Running head: Desirable properties for veto supertrees.

PhySIC: a Veto Supertree Method with Desirable Properties

Vincent Ranwez[1,*], Vincent Berry[2], Alexis Criscuolo[1,2], Pierre-Henri Fabre[1], Sylvain Guillemot[2], Celine Scornavacca[1,2] and Emmanuel J. P. Douzery[1]

[1] *Institut des Sciences de l'Evolution (ISEM, UMR 5554 CNRS), Université Montpellier II, Place E. Bataillon - CC 064 - 34095 Montpellier Cedex 5, France*

{ranwez,fabreph,douzery}@isem.univ-montp2.fr

tel: 00 33 4 67 14 36 97

fax: 00 33 4 67 14 36 10

[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*

{vberry,criscuol,sguillem,scornava}@lirmm.fr

[*] corresponding author.

Abstract

This paper focuses on veto supertree methods, i.e., methods that aim at producing a conservative synthesis of the relationships agreed upon by all source trees. We propose desirable properties that a supertree should satisfy in this framework, namely the *non-contradiction* property (PC) and the *induction* property (PI). The former requires that the supertree does not contain relationships that contradict one or a combination of the source topologies, while the latter requires that all topological information contained in the supertree is present in a source tree or collectively induced by several source trees. We provide simple examples to illustrate their relevance and that allow a comparison with previously advocated properties. We show that these properties can be checked in polynomial time for any given rooted supertree. Moreover, we introduce the *PhySIC* method (PHYlogenetic Signal with Induction and non-Contradiction). For k input trees spanning a set of n taxa, this method produces a supertree that satisfies the above-mentioned properties in $O(kn^3 + n^4)$ polynomial computing time. The polytomies of the produced supertree are also tagged by labels indicating areas of conflict as well as those with insufficient overlap. As a whole, *PhySIC* enables the user to quickly summarize consensual information of a set of trees and localize groups of taxa for which the data requires consolidation. Lastly, we illustrate the behaviour of *PhySIC* on primate datasets of various sizes, and propose a supertree covering 95% of all primate extant genera. The *PhySIC* algorithm is available at `http://atgc.lirmm.fr/cgi-bin/PhySIC/physic.cgi`.

Introduction

Building supertrees

Phylogenies are invaluable tools in various areas of biology to understand the evolution of genes and taxa. Trees that incorporate an exhaustive sampling of taxonomic biodiversity provide crucial information about systematics, genomics, and diversification patterns of species (e.g., Davies et al., 2004). Large trees can be built using various approaches, including supermatrices and supertrees. The former approach consists of combining the different source datasets into a *supermatrix* of characters, and then analyzing it under standard phylogenetic reconstruction criteria (e.g., Delsuc et al., 2005). The supertree approach is an alternative methodology using trees rather than character data as a primary source of information. It first involves inferring smaller, partially overlapping, source phylogenetic trees from initial character data, and then assembling them into a larger, more comprehensive *supertree* (Bininda-Emonds, 2004a). This approach is particularly convenient when dealing with heterogeneous character sources, e.g., those scored from morphological, transposable elements, DNA, or protein studies. Supertrees have become increasingly popular (e.g., Bininda-Emonds, 2004b), notably since the seminal work involving the reconstruction of the primate supertree (Purvis, 1995a). The widespread use of supertrees is explained by three useful applications (Wilkinson et al., 2004): (i) they provide large phylogenetic frameworks for broad comparative studies; (ii) they evaluate the congruence of sets of input trees, and reveal conflicts due to outlier/unstable taxa; and (iii) they identify insufficient overlap among leaf sets of input trees, and assign priorities for choosing the taxa to be subsequently sampled.

Different kinds of supertree methods

Supertree methods fall into three categories depending on their way of handling topological conflicts, i.e., different arrangements of the same leaves among labeled source trees.

The first suite of methods do not handle incompatible source trees. The pioneering methods that belong to this category are *Build* (Aho et al., 1981) and the strict consensus supertree (Gordon, 1986). Although they are important milestones, these methods appear "of limited use. As most systematics know, phylogenies usually conflict with one another" (Bininda-Emonds, 2004b, p4).

The second suite of methods handle conflicts among input trees in a liberal way: they apply a *voting* procedure. In order to extract their main phylogenetic signal, source trees are asked to vote on various parts of the phylogeny to be inferred, with the most supported candidates being elected and composing the output supertree. Voting methods are said to *resolve* conflicts (Thorley and Wilkinson, 2003): for each conflict, they use some optimization criterion to make a decision in favor of one of the topological alternatives. Most conflicts among input trees are expected to be resolved because relationships displayed by the supertree are guided by source topologies on the basis of weight of evidence. The most widespread voting method is Matrix Representation with Parsimony (MRP) whereby nodes of each source tree are encoded as binary characters of a matrix that is then analyzed with the maximum parsimony criterion to obtain the composite tree (Baum, 1992; Ragan, 1992). Analyzing this binary encoding of source topological information with other tree-building criteria leads to variants of MRP such as Matrix Representation with Flipping (MRF; Chen et al., 2003) and Matrix Representation with Compatibility (MRC; Ross and Rodrigo, 2004). Other methods of the voting kind, such as MinCut (MC; Semple and Steel, 2000) and ModifiedMinCut (MMC; Page, 2002) extend *Build*. They encode source trees in a graph that is progressively decomposed to get supertree clades. When conflicts hinder the decomposition,

the graph is cut by removing the least supported relationships. The Average Consensus (Lapointe and Cucumel, 1997) and Super Distance Matrix (Criscuolo et al., 2006) methods implement the voting approach in an alternative way. They average the initial distance matrices, converted from source characters or valued topologies, into a *superdistance* matrix; a tree-building distance-based approach is then used to infer a supertree from the matrix. Interestingly, voting methods like MRP may generate *novel* clades, i.e., clades not present in any input tree alone (Purvis, 1995b; Bininda-Emonds and Bryant, 1998; Sanderson et al., 1998). Unfortunately, when source trees conflict, novel clades that are contradicted by each of the source trees can be present in the supertree inferred by MRP (Goloboff and Pol, 2002; Goloboff, 2005; Cotton et al., 2006) and by MRF (Goloboff, 2005). The importance of this phenomenon is still debated, Bininda-Emonds (2003) reporting, on the basis of simulations, that this situation is not very frequent for MRP, while Goloboff (2005) shows selected case studies where "this situation is, clearly, not very unlikely".

The third suite of methods handle conflicts among input trees in a conservative way. They adopt a *veto* philosophy: the phylogenetic information of every source topology is to be respected, and the supertree is not allowed to contain any group that a source tree would vote against. These methods *remove* conflicts (Thorley and Wilkinson, 2003) because they either propose multifurcations in the supertree (Goloboff and Pol, 2002), or prune rogue taxa (Berry and Nicolas, 2004). In this framework, the supertree should not retain a single branching pattern within a given clade when several valid topological alternatives are present in the source trees. The full agreement required by veto methods provides an unambiguous phylogenetic framework that is, for instance, well suited for taxonomic revisions. More specifically, such a conservative approach may be applied to automatically build or update parts of the Tree of Life (`http://tolweb.org`). Several supertree methods akin to the veto philosophy have been proposed, all of which are inspired by consensus approaches that operate on trees with identical leaf sets. For example, extensions of

the strict consensus (Gordon, 1986; Huson et al., 1999), semi-strict consensus (Goloboff and Pol, 2002), and maximum agreement subtree consensus (Berry and Nicolas, 2004) have been proposed to infer veto supertrees.

Properties of supertree methods

To assess the relevance of supertree methods, it is most useful to have properties characterizing the extent to which the supertrees they infer are reliable syntheses of source trees (Bininda-Emonds and Bryant, 1998; Steel et al., 2000; Wilkinson et al., 2004; Goloboff, 2005). For instance, Steel et al. (2000) suggest that the output supertree should (i) encompass every source tree when possible, (ii) always contain every leaf (taxon) that occurs in at least one source tree, and (iii) be computed under a running time that grows polynomially with respect to the total number of leaves. These authors also showed that rooted input trees are more appealing than unrooted ones for supertree methods that aim to satisfy several desirable properties simultaneously. Yet, even if supertree methods satisfy some desirable properties, the inferred supertrees often contain polytomies which actually intermix two distinct phenomena: either a lack of overlap in the topological information among source trees, or the occurrence of topological conflicts among them, or a combination of these. We thus decided to develop a method that proposes supertrees with unambiguous resolutions, and provides biologists with explanations about causes of polytomies. For this purpose, we rely on two new formal properties.

On the one hand, we think that supertree methods should avoid arbitrary resolutions, i.e., resolutions that are not entailed by the source topologies. Indeed, novel relationships displayed by a supertree "are worrying if they are not implied by combinations of the input trees" (Wilkinson et al., 2005), and "should be identified as such, to highlight their lack of any known justification"

(Pisani and Wilkinson, 2002). Thus, we first request that every piece of phylogenetic information displayed in the supertree be present in one or several source topologies, or be induced by their interaction; we call this the *induction* property.

On the other hand, we focus on unanimous clades, thus adopting a veto point of view. This means that the supertree is not allowed to contain a clade that conflicts either directly with a source tree or indirectly with a combination of them. We call this the *non-contradiction* property. Such a supertree, that incorporates only uncontradicted input relationships, provides a reliable baseline for subsequent analyses (Goloboff and Pol, 2002; Goloboff, 2005).

Goloboff and Pol (2002) mentioned similar properties in a formal characterization involving triplets. They provide examples showing that supertree methods of the voting kind, such as MRP, MC, understandably do not respect these properties. Although being appealing, the characterization proposed by Goloboff and Pol (2002) can at times be too restrictive or permissive (see following sections). Recently, Grunewald et al. (2006) provided another characterization of a property related to arbitrary resolutions contained in the supertree with respect to source trees. In both cases, there does not seem to be any straightforward algorithm that would always allow for property verification. Note, however, that Goloboff and Pol (2002) proposed a supertree heuristic algorithm that satisfies the desired properties in most cases.

In this paper, we provide a characterization of non-contradiction and induction properties, that differ from those of Goloboff and Pol (2002) and Grunewald et al. (2006). We also describe simple and polynomial-time algorithms that enable users to check whether or not a given supertree satisfies these properties. Then we propose an algorithm called *PhySIC* that improves the *Build* algorithm (Aho et al., 1981) by always inferring a supertree and which, moreover, satisfies the non-contradiction and induction properties. As far as we know, this is the first time that a polynomial-time method is proposed that always satisfies properties related to induction and non-

contradiction. Moreover, improving the behavior of *Build* with respect to arbitrary decisions can benefit the various methods that extend this algorithm for supertree purposes, e.g., MC (Semple and Steel, 2000), MCC (Page, 2002), *AncestralBuild* (Daniel and Semple, 2004; Berry and Semple, 2006) and *RankedTree* (Bryant et al., 2004). Next, we pinpoint the difference between the behaviour of *PhySIC* and that of well-known supertree methods on a biological case study on Primates. Lastly, we illustrate *PhySIC* on the reconstruction of the primate supertree at the genus level from various source trees based on mitochondrial DNA, nuclear DNA and jumping gene sequences. The supertree reconstructed appears to be useful for displaying phylogenetic relationships among the major primate taxa (Goodman et al., 2005). Moreover, the produced supertree displays label(s) on each of its polytomous nodes which identifies the cause(s) of these polytomies (lack of cross-information and/or presence of contradictions). The *PhySIC* method has been implemented in `C++` using the `Bio++` library (Dutheil et al., 2006) and is freely available as a web service and for download at `http://atgc.lirmm.fr/cgi-bin/PhySIC/physic.cgi`.

Non-contradiction and induction properties

We first introduce vocabulary and notations required to formally define the properties of *non-contradiction* (PC) and of *induction* (PI). Simple examples are then used to illustrate the relevance of PC and PI as well as to relate them with previously proposed properties for supertree methods (Steel et al., 2000; Goloboff and Pol, 2002). Then we show how to check in polynomial time whether a supertree satisfies PC and PI for a given collection of source trees.

Topological description of trees

The definitions and notations used for trees and their topological description are mainly the same as those used by Semple and Steel (2003). We only consider rooted phylogenies, due to the fact that supertree methods cannot fulfill different desirable properties listed in Steel et al. (2000) when considering unrooted trees. Hereafter, the terms *phylogeny* and *tree* are considered synonymous. Given a tree T, $L(T)$ denotes the set of taxa associated to its leaves. More generally, given a collection $\mathcal{T}$ of trees, $L(\mathcal{T})$ denotes the set of taxa appearing in at least one tree of $\mathcal{T}$. Given two phylogenies T and T' on the same leaf set ($L(T) = L(T')$), we say that T *refines* T' whenever T contains all clades of T'. In other words, either T and T' are identical or T can be transformed into T' by collapsing some of its internal edges.

A rooted tree on three leaves A, B, C has only three possible binary shapes, called *triplets* and denoted by $AB|C$, resp. $AC|B$, resp. $BC|A$, depending on the innermost clade (AB, resp. AC, resp BC). Given a triplet t, $\bar{t}$ denotes any of the two other triplets on the same set of leaves. Alternatively, a tree on three leaves can be a star tree, i.e., a unique internal node connected to the leaves. Any rooted tree T can be equivalently described by the set of triplets homeomorphic to subtrees of T connecting three leaves (e.g., Grunewald et al., 2006), $rt(T)$ denotes this set. Given a collection $\mathcal{T}$ of phylogenies, $rt(\mathcal{T}) = \bigcup_{T_i \in \mathcal{T}} rt(T_i)$ denotes the set of triplets present in these phylogenies. Note that it is possible that $rt(\mathcal{T})$ contains two triplets t and $\bar{t}$, namely when $\mathcal{T}$ hosts two incompatible phylogenies. Clearly, two such triplets cannot be combined into a single supertree of the collection.

Given a set $\mathcal{R}$ of triplets, $L(\mathcal{R})$ denotes the set of taxa appearing in at least one tree in $\mathcal{R}$. A tree T is said to *display* a set $\mathcal{R}$ of triplets when $\mathcal{R} \subseteq rt(T)$; moreover, T *strictly displays* $\mathcal{R}$ if additionally $L(T) = L(\mathcal{R})$. A set $\mathcal{R}$ of triplets is *compatible* if there is a tree T that displays $\mathcal{R}$. To

find a tree displaying $\mathcal{R}$, it is useful to take into account that some triplets of the tree are *induced* by $\mathcal{R}$: a compatible set $\mathcal{R}$ of triplets *induces* a triplet t, denoted by $\mathcal{R} \vdash t$, if and only if $\mathcal{R} \cup \{\bar{t}\}$ is not compatible, or equivalently if any tree T that displays $\mathcal{R}$ contains t. For instance, any tree displaying $\{AB|C, BC|D\}$ also has to display the triplet $AC|D$, i.e., $\{AB|C, BC|D\} \vdash AC|D$. Bandelt and Dress (1986) and Dekker (1986) were among the first to investigate such induction rules. The set of all triplets induced by a compatible set $\mathcal{R}$ is called the *closure* of $\mathcal{R}$ and is denoted by $cl(\mathcal{R})$. Source trees considered for supertree building are sometimes incompatible, and then the set of triplets considered is incompatible. Nonetheless, we can characterize the set of triplets induced by these collections by extending the preceding definition: we will say that a set $\mathcal{R}$ of triplets *induces* a triplet t when there is a compatible subset $\mathcal{R}'$ of $\mathcal{R}$ that induces t.

Characterizing non-contradiction and induction by triplets

Here we describe two important properties that veto method supertrees should satisfy. They concern topological relationships that a supertree should not contain with respect to the input trees : first, it should not contain relationships *contradicting* the source trees (PC property); moreover, it should only contain relationships that are *induced* by the input trees (PI property). Below we detail these two properties.

There are several ways for a supertree to contradict a collection of source trees. The most direct contradiction occurs when only one resolution appears for a group of taxa in one or several source trees, and the supertree contains a different resolution for the group. When different resolutions appear in source trees, as soon as the supertree proposes a resolution for the concerned taxa it contradicts at least one input tree. Contradictions are less direct when the supertree proposes a resolution that contradicts no single input tree but does contradict a combination of

them.

A relationship contained in a supertree can present no contradiction with the input trees and still not be desirable. For instance, Fig. 1 shows a collection of two source trees and four possible supertrees, T', T'', T''' and T, for this collection. Both B and C are sister taxa of A in the source trees, but no information is present in these trees to resolve the clade A, B, C. Thus, the fully resolved supertrees T', T'', T''' all take arbitrary decisions by proposing one of the possible resolutions for this clade. Here, T is the sole supertree not proposing an arbitrary resolution for the clade. Arbitrary resolutions are misleading as they display relationships that are not entailed by the input trees.

The above properties can be formalized in different ways depending on the kind of topological relationship considered, e.g. clades, nestings, triplets, etc. Following Goloboff and Pol (2002) and Grunewald et al. (2006), we chose to focus on triplets. Given a collection $\mathcal{T}$ of input trees and a candidate supertree T, $\mathcal{R}(T, \mathcal{T})$ denotes the set of triplets of $\mathcal{T}$ for which T proposes a resolution. More formally, $\mathcal{R}(T, \mathcal{T}) = \big\{AB|C \in rt(\mathcal{T})$ such that $\{AB|C, AC|B, BC|A\} \cap rt(T) \neq \emptyset\big\}$. The set $\mathcal{R}(T, \mathcal{T})$ corresponds to all topological information present in collection $\mathcal{T}$ that is related to the information present in supertree T. Using this notation, we can express the induction property (PI) and the non-contradiction property (PC) as follows:

- T satisfies PI for $\mathcal{T}$ if and only if for all $t \in rt(T)$, it holds that $\mathcal{R}(T, \mathcal{T}) \vdash t$. In other words, PI requires that each and every triplet of T is induced by $\mathcal{R}(T, \mathcal{T})$.

- T satisfies PC for $\mathcal{T}$ if and only if for all $t \in rt(T)$ and all $\bar{t}$, it holds that $\mathcal{R}(T, \mathcal{T}) \nvdash \bar{t}$. This means that, for each and every triplet of T, $\mathcal{R}(T, \mathcal{T})$ induces no alternative resolution.

For instance, considering collection $\mathcal{T} = \{T_1, T_2\}$ in Fig. 2 and supertree T' of Fig. 3, the set $\mathcal{R}(T', \mathcal{T})$ is $\{AC|E, AC|F, AB|E, AB|F, BC|E, BC|F, EF|A, EF|B, EF|C\}$. Note that the

triplet $AD|C$ present in $rt(\mathcal{T})$ due to T_2 is not in this set because A, D, C are multifurcating in T'. When the source trees are incompatible, it is possible that $\mathcal{R}(T, \mathcal{T})$ contains two different triplets for the same three taxa. For example, consider the supertree T in Fig. 3 proposed by the MC and MMC voting methods (Semple and Steel, 2000; Page, 2002) on the collection $\mathcal{T} = \{T_1, T_2\}$. $\mathcal{R}(T, \mathcal{T})$ contains both $AB|C$ (resulting from T_2) and $AC|B$ (resulting from T_1). In this case, the supertree T that contains the triplet $t = AB|C$ does not satisfy PC, since $\bar{t} = AC|B$ is in $\mathcal{R}(T, \mathcal{T})$ (hence, $\mathcal{R}(T, \mathcal{T}) \vdash \bar{t}$). Indeed, in this example, supertree T includes topological information contained in T_2 that contradicts that of T_1. This situation indirectly results from a difference in the sizes of the clades of T_1 and T_2 which are incompatible: the clade containing more taxa (here (A, B, D) in T_2 versus (A, C) in T_1) is favored in the MC-MMC supertree. Such a size bias effect has been well-known in the field since Purvis (1995b) demonstrated it for the MRP voting method. Here it is illustrated for another voting method, and one might wonder whether this size bias is present in most voting methods. Note however that this size bias does not seem to have a major impact on MRP's accuracy (Baum and Ragan, 2004).

When the source trees are compatible, any reasonable method is expected to produce a supertree satisfying PC. However, some methods usually propose a supertree that does not satisfy PI. Indeed, compatible source trees can sometimes be displayed by an exponential number of supertrees, and some methods arbitrarily propose only one of them, thus selecting some triplets to the exclusion of other possible triplets. For instance, when considering the trivial case of two source trees $AB|E$ and $CD|E$, both MC and MMC propose the supertree ((A,B),(C,D),E), while numerous supertrees are possible, e.g., ((A,C),(B,D),E). In such a case, it seems preferable to output a consensus of all possible supertrees, as done by MRP (e.g. Bininda-Emonds and Bryant, 1998). Unfortunately, some topological information of the source trees (e.g. triplets) can be absent from the obtained consensus as it can contain highly multifurcating nodes.

However, for some compatible collections of trees, it is possible to find a supertree that displays all triplets of the collection and is also refined by all other possible supertrees. More formally, a set $\mathcal{R}$ of triplets is said to *identify* a tree T if and only if T strictly displays $\mathcal{R}$ and T is refined by every tree T' that strictly displays $\mathcal{R}$. A set $\mathcal{R}$ can identify at most one tree, thus when the triplet set $\mathcal{R} = rt(\mathcal{T})$ of a collection $\mathcal{T}$ of source trees identifies a tree, this tree is a *canonical* representation of all possible supertrees.

Considering practical collections $\mathcal{T}$ of source trees, $rt(\mathcal{T})$ will almost never identify a tree, either because this set is incompatible, or because it does not identify a particular tree. Nevertheless, it is possible that a subset of the triplets in $rt(\mathcal{T})$ identifies a tree T, and then the topological information contained in T exactly corresponds to a subset of the topological information contained in $\mathcal{T}$. Such a subset is most interesting when the triplets t it contains do not have an alternative resolution $\bar{t}$ in $rt(\mathcal{T})$. This situation occurs for the subset $\mathcal{R}(T, \mathcal{T})$ of $rt(\mathcal{T})$ when the supertree T satisfies PI and PC.

Proposition 1 *A tree T satisfies PI and PC for a collection $\mathcal{T}$ of trees if and only if $\mathcal{R}(T, \mathcal{T})$ identifies T.*

The proof is given in Appendix. It is based on the fact that a set $\mathcal{R}$ identifies a tree T if and only if $rt(T) = cl(\mathcal{R})$ (Grunewald et al., 2006, Lem. 2.1). This proposition confirms the relevance of PI and PC: having a supertree T that satisfies both of them highlights a part of $rt(\mathcal{T})$ (namely $\mathcal{R}(T, \mathcal{T})$) that exactly corresponds to a tree, i.e. does not contain arbitrary topological information, and moreover does not contradict any input tree. Such a feature is most desirable for supertrees inferred by veto methods.

Links with other advocated properties

Properties similar to PI and PC were described in (Goloboff and Pol, 2002, p.519) as "the property of [the supertree] displaying $AB|C$ if it is found in some input tree or implied by some combination of input trees and no input tree or combination of input trees displays or implies $AC|B$ or $BC|A$". These properties were also pointed out as being desirable by Grunewald et al. (2006). Using our formalism, they can be translated as follows for a supertree T representing a collection $\mathcal{T}$:

- PI': for any $t \in rt(T)$, it holds that $rt(\mathcal{T}) \vdash t$
- PC': for any $t \in rt(T)$ and for all $\bar{t}$, it holds that $rt(\mathcal{T}) \nvdash \bar{t}$.

The essential difference between PI'-PC' and PI-PC is whether we evaluate supertrees based on triplets in the original set of trees, $rt(\mathcal{T})$, or on the triplets commonly resolved the supertree and at least one of the source trees, $\mathcal{R}(T, \mathcal{T})$. From the statement of the properties, it is clear that PC' implies PC and PI implies PI'. It is thus natural to wonder which version of the properties is preferable. Below, we show an example where PC' is too restrictive, and an example where PI' is too permissive. In contrast, PI and PC behave correctly in these examples.

Example 1 *Let $\mathcal{T} = \{T_1, T_2\}$ with T_1 and T_2 as shown in Fig. 4. $rt(\mathcal{T})$ contains $AE|B$ and $AC|E$, therefore $rt(\mathcal{T}) \vdash AC|B$. We also have $rt(\mathcal{T}) \vdash AB|C$ since $AB|C \in rt(T_1)$. Thus any tree providing a triplet on $\{A, B, C\}$ does not satisfy PC'. For analogous reasons PC' does not allow us to propose any triplet in the supertree. Thus PC' rejects the tree T of Fig. 4. Yet T is a reasonable and informative supertree for $\mathcal{T}$ and satisfies both PI and PC.*

We note that T is not a plenary supertree, i.e. it does not contain all input taxa, but this example shows that removing rogue taxa is a way in which more informative supertrees can be obtained.

This is in line with the remark of Wilkinson et al. (2004), who stated that "non-plenary supertree methods might be most useful for identifying unstable leaves". For instance, such leaves might be involved in lateral transfers. This example easily generalizes to cases where the *supertree* actually contains more leaves than each source tree. Figure 5 depicts this generalization.

The next example shows a supertree satisfying both PI' and PC', while also displaying irrelevant triplets.

Example 2 *Let* $\mathcal{T} = \{T_1, T_2\}$ *with* T_1 *and* T_2 *as illustrated in Fig. 6.* $rt(\mathcal{T}) = \{AB|C, AB|X, BC|A\}$. *The tree* T *of Fig. 6 displays* $\{AB|X, BC|X, AC|X\}$. $AB|X$ *is present in (thus induced by)* $rt(\mathcal{T})$ *but the two other triplets can also be induced from* $rt(\mathcal{T})$*:* $\{AB|X, BC|A\} \vdash \{BC|X, AC|X\}$. *It follows that* T *satisfies PI'. Moreover, it is easily seen that no combination of triplets in* $rt(\mathcal{T})$*, other than* $\{AB|X, BC|A\}$*, induces triplets. Thus* T *also satisfies PC'. However,* T *is clearly not an ideal supertree for* $\mathcal{T}$ *as no information in* $\mathcal{T}$ *induces group* A, B, C *to nest inside group* A, B, C, X*. The property PI, not satisfied by* T*, detects this problem: here* $\mathcal{R}(T, \mathcal{T})$ *only contains the triplet* $AB|X$ *and thus it does not induce the triplet* $AC|X$ *present in* T*.*

The PI' property quoted by Goloboff and Pol (2002) is stronger than the *Pareto* property (Neumann, 1983; Wilkinson et al., 2004) on triplets, which requires that the output tree contain all triplets that occur in all source trees. The Pareto property is appealing in general and has also been advocated in the supertree context (property P6 of Steel et al., 2000). However imposing the Pareto property on triplets may be problematic, even in the case of compatible source trees (Thorley and Wilkinson, 2003). This is due to the possibility of having *several* candidate supertrees that are both compatible with source trees and respect the Pareto property. In this case, no *single* supertree exists that satisfies the Pareto property while having no arbitrary resolution. The strict consensus of these supertrees does not necessarily satisfy the Pareto property. A solution is then to

return several trees, either all candidate supertrees or their reduced consensus (Wilkinson, 1994). However, this solution may not well be suited when the aim is to summarize a collection of source trees into a single supertree that is more easily dealt with for further analysis by biologists.

When source trees are incompatible, it may even be impossible to have a supertree satisfying both the Pareto and non-contradiction properties (PC and PC') as shown in the following example.

Example 3 *Consider the collection* $\mathcal{T} = \{T_1, T_2\}$ *where* $T_1 = (((A, D), B), ((C, F), E))$ *and* $T_2 = (((A, E), (B, F)), (C, D))$. *Triplets* $AB|C$ *and* $EF|D$ *are displayed by both trees of* $\mathcal{T}$. *Thus any supertree* T *for* $\mathcal{T}$ *must include all leaves in* $\mathcal{T}$ *in order to satisfy the Pareto property. Since* $rt(\mathcal{T})$ *contains* $AB|D$ *and* $AD|B$, *any tree* T *displaying a triplet for the three leaves does not satisfy PC (hence PC'). For similar reasons, no supertree* T *can display a triplet on the taxa A, C and D. Thus, any supertree satisfying PC (or PC') and including all taxa of* $\mathcal{T}$ *contains a multifurcating node on taxa* A, B, C, D, *hence does not display the triplet* $AB|C$, *i.e. does not satisfy the Pareto property.*

In other words, imposing the Pareto property can lead the supertree to explicitly contradict relationships present in some input trees. This shows that the Pareto property on triplets is not compatible with the veto approach, where the proposed supertree must not contradict the source trees. However, the Pareto property can be considered for other topological relationships (Wilkinson et al., 2004). For example, there is always a supertree satisfying PI and PC as well as the Pareto property on partial or full splits contained in the source trees.

The Pareto property specifies relations that the supertree *must* contain. The complementary co-Pareto property specifies relations that the supertree *must not* contain. The co-Pareto property in the consensus context requires that the consensus tree contain no relationships that are not present in at least one input tree. However, Wilkinson et al. (2004) point out that this

statement is not reasonable for supertrees, since "they might contain relashionships that are entailed by the input trees in combination, but are not present in any of them singly". Then they propose a weaker version that requires that the supertree does not contain relationships that are contradicted by all the input trees whose leaf set makes a contradiction possible. Note that, any supertree satisfying PC also satisfies the latter version of co-Pareto.

Steel et al. (2000) list five other properties that might be requested from supertree methods: changing the order of the trees in the input collection does not change the supertree (P1); renaming the taxa of the source trees gives the same supertree, but with the taxa renamed accordingly (P2); the output tree displays the source trees when they are compatible (P3); each leaf (taxon) that occurs in at least one source tree is in the supertree (P4); the running time of the method grows polynomially with respect to the total number of taxa (P5). The following example shows that ensuring P3 can force the supertree to contain arbirtrary clades. Thus P3 can conflict with PI.

Example 4 *Let $\mathcal{T} = \{T_1, T_2\}$ with $T_1 = ((A, B), W)$ and $T_2 = ((A, B), (X, (Y, Z)))$. A supertree with taxon set $\{A, B, W, X, Y, Z\}$ that satisfies P3 must display T_2, hence must have a clade including Y, Z but not X. However, it will contain arbitrary clades, no matter where taxon W is attached. This is because any supertree satisfying PI must include a polytomy on W,X,Y,Z since source trees include no information on the relative position of W and the group X, Y, Z. For instance, the supertree $((A, B), (X, Y), W, Z)$ excludes the possibility for (A, B) and (X, Y) to be intermixed.*

Note that if polytomies of a supertree are interpreted in terms of an Adams consensus (Adams, 1972), then this example does not put P3 into question. However, this interpretation of polytomies does not prevail in phylogenetics, as we discuss in further detail in the case study paragraph.

Checking PC and PI in polynomial time

Existing supertree methods can sometimes output trees that do not satisfy PC or PI. For instance, the MC supertree obtained for the collection of Fig. 2 does not satisfy PC, while on that of Example 4, it fails to satisfy PI. In contrast, for the collection $\{AB|C, BC|D\}$ the MC supertree satisfies bothPI and PC. The MRP method sometimes outputs supertrees not satisying these properties (e.g. PC is not satisfied in Fig. 1 of Bininda-Emonds and Bryant, 1998), and sometimes provides supertrees that satisfy them – e.g. when the source trees are compatible (Steel, 1992). We now describe an algorithm to decide whether a candidate supertree satisfies both PI and PC together. In case of a negative answer, it pinpoints those parts of the supertree contradicting these properties. This algorithm relies on two properties equivalent to PC and PI, whose formulation is less intuitive but whose checking is easy.

Definition 1 *Let $\mathcal{T}$ be the collection of source trees and T be a proposed supertree for $\mathcal{T}$. Define PC_{eq} and PI_{eq} to be the following properties:*

- *PC_{eq}: $rt(T) \cup \mathcal{R}(T, \mathcal{T})$ is compatible.*
- *PI_{eq}: for any $t \in rt(T)$ and for all $\bar{t}$, the set $\{\bar{t}\} \cup \mathcal{R}(T, \mathcal{T})$ is incompatible.*

Proposition 2 *(PI_{eq} and PC_{eq}) $\Leftrightarrow$ (PI and PC).*

Proof.

- **$PC_{eq} \Rightarrow$ PC**: $PC_{eq} \Rightarrow \forall t \in rt(T)$, $\{t\} \cup \mathcal{R}(T, \mathcal{T})$ is compatible. This ensures that there is at least one tree T' that displays $\mathcal{R}(T, \mathcal{T}) \cup \{t\}$. It follows that T' displays $\mathcal{R}(T, \mathcal{T})$ but not $\bar{t}$. As $\bar{t}$ is not displayed by every tree that displays the compatible set $\mathcal{R}(T, \mathcal{T})$, it follows that $\mathcal{R}(T, \mathcal{T}) \nvdash \bar{t}$.

- **PC** $\Rightarrow$ **PC**$_{eq}$: $PC \Rightarrow \mathcal{R}(T,\mathcal{T}) \subseteq rt(T)$ (cf proof of Prop. 1), this ensures that $rt(T) \cup \mathcal{R}(T,\mathcal{T})$ is compatible (since displayed by T).

- **PI** + **PC** $\Rightarrow$ **PI**$_{eq}$: PI and PC $\Rightarrow cl(\mathcal{R}(T,\mathcal{T})) = rt(T)$ by Prop. 1. This ensures PI$_{eq}$.

- **PI**$_{eq}$ + **PC**$_{eq}$ $\Rightarrow$ **PI**: PC$_{eq}$ ensures that $\mathcal{R}(T,\mathcal{T})$ is compatible. PI$_{eq}$ is exactly the definition of the induction for a compatible set, thus ensuring PI.

□

Note that we do not prove a direct equivalence between PI and PI$_{eq}$ in this general case. The two properties are only equivalent for a compatible set. In fact, PI$_{eq}$ is relatively uninformative without PC$_{eq}$, since PI$_{eq}$ holds as soon as $\mathcal{R}(T,\mathcal{T})$ is incompatible. Note also that another formulation of PC, closer to that of PI$_{eq}$ but less concise, is as follows: *for any $t \in rt(T)$, the set $\{t\} \cup \mathcal{R}(T,\mathcal{T})$ is compatible.*

PI$_{eq}$ and PC$_{eq}$ can easily be checked by using the *Build* algorithm, which indicates in polynomial time whether a set of rooted trees is compatible or not. A similar procedure was proposed by Steel (1992), and refined by Daniel (2004), to compute the strict consensus of all supertrees displaying a collection of compatible source trees. The following lemma provides us with an even faster way to check PC$_{eq}$.

Definition 2 (Direct contradiction) *A tree T* directly contradicts *a set of triplets $\mathcal{R}$ when there is a triplet t in $rt(T)$ such that $\exists \bar{t} \in \mathcal{R}$. A supertree T is said to directly contradict a collection $\mathcal{T}$ of source trees if T directly contradicts $rt(\mathcal{T})$.*

Direct contradictions are linked with the PC property in the following way:

Lemma 1 *If a tree T does not directly contradict a collection $\mathcal{T}$ of source trees then the three following statements hold:*

1. $\mathcal{R}(T, \mathcal{T}) \subseteq rt(T)$;

2. $\mathcal{R}(T, \mathcal{T})$ *is compatible;*

3. T *satisfies* PC_{eq} *for* $\mathcal{T}$.

Proof. By definition, $\mathcal{R}(T, \mathcal{T})$ only contains triplets on 3-taxon sets for which there is a triplet in $rt(T)$. Since T does not directly contradict $\mathcal{T}$, the triplets of $\mathcal{R}(T, \mathcal{T})$ are resolved as those in T. It follows that $\mathcal{R}(T, \mathcal{T}) \subseteq rt(T)$ (proving 1). $\mathcal{R}(T, \mathcal{T})$ is therefore compatible (proving 2). Moreover, $\mathcal{R}(T, \mathcal{T}) \subseteq rt(T)$ ensures that $\mathcal{R}(T, \mathcal{T}) \cup rt(T)$ is compatible, which is exactly the formulation of PC_{eq} (proving 3). □

Thus, to check that a supertree T satisfies PC_{eq}, and hence PC, for a collection $\mathcal{T}$, it suffices to check that any triplet of $rt(T)$ is not resolved in a different way in a tree of $\mathcal{T}$. This can be done by computing the set $rt(T)$ of $O(n^3)$ triplets in T and then comparing $rt(T)$ with the set of triplets of each source tree T_i. If the collection $\mathcal{T}$ contains k source trees and a total of n taxa, then this simple implementation requires $O(kn^3)$ computing time. However, it is possible to check this condition in linear time for each pair T, T_i with $T_i \in \mathcal{T}$: first restrict in $O(n)$ time the trees T and T_i to the taxa they share; then apply the algorithm of Berry et al. (2005) that, given two trees with the same taxa, finds in $O(n)$ time a triplet resolved differently in the trees, or states that this situation does not arise. Thus, successively considering k source trees leads to a procedure that checks PC in $O(kn)$ computing time.

PhySIC : a polynomial-time veto supertree method

We introduced above the PI and PC properties, showed their relevance and described algorithms to check whether a given supertree T satisfies them. In this section, we show that it

is possible to design a method that always produces supertrees that satisfy PI and PC. However, this aim is not precise enough, as the *star tree* (the tree whose leaves are all children of a single internal node) trivially satisfies these properties – simply because it does not resolve any triplet. Thus, a reasonable aim is to design a method that always infers supertrees that satisfy PI and PC *and* that contain as much resolution as possible, e.g., resolve as many triplets as possible. More precisely, we require a method that, given any collection $\mathcal{T}$, proposes a supertree T such that $\mathcal{R}(T, \mathcal{T})$ identifies T and $\mathcal{R}(T, \mathcal{T})$ has maximum size over all such subsets of $rt(\mathcal{T})$. Such a subset of $rt(\mathcal{T})$ is called a maximum identifying subset of triplets (MIST).

The difficulty of this problem cannot be simply deduced from previously known theoretical results for optimization problems on triplets. Indeed, the MIST problem is a middle term between the NP-hard problem that consists of finding a maximum-sized *compatible* subset of triplets (Bryant, 1997) and the polynomial-time problem that asks for the maximum-sized *tree-like* subset of a complete set of triplets (Berry and Gascuel, 2000; Bryant and Berry, 2001). Unfortunately, the MIST problem is NP-hard (Guillemot and Berry, 2007). This shows that it is highly unlikely that a polynomial-time algorithm exists that could find the most resolved supertree satisfying PI and PC. However, we can still rely on heuristic algorithms to find reasonable (but potentially suboptimal) solutions, as is commonly done for other NP-hard problems such as finding a most parsimonious tree or a maximum likelihood tree for a character matrix.

We present below a polynomial-time heuristic method that always outputs a supertree that satisfies PI and PC. The method tries to produce a supertree that contains as many input triplets as possible under this constraint. The method is a variant of the well-known *Build* algorithm and is called *PhySIC– Phylogenetic Signal with Induction and non-Contradiction*. Supertrees inferred by the method have a degree of resolution that can be close to that of supertrees inferred by voting methods (see next section), while only containing clades that are not arbitrary with respect to the

source trees nor contradicting them as detected by PC.

Inferring a supertree that satisfies PC

This section introduces algorithms, based on the *Build* algorithm, to produce non-trivial trees that satisfy PC.

The *Build* algorithm

The *Build* algorithm is a yes-or-no algorithm that tells whether a collection of triplets or larger trees is compatible or not. To achieve its goal, the algorithm tries to build a tree displaying the triplets; if the process is blocked at some step, this means that the input triplets are not compatible. This tree is built recursively, from the root to the leaves. First, the largest clades are identified, then clades included in the first ones, and so on. The composition of the clades is guided by the structure of a *graph*, that is a set of objects (called *vertices*) with links (called *edges*) between pairs of them.

The graph used by *Build*, called here the *Aho Graph*, is defined as follows: let $\mathcal{R}$ be a collection of triplets on a taxon set X, the *Aho Graph* G for $\mathcal{R}$ is the undirected graph with vertices X and with edges (A, B) between two taxa A and B whenever there is a triplet $AB|C \in \mathcal{R}$. Thus, an edge between two taxa means that at least one triplet sees these two taxa in the same clade. The vertices of G are denoted by $v(G)$ (in the present description $v(G) = X$). For instance, Fig. 7a shows the Aho graph built from $\mathcal{R} = rt(\{T_1, T_2\})$, where T_1, T_2 are the source trees of Fig. 2, and e.g., the edge between taxa B and D is due to the triplet $bd|c \in rt(T_2)$.

A *connected component* C_i of a graph is a maximal set of taxa linked to one another, i.e., such that for any pair A, B of taxa in C_i, there is a set of edges that links A to B. For instance, the graph in Fig. 7a contains two connected components: $C_1 = \{E, F\}$ and $C_2 = \{A, B, C, D\}$.

The connected components of graph G are denoted by $CC(G)$. The vertices of a component C_i of G are denoted by $v(C_i)$. When the Aho graph contains several connected components, each of them corresponds to a clade of the tree representing the input collection of triplets (if such a tree exists). Once these clades are known, the clades contained in each of these primary clades are found by recursively processing Aho graphs for subsets of triplets that respectively concern the taxa of these clades: the restriction of $\mathcal{R}$ to taxa of a component C_i is denoted by $\mathcal{R}|v(C_i)$ and defined as $\big\{AB|C \in \mathcal{R}$ such that $\{A, B, C\} \subseteq v(C_i)\big\}$. For example Fig. 7b shows the Aho graph obtained from $\mathcal{R}|v(C_2)$ where $\mathcal{R}$ is the set of triplets due to source trees in Fig. 2, and C_2 is the component of the initial Aho graph shown in Fig. 7a. The recursive calls stop when dealing with components containing less than 3 taxa, since there is no triplet (hence incompatibility) on so few taxa. However, if at some point in the recursive process, the Aho graph for a set of at least three taxa has only one connected component, this means that the input trees are conflicting on the resolution of these taxa. When this happens, the algorithm states that the collection of source trees is incompatible. Otherwise, when all recursive calls return, the algorithm concludes that the source trees are compatible. For instance, when run on the collection of Fig. 2, *Build* first finds two connected components, $C_1 = \{E, F\}$ and $C_2 = \{A, B, C, D\}$, but the recursive call on C_2 leads to a graph containing only one connected component (Fig. 7b), which leads the algorithm to detect the incompatibility of the source trees.

A first simple modification of *Build*

We first describe here a simple modification of *Build* that infers a supertree from a collection of source trees $\mathcal{T}$. This subroutine, called $Build_{PC}$ (see Fig. 8), takes as input the triplet set $\mathcal{R} = rt(\mathcal{T})$ of a collection $\mathcal{T}$ of source trees and the list S of taxa contained in these trees. $Build_{PC}$ mainly differs from *Build* when the Aho graph contains one connected component on the

set S of taxa currently considered (line 1). In this case, $Build_{PC}$ returns the star tree on S (i.e., a single polytomy on S, thus contradicting no input triplet), whereas *Build* simply concludes that the sources trees are incompatible. This star tree is then grafted as a subtree of the tree built by the previous recursive call. Thus, we can now output a supertree even when the source trees are incompatible. As an example, from the collection of Fig. 2, $Build_{PC}$ infers the supertree T' displayed in Fig. 3.

Proposition 3 *Given a collection $\mathcal{R}$ of triplets on a taxon set S, $Build_{PC}$ returns a tree T on S that satisfies the PC property for $\mathcal{R}$.*

The proof of this proposition can be found in appendix.

A more involved algorithm to infer a supertree satisfying PC

$Build_{PC}$ sometimes produces poorly resolved trees due to multifurcations returned in cases where G contains a single connected component (i.e., when $\mathcal{R}$ contains conflicts covering the considered subset of taxa). In the most extreme (though unlikely) case, this situation occurs at the first step of the algorithm, which then outputs a star tree.

The most basic conflicts between triplets of $\mathcal{R}$ occurs when two different triplets t and $\bar{t}$ appear in $\mathcal{R}$ for a same set of three taxa. Such a direct contradiction cannot be present in a tree that satisfies PC. Given $\mathcal{R}_{dc}$, the set of triplets such that $t, \bar{t} \in \mathcal{R}$ it seems relevant to consider the subset $\mathcal{R}' = \mathcal{R} - \mathcal{R}_{dc}$ $\mathcal{R}$. We define a variant of $Build_{PC}$, called $PhySIC_{PC}$, that resorts to that subset whenever conflicts are detected. This enables the produced supertree T' to be generally much more resolved than the tree returned by $Build_{PC}$. For instance, Fig. 7b shows the graph obtained for $\mathcal{R}|v(C_2)$, where $\mathcal{R}$ are triplets of the collection in Fig. 2 and C_2 is the connected component shown in Fig. 7a. This graph is connected due to the direct conflicts

between $AB|C$ (resulting from T_1) and $BC|A$ (resulting ftom T_2). This situation leads $Build_{PC}$ to return a polytomy on A, B, C, D. In contrast, building the graph on the basis of $\mathcal{R}'$ results in two connected components, C_i and C_j, allowing $PhySIC_{PC}$ to propose a tree with two subtrees for taxa A, B, C, D.

The correctness of $Build_{PC}$ ensures that T' satisfies PC with respect to $\mathcal{R}'$ but without any guarantee that this also holds with respect to $\mathcal{R}$. To ensure the latter, and thus the correctness of $PhySIC_{PC}$, T' must not resolve any triplet of $\mathcal{R}_{dc}$. A way to ensure this is to collapse any branch of T' that resolves a triplet of $\mathcal{R}_{dc}$. The tree thus obtained is still always at least as resolved as the one proposed by $Build_{PC}$ and potentially contains supplementary branches. Indeed, direct contradictions at the root of a clade no longer prevent the proposition of clades on subsets of its taxa. For instance, on the collection of Fig. 2, the tree initially computed by $PhySIC_{PC}$ is the tree called T in Fig. 3. But as the branch leading to the clade (A, D, B) contradicts $AC|B \in \mathcal{R}_{dc}$, the branch above this clade is collapsed, and the final tree output by $PhySIC_{PC}$ is then the tree named T" in Fig. 3. This tree contains one clade more than the tree output by $Build_{PC}$ (the tree named T' in the figure).

These ideas are included in the $PhySIC_{PC}$ algorithm whose pseudo-code is detailed in Fig.9.

Theorem 1 *Given a triplet set $\mathcal{R} = rt(\mathcal{T})$ from a collection $\mathcal{T}$ of source trees on a taxon set S of n leaves, $PhySIC_{PC}$ returns in $O(n^4)$ time a tree T satisfying PC for $\mathcal{T}$.*

The proof of this result can be found in appendix.

Ensuring that the supertree satisfies PI

The supertree T_{PC} output by $PhySIC_{PC}$ does not usually satisfy the PI property. The $PhySIC_{PI}$ algorithm transforms T_{PC} so that it also satisfies PI. To that aim $PhySIC_{PI}$ recursively searches the tree and checks that for each branch each triplet is induced by $\mathcal{R}(T_{PC}, \mathcal{T})$. The theorem 3.1.1 of Daniel (2004) provides a useful characterisation to decide when a branch is justified, directly or indirectly, thanks to triplets present in $\mathcal{R}(T_{PC}, \mathcal{T})$. When considering the branch linking u to a subtree S_i, the theorem considers a graph G_{ij} for any sibling subtree S_j of S_i. Any such graph G_{ij} is the Aho graph with vertices $L(S_i)$, and with edges due to triplets of $\mathcal{R}(T_{PC}, \mathcal{T})$ whose three leaves are in $L(S_i) \cup L(S_j)$. The theorem states that the branch from u to the root of S_i is justified if and only if G_{ij} is connected, for any sibling subtree S_j.

Example 5 *Consider for instance the simple example where $\mathcal{T}$ contains the trees ((A,B),X) and ((E,F),X). The Aho graph for $rt(\mathcal{T}) = \{AB|X, EF|X\}$ is made of three connected components: $C_1 = \{A, B\}$, $C_2 = \{E, F\}$ and $C_3 = \{X\}$, therefore applying the $PhySIC_{PC}$ algorithm gives the tree $T_{PC} = ((A, B), (E, F), X)$. T_{PC} displays $AB|E$ even though this information is not induced by $\mathcal{T}$. Indeed, the branch defining the clade (A, B) is detected as not justified since the corresponding connected component, C_1, is not connected in the Aho graph when we consider only edges due to triplets with taxa in $C_1 \cup C_2$.*

This Theorem is the basis of a *decision* algorithm called *Identifies*, that states whether a given set of triplets identifies a given tree (Daniel, 2004). It is possible to design a simple variant of this algorithm that always returns a tree (not just a *yes* or *no* answer): when a branch between a node p and the root of a subtree S_i is not justified, the idea is to replace S_i by a star tree on the taxa of the corresponding clade. This crude variant removes the unjustified branches, but also potentially many other branches, i.e., those inside S_i, those leading to sibling subtrees S_j of S_i, and

those inside S_j subtrees. $PhySIC_{PI}$ is a more refined variant that only collapses the unjustified branches. See the pseudo-code in Fig. 10 for details. In this code, $PhySIC_{PI}$ is given a tree T in which unjustified branches are to be collapsed, and a collection $\mathcal{T}$ of source trees or, equivalently, the corresponding set of triplets (as written in the pseudo-code). $PhySIC_{PI}$ repeatedly calls the $Check_{PI}$ subroutine to detect unjustified branches that are then removed until none remain (note that in the pseudo-code of $Check_{PI}$, $S(T)$ denotes (complete) subtrees connected to the root of T, i.e., the subtrees corresponding to the largest clades under the root of T).

From the collection of Fig. 2, $PhySIC_{PC}$ infers the supertree T'' displayed in Fig. 3. and none of the three internal branches of T'' are collapsed by $Check_{PI}$. For instance, consider the step where $Check_{PI}$ checks the subtree ((A,D),B,C) of T'', whose child subtrees are (A,D) plus the two trivial subtrees on B and C. The sole branch that has to be checked in ((A,D),B,C) is the one defining the clade (A,D). Here, $Check_{PI}$ builds two Aho graphs with vertices $\{A, D\}$: one with edges due to triplets on $\{A, D\} \cup \{B\}$ and one with edges due to triplets on $\{A, D\} \cup \{C\}$. Both graphs are connected thanks to triplets of the source tree T_2; therefore, $Check_{PI}$ does not collapse any branch at this step.

Theorem 2 *Given a collection $\mathcal{T}$ of trees and a tree satisfying PC for $\mathcal{T}$, $PhySIC_{PI}$ returns in $O(n^4)$ time a tree T on $L(\mathcal{T})$ that satisfies both PC and PI for $\mathcal{T}$.*

The proof of this Theorem can be found in the appendix.

The *PhySIC* algorithm (see pseudo-code) builds a supertree for a collection of k source trees $\mathcal{T}$ by first computing the set $rt(\mathcal{T})$ and then successively calling $PhySIC_{PC}$ and $PhySIC_{PI}$. Since $rt(\mathcal{T})$ is computed in $O(kn^3)$, *PhySIC* runs in $O(kn^3 + n^4)$ time.

Theorem 3 *Given a collection $\mathcal{T}$ of k source trees on n leaves,* PhySIC *returns in $O(kn^3 + n^4)$ time a tree both satisfying PC and PI.*

Lastly, we note that similar procedures can be designed to modify the supertree proposed by any existing supertree method. If a method proposes a supertree T that does not satisfy PC and PI, it is possible in polynomial time to transform T into a tree T' that satisfies these properties. Indeed, the algorithm indicated previously to check PC indicates the triplets from which the incompatibility arises. Then the branches of T inducing these triplets can be collapsed to obtain a tree T' satisfying PC. Now, a procedure similar to $PhySIC_{PI}$ can be applied to T' to ensure that it also satisfies PI (without invalidating PC).

Biological case studies on Primates

To illustrate the impact of the PC and PI properties on supertree inference, and to compare the behavior of veto methods like *PhySIC* to that of voting methods like MRP and MMC, we present two case studies centered on Primates. This mammalian order is one of the first taxonomic groups for which a large-scale supertree approach has been conducted (Purvis, 1995a). The first example is designed to show the desirable properties of *PhySIC* compared to other supertree methods on a smaller, understandable taxonomic scale. The second example addressing the question of the primate supertree at the genus level shows how *PhySIC* performs on a larger taxonomic scale–approaching what supertree studies tend to be performed on–, and shows that varying degrees of resolution are achieved in the supertree depending upon the nodes retained from the input trees.

First example: illustration of supertree desirable properties

Source trees.—We focused on a subsample of Primates IRBP (Interphotoreceptor Retinoid Binding Protein) and ADRA2B (α2B-Adrenergic Receptor) gene sequences, respectively from

Poux and Douzery (2004) and Poux et al. (2006), with a rodent (*Mus*) and lagomorph (*Oryctolagus*) outgroup. For ADRA2B, the hominoid representative was *Pan*, with the sequence downloaded from the chimp ENSEMBL project. The ADRA2B and IRBP source trees were inferred by maximum likelihood (ML) analysis of the corresponding alignments, using PHYML (Guindon and Gascuel, 2003), version 2.4.4, under a GTR+Γ_4+INV model of DNA evolution. The node support was estimated after 1000 bootstrap replicates using the same software, and expressed as bootstrap percentages (BP). Denser taxonomic and phylogenetic information for Strepsirrhines (i.e., *Lemurs* and *Galagos*) was sought from a study of presence-absence of short interspersed nuclear elements (SINE) integrations in primate genomes (Roos et al., 2004, Fig. 2). Sixty-one monolocus SINE characters detected by these authors were subjected to a maximum parsimony analysis using PAUP* (Swofford, 2002), version 4b10, with 1000 bootstrap replicates using a heuristic search, with 10 random additions of taxa, and TBR branch swapping. We only retained the best supported nodes of source trees, i.e., those showing at least 50% bootstrap (*cf.* also Daubin et al., 2002).

Comparison of supertrees inferred from PhySIC*, MMC and MRP.*—Starting from the three source trees (Fig. 11), supertrees were built using the MMC, MRP, and *PhySIC* methods. For MRP, the matrix representation of the three source topologies resulted in 47 characters. Parsimony analysis was conducted under PAUP*, with a heuristic search with 1000 random addition sequences, and TBR branch swapping, resulting in 864 equally parsimonious trees, a strict consensus of which provided the MRP supertree. The MMC supertree was obtained using the program distributed by Rod Page. Fig. 12 shows the supertrees respectively reconstructed by MMC, MRP, and *PhySIC* with its $PhySIC_{PC}$ intermediate step.

The supertrees produced all contain some soft polytomies, each of them representing uncertainty about the resolution of a node's child subtrees or lineages. A soft polytomy can have two

distinct interpretations, differing in the set of admissible fully-resolved phylogenies it encompasses. Consider the case of the polytomous node P in the MMC tree of Fig. 12. This node has three child subtrees S_1 =(*Homo*, *Hylobates*), S_2=(*Pan*,(*Cercopithecus*, *Macaca*)) and S_3, the Platyrrhinii clade. The most widespread meaning of a soft polytomy accepts any fully-resolved tree on subtrees S_1, S_2, S_3 that keeps their monophyly: $((S_1, S_2), S_3)$, $((S_1, S_3), S_2)$ or $((S_2, S_3), S_1)$. Strict consensus, majority-rule consensus, and hence MRP, interpret polytomies in this way (Margush and McMorris, 1981). Polytomies proposed by $PhySIC_{PC}$ are also to be interpreted in this way. A second interpretation of soft polytomies was introduced by the Adams consensus (Adams, 1972) and is also intended by MC (Semple and Steel, 2000) and MMC (Page, 2002). This interpretation accepts as possible phylogeny any fully-resolved tree that maintains the structure of each subtree respectively, no matter whether or not S_1, S_2, and/or S_3 are kept monophyletic (i.e., their leaves can be interleaved). Thus, the polytomy P of the MMC tree in Fig. 12 can indeed give rise to fully-resolved trees grouping *Pan* and *Homo* without *Hylobates*, as long as *Pan* is kept outside the clades containing *Cercopithecus* and *Macaca* (which is the structure imposed by S_2). Under this interpretation, a soft polytomy represents a much wider range of fully-resolved phylogenies than with the first interpretation, and is harder to interprete in a phylogenetic context. (In particular, this means that simulation studies on supertree methods that use the Robinson and Foulds distance to evaluate the performance of MC or MMC are misleading: on the previous example, the MMC method would have been considered to propose the incorrect clades *Homo* + *Hylobates*, and *Pan* + *Hylobates*).

The contribution of $PhySIC_{PC}$ to the supertree inference may be illustrated by the situation among platyrrhines. Here, ADRA2B indicates that *Ateles* is the sister-group of *Pithecia*, *Callithrix*, and *Cebus*, whereas IRBP indicates that *Ateles* and *Cebus* are the closest relatives (Fig. 12: boxed areas). This conflict is detected by $PhySIC_{PC}$. As a result, the $PhySIC_{PC}$ and

PhySIC supertrees display all four platyrrhines within a multifurcation. By contrast, MRP and MMC give priority to the *Callithrix* + *Cebus* grouping present in the ADRA2B source tree, and thus contradicts the *Ateles* + *Cebus* grouping present in the IRBP source tree. Resolution of this conflict between the source trees reflects the voting approach followed by MRP and MMC. For instance, consider the case of MRP: the ADRA2B source topology comprises two nodes within platyrrhines, against one node for the IRBP topology. Therefore, MRP favors the node *Callithrix* + *Cebus* involved in a topological conflict but belonging to a larger and more resolved clade (Bininda-Emonds and Bryant, 1998). The behavior of MRP and MMC on platyrrhines is problematic. Indeed, it favors one source topology while contradicting another, just on the basis of their respective levels of resolution, and despite the fact that both contain the same number of taxa for the Platyrrhine subtree. MRP has already been criticized on this point (e.g., Goloboff, 2005). Note that source trees also conflict on the position of *Propithecus* with respect to *Microcebus* and *Lemur*. However, in this case, MRP behaves as $PhySIC_{PC}$ and *PhySIC*, i.e., displays a polytomy on groups containing these three taxa (Fig. 12: letters *A*-*B*-*C*). By contrast, MMC groups together the *Propithecus* and *Lemur* clades, following the SINE information, but contradicting the IRBP information.

The complementary contribution of $PhySIC_{PI}$ to the supertree inference may be illustrated by the situation among Catarrhines + Platyrrhines. Although not contradicting the source trees, the $PhySIC_{PC}$ supertree contains two topological errors. First, man and chimp do not group together relative to the gibbon, as would be expected from a plethora of data (Goodman et al., 2005). *Homo* is instead associated with *Hylobates*, whereas *Pan* branches with the two cercopithecoids, *Cercopithecus* and *Macaca*. This situation results from the taxon sampling of the source topologies. More precisely, man and chimp are not simultaneously present in any source tree, i.e., the former clusters with the gibbon (IRBP) and the latter with cercopithecoids (ADRA2B). These

two source clades are reproduced in $PhySIC_{PC}$ and MMC supertrees.

In the case of $PhySIC_{PC}$, these two clades are involved in a polytomy with the platyrrhines. This polytomy means that (*Homo*, *Hylobates*) is a sister clade of the clade containing *Pan*. However, although these clades are correct when considered separately, they should not be sister groups in the supertree. $PhySIC_{PI}$ detects this situation of arbitrary resolution and collapses the corresponding branches, thus the final *PhySIC* supertree allows for a group (*Pan*, *Homo*). MMC displays the same polytomy as $PhySIC_{PC}$ but with a different meaning: the interleaving interpretation of this soft polytomy means that MMC does not reject the expected resolution, namely grouping (*Pan*, *Homo*) as a sister clade of *Hylobates*. In conclusion, both MMC and *PhySIC* allows for the expected group (*Pan*, *Homo*), but note that the *PhySIC* supertree is more accurate, as its polytomy does not allow the catarrhine taxa *Homo*, *Hylobates* or *Pan* to branch within the platyrrhines. Here, MRP does not introduce arbitrary resolutions, and proposes a polytomy involving the 5 catarrhine taxa.

Another problem of MMC and $PhySIC_{PC}$ supertrees is that *Lepilemur* is the sister-group of all Lemuriformes but *Daubentonia*, whereas this topological information is not present in the only source tree (SINEs) for which *Lepilemur* is scored. This result is explained by the fact that the restriction of IRBP and ADRA2B source topologies to taxa lettered *A*-*B*-*C*-*X* leads to the situation described on Fig. 6. Thanks to the PI property, the *PhySIC* algorithm again corrects this problem, and displays a polytomy involving the major clades of lemuriformes, together with *Lepilemur* (Fig. 12). The same polytomy is also proposed by MRP. Overall, this first case study illustrates that the two properties introduced in the present work help to identify and manage the potential arbitrary and conflicting resolutions arising in supertrees when combining independent source topologies.

Second example: a *PhySIC* supertree of primate genera

Primary data and source tree inference.—We used 24 datasets to reconstruct the primate phylogeny: two mitochondrial DNA (mtDNA), 19 nuclear DNA, and three transposable elements datasets. All sequences used in this study were retrieved from EMBL-Genbank databases. The sampling of genes and other molecular markers is detailed in Table 1. The corresponding data are available under TreeBASE accession numbers XXXXX. This combined dataset encompasses 95% of all primate extant genera (Wilson and Reeder, 2005), i.e., 66 genera. Two subfossil genera from ancient DNA analyses were also included (Karanth et al., 2005). All genes were aligned with Clustal X (Thompson et al., 1997) with subsequent manual refinement. We used *Mus* and *Rattus* as outgroups in all analyses for which sequence data was available. Each gene was analyzed with the ML criterion under the best fitting model (Table 1). Separate ML phylogenetic reconstructions and bootstrap analyses were performed with PHYML (Guindon and Gascuel, 2003) as described in previous section. Maximum parsimony phylogenetic reconstruction and bootstrap analysis on the three transposable element datasets were also conducted using PAUP* as described in previous section. Clades of the source trees with BP values above a specified threshold were retained. To evaluate the influence of this parameter, five *PhySIC* supertrees were inferred by respectively considering BP $\geq$ 50%, 60%, 70%, 80%, and 90%. Each run of *PhySIC* took less than 4 seconds on an Intel MacBook.

The major clades of primate genera.—The most resolved supertree reconstructed by *PhySIC* is obtained when source trees were restricted to nodes supported by more than 70% bootstrap (Fig. 13, BP$\geq$ 70%). This topology conforms to current ideas on primate phylogeny, and is close to the informal supertree of Primates at the genus level proposed by Goodman et al. (2005). In addition, we here extend their taxon sampling with the three extant genera *Euoticus*, *Piliocolobus*,

and *Simias*, and the two subfossil genera *Megaladapis* and *Paleopropithecus*. Our supertree displays the fundamental dichotomy among Primates between Strepsirrhini and Haplorrhini. Strepsirrhines then split into Lorisiformes (*Lorises* and *Galagos*) and Lemuriformes (*lemurs* and *Daubentonia*, the aye-aye). Haplorrhines also split into Tarsiers and Anthropoids. The latter clade subsequently divides into monophyletic New World primates (Platyrrhini) and Old World primates (Catarrhini). Platyrrhini display a trifurcation involving the three families Atelidae (the *Ateles* + *Alouatta* clade), Pitheciidae (the *Pithecia* + *Callicebus* clade), and Cebidae (the *Cebus* + *Saimiri* + *Aotus* + *Saguinus* clade). Catarrhines split into Hominoidea (gibbons and apes) and Cercopithecoidea (colobines and cercopithecines).

Identifying and labeling the causes of supertree polytomies.—Since veto methods are used for evaluating the topological congruence of source trees, and for measuring their degree of leaf overlap, the *PhySIC* program outputs labels on each polytomous node. A label "`C`" (standing for *Contradiction*) indicates that the polytomy results from contradictions among the source trees on phylogenetic relationships of corresponding taxa: proposing a resolution for the polytomy would contradict at least one source tree, i.e., would not respect the PC property. A label "`I`" (standing for *Induction*) indicates a lack of cross-information in the source trees: any dichotomous resolution of the clade would be at least partially arbitrary, thus would not respect the PI property. Note that a given label applies only to the node to which it is assigned but not to other nodes in its subtrees. For instance, in the primate genera supertree (Fig. 13), the platyrrhine trifurcation (Atelidae, Pitheciidae, Cebidae) with a `C` label indicates that there is topological contradiction among the source trees about the sister-group relationships of these three families. However, the `C` label does not put the monophyly of Atelidae, Pitheciidae, and Cebidae into question. Note also that a same polytomy can be characterized by both `C` and `I` labels. This means that the unability of the supertree to propose a dichotomous resolution is partly due to a lack of taxonomic

overlap, and partly due to contradictions. For example, Fig. 13 shows that the clade *Cercopithecus*, *Erythrocebus*, *Chlorocebus*, and *Miopithecus* is tagged by both `C` and `I`, reflecting two problems. On the one hand, source topologies disagree about the placement of *Erythrocebus*: this genus is either related to *Cercopithecus* (as suggested by IRBP exon 1) or to *Chlorocebus* (*cf.* the TSPY and chromosome Xq13.3 markers, and the Alu of Xing et al. (2005)). On the other hand, the input trees analyzed here do not provide the information required to know whether *Miopithecus* is the sister group of *Cercopithecus*, or is that of *Erythrocebus + Chlorocebus*, or is the most basal genus in the clade.

Impact of the robustness of source trees on veto supertree resolution.—The number of clades retained from the original source trees depends on the bootstrap threshold imposed to select them for supertree inference. Choosing a low threshold thus increases the number of retained source clades, hence lowers the number of polytomies due to a lack of cross-information among source trees, but increases the number of polytomies due to conflicts among source trees. Increasing the threshold has the opposite effect. The primate supertree of Fig. 13 was obtained with BP $\geq 70\%$. Lowering the threshold to BP $\geq 50\%$ or BP $\geq 60\%$, *PhySIC* yields a completely multifurcating supertree, due to weakly supported clades that conflict among source trees. When the bootstrap stringency is increased from the BP $\geq 70\%$ to BP $\geq 80\%$ threshold, a similar level of resolution in the genus level phylogeny is obtained with the exception of two additional polytomies: the first involves Indriidae (*Indri + Avahi + Propithecus + Paleopropithecus*) relative to other lemuriformes, and the second involves *Allenopithecus* relative to the *Cercopithecus* clade (white stars in Fig. 13 refer to disappearing branches). Interestingly, increasing the threshold removes a topological conflict among *Lophocebus*, *Papio* and *Theropithecus*: with the PC property being satisfied, then the *PhySIC* supertree groups together the latter two genera. At the BP $\geq 90\%$ threshold, 7 additional polytomies with respect to the $BP \geq 70\%$ topology appear (Fig. 13: black stars refer

to node collapsing). This reflects the fact that less source nodes (i.e., the nodes of source trees) are available for supertree inference. The PI property is thus less often satisfied in the $PhySIC_{PC}$ supertree, leading to a greater number of irresolutions in the *PhySIC* supertree.

Overall, two reasons can lead the *PhySIC* method to propose a poorly resolved supertree. First, it is possible that the source trees contain too little cross information for the method to decide how the taxa of the respective source trees branch relatively to each other. In this case, all methods, including voting methods, will produce unresolved supertrees. Obtaining more resolved supertrees can then only be achieved by adding new source trees containing new clades on the key taxa. The second reason why the *PhySIC* supertree can lack resolution is the presence of topological conflicts among source trees. Like other veto methods, *PhySIC* is very sensitive to incongruences in the source trees. Thus, to obtain a well-resolved tree, a preliminary process whereby unreliable clades are collapsed in the source trees is usually necessary before applying the method. This collapsing can be done on the basis of the support values provided on the clades by most phylogenetic inference methods (e.g., bootstrap values, Bayesian posterior probabilities, Bremer support). We showed that a well-resolved supertree of Primates can be obtained with such an approach from a non-trivial number of gene trees. Note that on some datasets, contradicting clades showing high support values can occur, e.g., due to lateral gene transfers. In such cases, veto methods will still produce unresolved supertrees (as long as they are not allowed to exclude rogue taxa). This can be seen as a drawback or as a way to pinpoint such events. In such cases, outlier source trees can be identified (Shimodaira and Hasegawa, 1999; Lerat et al., 2003) and then curated or removed from the collection of source trees, leading to a more resolved supertree.

Conclusion

Veto supertree methods are of interest for combining source topologies containing reliable clades. Their study also brings insight for the characterization of what we expect from voting methods. Indeed, when source trees are not conflicting, there is no fundamental difference between the two approaches. In such cases, veto and voting approaches should lead to *reasonable* supertrees. What *reasonable* means can be characterized by several formal properties. In the present work, we showed pitfalls of some previously proposed supertree properties, and also proposed new properties. In the general case of conflicting source trees, we believe there is still room for improvement, *e.g.,* detecting arbitrary clades of a supertree even when it partially conflicts with some source trees, as usually happens in the voting context. With the new theoretical material at hand we believe that this is a reasonable goal.

References

Adams, E. 1972. Consensus techniques and the comparison of taxonomic trees. Syst. Zool. 21:390–397.

Aho, A. V., Y. Sagiv, T. G. Szymanski, and J. D. Ullman. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comp. 10:405–421.

Bandelt, H.-J. and A. W. M. Dress. 1986. Reconstructing the shape of a tree from observed dissimilarity data. Adv. in Appl. Math. 7:309–343.

Baum, B. R. 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. Taxon 41:3–10.

Baum, B. R. and M. A. Ragan. 2004. The MRP method. Pages 17–34 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. Bininda-Emonds, ed.). Kluwer.

Berry, V. and O. Gascuel. 2000. Inferring evolutionary trees with strong combinatorial evidence. Theor. Comput. Sci. 240:217–298.

Berry, V., S. Guillemot, F. Nicolas, and C. Paul. 2005. On the approximation of computing evolutionary trees. *in* Proceedings of the 11th International Computing and Combinatorics Conference (COCOON'05) (L. Wang, ed.) LNCS.

Berry, V. and F. Nicolas. 2004. Maximum agreement and compatible supertrees. Pages 205–219 *in* Proceedings of CPM (S. C. Sahinalp, S. Muthukrishnan, and U. Dogrusoz, eds.) vol. 3109 of *LNCS*.

Berry, V. and C. Semple. 2006. Fast computation of supertrees for compatible phylogenies with nested taxa. Syst. Biol. 55:U108–U126.

Bininda-Emonds, O. R. P. 2003. Novel versus unsupported clades: assessing the qualitative support for clades in mrp supertrees. Syst. Biol. 52:839–848.

Bininda-Emonds, O. R. P. 2004a. The evolution of supertrees. Trends Ecol. Evol. 19:315–322.

Bininda-Emonds, O. R. P. 2004b. Phylogenetic supertrees (combining information to reveal the tree of life) vol. 4 of *computational biology series*. Kluwer academic publishers.

Bininda-Emonds, O. R. P. and H. N. Bryant. 1998. Properties of matrix representation with parsimony analyses. Syst. Biol. 47:497–508.

Bryant, D. 1997. Building trees, hunting for trees and comparing trees: theory and method in phylogenetic analysis. Ph.D. thesis University of Canterbury, Department of Mathemathics.

Bryant, D. and V. Berry. 2001. A structured family of clustering and tree construction methods. Adv. in Appl. Math. 27:705–732.

Bryant, D., C. Semple, and M. Steel. 2004. Supertree methods for ancestral divergence dates and other applications. chap. 6, Pages 129–150 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. Bininda-Emonds, ed.) vol. 4 of *Computational Biology Series*. Kluwer academic publishers.

Chen, D., L. Diao, O. Eulenstein, D. Fernández-Baca, and M. J. Sanderson. 2003. Flipping: A supertree construction method. Pages 135–160 *in* Bioconsensus vol. 61 of *Series in Discrete Math. and Theoretic Computer Science* DIMACS Am. Math. Soc., Providence.

Cotton, J. A., C. S. C. Slater, and M. Wilkinson. 2006. Discriminating supported and unsupported relationships in supertrees using triplets. Syst. Biol. 55:345–350.

Criscuolo, A., V. Berry, E. J. P. Douzery, and O. Gascuel. 2006. SDM: a fast distance-based approach for (super)tree building in phylogenomics. Syst. Biol. 55:740–755.

Daniel, P. 2004. Supertree methods: some new approaches. Master's thesis University of Canterbury.

Daniel, P. and C. Semple. 2004. Supertree algorithms for nested taxa. chap. 7, Pages 151–171 *in* Phylogenetic supertrees: combining information to reveal the Tree of Life (O. Bininda-Emonds, ed.) vol. 4. Kluwer academic publishers.

Daubin, V., M. Gouy, and G. Perrière. 2002. A phylogenomic approach to bacterial phylogeny: Evidence of a core of genes sharing a common history. Genome Res. 12:1080–1090.

Davies, T. J., T. G. Barraclough, M. W. Chase, P. S. Soltis, D. E. Soltis, and V. Savolainen. 2004.

Darwin's abominable mystery: Insights from a supertree of the angiosperms. Proc. Natl. Acad. Sci. USA 101:1904–1909.

Dekker, M. C. 1986. Reconstruction methods for derivation trees. Master's thesis University of Amsterdam.

Delsuc, F., H. Brinkmann, and H. Philippe. 2005. Phylogenomics and the reconstruction of the tree of life. Nat. Rev. Genet. 6:361–375.

Dutheil, J., S. Gaillard, E. Bazin, S. Glemin, V. Ranwez, N. Galtier, and K. Belkhir. 2006. Bio++: a set of C++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. BMC Bioinformatics 7:188.

Goloboff, P. A. 2005. Minority-rule supertrees? MRP, compatibility, and MinFlip may display the least frequent groups. Cladistics 21:282–294.

Goloboff, P. A. and D. Pol. 2002. Semi-strict supertrees. Cladistics 18:514–525.

Goodman, M., L. I. Grossman, and D. E. Wildman. 2005. Moving primate genomics beyond the chimpanzee genome. Trends Genet. 21:511–517.

Gordon, A. G. 1986. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labelled leaves. J. Classif. 3:335–348.

Grunewald, S., M. A. Steel, and M. S. Swenson. 2006. Closure operations in phylogenetics mathematical Biosciences (in press).

Guillemot, S. and V. Berry. 2007. Finding a largest subset of rooted triples identifying a tree is an NP-hard task. Tech. rep. LIRMM, Univ. Montpellier 2.

Guindon, S. and O. Gascuel. 2003. A simple, fast and accurate method to estimate large phylogenies by maximum-likelihood. Syst. Biol. 52:696–704.

Huson, D. H., S. M. Nettles, and T. J. Warnow. 1999. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. J. Comput. Biol. 6:369–386.

Karanth, K. P., T. Delefosse, B. Rakotosamimanana, T. J. Parsons, and A. D. Yoder. 2005. Ancient DNA from giant extinct lemurs confirms single origin of Malagasy primates. Proc. Natl. Acad. Sci. USA 102:5090–5095.

Lapointe, F.-J. and G. Cucumel. 1997. The average consensus procedure: Combination of weighted trees containing identical or overlapping sets of taxa. Syst. Biol. 46:306–312.

Lerat, E., V. Daubin, and N. A. Moran. 2003. From gene trees to organismal phylogeny in prokaryotes: The case of the gamma-proteobacteria. PLoS Biology 1:101–109.

Margush, T. and F. McMorris. 1981. Consensus n-trees. Bull. Math. Biol. 43:239–244.

Neumann, D. A. 1983. Faithful consensus methods for n-trees. Matematical Biosciences 63:271–287.

Page, R. D. M. 2002. Modified mincut supertrees. Pages 537–552 *in* Proceedings of the 2nd International Workshop on Algorithms in Bioinformatics (WABI'02) (R. Guigó and D. Gusfield, eds.).

Pisani, D. and M. Wilkinson. 2002. Matrix representation with parsimony, taxonomic congruence, and total evidence. Syst. Biol. 51:151–155.

Poux, C., P. Chevret, D. Huchon, W. W. de Jong, and E. J. P. Douzery. 2006. Arrival and

diversification of caviomorph rodents and platyrrhine primates in South America. Syst. Biol. 55:228–244.

Poux, C. and E. J. P. Douzery. 2004. Primate phylogeny, evolutionary rate variations, and divergence times: A contribution from the nuclear gene IRBP. Am. J. Phys. Anthropol. 124:1–16.

Purvis, A. 1995a. A composite estimate of primate phylogeny. Philos. Trans. R. Soc. Lond. B Biol. Sci. 348:405–421.

Purvis, A. 1995b. A modification to baum and ragan's method for combining phylogenetic trees. Syst. Biol. 44:251–255.

Ragan, M. A. 1992. Matrix representation in reconstructing phylogenetic relationships among the eukaryots. Biosystems 28:47–55.

Roos, C., J. Schmitz, and H. Zischler. 2004. Primate jumping genes elucidate strepsirrhine phylogeny. Proc. Natl. Acad. Sci. USA 101:10650–10654.

Ross, H. A. and A. G. Rodrigo. 2004. An assessment of matrix representation with compatibility in supertree construction. *in* Phylogenetic supertrees (combining information to reveal the tree of life) (O. R. P. Bininda-Emonds, ed.) vol. 4. Kluwer academic publishers.

Sanderson, M. J., A. Purvis, and C. Henze. 1998. Phylogenetic supertrees: assembling the trees of life. Trends Ecol. Evol. 13:105–109.

Semple, C. and M. Steel. 2000. A supertree method for rooted trees. Discrete Appl. Math. 105:147–158.

Semple, C. and M. A. Steel. 2003. Phylogenetics vol. 24 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press.

Shimodaira, H. and M. Hasegawa. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. Mol. Biol. Evol. 16:1114–1116.

Singer, S. S., J. Schmitz, C. Schwiegk, and H. Zischler. 2003. Molecular cladistic markers in New World monkey phylogeny (Platyrrhini, Primates). Mol. Phylogenet. Evol. 26:490–501.

Steel, M. A. 1992. The complexity of reconstructing trees from qualitative characters and subtree. J. Classif. 9:91–116.

Steel, M. A., A. W. M. Dress, and S. Böcker. 2000. Simple but fundamental limitations on supertree and consensus tree methods. Syst. Biol. 49:363–368.

Swofford, D. L. 2002. PAUP*. Phylogenetic Analysis Using Parsimony (* and Other Methods). Version 4b10. Sinauer Associates, Sunderland, Massachusetts version 4.0b2.

Thompson, J. D., T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins. 1997. The clustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. Nucl. Acids Res. 24:4876–4882.

Thorley, J. L. and M. Wilkinson. 2003. A view of supertrees methods. Pages 185–194 *in* Bioconsensus (M. F. Janowitz, F.-J. Lapointe, F. R. McMorris, and F. S. Roberts, eds.) vol. 61 of *Discrete Mathematics and Theoretical Computer Science* DIMACS.

Wilkinson, M. 1994. Common cladistic information and its consensus representation: reduced adams and reduced cladistic consensus trees and profiles. Syst. Biol. 43:343–368.

Wilkinson, M., D. Pisani, J. A. Cotton, and I. Corfe. 2005. Measuring support and finding unsupported relationships in supertrees. Syst. Biol. 54:823–831.

Wilkinson, M., J. Thorley, D. Pisani, F.-J. Lapointe, and O. McInerney. 2004. Some desiderata for liberal supertree. Pages 227–246 *in* Phylogenetic supertrees (combining information to reveal the tree of life) (O. R. P. Bininda-Emonds, ed.) vol. 4. Kluwer academic publishers.

Wilson, D. E. and D. M. Reeder. 2005. Mammal species of the world. Johns Hopkins University Press, Baltimore.

Xing, J., H. Wang, K. Han, D. A. Ray, C. H. Huang, L. G. Chemnick, C.-B. Stewart, T. R. Disotell, O. A. Ryder, and M. A. Batzer. 2005. A mobile element based phylogeny of old world monkeys. Mol. Phylogenet. Evol. 37:872–880.

Acknowledgments

We thank J. Cotton, R. Page, O. Bininda-Emonds and an anonymous reviewer for many invaluable remarks on a first version of this manuscript. This work has been supported by the "ACI Informatique-Mathématique-Physique en Biologie Moléculaire [ACI IMP-Bio]", by the "Action incitative BIOSTIC-LR", by IFR119 "Biodiversité Continentale Méditerranéenne et Tropicale" (Montpellier) and by the Research Networks Program in BIOINFORMATICS of the High Council for Scientific and Technological Cooperation between France and Israël. This publication is the contribution N^o 2007-053 of the Institut des Sciences de l'Évolution de Montpellier (UMR 5554 - CNRS).

Appendix

Proof of Proposition 1

Proof. The proof is based on the fact that a set $\mathcal{R}$ identifies a tree T if and only if $rt(T) = cl(\mathcal{R})$ (Grunewald et al., 2006, Lem. 2.1).

$\Rightarrow$ By definition, $\mathcal{R}(T, \mathcal{T})$ only contains triplets on 3-taxon sets for which there is a resolution in T. Moreover, PC ensures that any such triplet cannot be resolved in $\mathcal{R}(T, \mathcal{T})$ differently than that in T. It follows that $\mathcal{R}(T, \mathcal{T}) \subseteq rt(T)$. $\mathcal{R}(T, \mathcal{T})$ is therefore compatible and $cl(\mathcal{R}(T, \mathcal{T})) \subseteq cl(rt(T)) = rt(T)$. Meanwhile, having proved that $\mathcal{R}(T, \mathcal{T})$ is compatible, it is clear from PI that $cl(\mathcal{R}(T, \mathcal{T})) \supseteq rt(T)$.

$\Leftarrow$ Having $\mathcal{R}(T, \mathcal{T})$ identifying T ensures that $\mathcal{R}(T, \mathcal{T})$ is compatible. On one hand, $cl(\mathcal{R}(T, \mathcal{T})) = rt(T)$ implies that for all $t \in rt(T)$, $t \in cl(\mathcal{R}(T, \mathcal{T}))$, i.e., $\mathcal{R}(T, \mathcal{T}) \vdash t$ and therefore PI holds. On the other hand, given $t \in rt(T)$, if $\mathcal{R}(T, \mathcal{T}) \vdash \bar{t}$ then, by definition of the closure, $\bar{t} \in cl(\mathcal{R}(T, \mathcal{T})) = rt(T)$ so that both t and $\bar{t}$ are in $rt(T)$, which is not possible. This proves that $\mathcal{R}(T, \mathcal{T}) \nvdash \bar{t}$ for any $t \in rt(T)$, i.e., PC holds.

□

Proof of Proposition 3

Proof. Let $AB|C \in rt(T)$ be a triplet of the output supertree T and consider the recursive step where the tree returned in line 3 hosts A, B in the same subtree T_i, while C is in another subtree, say T_j. This means that A, B are vertices in a connected component C_i of the current Aho graph G, while C is in another component C_j of $CC(G)$. If $\mathcal{R}$ contained $AC|B$ or $BC|A$, then C_i and C_j

would not have been distinct connected components (in such cases, graph G would have contained the edge (A, C) or (B, C)). Thus, for any triplet t of $rt(T)$, we know that no triplet $\bar{t}$ is in $\mathcal{R}$, i.e., T does not directly contradict $\mathcal{R}$. This ensures that $Build_{PC}$ returns a tree satisfying PC (lem. 1).

□

Proof of Theorem 1

Proof. Correctness of the algorithm: The triplets present in the tree T_{PC} returned by $PhySIC_{PC}$ depend on the internal branches of this tree. Any internal branch of T_{PC} is created at line `10` during some recursive step, thus linking a subtree T_i to the root of the tree returned by this step. Thanks to Lem. 1, we just have to prove that every branch created at this line does not generate a triplet that directly contradicts $\mathcal{R}$.

At this line, either (i) $\mathcal{C}_{PC}$ corresponds exactly to the connected components of G (we have $|CC(G)| > 1$) and then, from Prop. 3, the triplets created are not in direct contradiction with $\mathcal{R}$; or (ii) $\mathcal{C}_{PC}$ corresponds to a partition of the vertices of G based on $\mathcal{R}'$. Two cases are then possible. In the first case (line `4`), G' is connected and the current call returns a multifurcation that generates no triplet and hence does not invalidate PC. In the second case, the `Repeat` loop ensures that for each set of three taxa $A, B, C \in v(G)$, $\mathcal{C}_{PC}$ does not contain two elements C_i and C_j with $A, B \in C_i$ and $C \in C_j$ such that either $AC|B$ or $BC|A$ belongs to $\mathcal{R} = \mathcal{R}' \cup \mathcal{R}_{dc}$. Indeed, $AC|B$ (and $BC|A$) cannot be in $\mathcal{R}_{dc}$ since C_i would have been removed from $\mathcal{C}_{PC}$ (line `9`). Moreover, if $AC|B$ (resp. $BC|A$) was in $\mathcal{R}'$, then A and C (resp. B and C) would have been in the same component of $\mathcal{C}_{PC}$ contradicting $C_i \neq C_j$. This proves that the tree T_{PC} built in line `10`, and whose subtrees bijectively correspond to the elements of $\mathcal{C}_{PC}$, is such that no triplet of $rt(T_{PC})$

directly contradicts $\mathcal{R}$.

Time complexity of the algorithm. The most time consuming operations in $PhySIC_{PC}$ are the computation of $\mathcal{R}'|v(C_i)$ and G'_i (line `8`), and that of the connected components of this graph (line `9`). Obtaining $\mathcal{R}'|v(C_i)$ and constructing G'_i requires considering each triplet of $\mathcal{R}$ at most once and thus has a time complexity of $O(n^3)$. Determining the $CC(G'_i)$s costs $O(n^2)$ (which is the maximum number of edges for a graph with n vertices). During the whole set of recursive calls to $PhySIC_{PC}$, $\mathcal{C}_{PC}$ is modified at most $O(n)$ times (proportional to the number of clades of a tree with n leaves). Lines `8` and `9` are executed as many times as $\mathcal{C}_{PC}$ is modified, i.e., $O(n)$ times. Thus, for the whole set of recursive calls to $PhySIC_{PC}$, the computation time required by these critical lines is $O(n^4)$, which is also the complexity of the entire procedure. □

Proof of Theorem 2

Proof. Correctness of the algorithm: we first prove that $PhySIC_{PI}$ always returns a tree, denoted by T_{PI}, and then that T_{PI} satisfies both PC and PI. By hypothesis, $PhySIC_{PI}$ is called with a tree T_{PC} that satisfies PC. Since $PhySIC_{PI}$ modifies this tree by only collapsing some of its branches (possibly none), the tree considered in any execution $Check_{PI}$ never directly contradicts $\mathcal{T}$. This ensures that the $Check_{PI}$ subroutine never exits in line `12`, i.e., $PhySIC_{PI}$ always returns a tree. Moreover, being a contraction of T_{PC}, this tree satisfies PC.

$Check_{PI}$ differs from the *Identifies* algorithm in that `Return "tree not identified"` in the latter is replaced by line `14` in the former. Given a tree T and a set $\mathcal{R}$ of triplets, $Identifies(T, \mathcal{R})$ returns `yes` iff $\mathcal{R}$ identifies T (otherwise it returns `no`) (Daniel, 2004, Thm. 3.1.1). This ensures that when a call to $Check_{PI}(T, \mathcal{R}_{PI})$ issued by $PhySIC_{PI}$ in line `11` collapses no branch of T_{PI},

then the set $\mathcal{R}_{PI}$ identifies this tree. Since the tree T_{PI} returned by $PhySIC_{PI}$ is such that it is not modified by the last run of $Check_{PI}$, then T_{PI} is identified by $\mathcal{R}(T_{PI}, \mathcal{T}) = \mathcal{R}_{PI}$. In other words, T_{PI} satisfies both PI and PC for $\mathcal{T}$ (Prop. 1).

Time complexity of the algorithm. As for $PhySIC_{PC}$, the most time consuming operations done by $PhySIC_{PI}$ are the construction of the Aho graph G_{ij} and the computation of its connected components in the $Check_{PI}$ subroutine. The G_i graphs that may be used in $Check_{PI}$ can be precomputed in the $PhySIC_{PI}$ part of the pseudo-code (i.e., before calling $Check_{PI}$), knowing $rt(\mathcal{T})$ and the current tree T_{PI} to be examined in $Check_{PI}$. This preprocess clearly requires $O(n^4)$ time, since there are $O(n)$ such graphs (one for each clade of T), each of which is obtained by examining the $O(n^3)$ triplets of $\mathcal{R}(T_{PI}, \mathcal{T})$. Each G_{ij} graph can be obtained from a copy of the corresponding G_i graph, completed by the edges due to triplets $ab|c$ having $a, b \in C_i$ and $c \in C_j$. All the G_{ij} graphs required during the recursive calls to $Check_{PI}$ resulting from an execution of line `11` in $PhySIC_{PI}$ can also be precomputed in the $PhySIC_{PI}$ pseudo-code part. This can be done just before line `11`, provided that $Check_{PI}$ is modified to end as soon as an edge is collapsed (line `14`) – it is clear that this slight modification does not modify the correctness of the algorithm. Indeed, the only G_{ij}s that are then required by $Check_{PI}$ are those corresponding to two sibling clades C_i and C_j of the current T_{PI} tree. Computing all of these G_{ij}s before line `11` of $PhySIC_{PI}$ is done in $O(n^3)$ since each triplet $ab|c$ of $rt(T_{PI})$ adds an edge between A and B in the one and only graph G_{ij}, such that C_i and C_j are sibling clades in T_{PI} and $A, B \in C_i$ and $C \in C_j$.

Note also that the only information used by $Check_{PI}$ on graph G_i and G_{ij} is the number of their connected components. The total number of edges present in the G_{ij} graphs is in $O(n^3)$: precomputation of the number of connected components for this set of graphs is thus globally $O(n^3)$ time. As this has to be done at each pass of the `Repeat` loop, and as this loop is done at

most $O(n)$ times (each pass results in the collapsing of one of the $O(n)$ clades of T), this part of the computation is globally (on the whole for $PhySIC_{PI}$) in $O(n^4)$ time. Determination of the number of connected components of each G_i is done only once just before the `Repeat` loop. For each of these $O(n)$ graphs, this requires examining $O(n^3)$ triplets. Thus, this preprocess also costs $O(n^4)$ time. The preprocesses done for G_i and G_{ij} graphs thus requires $O(n^4)$ time and reduces the running time of $Check_{PI}$. The modification of $Check_{PI}$, consisting of returning to $PhySIC_{PI}$ as soon as an edge is collapsed, also simplifies the algorithm (e.g., the `Repeat` loop is no longer required).

Thanks to the preprocessing, the only time-consuming operation in $Check_{PI}$ for the current tree T_{PI} is the examination of the $O(n^2)$ pairs of sibling clades C_i and C_j of this tree. Operations performed for each of this pair of clades is in $O(1)$ (the number of connected components of useful graphs G, G_i and G_{ij} have been preprocessed). Since a new tree T_{PI} can only be obtained by collapsing one of the $O(n)$ edges (line `14`) of T_{PI}, this can at most occur $O(n)$ times. Therefore, all executions of $Check_{PI}$ issued by a run of $PhySIC_{PI}$ are in $O(n^3)$ time. Thus, the whole complexity of the $PhySIC_{PI}$ algorithm is no more than the cost of the preprocessing, i.e., $O(n^4)$ time.

□

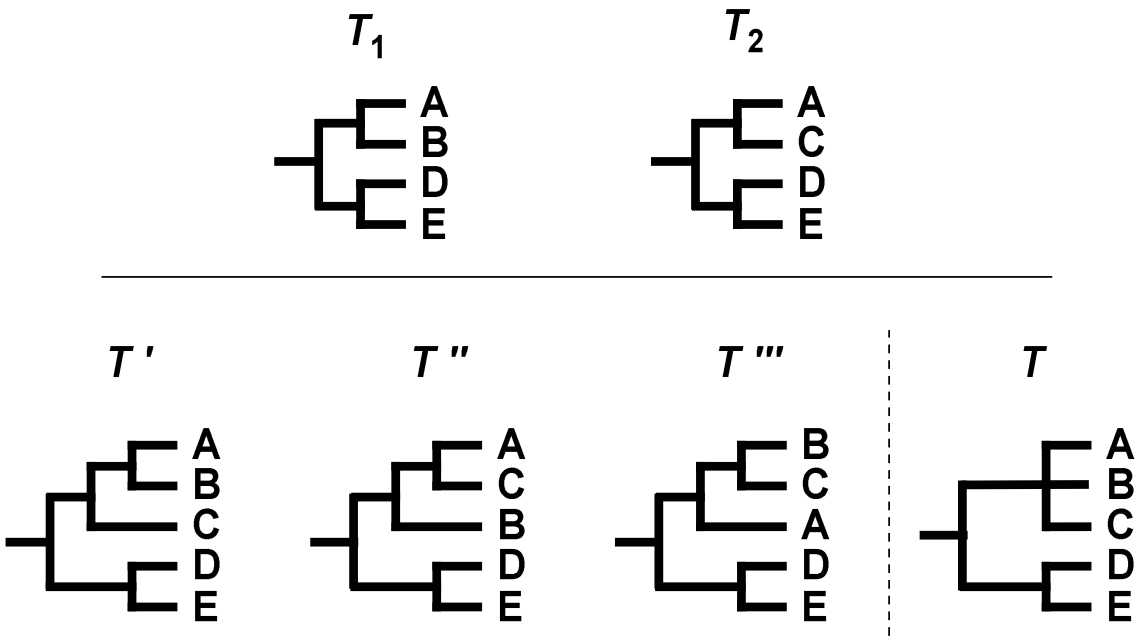

Figure 1:

Figure 1: Supertrees can contain arbitrary resolution.

Example of a collection $\mathcal{T} = \{T_1, T_2\}$ of two source trees and several possible supertrees T', T'', T''' and T. Unlike T, the supertrees T', T'', T''' propose an arbitrary resolution for the clade A, B, C.

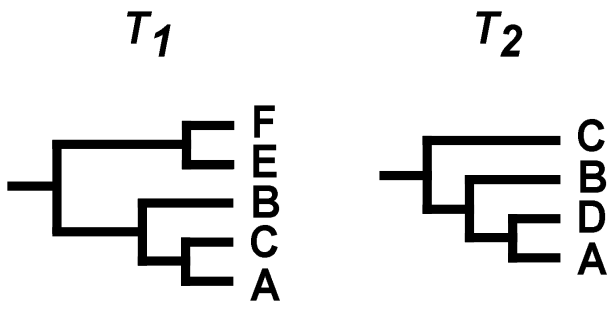

Figure 2:

Figure 2: Example of a collection $\mathcal{T} = \{T_1, T_2\}$ of two source trees.

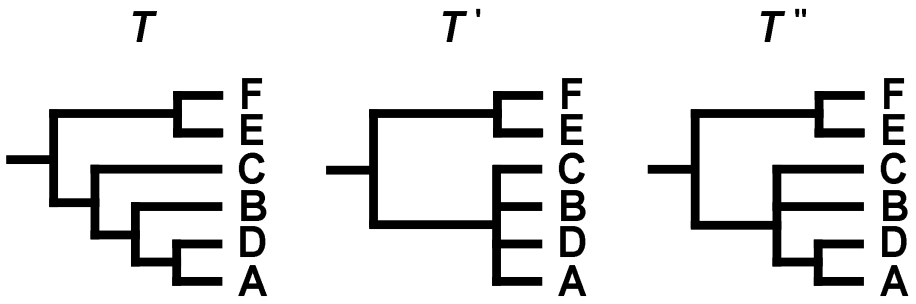

Figure 3:

Figure 3: Various supertrees for the collection of Fig. 2.

T is the supertree proposed by the MC (Semple and Steel, 2000) and MMC methods (Page, 2002). T' and T'' are the supertrees respectively proposed by the $Build_{PC}$ and $PhySIC_{PC}$ algorithms described in this paper.

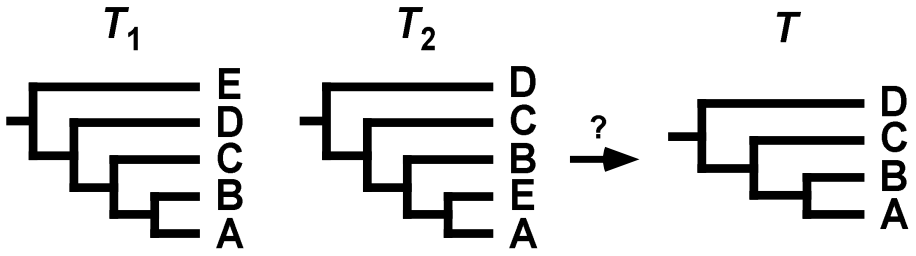

Figure 4:

Figure 4: Excluding rogue taxa from the analysis can lead to informative supertrees.

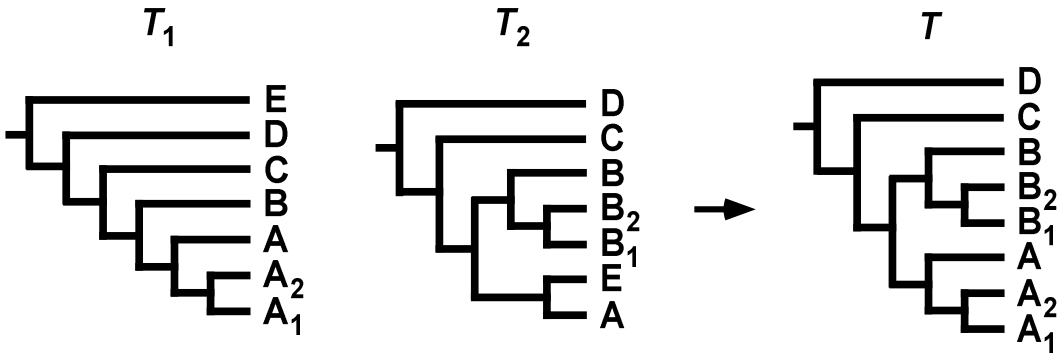

Figure 5:

Figure 5: Another example with rogue taxa.

This figure presents a generalization of the example displayed in Fig. 4 to the case of a supertree containing more taxa than each input tree.

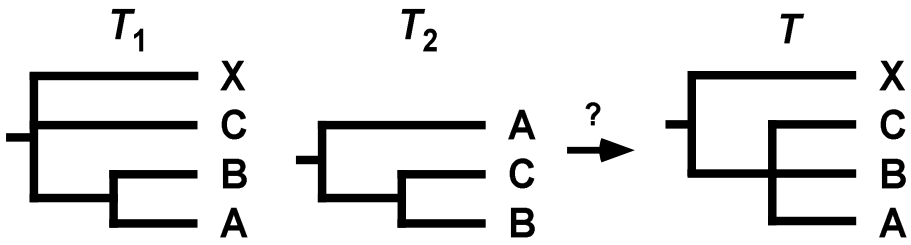

Figure 6:

Figure 6: Contradiction in the source trees can lead to arbitrary resolution. An example where the presence of contradiction in the source trees (namely, $AB|C$ in T_1 versus $BC|A$ in T_2) can lead to the inferrence of arbitrary clades (namely excluding X from the clade $\{A, B, C\}$ in the supertree T). This problem is detected by PI but not by PI' nor PC'.

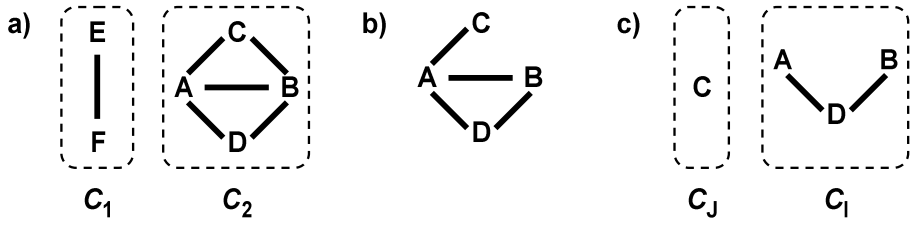

Figure 7:

Figure 7: Examples of graphs.

(a) The initial Aho graph G created from the triplets $rt(\mathcal{T})$ of the collection $\mathcal{T}$ displayed in Fig. 2. The two connected components of G are $C_1 = \{E, F\}$ and $C_2 = \{A, B, C, D\}$. (b) the Aho graph obtained from $\mathcal{R}|v(C_2)$. This graph is connected, showing that the input trees conflict on the resolution of $\{A, B, C, D\}$, hence are incompatible. (c) the Aho graph obtained from $\mathcal{R}|v(C_2)$ when removing the triplets $\mathcal{R}_{dc} = \{AB|C, AC|B\}$.

Algorithm $Build_{PC}\,(S, \mathcal{R})$

if *S contains less than 3 taxa* **then** return the trivial tree on S
Let G denote the Aho graph for $\mathcal{R}$
if *G has only one connected component* **then**
1 return the star tree on $L(\mathcal{R})$
else
$\mathcal{C}_{PC} \leftarrow CC(G)$
foreach $C_i \in \mathcal{C}_{PC}$ **do**
2 **if** $(\mathcal{R}|v(C_i)) = \emptyset$ **then** $T_i \leftarrow$ star tree on $v(C_i)$
else $T_i \leftarrow Build_{PC}\,(v(C_i), \mathcal{R}|v(C_i))$
3 Return the tree made of a root node connected to $T_1, T_2, ..., T_{|\mathcal{C}_{PC}|}$

Figure 8:

Figure 8: Details of the $Build_{PC}$ subroutine taking a set S of taxa and a set $\mathcal{R}$ of triplets on S as input.

Algorithm $PhySIC_{PC}(S, \mathcal{R})$

if *S contains less than 3 taxa* **then** return the trivial tree on S
Let G denote the Aho graph for $\mathcal{R}$
if *G has several connected components* **then** $\mathcal{C}_{PC} \leftarrow CC(G)$
else
  Let $\mathcal{R}_{dc}$ be the set of triplets t s.t. $t, \bar{t} \in \mathcal{R}$
  $\mathcal{R}' \leftarrow \mathcal{R} - \mathcal{R}_{dc}$
  Let G' be the Aho graph for $\mathcal{R}'$
4 **if** *G' is connected* **then** $\mathcal{C}_{PC} \leftarrow v(G)$
  else
    $\mathcal{C}_{PC} \leftarrow CC(G')$
5   **repeat**
6   **foreach** $AB|C \in \mathcal{R}_{dc}$ **do**
7   **if** *$A, B \in C_i$ and $C \in C_j$ (with $C_i, C_j \in \mathcal{C}_{PC}$ and $i \neq j$)* **then**
8   Build G'_i the Aho graph for $\mathcal{R}'|v(C_i)$
           if *G'_i is connected* **then** $\mathcal{C}_{PC} \leftarrow (\mathcal{C}_{PC} - \{C_i\}) \cup v(C_i)$
9   **else** $\mathcal{C}_{PC} \leftarrow (\mathcal{C}_{PC} - \{C_i\}) \cup CC(G'_i)$
      until *$\mathcal{C}_{PC}$ no longer changes*
foreach $C_i \in \mathcal{C}_{PC}$ **do**
  if $(\mathcal{R}|v(C_i)) = \emptyset$ **then** $T_i \leftarrow$ star tree on $v(C_i)$
  else $T_i \leftarrow PhySIC_{PC}(v(C_i), \mathcal{R}|v(C_i))$
10 Return the tree made of a root node connected to $T_1, T_2, ..., T_{|\mathcal{C}_{PC}|}$

Figure 9:

Figure 9: Details of the $PhySIC_{PC}$ subroutine taking a set S of taxa and a set $\mathcal{R}$ of triplets on S as input.

Algorithm $PhySIC_{PI}(T, \mathcal{T})$

$T_{PI} \leftarrow T$
repeat
  $\mathcal{R}_{PI} \leftarrow \mathcal{R}(T_{PI}, \mathcal{T})$
  $T_{PI} \leftarrow Check_{PI}\ (T_{PI}, \mathcal{R}_{PI})$
until T_{PI} *no longer changes*
Return T_{PI}

Algorithm *PhySIC* $(\mathcal{T})$

Let S be the taxa appearing in $\mathcal{T}$
$\mathcal{R} \leftarrow rt(\mathcal{T})$
$T_{PC} \leftarrow PhySIC_{PC}(S, \mathcal{R})$
Return $PhySIC_{PI}(T_{PC}, \mathcal{R})$

Algorithm $Check_{PI}(T, \mathcal{R})$

if T *is made of a single leaf* **then** return T
Let G be the Aho graph for $\mathcal{R}$
if $|CC(G)| = 1$ **then** return "error, $\mathcal{R}$ is incompatible"
repeat
  foreach $T_i \in S(T)$ **do**
    Let G_i be the Aho graph for $\mathcal{R}|L(T_i)$
    foreach $T_j \in S(T)$ *s.t.* $T_i \neq T_j$ **do**
      Build G_{ij} from G_i and $\mathcal{R}|(L(T_i) \cup L(T_j))$
      if G_{ij} *is not connected* **then**
        Collapse the branch between the root of T and T_i
until *no branch of* T *is collapsed*
foreach $T_i \in S(T)$ **do**
  $T'_i \leftarrow Check_{PI}\ (T_i, \mathcal{R}|L(T_i))$
Return the tree made of a root node connected to $T'_1, T'_2, ..., T'_{|S(T)|}$

Figure 10:

Figure 10: Details of the *PhySIC* algorithm and $PhySIC_{PI}$ and $Check_{PI}$ subroutines.

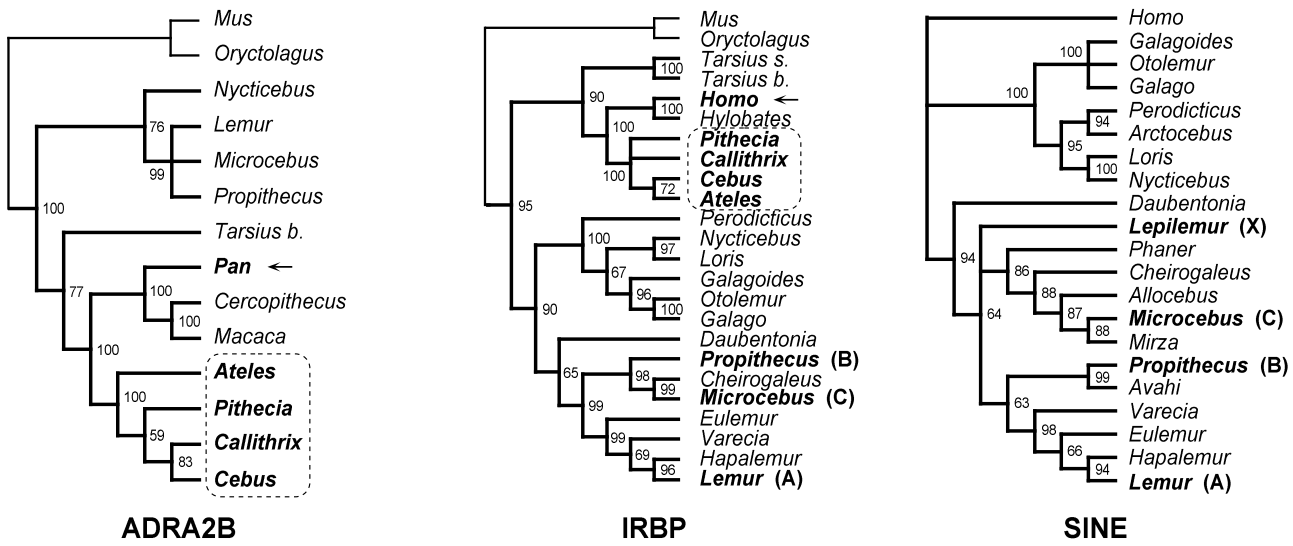
Mus
Oryctolagus
Nycticebus
Lemur
Microcebus
Propithecus
Tarsius b.
Pan
Cercopithecus
Macaca
Ateles
Pithecia
Callithrix
Cebus
ADRA2B
Mus
Oryctolagus
Tarsius s.
Tarsius b.
Homo
Hylobates
Pithecia
Callithrix
Cebus
Ateles
Perodicticus
Nycticebus
Loris
Galagoides
Otolemur
Galago
Daubentonia
Propithecus (B)
Cheirogaleus
Microcebus (C)
Eulemur
Varecia
Hapalemur
Lemur (A)
IRBP
Homo
Galagoides
Otolemur
Galago
Perodicticus
Arctocebus
Loris
Nycticebus
Daubentonia
Lepilemur (X)
Phaner
Cheirogaleus
Allocebus
Microcebus (C)
Mirza
Propithecus (B)
Avahi
Varecia
Eulemur
Hapalemur
Lemur (A)
SINE

Figure 11:

Figure 11: Three source trees for the case study on primates.

Majority rule consensus source trees derived from the bootstrap analysis of ADRA2B and IRBP sequences (in maximum likelihood), and SINE characters (in maximum parsimony). Bootstrap percentages are indicated on nodes, and only nodes defined by more than 50% are considered. Thin branches lead to the outgroup. Taxa in bold are handled differently by the different supertree algorithms and illustrate three different situations (arrows, letters, box: see main text). The two *Tarsius* species are *T. bancanus* (*B*) and *T. syrichta* (*S*).

Figure 12: Comparison of supertrees inferred by three methods: MMC, MRP and *PhySIC* (the $PhySIC_{PC}$ intermediate step of the latter is also displayed).

Thin branches lead to the outgroup. Taxa in bold are handled differently by the different supertree algorithms and illustrate three different situations. (i) Arrows indicate the surprising positioning of *Homo* and *Pan* under the MMC and $PhySIC_{PC}$ algorithms ; (ii) A-B-C-X letters correspond to taxa arbitrarily grouped by MMC and $PhySIC_{PC}$ (*cf.* Fig. 2); (iii) boxes contain platyrrhine taxa for which MMC and MRP contradict the IRBP source topology. The P label on MMC and $PhySIC_{PC}$ supertrees refers to the polytomy involving catarrhine and platyrrhine clades. The taxonomic frame for Primates is given on the *PhySIC* supertree. Hatched rectangles represent Anthropoidea (Catarrhini + Platyrrhini). White and black rectangles respectively represent Haplorrhini (Tarsiiformes + Anthropoidea) and Strepsirrhini (Lorisiformes + Lemuriformes).

Figure 13: Primate *PhySIC* supertree including 95% of all extant genera and containing no contradiction nor arbitrary resolution with respect to the source trees, as defined by the PI and PC properties.

The genus-level primate supertree has been reconstructed from 24 molecular source trees restricted to nodes supported by more than 70% bootstrap. When the bootstrap threshold is increased to 80% – respectively 90% – the supertree topology changes: disappearing branches as well as one appearing clade (*Papio* + *Theropithecus*) are indicated by white – respectively black – stars. Polytomies are labelled by tags pointing out the properties (PI, PC, or both) that would not be satisfied if the corresponding clade was more resolved. The taxonomic frame and clade names for Primates are given. Hatched rectangles represent Anthropoidea (Platyrrhini + Catarrhini). White and black rectangles respectively represent Haplorrhini (Tarsiiformes + Anthropoidea) and Strepsirrhini (Lorisiformes + Lemuriformes).

Table 1: Molecular markers used to infer the primate source trees.

The best-fitting model and the number of primate genera per marker are indicated. (mtDNA: mitochondrial DNA).

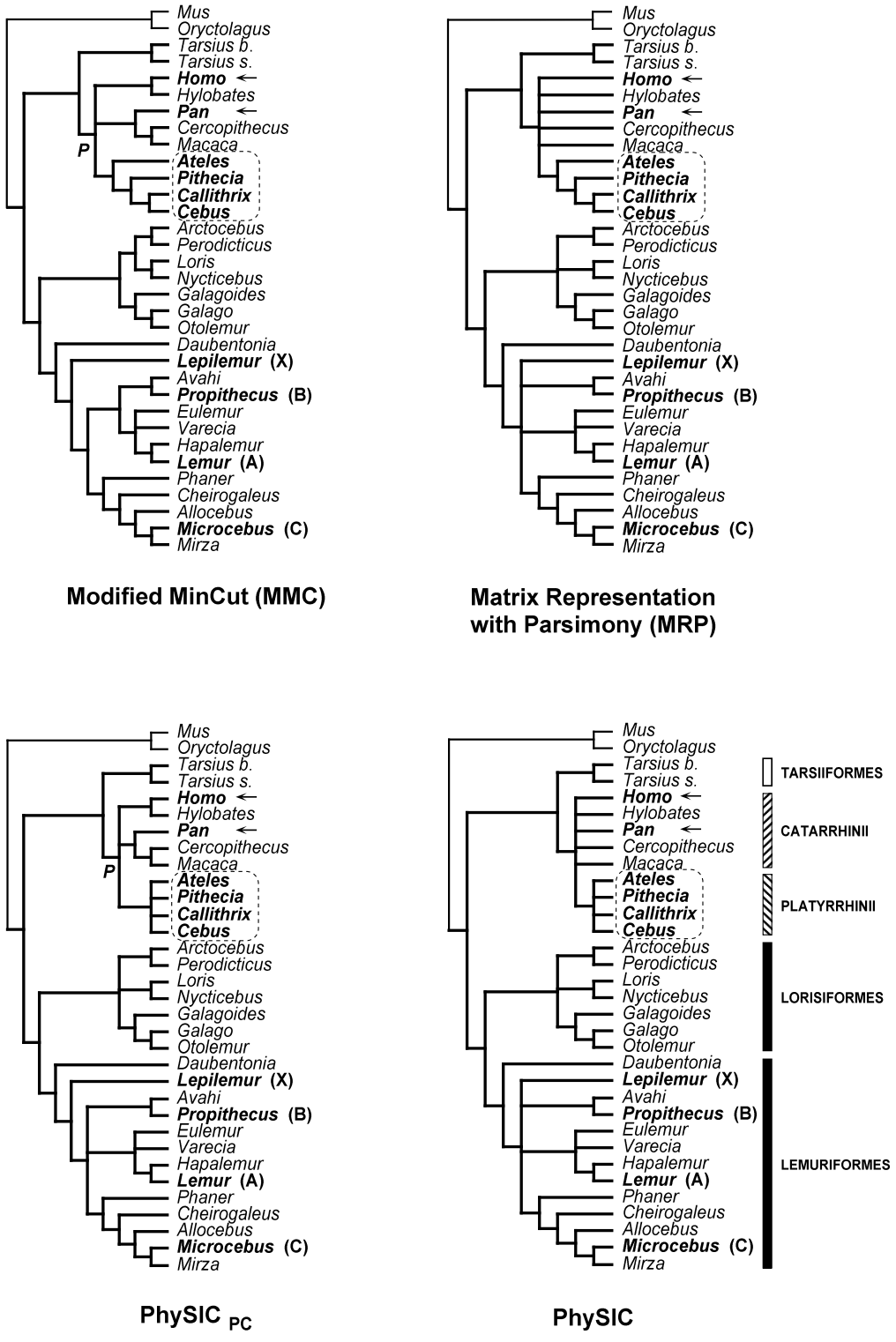
Mus
Oryctolagus
Tarsius b.
Tarsius s.
Homo
Hylobates
Pan
Cercopithecus
Macaca
P
Ateles
Pithecia
Callithrix
Cebus
Arctocebus
Perodicticus
Loris
Nycticebus
Galagoides
Galago
Otolemur
Daubentonia
Lepilemur (X)
Avahi
Propithecus (B)
Eulemur
Varecia
Hapalemur
Lemur (A)
Phaner
Cheirogaleus
Allocebus
Microcebus (C)
Mirza
Modified MinCut (MMC)
Matrix Representation with Parsimony (MRP)
PhySIC PC
PhySIC
TARSIIFORMES
CATARRHINII
PLATYRRHINII
LORISIFORMES
LEMURIFORMES

Figure 12:

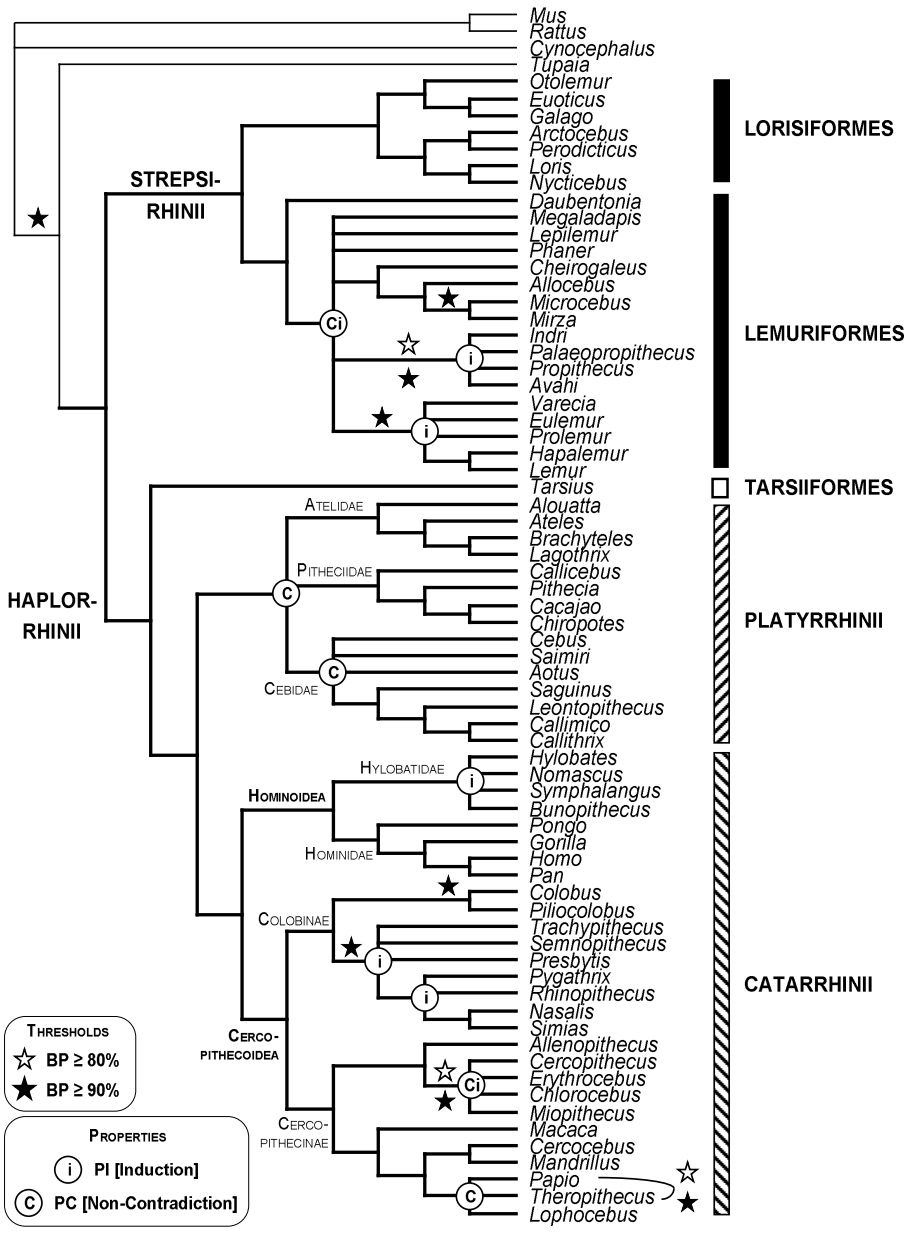
Mus
Rattus
Cynocephalus
Tupaia
Otolemur
Euoticus
Galago
Arctocebus
Perodicticus
Loris
Nycticebus
LORISIFORMES
STREPSI-
RHINII
Daubentonia
Megaladapis
Lepilemur
Phaner
Cheirogaleus
Allocebus
Microcebus
Mirza
Indri
Palaeopropithecus
Propithecus
Avahi
Varecia
Eulemur
Prolemur
Hapalemur
Lemur
LEMURIFORMES
Tarsius
TARSIIFORMES
Atelidae
Alouatta
Ateles
Brachyteles
Lagothrix
Pitheciidae
Callicebus
Pithecia
Cacajao
Chiropotes
HAPLOR-
RHINII
Cebus
Saimiri
Aotus
Cebidae
Saguinus
Leontopithecus
Callimico
Callithrix
PLATYRRHINII
Hylobatidae
Hylobates
Nomascus
Hominoidea
Symphalangus
Bunopithecus
Pongo
Hominidae
Gorilla
Homo
Pan
Colobus
Colobinae
Piliocolobus
Trachypithecus
Semnopithecus
Presbytis
Pygathrix
Rhinopithecus
Nasalis
Simias
Cerco-
pithecoidea
Allenopithecus
Cercopithecus
Erythrocebus
Chlorocebus
Miopithecus
Cerco-
pithecinae
Macaca
Cercocebus
Mandrillus
Papio
Theropithecus
Lophocebus
CATARRHINII
Thresholds
BP ≥ 80%
BP ≥ 90%
Properties
PI [Induction]
PC [Non-Contradiction]

Figure 13:

MARKERS	MODEL	GENERA
α-2B Adrenergic Receptor	HKY+Γ	16
Albumin gene introns 3 and 4	K80+I	20
ATPase 7A	HKY+Γ	11
Breast and Ovarian Cancer Susceptibility 1	HKY+Γ	12
Cytochrome b [mtDNA]	GTR+Γ+I	68
α-1,2 fucosyltransferase	HKY+Γ	21
β globin	GTR+I+Γ	22
γ globin exons 1 and 2	HKY+Γ	26
ϵ globin	HKY+Γ	33
Glucose-6-Phosphate Dehydrogenase introns 4 and 5	HKY+Γ	18
Glucose-6-Phosphate Dehydrogenase introns 7 and 8	HKY+Γ	12
Interphotoreceptor Retinoid Binding Protein exon 1	HKY+Γ	34
Interphotoreceptor Retinoid Binding Protein intron 1	K80+Γ	24
Lysozyme Gene	HKY+Γ	19
Membrane Cofactor Protein gene	HKY+Γ	13
β2 microglobulin precursor exons 1 and 2	HKY+Γ	18
NADH dehydrogenase subunit 5 [mtDNA]	GTR+Γ+I	24
Phospholipase C β4 gene	GTR	13
Testis-Specific Protein Gene	HKY+Γ	21
von Willebrand gene introns 11 and 12	HKY+Γ	37
9.3 kb chromosome Xq13.3 fragment	HKY+Γ	13
SINE (Roos et al., 2004)	–	20
ALU (Singer et al., 2003)	–	8
SINE (Xing et al., 2005)	–	15

Table 1: