determine the most informative sequence set for a given gene. As long as the gene tree contains odn, it will also contain leaves representing paralogous sequences. Yet, if for each node $v \in \mathcal{D}(M)$ of M we choose to keep either v_1 or v_2, we obtain a pruned single-labeled tree containing only apparent orthologous sequences (observed paralogous have been removed by pruning nodes). Note that the so obtained single-labeled tree is auto-coherent by definition.

Definition 7. *Let M be a MUL tree. We say that T is obtained by (duplication)* pruning *M iff T is obtained from M choosing for each odn v either v_1 or v_2. We denote this operation by the symbol $\lesssim$.*

One can wonder, for a non auto-coherent MUL tree M, what is the most informative single-labeled tree T *s.t.* $T \lesssim M$. We define this problem as the $MIPT$ (Most Informative Pruned Tree) problem.
To evaluate the informativeness of a tree we can use either the number of triplets of T (see [24, 25, 11]) that, for binary trees, depends only on the number of leaves, or the CIC criterion (see [26, 12]). The CIC of a not fully resolved and incomplete[3] tree T with $|L(T)|$ leaves among the n possible is a function of both the number $n_R(T, n)$ of fully resolved trees T' on $L(T)$ such that $\mathcal{R}(T) \subseteq \mathcal{R}(T')$ and the number $n_R(n)$ of fully resolved trees on n leaves. More precisely,

$$CIC(T, n) = -\log\Big(n_R(T, n)/n_R(n)\Big)$$

In the case of binary trees, $n_R(T, n)$ depends only on the number of source taxa missing in T since T does not contain multifurcations. Thus, dealing with binary trees, maximizing the information of a tree (*i.e.*, maximizing the number of triplets or minimizing the CIC value) consists in finding the tree with the

Algorithm 4: pruning(v,M,$\mathcal{D}(M)$)

Data: A node v, a MUL tree M, and a set of odn $\mathcal{D}(M)$.
Result: The most informative MUL tree M' *s.t.* $M'_v \lesssim M_v$.
foreach ($m \in sons(v)$) **do** pruning(m,M,$\mathcal{D}(M)$);
if ($v \in \mathcal{D}(M)$) **then**
  if ($|L(v_1)| > |L(v_2)|$) **then** $v \leftarrow v_1$;
  else $v \leftarrow v_2$;
  $\mathcal{D}(M) \leftarrow \mathcal{D}(M) - \{v\}$;
return M;

largest number of leaves. A natural approach for the MIPT problem for binary MUL trees is an algorithm that, after having computed $\mathcal{D}(M)$, uses the bottom-up algorithm 4 starting from root(M), to keep the most informative subtree between M_{v_1} and M_{v_2}, for each odn v.

Theorem 3. *Let M a MUL tree on a set of leaves of cardinality n. In time $O(n)$, pruning(M,root(M),$\mathcal{D}(M)$) returns the most informative tree T* s.t. *$T \lesssim M$.*

Proof. First of all, it's obvious that pruning(M,root(M),$\mathcal{D}(M)$) returns a tree. Indeed, if for each odn v only one node between v_1 and v_2 is kept, at the end of the bottom-up procedure one copy of each duplicated leaf is present in the modified M. Now, we have to show that the resulting tree is the most informative tree *s.t.* $T \lesssim M$, *i.e.*, the tree with as many leaves as possible. For an odn v that is the ancestor of other duplication nodes, the choices made for v_1 do not influence the choices for v_2 since for each duplication node we can keep only one of the two subtrees, the most populous one. Thus we can search for the best set of choices left/right for v_1 and v_2 independently and then choose the most populous pruned subtree between v_1 and v_2. Iterating recursively this reasoning, we demonstrate that the tree obtained by Algorithm 4 is the most informative tree T *s.t.* $T \lesssim M$. The computation of the set of odn $\mathcal{D}(M)$ takes linear time. The subroutine $pruning(M, root(\mathrm{M}), \mathcal{D}(M))$ requires a tree walk, thus the time complexity of Algorithm 4 is $O(n)$. □

6 The Compatibility Issue of Single-labeled Subtrees Obtained from MUL Trees

We can also ask if it is possible, given a collection of MUL tree $\mathcal{M}$, to discriminate leaves representing orthologous and paralogous sequences in a gene tree using the information contained in the other gene trees to obtain a compatible forest $\mathcal{T}$, *i.e.*, a forest for which there exists a tree T *s.t.* $\cup_{T_i \in \mathcal{T}} \mathcal{R}(T_i) \subseteq \mathcal{R}(T)$. We denote this problem by EPCF, Existence of a Pruned and Compatible Forest.

[3] A tree is called incomplete when it misses some taxa.

Unfortunately, the EPCF problem is NP-complete.

EPCF	**Instance** : A set of leaves X and a collection $\mathcal{M}=\{M_1, \cdots M_k\}$ of MUL trees on X. **Question** : $\exists$ a set S of choices left/right, $S : \mathcal{M} \to \mathcal{T}$, with $\mathcal{T}=\{T_1, \cdots T_k\}$ *s.t.* $T_i \lesssim M_i$ and $\mathcal{T}$ is compatible?

Theorem 4. *The EPCF problem is NP-complete.*

Proof. We start by proving that EPCF is in NP, *i.e.*, checking if a set S of choices left/right is a solution for the instance $\mathcal{I} = (\mathcal{M}, X)$ can be done in polynomial time. First of all, for each MUL tree $M_j \in \mathcal{M}$, we place the choices left/right on M_j, *i.e.*, we discard the subtrees not chosen, obtaining a forest of trees $\mathcal{T}$. We check then the compatibility of $\mathcal{T}$ with the Aho graph[14]. Constructing this graph can be done in polynomial time.

Given that EPCF is in NP, we use a reduction of 3-SAT to EPCF to demonstrate that it is NP-complete.

3-SAT	**Instance** : A boolean expression $\mathcal{C}=(C_1 \wedge C_2 \wedge \cdots \wedge C_n)$ on a finite set $L=\{l_1, l_2, \cdots, l_m\}$ of variables with $C_j=$ $(a \vee b \vee c)$ where $\{a, b, c\} \in \{l_1, l_2, \cdots, l_m, \overline{l_1}, \overline{l_2} \cdots, \overline{l_m}\}$ **Question** : $\exists$ a truth assignment for L that satisfies all C_j in C ?

We need to show that every instance of 3-SAT can be transformed into an instance of EPCF; then we will show that given an instance $\mathcal{I} = (\mathcal{C}, L)$ of 3-SAT, $\mathcal{I}$ is a positive instance, *i.e.*, an instance for which a solution exists, iff the corresponding instance for EPCF is positive.

Given an instance $\mathcal{I} = (\mathcal{C}, L)$ of 3-SAT, we build an instance $\mathcal{I}' = (\mathcal{M}, X)$ of EPCF associating to each l_i in L the binary tree[4] $T(l_i) = (((x_i, y_i), z_i), d)$ and to $\overline{l_i}$ the binary tree $T(\overline{l_i}) = (((z_i, y_i), x_i), d)$ (see Fig. 4 for an example). The set of subtrees $\left\{T(a) \mid a \in \{l_1, l_2, \cdots, l_m, \overline{l_1}, \overline{l_2}, \cdots, \overline{l_m}\}\right\}$ is denoted by $\mathcal{T}_L$. Then, for each clause $C_j = (a \vee b \vee c)$ in C, a binary MUL tree M_j is built, formed by three subtrees $((T(a), T(b)), T(c))$. Note that M_j has exactly two duplication nodes due to the presence of d in $T(a)$, $T(b)$ and $T(c)$, so that any left/right choice of M_j will reduce it to either $T(a)$, $T(b)$ or $T(c)$. In Fig. 5 an example of a MUL tree built from a clause. In this way we obtain a forest of MUL trees $\mathcal{M}$ on the leaf set $X = \left\{\left\{\bigcup_{i=1}^{m}\{x_i, y_i, z_i\}\right\} \cup \{d\}\right\}$, *i.e.*, an instance of the EPCF problem. Clearly $\mathcal{M}$ can be built in polynomial time.

[4] $T(l_i)$ is expressed in the Newick format.

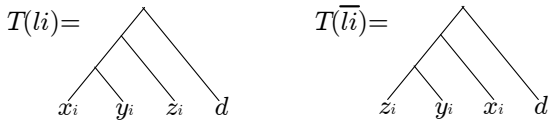

Fig. 4. Binary trees on four leaves associated to l_i and to $\overline{l_i}$

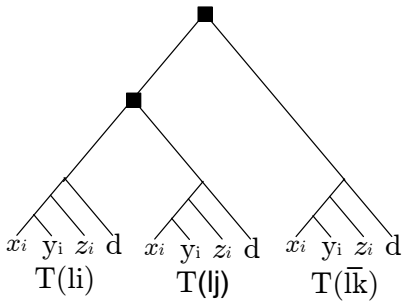

Fig. 5. MUL tree built from the clause $\{l_i \vee l_j \vee \overline{l_k}\}$. Odn are indicated by black squares

We now need to show that a positive instance of 3-SAT gives a positive instance of EPCF through the previous transformation. Having a positive instance for 3-SAT implies that for each $C_j \in \mathcal{C}$ with $C_j = (a \vee b \vee c)$, at least one of the three literals is TRUE. Without loss of generality, let us suppose that a is TRUE. Then in the MUL tree M_j corresponding to C_j we set the choice left/right so that only the subtree $T(a)$ is kept. We then obtain a forest $\mathcal{T}$ that is a subset of $\mathcal{T}_L$. We need to prove that $\mathcal{T}$ is compatible. Let $\tilde{T}(a)$ denote the tree $T(a)|(L(T(a)) - \{d\})$[5] and $\tilde{\mathcal{T}}$ the forest composed by all trees $\{\tilde{T}(a)|T(a) \in \mathcal{T}\}$. Then, we can build a tree $T_s = (\tilde{T}_1, \tilde{T}_2, \cdots, \tilde{T}_{|\tilde{\mathcal{T}}|}, d)$. Since l_i cannot have the value TRUE and FALSE at the same time, we have either $T(l_i)$ or $T(\overline{l_i})$ in $\mathcal{T}$. The tree T_s is therefore a single-labeled tree. Moreover, by construction $T_s|(L(T(a))$ is identical to $T(a)$, for all $T(a)$ in $\mathcal{T}$ ensuring that $\bigcup_{T_i \in \mathcal{T}} \mathcal{R}(T_i) \subseteq \mathcal{R}(T_s)$. Thus $\mathcal{T}$ is compatible.

Now, the only thing left to prove is that a positive instance of EPCF gives a positive instance of 3-SAT.
The repetition of the taxon d in each subtree makes the two nodes connecting the subtrees in each M_j be odn. Thus a left/right choice set S reduces each M_j in $\mathcal{M}$ into a tree $T(a) \in \mathcal{T}_L$, providing the forest $\mathcal{T}$. Setting the value of a to TRUE ensures that the clause C_j corresponding to M_j is TRUE. This can be done simultaneously for all clauses $\in \mathcal{C}$ since the forest compatibility implies that there is no contradiction among the trees in $\mathcal{T}$, all the more so direct contradictions. Then, either $T(l_i)$ or $T(\overline{l_i})$ is in $\mathcal{T}$. This ensures us that either l_i or $\overline{l_i}$ is assigned to TRUE, but not both. □

[5] Given a tree T and a label set S, we denote by $T|S$ the restriction of T to the set S.

Note that the problem to find the most informative forest $\mathcal{T} = \{T_1, \cdots T_k\}$ *s.t.* $T_i \lesssim M_i$ and $\mathcal{T}$ is compatible, denoted by MIPCF (Most Informative Pruned and Compatible Forest) is FPT. Indeed, analyzing all possible scenarios of choices left/right is FPT on the total number of duplication nodes in $\mathcal{M}$.

7 Experiments

We use here the algorithms described in this paper to analyse the HOGENOM database release 4 [27]. HOGENOM is a database of homologous genes from 381 fully sequenced genomes, containing 147,586 gene families for which alignments and trees are available. We focused on building trees at the species level, thus we only retained the 46,335 families containing taxa spanning more than two species. Other gene families concern different strains of the same few species, which can be of use when studying macro-evolutionary events but are of no use when building the species tree.

The 46,335 families span 376 species and 33,198 of these families have several sequences from the same species, their gene tree being hence a MUL-tree. This first observation shows that only 28.9% of the gene families can be used directly by supertree methods. This echoes, though less severely, the critic of [28] who called "Trees of 1%" the species trees built by the first phylogenomic works that could rely only on single-labeled trees [29, 30]. We note that as more complete genomes will be available, the percentage of MUL-trees will only increase.

In this paper, we propose fast algorithms that allow to process MUL-trees in order to distinguish and extract the speciation signal from the signal due to non-speciation events such as duplications and transfers. The significant increase in the number of gene families whose phylogenetic signal can then be used is expected to allow phylogenomic methods to obtain a more accurate picture of the estimated species trees. Targeted phylogenomic methods are both supermatrix and supertree approaches, though here we will focus on the latter due to our previous experience in the field.

7.1 Enlarging the amount of gene families to be used for species tree building

To explore the impact of our algorithms on the huge HOGENOM gene tree collection, we distinguished several sets of single-labeled gene trees:

- F_1, the forest of single-labeled gene trees;
- F_2, the forest of MUL-trees that can be turned into single-labeled trees when removing a copy of each pair of isomorphic sibling subtrees (section 3);
- F_3, the forest of MUL-trees that are still multi-labeled after applying the isomorphic simplification, but are auto-coherent (section 4). This third set of trees can be turned into single-labeled trees by two alternative ways:
  - F_3^p is the set of trees obtained from F_3 by applying the algorithm of section 5 (*i.e.*, by keeping for each duplication node, the largest subtree);

- F_3^s is the set of trees obtained when summarizing each tree T of F_3 by another tree containing only its speciation signal. This is done by first computing the linear triplet decomposition $R^l_{wd}(T)$ of the tree, then obtaining a tree T' that represents as much as possible this set of triplets while not containing at all additional, hence arbitrary, triplets. For building T' we rely on the PhySIC heuristic algorithm [24].

Note that F_1, F_2 and F_3 correspond to mutually exclusive sets of HOGENOM families, while F_3^p and F_3^s are composed of alternative single-labeled trees that correspond to same families. Then we considered the largest datasets that can be composed by combining these forests, *i.e.*, $F^s_{all} = F_1 \cup F_2 \cup F_3^s$ and $F^p_{all} = F_1 \cup F_2 \cup F_3^p$. All these forests are all composed of single-labeled trees, thus can be analyzed by supertree methods to produce species trees. For this purpose, the most informative forest is obviously the union of F_1, F_2 with either F_3^p or F_3^s. Note that F_3^s and F_3^p cannot be used at the same time, which would bias the supertree inference toward the phylogenetic signal contained in families of F_3.

	F_1	F_2	F_3^s	F_3^p	F^s_{all}	F^p_{all}
# trees	13,378	11,891	17,674	16,148	42,943	41,417
# triplets in total	151,287	2×10^6	421×10^6	424×10^6	423×10^6	426×10^6
# triplets/tree on avg.	11	169	23,819	26,261	18,472	10,291
# distinct triplets	68,538	601,429	22.9×10^6	22.2×10^6	22.9×10^5	22.3×10^6
% of input triplets	0.3%	2.3%	86.8%	84.4%	86.9 %	84.4%

Table 1. Information contained in the six considered forests to build the species tree for the 376 species present in HOGENOM. The first row reports the number of trees in each forest, while other rows give indications on the amount of information contained in the forests in terms of triplets. Considered triplets are *speciation triplets* as defined earlier in this paper. The second row reports the total number of triplets (with repetitions) for each forest (*i.e.*, the sum of $|R_{wd}(M_i)|$ for all MUL-trees M_i in the forest). The third row deduces from it the average number of triplets induced by a tree on average. The fourth row displays the number of distinct triplets, *i.e.*, when not considering the fact that some triplets are found several times. The fifth row details the percentage of speciation triplets available as input to the methods in proportion of the number of possible triplets for building a supertree of that size.

We first report on characteristics of the forests detailed above (see Table 1). This allows to measure the phylogenetic signal contained in each part of the initial tree collection and the gain obtained by the possible enlargements of the F_1 forest. This is measured here both in number of trees in the forests and number of triplets they contain. To that aim, we report sizes of R_{wd} sets, rather than that of R^l_{wd} sets, because this gives a more extent idea of the information contained in the collections.

From the number of trees in the different collections displayed in Table 1, it can be observed that the algorithms proposed in this paper allow to use up to 43k gene families instead of the 13k trees corresponding to orthologous genes with no detected paralogs. These 43k trees represent more than 90% of HOGENOM gene families, *i.e.*, more than three times the number of gene families that can be used in phylogenomic studies.

What is even more impressive is the gain in the amount of topological information for building the species tree. Indeed, from the second and third row of the table, it can be seen that trees in F_1 include on average few species. This is due to the fact that most of the large trees contain duplication nodes. Indeed, widening the scope of considered species for a same family increases the probability of observing duplicated sequences. This is particularly true for some species that are known to have undergone ancient duplications of their whole genome. Taking the presence of duplications into account, even in a very simple way as done to obtain F_2, allows a significant increase in the expressed phylogenetic signal. Indeed, though F_2 contains roughtly the same number of trees than F_1, it contains 10 times more speciation triplets. However, as F_2 only allows identical resolution of duplicated sequences, most trees translating the presence of several duplication and/or transfer events can only be represented in the F_3 forests. The table shows that the more refined analyses conducted to compose F_3^p and F_3^s lead to a very significant increase in the number of speciation information extracted (about 2,000 times more speciation triplets than F_1 and 300 times more distinct speciation triplets).

Moreover, the increase of the additionally available information covers harmoniously the set of all possible triplets, as the number of distinct triplets for which the input forest contains a resolution goes from 68.5k to almost 23 millions. In terms of percentage of information available to build a species tree, the last row of Table 1 shows that the critic of Bapteste et al [28] was well founded since less than 1% triplets of all possible triplets are contained in the F_1 forest. In contrast, this increases up to 86.9% in the best case that we can now consider (forest F_{all}^s).

7.2 Running times

All algorithms have been implemented in C++. In table 7.2 we report the running times of the algorithms presented in Sections 3-5 on the HOGENOM data base on a Linux-based machine running with 3 GHz processor and 4 GB RAM.

7.3 Building supertrees

It now remains to be seen whether the increase in the amount of available information benefits the species tree construction step, *i.e.*, whether the extracted information is of good quality. This is the question we now address. To build supertrees, we composed several datasets from the above forests: the four forests F_1, F_2, F_3^s, F_3^p were each considered separately, then we considered the two largest forests that could be composed from these basic ones, namely F_{all}^s

	applied algorithms	running time
obtaining F_1	checking if $\mathcal{D}(M) \neq \emptyset$ (algorithm 1)	2m20s
obtaining F_2	algorithm 2	5m1s
obtaining F_3	AncestralBuild algorithm	14m40s
obtaining F_3^p	algorithm 4	0m14s
obtaining F_3^s	PhySIC algorithm	21m14s

Table 2. Running times of the algorithms presented in Sections 3- 5 on the HOGENOM gene tree collection.

and F_{all}^p. Two supertree methods were considered: the well-known MRP method [10] and the more recent PhySIC_IST method [12]. The two methods differ in the way they deal with contradictory topological signals found in the source trees. MRP is a *voting* method, *i.e.*, arbitrating between conflicting signals in favor of the most frequent one, that relies on the maximum parsimony criterion. In contrast, PhySIC_IST is a method merely built from a *veto* principle, *i.e.*, allowing pieces of contradictory signal to vote against resolution of some branches in the supertree. As a result, PhySIC_IST infers more reliable but less resolved supertrees. This veto behavior can be tempered by removing the less frequent triplets from the input trees. This preprocess is regulated by a parameter called STC (*source tree correction*), for which we used different values in our experiments: 0.9, 0.8, 0.5, ordered by increasing tolerance to contradictory signal.

	F_1	F_2	F_3^s	F_3^p	F_{all}^s	F_{all}^p
CIC of PhySIC_IST (0.9)	2%	12%	48%	46%	47%	44%
# species PhySIC_IST (0.9)	22	67	204	198	200	189
CIC of PhySIC_IST (0.8)	3%	16%	59%	54%	57%	51%
# species PhySIC_IST (0.8)	22	81	241	225	234	213
CIC of PhySIC_IST (0.5)	3%	19%	81%	79%	60%	61%
# species PhySIC_IST (0.5)	23	96	323	318	246	248
CIC of MRP supertree	N/A	N/A	98.01%	99.90%	99.73%	99.95%
# of most pars. trees for MRP	N/A	N/A	510	2	4	1

Table 3. Characteristics of the supertrees built by MRP and PhySIC_IST from investigated forests. CIC values (*i.e.*, resolution degree [12]) of the inferred supertrees are detailed, as well as the number of species in the supertrees for the non-complementary PhySIC_IST method. The latter method was run for three different values of its STC threshold (ie. contradiction intolerance, see main text): 0.5, 0.8 and 0.9. Last row details the number of most parsimonious trees found by MRP in each case.

A first general observation is that, as expected, the resolution degree (CIC value) of the supertrees proposed by all methods increases as the amount of

available information increases. This is shown by the first line of Table 3 when going from F_1 to F_2 and from F_2 to F_3 forests. When going from F_3 forests to the corresponding F_{all} ones, the MRP method follows again the same tendency, while the PhySIC_IST method does not. This is however explained by an increase in the level of contradictory signal present in the information that PhySIC_IST extracts from the forests when going from F_3^s to F_{all}^s and similarly from F_3^p to F_{all}^p (data not shown).

We first analyze results of the MRP method. On datasets F_1 and F_2, the method was interrupted after a week computation. Most probably, the method couldn't give any supertree in these cases[6] due to too poor phylogenetic signal contained in the forests (as can be checked in Table 1). As a result, the parsimony criterion could not distinguish between candidate supertrees due to a huge number of most parsimonious trees. Other datasets did not suffer from this problem as they contained several thousand times more signal. However, even for the relatively large datasets F_3^p and F_3^s, the parsimony analysis found several most parsimonious trees. The number of most parsimonious tree was always reduced when completing these forests with the relatively small F_1 and F_2 forests (*i.e.*, datasets F_{all}^s and F_{all}^p). This shows how important it is to use every possible bit of information that can be extracted from the data when dealing with such large phylogenies spanning the origins of life.

When observing the structure of the inferred supertrees, for all datasets it can be observed that super-kingdoms are respected up to 5 taxa over the 376 considered: Archaea and Eukaryotes are monophyletic, while Bacteria are splitted into several paraphyletic groups. Moreover, the number of badly placed species always decreases when going from F_3^p, F_3^s to F_{all}^p, F_{all}^s forests, again showing the interest in using all possibly available information.

Problematic species are the following:

- the *Candidatus Carsonella ruddii* bacteria groups with Archaea for the F_3^s dataset and within Eukaryotes with F_{all}^s. It is however placed just outside Eukaryotes in other datasets;
- the *Encephalitozoon cuniculi* (a.k.a. *microsporidians*) eukaryote groups with bacteria when building supertrees from F_3^s and F_3^p, however it correctly goes to the root of eukaryotes when resorting to F_{all}^s and F_{all}^p;
- the *Guillardia theta* eukaryote behaves like *Encephalitozoon cuniculi* except that it is correctly placed only when using F_{all}^s. This species is well known to be problematic from a phylogenetic point of view, as it results from a long branch.
- the two bacteria *Aquiflex Aeolicus* and *Thermotoga* branch from a polytomous node at the root of archaea when analyzing F_{all}^s but are within bacteria for other datasets. It is believed that these taxa are indeed the closest bacteria from archaea (e.g., [31]).

The fact that bacteria are paraphyletic could be due to several effects. Firstly, perturbations introduced by an incorrect rooting of gene trees in general: the

[6] even when asked to restrict to a small number of most parsimonious trees

midpoint rooting procedure was used in HOGENOM without manual curation. Second, it is has been established that some genes in eukaryotes have an endosymbiotic origin: mitochondria from an alpha proteobacteria and plastids from cyanobacteria [32, 33]. Thus, it is likely such eukaryotic genes vote for an incorrect placement of eukaryotes inside bacteria, making the latter paraphyletic. Once again a deeper analysis is of these effects is needed and we postpone it to a further study.

Nonetheless, species from the three super-kingdoms are overall well separated in inferred supertrees. This shows the good quality of the speciation information that we extracted from HOGENOM multigene families thanks algorithms presented here. That is, not only one can now extract more phylogenetic signal from phylogenomic databases, but this signal seems to be useful to build species trees. The next step is looking into details of the changes induced in the species tree inferred when going from F_3^p, resp. F_3^s to F_{all}^p, resp. F_{all}^s, but this deeper analysis is beyond the scope of the current paper.

The results obtained by the PhySIC_IST supertree method are complementary to those obtained by MRP. Overall, the supertrees output by PhySIC_IST are less resolved (as can be observed by CIC values of Table 1, but more correct phylogenies seem to be inferred in return as far as our analysis went, *i.e.*, mostly looking at the separation between eukaryotes, bacteria and archaea. In all inferences from F_1, $F_3^s, F_3^p, F_{all}^s, F_{all}^p$, eukaryotes were always monophyletic, as well as archaea. Bacteria were monophyletic in 13 of these trees, while one group of bacteria went to the root of the tree for the dataset F_{all}^s analysed with threshold 0.8 and one group of bacteria went to the root of the archaea in the supertree inferred from F_3^s with threshold 0.8. Supertrees proposed from forest F_2 are form a less idyllic picture, since we observe the same problems as for MRP supertrees, *i.e.*, several bacteria branching into the eukaryotic group.

We note that the smaller CIC values obtained by PhySIC_IST in comparison to MRP is almost exclusively explained by the fact that some taxa species are not inserted, *i.e.*, the PhySIC_IST supertree contain very few polytomies (unresolved nodes), most trees being binary. This goes to an extreme for the smallest forest, where PhySIC_IST supertrees contain less than 10% species, and only eukaryotes. This indicates that the method finds the positioning of bacteria and archaea too difficult given the small amount of information available in F_1. Recall also that MRP could not terminate for this forest. The supertrees proposed by PhySIC_IST in this case conforms mostly to what is known on eukaryotes, e.g. encoded in the NCBI taxonomy. The two differences are *Encephalitozoon cuniculi* going to the root of the eukaryotes, and the group composed of *Leishmania major* and *Trypanosoma brucei* that goes into the *coelomata* group instead of being at the root of eukaryotes. Recall that the eukaryote *Encephalitozoon cuniculi* is a problematic species for MRP. As an improvement, PhySIC_IST places it most often at the basis of the eukaryotic group, and not among bacteria. Though, the acknowledged position for this taxa is deeper in the eukaryotes. All in all, this confirms the hypothesis of a problematic positioning of this taxa in the HOGENOM gene trees.

In contrast to what happens for F_1, supertrees inferred from other forests contain species from the three super kindgoms, most usually well-separated as indicated above. Lastly, we note that the resolution proposed by PhySIC_IST supertrees for these groups oscillates between the two possible topologies, *i.e.*, the two grouped ones being different depending on the forests, and sometimes also depending on the STC thresholds used. This confirms that contradictory signal exist in HOGENOM data for deciding how to root the Tree of Life, likely due to a too crude rooting procedure of the gene trees.

8 Conclusions

In this paper we presented several algorithms to transform multi-labeled evolutionary trees into single-labeled ones so that they can be used by all existent supertree methods. We studied the impact of these algorithms on a phylogenomic database. Results showed that not only these algorithms allow to extract more information, but that supertrees inferred from this extra information are much more resolved and, at a first rough level of analysis, globally in accordance to phylogenetic knowledge. Moreover, the algorithms shown very inexpensive running times, e.g. processing several million trees in a few minutes for some of them.

Future work includes a more thorough analysis of the inferred supertrees, *i.e.*, to look at the proposed phylogeny for major bacterial groups. However, this could only be done after refining the rooting procedure applied to HOGENOM gene trees. Further theoretical developments needs to be done to obtain more involved FPT algorithms to solve the MIPCF problem.

Acknowledgment

This work was funded by the ANR-08-EMER-011 project.

References

1. Cotton, J., Page, R.: Rates and patterns of gene duplication and loss in the human genome. In of London. Biological science, R.S., ed.: Proceedings. Volume 272. (2005) 277–283
2. Ma, B., Li, M., Zhang, L.: From gene trees to species trees. SIAM J. Comput **30** (2000) 729—752
3. Hallett, M.T., Lagergren, J.: New algorithms for the duplication-loss model. In: RECOMB2000, Fourth Annual International Conference on Computational Molecular Biology. (2000) 138–146
4. Chen, K., Durand, D., Farach-Colton, M.: Notung: a program for dating gene duplications and optimizing gene family trees. J Comput Biol **7** (2000) 429—444
5. Vernot, B., Stolzer, M., Goldman, A., Durand, D.: Reconciliation with non-binary species trees. J Comput Biol **15**(8) (2008) 981—1006

6. Chauve, C., Doyon, J.P., El-Mabrouk, N.: Gene family evolution by duplication, speciation, and loss. J Comput Biol **15**(8) (2008) 1043—1062
7. Goodman, M., Czelusniak, J., Moore, G., A.E., R.H., Matsuda, G.: Fitting the Gene Lineage into its Species Lineage, a Parsimony Strategy Illustrated by Cladograms Constructed from Globin Sequences. Systematic Zoology **28**(2) (1979) 132—163
8. Arvestad, L., Berglund, A., Lagergren, J., Sennblad, B.: Bayesian gene/species tree reconciliation and orthology analysis using MCMC. Bioinformatics **19 Suppl 1** (2003) 7—15
9. Durand, D., Halldórsson, B., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. J. Comput. Biol. **13** (Mar 2006) 320—335
10. Baum, B.R., Ragan, M.A.: The MRP method. In Bininda-Emonds, O., ed.: Phylogenetic supertrees: combining information to reveal the Tree of Life. Kluwer (2004) 17–34
11. Page, R.D.M.: Modified mincut supertrees. In Guigó, R., Gusfield, D., eds.: Proceedings of the 2nd International Workshop on Algorithms in Bioinformatics (WABI'02). (2002) 537–552
12. Scornavacca, C., Berry, V., Lefort, V., Douzery, E.J.P., Ranwez, V.: Physic_ist: cleaning source trees to infer more informative supertrees. BMC Bioinformatics **9**(8) (2008) 413
13. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks **21** (1991) 12–28
14. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comp. **10**(3) (1981) 405–421
15. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. **13**(2) (1984) 338–355
16. Bender, M.A., Farach-Colton, M.: The lca problem revisited. In Springer-Verlag, ed.: LATIN '00: Proceedings of the 4th Latin American Symposium on Theoretical Informatics. (2000) 88–94
17. Berry, V., Semple, C.: Fast computation of supertrees for compatible phylogenies with nested taxa. Syst. Biol. **55** (2006) 270–288
18. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification **9**(1) (January 1992) 91–116
19. Semple, C., Steel, M.A.: Phylogenetics. Volume 24 of Oxford Lecture Series in Mathematics and its Applications. Oxford University Press (2003)
20. Grunewald, S., Steel, M.A., Swenson, M.S.: Closure operations in phylogenetics. Mathematical Biosciences **208**(2) (2007) 521–37
21. Bryant, D., Steel, M.: Extension operations on sets of leaf-labelled trees. Adv. Appl. Math. **16**(4) (1995) 425–453
22. Berry, V., Nicolas, F.: Improved parameterized complexity of the maximum agreement subtree and maximum compatible tree problems. IEEE/ACM Trans. Comput. Biol. Bioinformatics **3**(3) (2006) 289–302
23. Aho, A., Hopcroft, J., Ullman, J.: The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1974)
24. Ranwez, V., Berry, V., Criscuolo, A., Fabre, P., Guillemot, S., Scornavacca, C., Douzery, E.: PhySIC: a veto supertree method with desirable properties. Syst. Biol. **56** (Oct 2007) 798–817
25. Semple, C., Steel, M.: A supertree method for rooted trees. Discrete Appl. Math. **105** (2000) 147–158

26. Thorley, J., Wilkinson, M., Charleston, M.: The information content of consensus trees. In Rizzi, A., Vichi, M., Bock, H.H., eds.: Advances in Data Science and Classification. Studies in Classification, Data Analysis, and Knowledge Organization. (1998) 91–98
27. Penel, S., Arigon, A.M., Dufayard, J.F., Sertier, A.S., Daubin, V., Duret, L., Gouy, M., Perrière, G.: Databases of homologous gene families for comparative genomics. BMC Bioinformatics **10 Suppl 6** (2009) S3
28. Bapteste, E., Susko, E., Leigh, J., Ruiz-Trillo, I., Bucknam, J., Doolittle, W.F.: Alternative methods for concatenation of core genes indicate a lack of resolution in deep nodes of the prokaryotic phylogeny. Mol Biol Evol **25**(1) (Jan 2008) 83–91
29. Brochier, C., Forterre, P., Gribaldo, S.: An emerging phylogenetic core of archaea: phylogenies of transcription and translation machineries converge following addition of new genome sequences. BMC Evol Biol **5**(1) (2005) 36–36
30. Ciccarelli, F.D., Doerks, T., von Mering, C., Creevey, C.J., Snel, B., Bork, P.: Toward automatic reconstruction of a highly resolved tree of life. Science **311**(5765) (Mar 2006) 1283–1287
31. Henz, S.R., Huson, D.H., Auch, A.F., Nieselt-Struwe, K., Schuster, S.C.: Whole-genome prokaryotic phylogeny. Bioinformatics **21**(10) (2005) 2329–2335
32. Gray, M.W.: The endosymbiont hypothesis revisited. Int.Rev.Cytol. **141**(233-357) (1992)
33. Margulis, L.: Symbiosis in Cell Evolution. W.H. Freeman and Company, New-York (1993)